ESCUELA SUPERIOR POLITECNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"ARQUITECTURA CLIENTE / SERVIDOR"

Control de Ventas e Inventario en una Farmacia

TRABAJO DE GRADUACION

Previo a la obtención del Título de:

INGENIERO EN COMPUTACION

Presentado por:

John Pacheco Rubio

Lusitania Rosero Salazar

Luis Toala Meza

Guayaquil - Ecuador

1999

AGRADECIMIENTO

A nuestros Padres por su incondicional ayuda y a la Universidad por habernos dado la oportunidad de obtener nuestro titulo profesional.

DEDICATORIA

A mis padres que con su apoyo, sacrificio y ayuda permitieron que culmine mi carrera. A mis amigos y a mi hermana Alicia, y en especial a mi amado esposo Jaime.

Lusitania Rosero Salazar

TRIBUNAL DE GRADUACION

Ing. Carlos Valero
Director de Tópico

Ing. Carlos Monsalve A.
Subdecano de la FIEC

Ing. Guido Caicedo R.
Miembro del Tribunal

Ing. Rebeca Estrada P.
Miembro del Tribunal

DECLARACION EXPRESA

"La responsabilidad por los hechos, ideas y doctrinas expuestos en esta tesis, nos corresponde exclusivamente; y el patrimonio intelectual de la misma, a la ESCUELA SUPERIOR POLITECNICA DEL LITORAL"

(Reglamentos de Exámenes y Títulos profesionales de la ESPOL)

John Pacheco Rubio **Lusitania Rosero Salazar**

Luis Toala Meza

RESUMEN

El presente trabajo implementa un sistema de Control de Ventas e Inventario en una Farmacia, basándose en el modelo 3-Tier Cliente/Servidor bajo un ambiente distribuido, utilizando el lenguaje de desarrollo Java y la arquitectura CORBA.

En los primeros capítulos se revisan las base teóricas de Cliente/Servidor y CORBA, para luego empezar a detallar los requerimientos del sistema y realizar el análisis y diseño orientado a objetos del mismo.

Para el desarrollo del lado del cliente se utilizó Applets usando la herramienta Visual Age. El desarrollo y procesamiento en el lado el servidor está basado en ORB's y connectors para lo cual se utilizaron los productos Visibroker for Java y SQL Server. El Middleware está basado en la arquitectura CORBA. El sistema está desarrollado en un ambiente distribuido para ser puesto en Internet.

El sistema será capaz de realizar las siguientes transacciones: Ventas, Consulta/Emisión de Factura, Ingreso de productos, Dar de baja a productos, Creación de nuevos productos y Eliminación de productos.

INDICE GENERAL

GLOSARIO

API: Application Protocol Interface.

Applets: Clase o subrutina que encapsula su funcionalidad, escrita en Java.

AWT: Clase de Java que permite implementación de una interfaz gráfica.

Back-end: Denominación del servidor en una plataforma Cliente/Servidor.

Browser: Programa que funciona bajo el protocolo HTTP, cuya función es recuperar el documento desde un servidor, interpretar el código HTML, y presentarlo de manera gráfica.

Compilador: Programa encargado de convertir instrucciones de alto nivel en instrucciones de bajo de nivel de un programa.

CORBA: Common Object Request Broker Architecture. Plataforma de implementación de ORB.

DCOM: Arquitectura de Microsoft para la implementación de objetos distribuidos.

DII: Dynamic Invocation Interface. Interface para invocar los objetos, métodos y parámetros de un objeto.

DSI: Dynamic Skeleton Interface. Interface de acceso a métodos para el lado del servidor.

Front-end: Denominación del cliente en una plataforma Cliente/Servidor.

Gatekeeper: Servicio que permite a los usuarios Web accesar a servicios provistos por objetos que son parte de un sistema interno.

Hardware: Componentes electrónicos de un equipo de computación.

HTML: Codificación aplicada a archivos de textos que permiten su acceso como paginas con formato.

IDL: Interface Definition Lenguage. Lenguaje algorítmico para definir los objetos base en la arquitectura ORB.

IIOP: Inter Internet Object Protocol. Protocolo de intercambio de objetos.

Interface Repository: Servicio de Visibroker que sirve como repositorio de interfaces para sus invocaciones desde los clientes.

Interface: Representación gráfica para interacción de los actores de un sistema.

Java: Lenguaje de programación orientada a objetos.

JaveBeans: Objeto de tipo multimedia creado con Java.

JDBC: Driver de conexión a bases de datos por medio de Java.

Multitarea: Sistema que permite ejecutar varia tareas paralelas a la vez.

Naming Services: Servicio VisiBroker que permite a las aplicaciones Clientes atar objetos usando nombres significativos y lógicos sin tener que direccionar por medio de cualquier convención de nombre de una plataforma específica.

NOS: Network operating system. Sistema operativo de red.

OLE: Object Linking and Embeding. Arquitectura básica de Microsoft para invocación e interacción entre objetos.

OMG: Object Management Group. Grupo encargado de establecer estándares para la invocación de objetos.

ORB: Object Request Broker. Arquitectura de soporte de aplicaciones orientadas a objetos bajo los estándares de la OMG.

OS Agent: Servicio Visibroker para atender requerimientos de los objetos CORBA.

Protocolo: Códigos y procedimientos que hacen posible que se intercambien información entre dos nodos.

Skeletons: Esqueletos. Estructura inicial de operabilidad del lado de un servidos CORBA..

Software: Componente lógico de un equipo de computación.

SQL: Structured Query Language. Lenguaje estructurado de petición de requerimientos estándares a servidores de bases de datos.

Stored Procedures: Procedimientos internos a una base de datos que proporcionan coherencia.

Stubs: Estructura inicial de operabilidad del lado de un Cliente CORBA.

Threads: Hilos de ejecución de procesos.

Tier: Capa lógica de servicios intermediarios entre cliente y servidor.

WEB: Forma de tener acceso a información a traves de una red Internet.

INTRODUCCION

El presente trabajo tiene como objetivo el análisis, diseño e implementación de un sistema farmacéutico basado en la tecnología Cliente/Servidor utilizando la arquitectura Corba.

La tecnología Cliente-Servidor, usada en este sistema, emplea el esquema “Three-Tier” o “Multi-tier” desarrollado en un ambiente distribuido.

Para el desarrollo del sistema, se usó el análisis y diseño orientado a objetos utilizando las metodologías de OMT Rumbaugh, Booch y Jacobson, que permite independencia del lenguaje de desarrollo, implementación de objetos y su reusabilidad. El lenguaje de desarrollo utilizado es Java que se acopla con la arquitectura CORBA.

El software para el desarrollo del sistema fue: Visibroker para Java 3.4, JDK 1.1.7, Enterprise Visual Age 2.0, Netscape Communicator 4.04, para la base de datos: SQL Server 7, el Web Server: Domino Lotus.

CAPITULO 1

1. TECNOLOGIA CLIENTE-SERVIDOR

1.1 Definición.

La tecnología Cliente/Servidor se refiere a una arquitectura de diseño de software de aplicación que es el resultado de la subdivisión de un sistema de información, en conjunto de procesos servidores, generalmente especializados, que pueden ejecutarse en variadas plataformas, tanto hardware como software, y que provee a un gran número de procesos cliente, sobre diferentes plataformas físicamente interconectadas por una red local o de área extendida, utilizando uno o varios protocolos de comunicación.

Los clientes y servidores son dos entidades lógicamente separadas que trabajan sobre una misma red para elaborar una tarea específica por medio de un grupo de servicios intermediarios llamado middleware (figura 1.1).

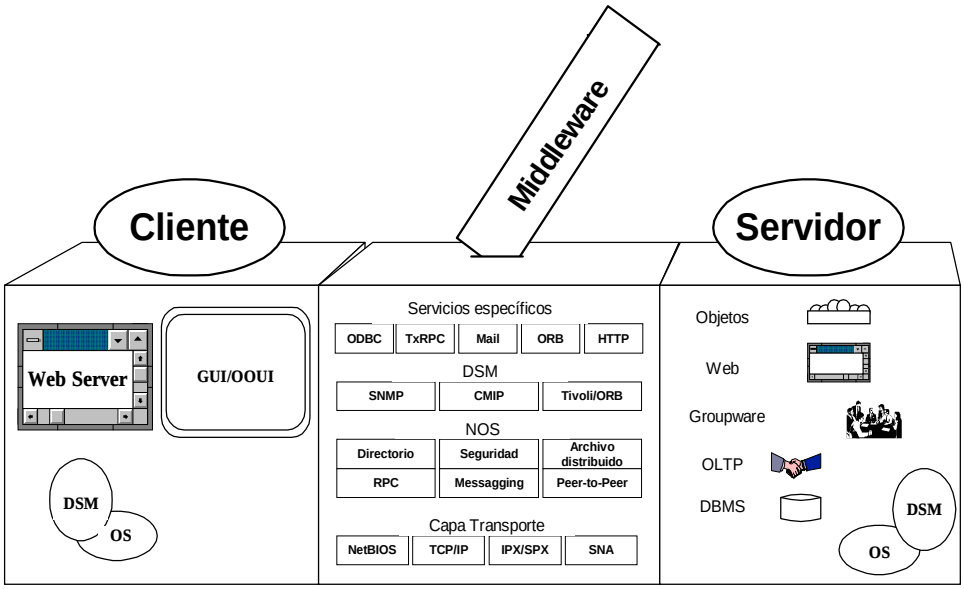

Figura 1.1 Infraestructura Cliente/Servidor

Este esquema nos permite tener sistemas distribuidos con diferentes arquitecturas, en el cual la parte servidor(back-end) intercambia información o provee servicios a muchos clientes (front-end).

El desarrollo de aplicaciones cliente/servidor, incluye procesamiento de transacciones, diseño de base de datos, experiencia en comunicaciones y una interface gráfica amigable. Las aplicaciones mas avanzadas requieren conocimiento de objetos distribuidos e internet.

1.2 Procesos Clientes.

Son procesos que piden o requieren servicios. Hay dos tipos básicos: Clientes independientes que son desarrollados con diversas herramientas que generan productos compilados, ejemplo: Visual Basic, y Clientes basados en Browser's que sólo pueden ser ejecutados dentro de un browser, ejemplo: Java Applets.

1.2.1 Java Applets

Un applet es una pieza de código que se almacena en un servidor Web (HTTP) para que pueda ser accedida por una estación "cliente" (PC), transmitida a través de la red (Internet/Intranet) y la que se instala automáticamente y es ejecutada por el programa navegador de Web ("browser") como parte de un documento (HTML).

1.3 Middleware

Middleware es un conjunto de procesos entre clientes y servidores que permite la comunicación entre ambos. Este comienza con el conjunto API del lado del cliente que es usado para invocar un servicio, cubre la transmisión del requerimiento sobre la red y la respuesta. El middleware no incluye el software que proporciona el servicio, ni la interface del usuario.

Tiene tres formas o componentes:

1. **Mecanismos de Comunicación.-** Se refiere a los protocolos de Comunicación de nivel de Sesión, Transporte y Red, que permiten establecer sesiones o conexiones entre Clientes y Servidores.

2. **Funciones especiales de los NOS.-** Funciones y procesos que extienden el alcance y la capacidad de los Sistemas operativos de los computadores empleados en clientes y servidores.

3. **Middleware para servicios específicos.-** Incluyen todas las otras funciones utilizadas para la resolución de problemas específicos. Especialmente funciones muy ligeras y dependientes principalmente del tipo de servidor con el que se está operando.

1.4 Procesos Servidores.

Existen varios tipos disponibles: File Servers, Database Servers, Transaction Processing Servers, Groupware Servers, Web Servers y Objects Servers.

Un mismo computador podría albergar a más de un proceso servidor.

1.4.1 Object Servers.

Es la nueva tendencia en el desarrollo de sistemas con arquitectura Cliente/Servidor. Facilitan el desarrollo de sistemas con procesamiento distribuido con esquemas 3-Tier y n-Tier. Están basados en el concepto de ORB (object Request Broker).

Pueden funcionar ya sea en conjunto con Web Servers o completamente independientes de ellos. Están sujetos a dos arquitecturas básicas: CORBA (Common ORB Arquitecture) y DCOM (Distributed Common Object Model).

1.5 Esquema Three Tier

Este esquema nos permite identificar claramente los componentes del sistema, la primera capa muestra al objeto que se está negociando en forma gráfica, la segunda capa o nivel contiene objetos definidos en los servidores y las funciones que permiten su negociación y en la ultima capa o nivel están las bases de datos y su interface de comunicación.

1.5.1 Esquema Two Thier versus Three Thier

En los esquemas Cliente-Servidor two-tier, la lógica de aplicación está oculta dentro de la interface del usuario en el cliente o dentro de la base de datos en el servidor, o en ambas. En cambio en los esquemas three-tier, la lógica de aplicación (o procesos) residen en el middle-tier(capa intermedia), que está separado de los datos y la interface del usuario.

Los sistemas Cliente/Servidor son más robustos, escalables y flexibles. Ellos pueden integrar datos desde múltiples fuentes. Ejemplos de estos sistemas son: TP-Monitor, objetos distribuidos, y el Web. Ejemplos de sistemas two-tier: servidores de archivo y servidores de base de datos con stored procedures.

CAPITULO II

2. CORBA

2.1 Definición.

Corba(Common Object Request Broker Architecture), es una tecnología de integración de sistemas distribuidos, permite que las aplicaciones se comuniquen entre sí sin importar donde estén localizadas o por quien hayan sido diseñadas.

En un ambiente distribuido se necesita de un intermediario que ayude a los clientes a encontrar los servidores que puedan atender sus requerimientos. El intermediario o componente central de Corba es el ORB(Object Request Broker). Abarca todo lo referente a la infraestructura de comunicación necesaria para identificar y localizar objetos, maneja la conexión y libera datos.

2.2 ORB

El ORB (Object Request Broker), intermediador de requerimientos de servicios, es el middleware que permite la comunicación entre cliente y servidor por medio de objetos. Usando un ORB, un cliente puede transparentemente invocar un método en un objeto servidor, el cual puede estar en una misma máquina o en algún lugar de la red. El ORB intercepta la llamada y es el responsable de encontrar un objeto que pueda implementar el requerimiento, pasar los parámetros, invocar el método, y retornar los resultados. El cliente no necesita conocer donde está localizado el objeto, su lenguaje de programación, sistema operativo, o algún otro aspecto del sistema que no sea parte de la interface del objeto.

El ORB es simplemente un proceso cuya función es localizar los procesos que proveerán los servicios requeridos. Una vez localizados el cliente puede conectarse con el servidor.

Los objetos en el ORB pueden actuar tanto como cliente o servidor, dependiendo de la ocasión.

La comunicación entre los ORBs se lo realiza por medio del protocolo IIOP (Internet Inter-ORB Protocol).

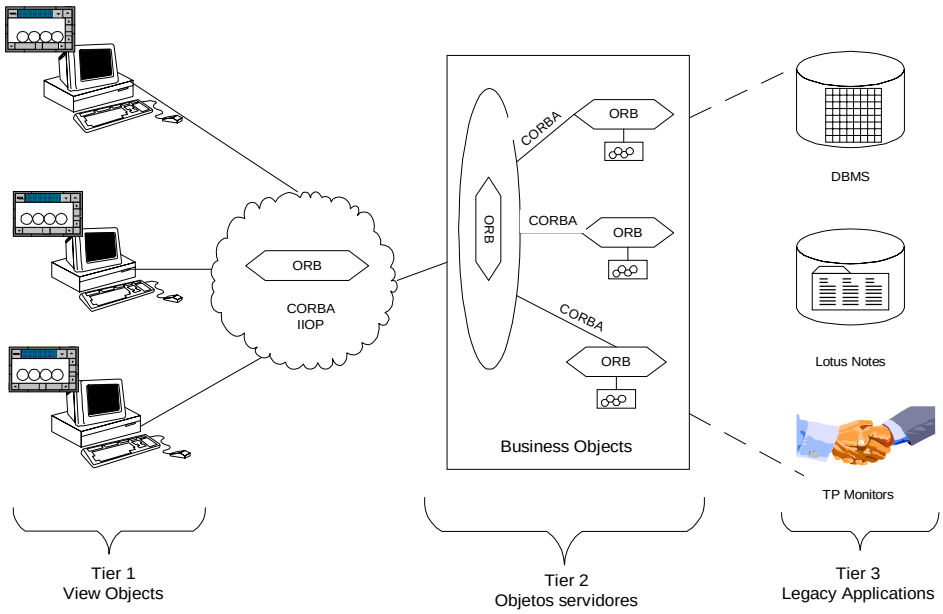

Figura 2.1 3-Tiered Client/Server, Object-Style

2.2.1 Estructura de un ORB de CORBA

a) En el Cliente

- **Stubs del lado del cliente**: Los stubs precompilados definen como los clientes invocan a los correspondientes servicios en los servidores. Del lado del cliente el stub actúa como una llamada local. Los servicios son definidos usando IDL(lenguaje de definición de interfaces), y ambos stubs tanto del cliente como del servidor son generados por el compilador IDL.

- **Dynamic Invocation Interface (DII):** Permite descubrir métodos para ser invocados en tiempo de corrida.

- **Interface Repository:** el depósito de interfaces es una base de datos que almacena información de las interfaces definidas. Permite obtener y modificar las descripciones de todas las interfaces de componentes registrados, los métodos que soportan y los parámetros que ellos requieren.

- **ORB interface:** Consiste de unos pequeños APIs para servicios locales que pueden ser útiles en una aplicación. Por ejemplo APIs para convertir una referencia de objeto a string y viceversa.

b) En el Servidor

- **El Server IDL Stubs:** (OMG los llama skeletons) proporcionan interfaces estáticas para cada servicio exportado por el servidor. Son creados usando el compilador IDL.

- **Implementation Repository:** almacena información que permite al ORB localizar implementación de objetos.

- **Dynamic Skeleton Interface(DSI):** Fueron introducidos por Corba 2.0, proporcionan un mecanismo de enlace en tiempo de

ejecución para servidores que necesitan manejar invocación de métodos de entrada para componentes que no tienen skeletons compilados basados en IDL (o stubs). Los Dynamic Skeletons son muy usados para implementar los enlaces entre ORBs.

- **Object Adapter:** es el medio que permite la comunicación entre el ORBCore y el OI (object implementation).

- **ORB Interface:** Consiste de unos pocos APIs para servicios locales que son idénticos a los proporcionados en el cliente.

2.2.2 Tipos de invocaciones del ORB

El Object Request Broker (ORB) realiza dos tipos de Invocaciones:

- Invocación Estática
- Invocación Dinámica

2.2.2.1 Invocación Estática

Se la define como una interface, la cual contiene descripción de: atributos(variables), métodos que van a ser invocados por el cliente y los argumentos de los métodos.

La invocación estática es implementada en IDL(Interface Definition Language), lenguaje que es puramente declarativo donde se define el nombre del método junto con sus argumentos que van ser invocados por el cliente.

2.2.2.2 Invocación Dinámica.

Antes de llamar a un objeto dinámicamente debemos primeramente obtener el objeto y su referencia, dicha referencia nos servirá para obtener las interfaces de los objetos y construir la petición de forma dinámica teniendo en cuenta que debe ser especificada para poderla ejecutar.

Corba provee 4 interfaces que brindan los servicios que se necesitan para invocar dinámicamente un objeto:

- **CORBA::Object:** Define operaciones que todo objeto CORBA debe soportar.

- **CORBA::Request:** Define las operaciones sobre un objeto remoto.

- **CORBA::NVList:** Provee métodos que ayudan a construir la lista de parámetros y a manipularla.

- **CORBA::ORB:** define los métodos del ORB de propósito general.

2.3 DCOM VS CORBA

Microsoft ha creado el modelo de objetos distribuido DCOM (Distributed Component Object Model) que parece ser un serio competidor de CORBA.

CORBA es visto como un middleware orientado a objetos, que soporta aplicaciones distribuidas que corren en una variedad de sistemas operativos.

DCOM tiene su origen en las estaciones de trabajo y en programación en Windows, esto incluye librerías DLL's, su modelo binario y, el mecanismo RPC delineado para distribución tipo COM, íntimamente ligado con OLE.

DCOM especifica interfaces entre componentes de objetos dentro de una aplicación simple o entre aplicaciones. Como CORBA,

DCOM separa la interface del objeto de su implementación y requiere que todas las interfaces sean declaradas usando un lenguaje de definición de interfaces(IDL).

A diferencia de CORBA, el modelo de objetos DCOM no soporta herencia múltiple, pero puede soportar múltiples interfaces y proporcionar reusabilidad a través de una técnica conocida como aggregation.

Tanto CORBA como DCOM son buenos para invocaciones dinámicas pero CORBA con su Repositorio de Interfaces provee una mejor solución.

Los objetos CORBA tienen referencias únicas y persistentes, además de tener estado. Los objetos DCOM son punteros a interfaces sin estado.

CORBA corre sobre la mayoría de las plataformas mientras que DCOM no.

CAPITULO 3

3. DESCRIPCION DEL SISTEMA

3.1. Objetivos.

El sistema debe cumplir con los siguientes requisitos: debe basarse en el paradigma Cliente – Servidor, el servidor debe ser Orientado a Conexión y manejar concurrentemente a los clientes.

La función principal de la aplicación servidora es servir de puente de conexión entre el applet cliente y la base de datos, para dar la flexibilidad de que la base se encuentre en otro ambiente o red, en otras palabras, funcionará como un “manejador de transacciones”.

El cliente debe ejecutarse en un browser y establecer una sesión con el servidor a través de INTERNET, debe además proveer la

interface necesaria para brindarle al usuario las siguientes opciones:

- Venta de un producto.
- Consulta de factura.
- Ingreso de un producto.
- Baja de un producto.
- Creación de nuevos productos.
- Eliminación de productos.

3.2 Funciones.

El proyecto Control de Ventas e Inventario en una farmacia tiene las siguientes características:

Ventas.- Especificar el código del producto. Los códigos pueden ya estar creados en la Base SQL Server. Si se vende más de un producto, para cada producto nuevo se debe verificar tanto su código como su existencia en el Inventario. La transacción retorna el respectivo precio del producto. Una vez ingresados todos los productos, la transacción hará un solo envío final (hay una opción de confirmación). Cada venta genera una Factura que tiene un detalle de los productos vendidos en esa Transacción.

Consulta de Factura.- Esta transacción simplemente muestra en pantalla los datos de la venta de productos, cantidades y precios.

Ingreso de Productos.- Si bien el inventario inicial puede estar ya cargado en la Base SQL Server, esta transacción permite incrementarlo. Se especifica el código del producto ingresado, la respectiva cantidad, el precio unitario de venta, y la fecha de caducidad del producto.

Baja de Productos.- Se especifica el código del respectivo producto, la cantidad de productos dados de baja, la fecha y el motivo de la baja.

Existen otras transacciones opcionales como: *Creación de nuevos Productos* y *Eliminación de Productos.* Éstas permiten respectivamente crear nuevos códigos de productos, cuyo inventario será luego incrementado con la respectiva transacción, o eliminarlos del Inventario. También se tiene una opción de “Revisión Automática de Caducidad”. Esta transacción opcional implica que si al ingresar productos de un mismo tipo, éstos tienen diversa fecha de caducidad, entonces hay que crear registros separados para cada grupo con fecha distinta. La transacción compara la fecha de todos los productos en el inventario con la fecha actual.

La arquitectura de la aplicación esta basada en la tecnología *Cliente/Servidor,* la cual opera sobre una red de *Area Local y Centralizada* que tiene capacidad de crecer en ambiente Distribuido, el servidor *Web* tiene distintos niveles de acceso y cada sistema fue diseñado con una interface gráfica de applets, los cuales operan sobre objetos definidos en *Corba.*

3.3 Herramientas utilizadas.

3.3.1 Visual Age for Java.

Visual Age para Java es un producto de IBM. Esta herramienta es un entorno visual integrado que soporta el ciclo completo de desarrollo de programas Java.

Entre sus características tenemos: es 100% compatible con Java, presenta un ambiente gráfico de desarrollo integrado, maneja múltiples proyectos, administra archivos y controla versiones, permite el desarrollo de aplicaciones corporativas cliente/servidor n-tier, puede ser usado para una amplia gama de proyectos.

Existen dos versiones: VAJava Professional que combina la programación visual y por eventos con el lenguaje Java en un poderoso IDE (ambiente de desarrollo por sus siglas en inglés) gráfico, y VAJava Enterprise que agrega a este IDE, las

capacidades para desarrollar aplicaciones escalables y simplifica la labor del desarrollo de aplicaciones cliente/servidor. Para la realización del sistema se utilizó la versión VAJava Enterprise.

VAJava es una real solución para empresas que quieran extender su alcance a través de la Web y con tecnología independiente de la plataforma.

3.3.2 Visibroker for Java

Es una herramienta para el desarrollo de aplicaciones Cliente/Servidor basadas en la arquitectura Corba. Emplea Java como su principal lenguaje de programación.

Esta herramienta soporta métodos de invocaciones Corba tanto estáticos como dinámicos. Los métodos del servidor pueden ser invocados por una aplicación Cliente Java o por applets dentro de un browser.

El visibroker para Java incluye un completo repositorio de interfaces Corba, escritos en Java. El compilador IDL Visibroker genera tanto el código skeleton para los objetos servidores en C++ o Java, como los stubs Java para el lado del cliente.

Todo Visigenic ORBs implementa completamente el protocolo IIOP, lo cual hace más fácil para los objetos C++ invocar métodos en objetos Java y viceversa.

El Visibroker viene con un servicio de nombramiento de objeto llamado OSAgent. Múltiples OSAgents corriendo en la misma red se localizan unos a otros y automáticamente particionan el espacio de nombres entre ellos. Esto les permite replicar y cargar objetos entre diferentes máquinas servidoras. En el caso de una excepción al sistema, el ORB redirecciona las llamadas de los clientes a una réplica.

3.3.3 Java Development Kit

El Java Development Kit es un entorno de desarrollo para escribir applets y aplicaciones que conforman el API central de Java 1.1, este compilador y otras herramientas corren desde el shell y no tienen una interface gráfica de usuario. Contiene herramientas como: compilador java (javac), interpretador java(java), archivos de extensión jar, el JDBC, etc. Para el desarrollo del sistema se utilizó el Java Development Kit 1.1.7.

3.4 Características del sistema usando Corba.

El proyecto contiene las siguientes características:

1. Sigue el modelo de diseño de objetos.
2. Utiliza la sintaxis del IDL.
3. Se implementó las siguientes interfaces:

- **DII** Dynamic Invocation Interface.
- **DSI** Dynamic Skeleton Interface.
- Interface Repository.
- **ORB** Interface.
- Basic Object Adapter.

CAPITULO 4

4. ANALISIS Y DISEÑO DEL SISTEMA

Los modelos (*abstracción que se construye para entender y resolver los problemas del sistema*) utilizados en este proceso de análisis y diseño fueron: Casos de Uso, Escenarios, Diagrama de Interacción de Objetos, Diagrama de Clases y modelo de objetos.

4.1 Modelo de requerimientos.

Describe las capacidades del sistema, en este caso, el sistema ***Control de Ventas e Inventario de una Farmacia*** al momento de ser puesto en ejecución será capaz de: realizar con éxito el proceso de simulación de ventas , ingreso, consultas, eliminación, dar de baja a productos farmacéuticos , esto permitirá optimizar a la empresa el manejo de sus recursos en esta área.

4.2 Requerimientos Funcionales.

Se refiere a comportamiento observable de lo que hará el sistema. Desde el punto de vista del sistema, los tipos de transacciones serán de: consulta, actualización, ingreso, creación y baja de productos orientados a una empresa farmacéutica. Este enfoque se complementa con la definición de los casos de uso.

4.2.1 Secuencia de Casos de Uso

Un caso de uso es una secuencia de transacciones en un sistema cuya tarea es producir un resultado con valor medible para un actor del sistema. A continuación se detallan todos los casos de uso del sistema.

Lista de casos de uso del servidor:

1. Se realiza una Venta.
2. Se ingresan productos.
3. Se realiza una Consulta.
4. Se da de Baja un producto
5. Se crea un nuevo producto.
6. Se elimina un producto.
7. Se inicia proceso de caducidad.

Lista de actores:

Primario: **Applet Cliente**

Secundario: **Base de Datos**

4.2.2 Diagrama de contexto de casos de usos del servidor

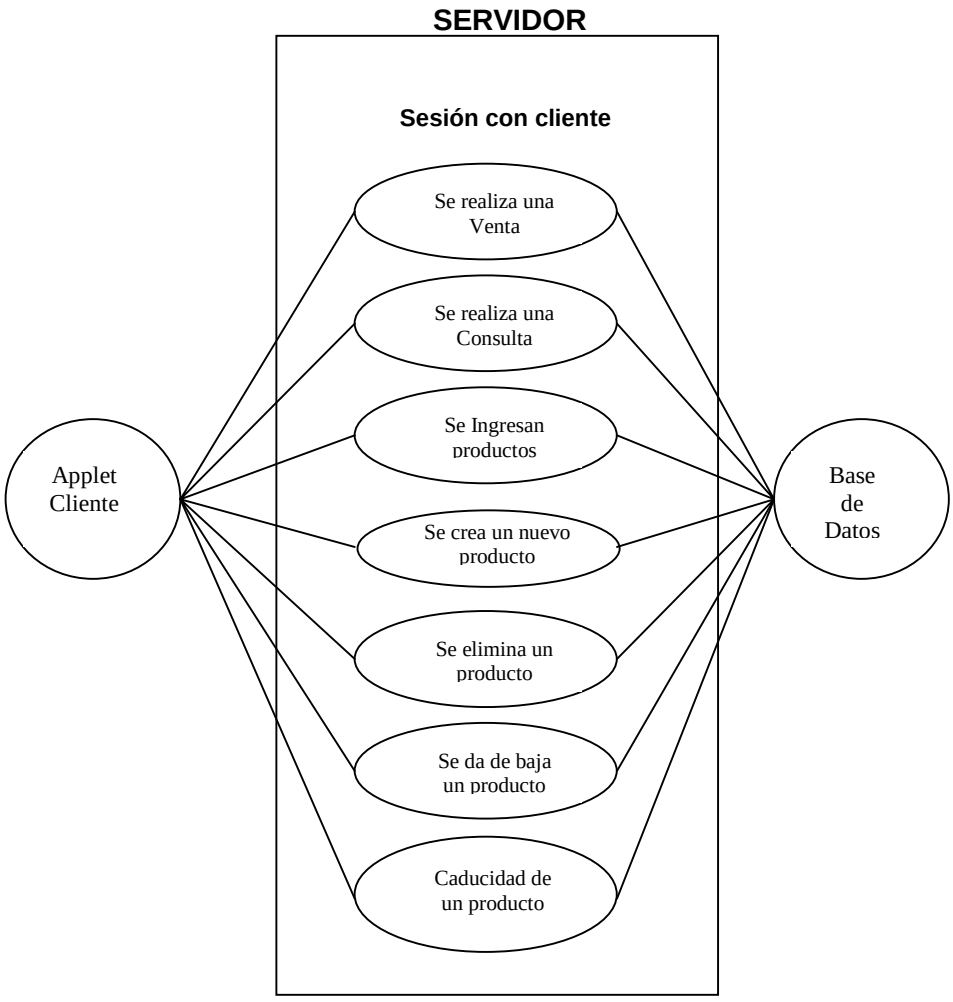

Figura 4.1 Diagrama de contexto de casos de uso

4.2.3 Descripción de Casos de Uso

Caso de Uso 1: Se realiza una venta

Descripción: Este caso de uso describe todo el proceso en el cual el applet CLIENTE abre una sesión en el servidor para poder realizar una venta llevando a cabo una serie de operaciones necesarias para el proceso.

El servidor debe encargarse de mapear los requerimientos con la tabla de transacciones, acceder a la base y reenviar las respuestas al cliente.

Valor medible: Actualización de la base de datos en los productos vendidos (Inventario) y registrar la operación efectuada (OPERACIONES Y DETALLE DE OPERACIÓN).

Notas:

1. El cliente se puede conectar desde una red remota usando la INTERNET.
2. La arquitectura de red es TCP/IP.
3. La Base de Datos puede encontrarse en otra máquina e inclusive en otra red.

4. Los datos de la farmacia están almacenados en una base SQL SERVER.
5. En forma de ayuda se desea tener un listado de todos los códigos y nombres de los productos en inventario, esta transacción debe realizarse al principio de la sesión.
6. Una vez elegido el producto que se desea vender, se debe filtrar de la lista de productos los lotes existentes para que el vendedor pueda elegir de qué lote va a vender. Si el lote no tiene la cantidad suficiente y existen mas en otro, se realizarán dos ventas, ya que se maneja por lotes mas no por código.
7. Se debe realizar una verificación de la cantidad y existencia del producto en venta.
8. Al finalizar la venta debe actualizar la cantidad en inventario de los productos vendidos en la Base de Datos.

Actor: Applet Cliente

Descripción:

- Toma el papel de usuario y es quien hace el requerimiento de conexión del servidor.
- Envía al servidor los códigos de transacciones necesarios para ejecutar la venta.

Notas:

- Debe presentar al inicio de la opción de ventas un listado con todos los códigos y nombres de los productos en inventario.
- Debe presentar los lotes con la cantidad, precio unitario y fecha de caducidad para ser elegido por el usuario del producto en venta.
- Debe pedir una verificación de la cantidad vendida al servidor.
- Debe ingresar la operación realizada en la base.
- Una vez terminada la sesión, pide al servidor la actualización de los datos en la base.

Actor: **Base de datos**

Descripción:

- Contiene los datos del sistema FARMACIA.
- Maneja las tablas:

PRODUCTOS.- Código y nombre del producto.

INVENTARIO.- # lotes, código del producto, cantidad, caducidad.

OPERACIONES.- # operación, factura.

DETALLE DE OPERACIÓN.- # operación, código, nombre, cantidad.

Notas: La base está elaborada en SQL SERVER.

Caso de Uso 2: Se ingresan productos

Descripción: Este caso de uso describe todo el proceso en el cual el applet CLIENTE abre una sesión en el servidor para poder ingresar uno o más productos al inventario de FARMACIA, llevando a cabo una serie de transacciones necesarias para el proceso.

El servidor debe encargarse de mapear los requerimientos con la tabla de transacciones, acceder a la base y reenviar las respuestas al cliente.

Valor medible: Actualización de la Base de Datos en los productos ingresados(INVENTARIO).

Notas:

1. El Cliente se puede conectar desde una red remota usando la INTERNET.
2. La arquitectura de red es TCP/IP.
3. La Base de Datos puede encontrarse en otra máquina e inclusive en otra red.
4. Los datos de la farmacia están almacenados en una base en SQL SERVER.

5. En forma de ayuda se desea tener un listado de todos los códigos y nombres de los productos en inventario esta transacción debe realizarse al principio de la sesión.
6. Una vez elegido el producto que se desea ingresar, se añade los datos a la tabla inventario.
7. Hay que tener en cuenta que por cada producto ingresado genera un nuevo registro basado en el número de lote.

Actor: **Applet Cliente**

Descripción:

- Toma el papel de usuario y es quien hace el requerimiento de conexión al servidor.
- Envía al servidor los códigos de transacciones necesarios para ingresar un producto al inventario.

Notas:

- Debe presentar al inicio un listado con todos los códigos y nombres de los productos en inventario.
- Al aceptar el ingreso del producto debe añadir los registros en la base.

- Una vez terminada la sesión, pide al servidor la actualización de los datos en la base.

ACTOR: Base de datos

Descripción:

- Contiene los datos del sistema FARMACIA.
- Maneja las tablas:

PRODUCTOS.- Código y nombre del producto.

INVENTARIO.- # lotes, código del producto, cantidad, caducidad.

OPERACIONES.- # operación, factura.

DETALLE DE OPERACIÓN.- # operación, código, nombre, cantidad.

Notas: La base está elaborada en SQL SERVER.

Caso de Uso 3: Se realiza una consulta

Descripción: Este caso de uso describe todo el proceso en el cual el applet CLIENTE abre una sesión en el servidor para poder realizar una CONSULTA llevando a cabo una serie de transacciones necesarias para el proceso.

El servidor debe encargarse de mapear los requerimientos con la tabla de transacciones, acceder a la base y reenviar las respuestas al cliente.

Valor medible: Registros con los datos de la venta efectuada dependiendo del número de la factura.

Notas:

1. Dependiendo del número de factura se presentarán los registros pertenecientes a la operación de venta efectuada.
2. La operación sólo es de consulta, así es que no es necesario actualizar la base.

Actor: **Applet Cliente**

Descripción:

- Toma el papel de usuario y es quien hace el requerimiento de conexión al servidor.
- Envía al servidor los códigos de transacciones necesarios para ejecutar la venta.

Notas:

- Debe permitir al usuario ingresar el numero de la factura que se desea consultar.
- Debe presentar todos los registros que pertenecen a esta factura.

Actor: **Base de datos**

Descripción:

- Contiene los datos del sistema FARMACIA
- Maneja las tablas:

PRODUCTOS.- Código y nombre del producto.

INVENTARIO.- # lotes, código del producto, cantidad, caducidad.

OPERACIÓNES.- # operación, factura.

DETALLE DE OPERACIÓN.- # operación, código, nombre, cantidad

Caso de uso 4: Se da de baja un producto.

Descripción: Este caso de uso describe todo el proceso en el cual el applet CLIENTE abre una sesión en el servidor para poder DAR DE BAJA UN PRODUCTO llevando a cabo una serie de transacciones necesarias para el proceso.

El servidor debe encargarse de mapear los requerimientos con la tabla de transacciones, acceder a la base y reenviar las respuestas al cliente.

Valor medible: Actualización de la Base de Datos eliminando los productos dados de baja (INVENTARIO).

Notas:

1. En forma de ayuda se desea tener un listado de todos los códigos y nombres de los productos en inventario por lotes, esta transacción debe realizarse al inicio de la sesión.
2. Se debe realizar una verificación de la cantidad a eliminar.
3. Hay que tener en cuenta que todos los productos en inventario están indexados por el número del lote de compra y éste es un número único.
4. La eliminación no se realizará sobre la cantidad total de unidades de un producto, sino sobre la cantidad existente en un lote de compra y puede ser parcial o total.
5. Al finalizar se debe actualizar la cantidad en inventario de los productos eliminados en la Base de Datos.

Actor: **Applet cliente**

Descripción:

- Toma el papel de usuario y es quien hace el requerimiento de conexión al servidor.
- Envía al servidor los códigos de transacciones necesarios para ejecutar la eliminación de los productos dados de baja.

Notas:

- Debe presentar un listado con todos los códigos y nombres de los productos en inventario.
- Debe presentar los lotes con la cantidad, precio unitario y fecha de caducidad para ser elegido por el usuario para dar de baja.
- Debe pedir una verificación de la cantidad eliminada al servidor.
- Una vez terminado, pide la servidor la actualización de los datos en la base.

Actor: **Base de datos**

Descripción:

- Contiene los datos del sistema FARMACIA.
- Maneja las tablas:

PRODUCTOS.- Código y nombre del producto.

INVENTARIO.- # lotes, código del producto, cantidad, caducidad.

OPERACIONES.- # operación, factura.

DETALLE DE OPERACIÓN.- # operación, código, nombre, cantidad.

Caso de uso 5: **Se crea un nuevo producto.**

Descripción: Este caso de uso describe todo el proceso en el cual el applet CLIENTE abre una sesión en el servidor para poder crear

un nuevo producto llevando a cabo una serie de transacciones necesarias para el proceso.

El servidor debe encargarse de mapear los requerimientos con la tabla de transacciones, acceder a la base y reenviar las respuestas al cliente.

Valor medible: Actualización de la base de datos en los productos existentes (PRODUCTOS).

Notas:

1. Se tiene un listado de todos los códigos y nombre de los productos en inventario, como guía para evitar códigos repetidos, esta transacción debe realizarse al inicio de la sesión.
2. El código es alfanumérico.
3. Se debe realizar una verificación de que el nuevo código no exista previamente en otro producto.
4. Al finalizar se debe insertar el nuevo producto en la Base de Datos.

Actor: Applet cliente

Descripción:

- Toma el papel de usuario y es quien hace el requerimiento de conexión al servidor.
- Envía al servidor los códigos de transacciones necesarios para crear nuevos productos en el inventario.

Notas:

- Debe presentar un listado con todos los códigos y nombres de los productos en inventario como ayuda.
- Debe permitir al usuario ingresar un código y un nombre para el nuevo producto, el código es alfanumérico.
- Debe pedir una verificación del código.

Actor: Base de datos.

Descripción:

- Contiene los datos del sistema FARMACIA.
- Maneja las tablas:

PRODUCTOS.- Código y nombre del producto.

INVENTARIO.- # lotes, código del producto, cantidad, caducidad.

OPERACIONES.- # operación, factura.

DETALLE DE OPERACIÓN.- # operación, código, nombre, cantidad.

Caso de uso 6: Se elimina un producto.

Descripción: Este caso de uso describe todo el proceso en el cual el applet cliente abre una sesión en el servidor para poder eliminar un producto llevando a cabo una serie de transacciones necesarias para el proceso.

El servidor debe mapear los requerimientos con la tabla de transacciones, acceder a la base de datos y reenviar las respuestas al cliente.

Valor medible: Actualización de la base de datos en los productos eliminados (PRODUCTOS).

Notas:

1. Sólo se pueden eliminar productos que no tienen unidades en inventario.
2. En forma de ayuda se desea tener un listado de todos los códigos y nombres de los productos en inventario que puedan ser eliminados, esta transacción se debe realizar al inicio de la sesión.
3. Una vez elegido el producto que se desea eliminar se actualiza la base.

Actor: **Applet cliente**

Descripción:

- Toma el papel del usuario y es quien hace el requerimiento de conexión al servidor.
- Envía al servidor los códigos de transacciones necesarios para eliminar el producto.

Notas:

- Debe presentar un listado con todos los códigos y nombres de los productos que puedan ser eliminados, es decir que no tengan unidades en inventario.
- Debe permitir al usuario ingresar el código del producto a eliminar directamente.
- Finalmente pedir al servidor eliminar el producto de la base de datos.

Actor: **Base de datos**

Descripción:

- Contiene los datos del sistema FARMACIA.
- Maneja las tablas:

PRODUCTOS.- Código y nombre del producto.

INVENTARIO.- # lotes, código del producto, cantidad, caducidad.

OPERACIONES.- # operación, factura.

DETALLE DE OPERACIÓN.- # operación, código, nombre, cantidad.

Caso de uso 7: Se inicia proceso de caducidad.

Descripción: Este caso de uso describe todo el proceso en el cual el applet cliente abre una sesión en el servidor para poder arrancar el proceso de caducidad de productos ya vencidos, llevando a cabo una serie de transacciones necesarias para el proceso.

El servidor mapea los requerimientos con la tabla de transacciones, acceder a la base y reenviar las respuestas al cliente.

Valor medible: Actualización de la base de datos en los productos ya vencidos (INVENTARIO).

Notas:

1. Se debe realizar una actualización en las tablas de inventario con respecto a la fecha de vencimiento de las unidades del lote.
2. Para esta actualización se tomará como referencia la fecha donde corre el servidor.

Actor: **Applet cliente**

Descripción:

- Toma el papel de usuario y es quien hace el requerimiento de conexión al servidor.
- Envía al servidor los códigos de transacciones necesarios para arrancar el proceso de caducidad de productos.

Notas: Debe presentar al usuario los elementos vencidos en el último proceso.

Actor: **Base de datos**

Descripción:

- Contiene los datos del sistema FARMACIA.
- Maneja las tablas:

PRODUCTOS.- Código y nombre del producto.

INVENTARIO.- # lotes, código del producto, cantidad, caducidad.

OPERACIONES.- # operación, factura.

DETALLE DE OPERACIÓN.- # operación, código, nombre, cantidad.

4.2.3.1 Escenarios

Los escenarios detallan cada caso de uso para definir los requerimientos internos del sistema desde la perspectiva de los objetos dentro del sistema. Identifican los objetos participantes en cada contexto y los resultados en términos de estos objetos.

Escenarios del caso de uso 1: Se realiza una venta.

Lista de escenarios:

1.1 Venta de productos sin conflictos.

1.2 Venta de un producto usando dos lotes.

1.3 Trata de vender una cantidad no disponible.

Escenario 1.1: Venta de productos sin conflictos

Asunciones:

1. El cliente se conecta desde un browser y crea el hilo para atenderlo.
2. La conexión con la base de datos está levantada.
3. El cliente envía el código que indica que se encuentra en una transacción venta (Cod.0001).
4. El número de factura es # 0005
5. Elige el código #0001 perteneciente a aspirinas.
6. Existen dos lotes de aspirinas:

   #005001 con 20 aspirinas

#005005 con 100 aspirinas.

7. Elige el lote #005005 de aspirinas que contiene una cantidad de 100 aspirinas en inventario.
8. El cliente vende 60 aspirinas.
9. El cliente acepta la venta.

Resultados:

1. Se envía el número de factura correspondiente.
2. Se envía la lista de códigos y nombres de los productos al cliente para presentarlos por pantalla, esto se realiza en el instante que el usuario elige la opción de ventas, lo que significa que siempre tendrá disponible la lista actualizada de productos.
3. Se envía lista de lotes existentes con el producto a ser vendido para elegir de qué lote se va a vender, y como ayuda para visualizar la cantidad existente.
4. Se verifica la cantidad vendida por el usuario en la tabla de inventario.
5. Se insertan en las tablas de operaciones y detalle de operación la venta efectuada basándose en el número de factura.
6. Se actualiza la tabla de inventario en la base de datos.

Objetos:

- Servidor_Farmacia
- Conexión
- Base_datos
- TranVenta.

Especificaciones del escenario 1.1.- Venta de productos sin conflictos.

Supuestos:

- El cliente se conectó desde un browser y se creó el hilo para atenderlo.
- La conexión con la base de datos está levantada.
- El estado de las tablas en la base son las siguientes:

PRODUCTOS:

Cod	Nombre
0001	Aspirina
0002	Ampollas
0003	Jarabe

INVENTARIO:

#lote	#código	Cant.	Precio	Status	Fec. compra	Fec.venc.
005001	**0001**	**20**	**1000**	**D**	**1999-03-20**	**1999-07-24**
005002	0003	100	30000	D	1999-03-20	1999-07-30
005003	0002	50	10000	D	1999-03-20	1999-08-24
005004	0003	50	30000	D	1999-04-14	1999-09-10
005005	**0001**	**100**	**1000**	**D**	**1999-04-14**	**1999-10-24**

OPERACIONES:

# Factura	Venta Total	Fecha
00001	50000	1994-04-20
00002	10000	1994-04-20
00003	10000	1994-04-22
00004	30000	1999-04-23

DETALLE DE OPERACION:

#factura	#lote	#código	Nombre del producto	cantidad	total
00001	005001	00001	Aspirinas	10	10000
00002	005002	00003	Jarabe	1	30000
00003	005001	00001	Aspirinas	5	5000
00003	005003	00002	Ampollas	2	20000
00004	005001	00001	Aspirinas	30	30000

- El cliente envía el código que indica que se encuentra en una transacción de venta (**COD.001**).
- La última factura tiene el #**00004**
- Elige el código #**00001** perteneciente a aspirinas.
- Existen dos lotes de aspirinas:

#005001 con 20 aspirinas

#005005 con 100 aspirinas

- Elige el lote #**005005** de aspirinas que contiene una cantidad de **100** aspirinas en inventario.
- El cliente vende **60** aspirinas
- El cliente acepta la venta.

Resultados:

- Se envía el número de factura correspondiente. (**#00005**)
- Se envía la lista de códigos y nombres de los productos al cliente para presentarlos por pantalla, esto se realiza en el instante que el usuario elige la opción de ventas, lo que significa que siempre tendrá disponible la lista actualizada de productos.
- Se envía lista de lotes existentes con el producto a ser vendido para elegir de qué lote se va a vender, y como ayuda para visualizar la cantidad existente.
- Se verifica la cantidad vendida por el usuario en la tabla de inventario.
- Se insertan en las tablas de operaciones y detalle de operación la venta efectuada basándose en el número de factura.

OPERACIONES:

# Factura	Venta Total	Fecha
00001	50000	1994-04-20
00002	10000	1994-04-20
00003	10000	1994-04-22
00004	30000	1999-04-23
00005	**60000**	**1999-04-24**

DETALLE DE OPERACION:

#factura	#lote	#código	Nombre del producto	cantidad	total
00001	005001	00001	Aspirinas	10	10000
00002	005002	00003	Jarabe	1	30000
00003	005001	00001	Aspirinas	5	5000
00003	005003	00002	Ampollas	2	20000
00004	005001	00001	Aspirinas	30	30000
00005	**005005**	**00001**	Aspirinas	**60**	**60000**

- Se actualiza la tabla de inventario en la base de datos.

INVENTARIO:

#lote	#código	Cantidad	Precio	Status	Fec. compra	Fec. vencim.
005001	0001	20	1000	D	1999-03-20	1999-07-24
005002	0003	100	30000	D	1999-03-20	1999-07-30
005003	0002	50	10000	D	1999-03-20	1999-08-24
005004	0003	50	30000	D	1999-04-14	1999-09-10
005005	**0001**	**40**	**1000**	**D**	**1999-04-14**	**1999-10-24**

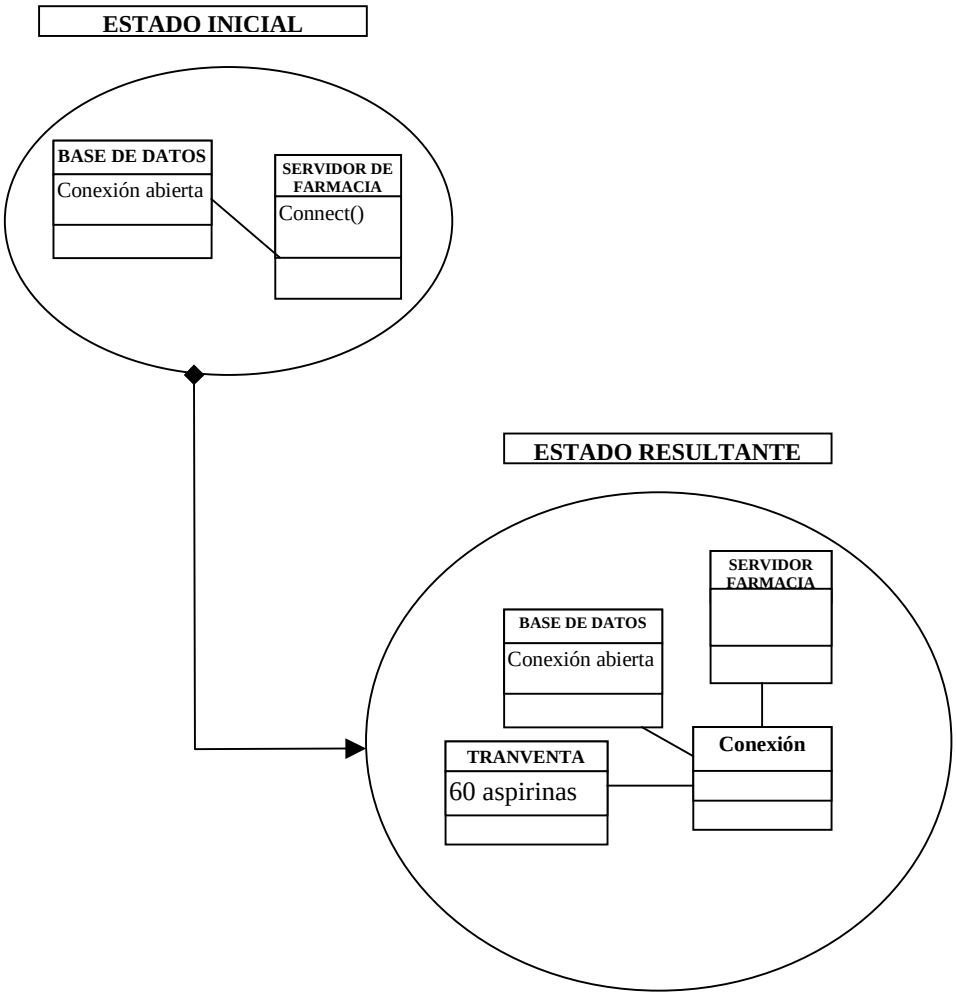

Figura 4.2 Estados del escenario 1.1

ESCENARIO 1.2: Venta de un producto usando dos lotes

Asunciones:

1. El cliente se conecta desde un browser y se crea el hilo para atenderlo.
2. La conexión con la base de datos está levantada.
3. El cliente envía el código que indica que se encuentra en un proceso de venta (COD 001).
4. El número de factura es #00005
5. Elige el código #00001 perteneciente a aspirinas.
6. El cliente quiere vender 60 aspirinas.
7. Existen dos lotes de aspirinas:

#005001 con 40 aspirinas.

#005005 con 30 aspirinas.

8. El vendedor realiza la venta de 60 aspirinas como dos ventas separadas, una de 40 y otra de 20.
9. El vendedor acepta la venta.

Resultados:

1. Se envía el número de factura correspondiente.
2. Se envía la lista de códigos y nombres de los productos al cliente para presentarlos por pantalla, esto se realiza en el

instante que el usuario elige la opción de ventas, lo que significa que siempre tendrá disponible la lista actualizada de productos.

3. Se envía la lista de lotes existentes con el producto a ser vendido para elegir de qué lotes se va a vender, como ayuda para visualizar la cantidad existente.
4. Se verifica la cantidad vendida por el usuario en la tabla de inventario.
5. Se insertan en las tablas de operaciones y detalle de operación la venta efectuada basándose en el número de factura.
6. Se actualiza la tabla de inventario en la base de datos.

Objetos:

- Servidor_Farmacia.
- Conexión.
- Base_Datos.
- TranVentas.

Especificaciones del escenario 1.2.- Venta de un producto usando dos lotes.

Supuestos:

- El cliente se conectó desde un browser y se creó el hilo para atenderlo.

- La conexión con la base de datos está levantada.
- El estado de las tablas en la base son las siguientes:

PRODUCTOS:

Cod	Nombre
0001	Aspirina
0002	Ampollas
0003	Jarabe

INVENTARIO:

#lote	#código	Cant.	Precio	Status	Fec. compra	Fec. vencim.
005001	**0001**	**40**	**1000**	**D**	**1999-03-20**	**1999-07-24**
005002	0003	100	30000	D	1999-03-20	1999-07-30
005003	0002	50	10000	D	1999-03-20	1999-08-24
005004	0003	50	30000	D	1999-04-14	1999-09-10
005005	**0001**	**30**	**1000**	**D**	**1999-04-14**	**1999-10-24**

OPERACIONES:

# Factura	Venta Total	Fecha
00001	50000	1994-04-20
00002	10000	1994-04-20
00003	10000	1994-04-22
00004	30000	1999-04-23

DETALLE DE OPERACION:

#factura	#lote	#código	Nombre del producto	cantidad	total
00001	005001	00001	Aspirinas	10	10000
00002	005002	00003	Jarabe	1	30000
00003	005001	00001	Aspirinas	5	5000
00003	005003	00002	Ampollas	2	20000
00004	005001	00001	Aspirinas	30	30000

- El cliente envía el código que indica que se encuentra en un proceso de venta (**COD.001**).

- La última factura tiene el #**00004**
- Elige el código #**00001** perteneciente a aspirinas.
- Se desea vender 60 aspirinas.
- Existen dos lotes de aspirinas:

  #005001 con 40 aspirinas

  #005005 con 30 aspirinas
- Elige el lote #**005001** de aspirinas que contiene una cantidad de **40** aspirinas en inventario.
- Realiza la primera venta por 40 aspirinas.
- Luego añade otro registro de venta y elige nuevamente el código de aspirinas y elige el lote #**005005** de aspirinas que contiene una cantidad de **30** aspirinas en inventario.
- En total se venden las **60** aspirinas
- El cliente acepta la venta.

Resultados:

- Se envía el número de factura correspondiente. (**#00005**)
- Se envía la lista de códigos y nombres de los productos al cliente para presentarlos por pantalla, esto se realiza en el instante que el usuario elige la opción de ventas, lo que significa que siempre tendrá disponible la lista actualizada de productos.

- Se envía lista de lotes existentes con el producto a ser vendido para elegir de qué lote se va a vender, como ayuda para visualizar la cantidad existente.
- Se verifica la cantidad vendida por el usuario en la tabla de inventario en cada venta realizada.
- Se insertan en las tablas de operaciones y detalle de operación la venta efectuada basándose en el número de factura las dos ventas

OPERACIONES:

# Factura	Venta Total	Fecha
00001	50000	1994-04-20
00002	10000	1994-04-20
00003	10000	1994-04-22
00004	30000	1999-04-23
00005	**60000**	**1999-04-24**

DETALLE DE OPERACION:

#factura	#lote	#código	Nombre del producto	cantidad	total
00001	005001	00001	Aspirinas	10	10000
00002	005002	00003	Jarabe	1	30000
00003	005001	00001	Aspirinas	5	5000
00003	005003	00002	Ampollas	2	20000
00004	005001	00001	Aspirinas	30	30000
00005	**005001**	**00001**	**Aspirinas**	**40**	**40000**
00005	**005005**	**00001**	**Aspirinas**	**20**	**20000**

- Se actualiza la tabla de inventario en la base de datos.

INVENTARIO:

#lote	#codigo	Cant.	Precio	Status	Fec. compra	Fec. vencim.
005001	**0001**	**0**	**1000**	**A**	**1999-03-20**	**1999-07-24**
005002	0003	100	30000	D	1999-03-20	1999-07-30
005003	0002	50	10000	D	1999-03-20	1999-08-24
005004	0003	50	30000	D	1999-04-14	1999-09-10
005005	**0001**	**10**	**1000**	**D**	**1999-04-14**	**1999-10-24**

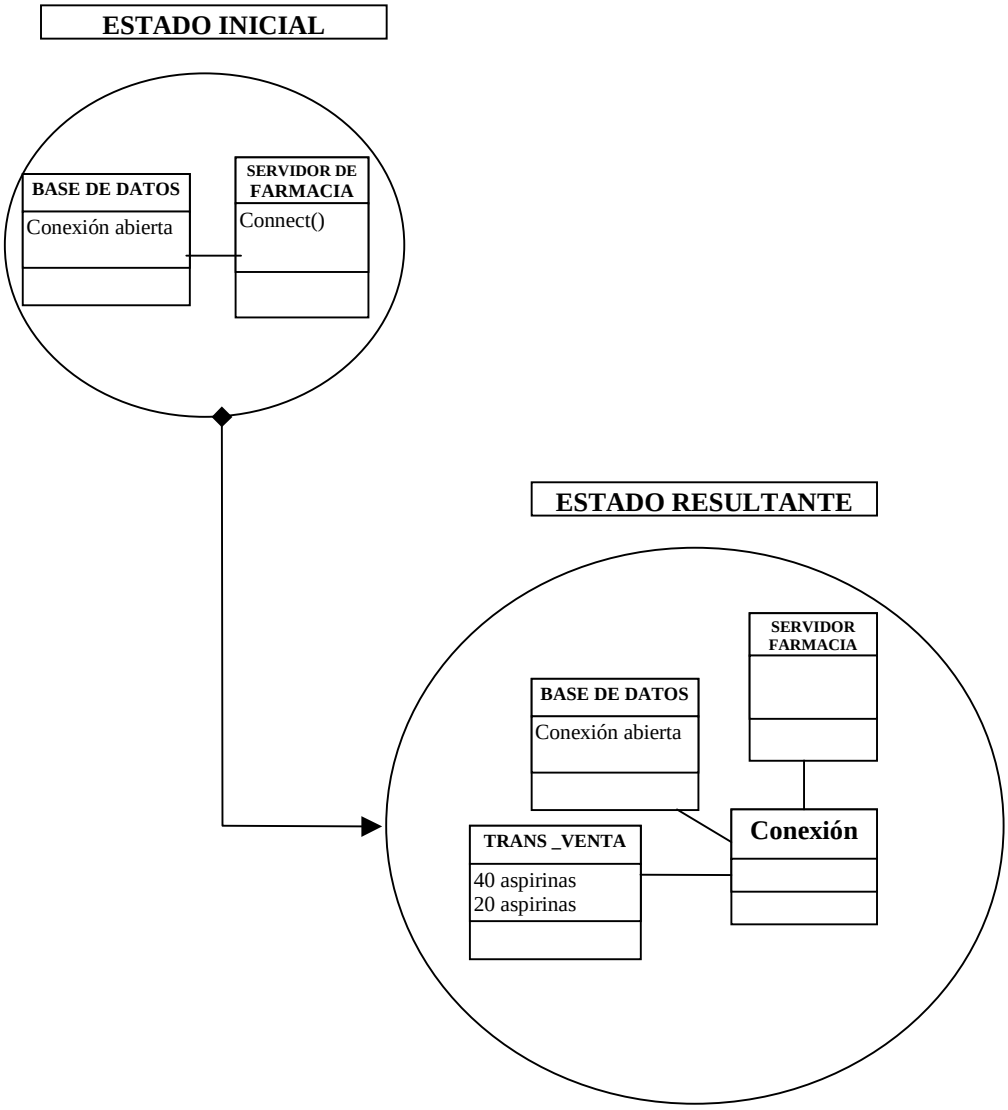

Figura 4.3 Estados del escenario 1.2

Escenario 1.3: Tratar de vender una cantidad no disponible

Asunciones:

1. El cliente se conecta desde un browser y se crea el hilo para atenderlo.
2. La conexión con la base de datos está levantada
3. El cliente envía el código que indica que se encuentra en un proceso de venta(COD 001)
4. El número de factura es #00005
5. Elige el código #00001 perteneciente a aspirinas.
6. El cliente quiere vender 60 aspirinas.
7. Existe un lote de aspirinas:

   #005001 con 10 aspirinas.

Resultados:

1. Se envía el número de factura correspondiente.
2. Se envía la lista de códigos y nombres de los productos al cliente para presentarlos por pantalla, esto se realiza en el instante que el usuario elige la opción de ventas, lo que significa que siempre tendrá disponible la lista actualizada de productos.
3. Se envía la lista de lotes existentes con el producto a ser vendido para elegir de qué lotes se va a vender, como ayuda para visualizar la cantidad existente.

4. Se verifica la cantidad vendida por el usuario en la tabla de inventario.
5. No se realiza la venta.

Objetos:

- Servidor_Farmacia.
- Conexión.
- Base_Datos.
- TranVenta.

Especificaciones del escenario 1.3.- Trata de vender una cantidad no disponible.

Supuestos

- El cliente se conectó desde un browser y se creó el hilo para atenderlo.
- La conexión con la base de datos está levantada.
- El estado de las tablas en la base son las siguientes:

PRODUCTOS:

Cod	Nombre
0001	Aspirina
0002	Ampollas
0003	Jarabe

INVENTARIO:

#lote	#código	Cant.	Precio	Status	Fec. Compra	Fec. vencim.
005001	**0001**	**10**	**1000**	**D**	**1999-03-20**	**1999-07-24**
005002	0003	100	30000	D	1999-03-20	1999-07-30
005003	0002	50	10000	D	1999-03-20	1999-08-24
005004	0003	50	30000	D	1999-04-14	1999-09-10

- El cliente envía el código que indica que se encuentra en un proceso de venta (**COD.001**).
- La última factura tiene el #**00004**
- Elige el código #**00001** perteneciente a aspirinas.
- Se desea vender 60 aspirinas.
- Existe un lote de aspirinas **#005001 con 10 aspirinas**
- Trata de realizar la venta por **60 aspirinas.**

Resultados:

- Se envía el número de factura correspondiente. (**#00005**)
- Se envía la lista de códigos y nombres de los productos al cliente para presentarlos por pantalla, esto se realiza en el instante que el usuario elige la opción de ventas, lo que significa que siempre tendrá disponible la lista actualizada de productos.
- Se envía lista de lotes existentes con el producto a ser vendido para elegir de qué lote se va a vender, como ayuda para visualizar la cantidad existente.

- Se verifica la cantidad que trata de vender el usuario en la tabla de inventario.
- La venta no se realiza y no se actualizan las tablas.

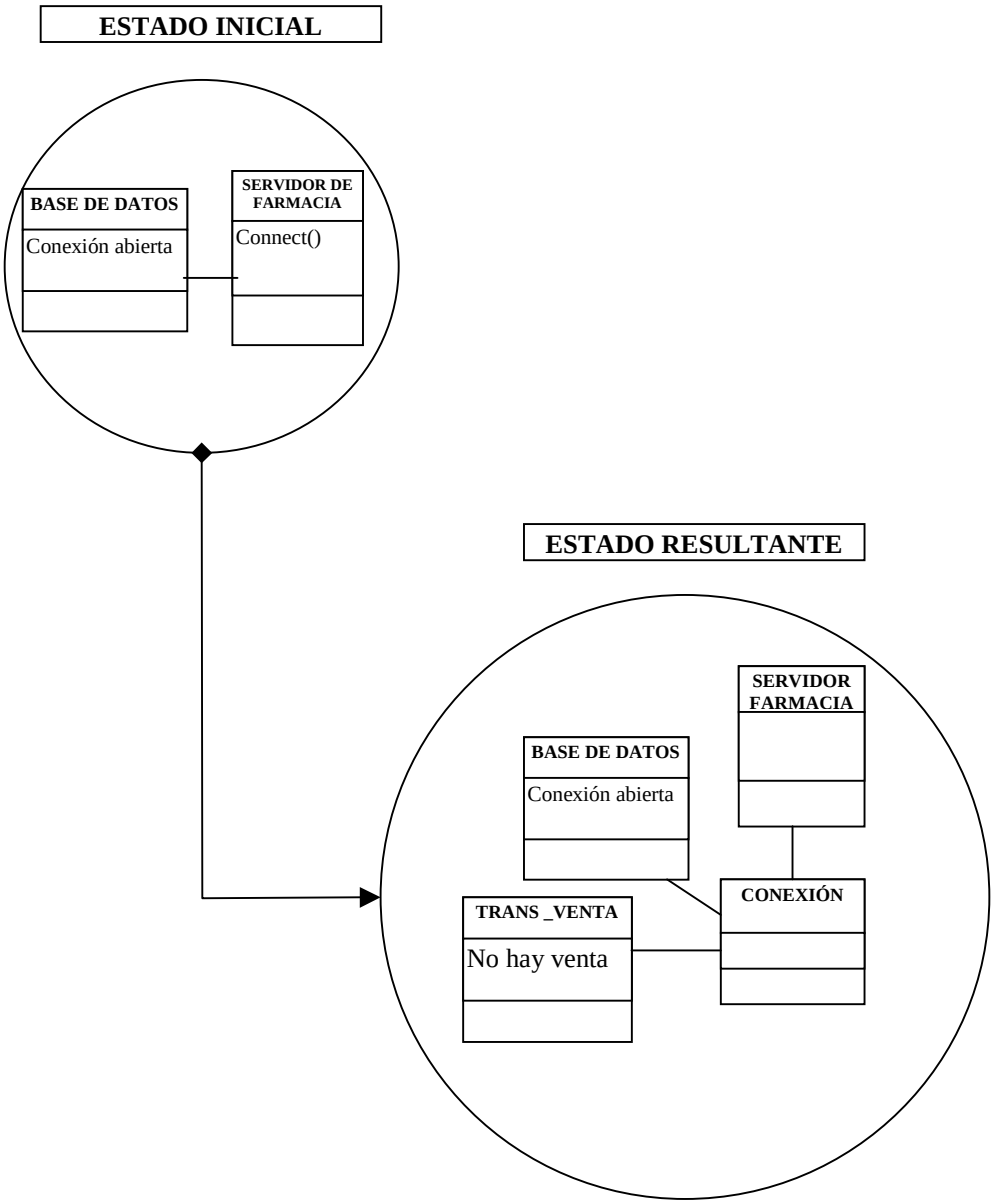

Figura 4.4 Estados del escenario 1.3

Escenarios del caso de uso 2: Se ingresan productos.

Lista de escenarios:

2.1 Se ingresan productos sin conflictos.

2.2 Se desea ingresar un producto que no existe.

Escenario 2.1: Ingreso de productos sin conflictos.

Asunciones:

1. El cliente se conecta desde un browser y se crea el hilo para atenderlo.
2. La conexión con la base de datos está levantada.
3. El cliente envía el código que indica que se encuentra en una transacción de ingreso de productos (COD.002).
4. Elige el código #00001 perteneciente a aspirinas.
5. Existen dos lotes de aspirinas:

   #005001 con 20 aspirinas.

   #005005 con 100 aspirinas.
6. El cliente ingresa un lote con 200 aspirinas.
7. El cliente acepta la transacción.

Resultados:

1. Se envía la lista de códigos y nombres de los productos al cliente para presentarlos por pantalla, esto se realiza en el

instante que el usuario elige la opción de ingreso de productos, lo que significa que siempre tendrá disponible la lista actualizada de productos.

2. Se actualiza la tabla de inventario en la base de datos.

Objetos:

- Servidor_Farmacia.
- Conexión.
- Base_Datos.
- TranIngresa.

Especificaciones del escenario 2.1.- Ingreso de productos sin conflictos.

Supuestos:

- El cliente se conectó desde un browser y se creó el hilo para atenderlo.
- La conexión con la base de datos está levantada.
- El estado de las tablas en la base son las siguientes:

  PRODUCTOS:

Cod	Nombre
0001	Aspirina
0002	Ampollas
0003	Jarabe

INVENTARIO:

#lote	#código	Cant.	Precio	Status	Fec. compra	Fec. vencim.
005001	**0001**	**20**	**1000**	**D**	**1999-03-20**	**1999-07-24**
005002	0003	100	30000	D	1999-03-20	1999-07-30
005003	0002	50	10000	D	1999-03-20	1999-08-24
005004	0003	50	30000	D	1999-04-14	1999-09-10
005005	**0001**	**100**	**1000**	**D**	**1999-04-20**	**1999-07-24**

- El cliente envía el código que indica que se encuentra en una transacción de venta (**COD.002**).
- Elige el código #**00001** perteneciente a aspirinas.
- Existen dos lotes de aspirinas:

#005001 con 20 aspirinas

#005005 con 100 aspirinas

- El cliente ingresa **200** aspirinas
- El cliente acepta el ingreso.

Resultados:

- Se envía la lista de códigos y nombres de los productos al cliente para presentarlos por pantalla, esto se realiza en el instante que el usuario elige la opción de ventas, lo que significa que siempre tendrá disponible la lista actualizada de productos.
- Se actualiza la tabla de inventario en la base de datos.

INVENTARIO:

#lote	#código	Cant.	Precio	Status	Fec. compra	Fec. vencim.
005001	**0001**	**20**	**1000**	**D**	**1999-03-20**	**1999-07-24**
005002	0003	100	30000	D	1999-03-20	1999-07-30
005003	0002	50	10000	D	1999-03-20	1999-08-24
005004	0003	50	30000	D	1999-04-14	1999-09-10
005005	**0001**	**100**	**1000**	**D**	**1999-04-20**	**1999-07-24**
005006	**0001**	**200**	**1000**	**D**	**1999-04-24**	**1999-08-24**

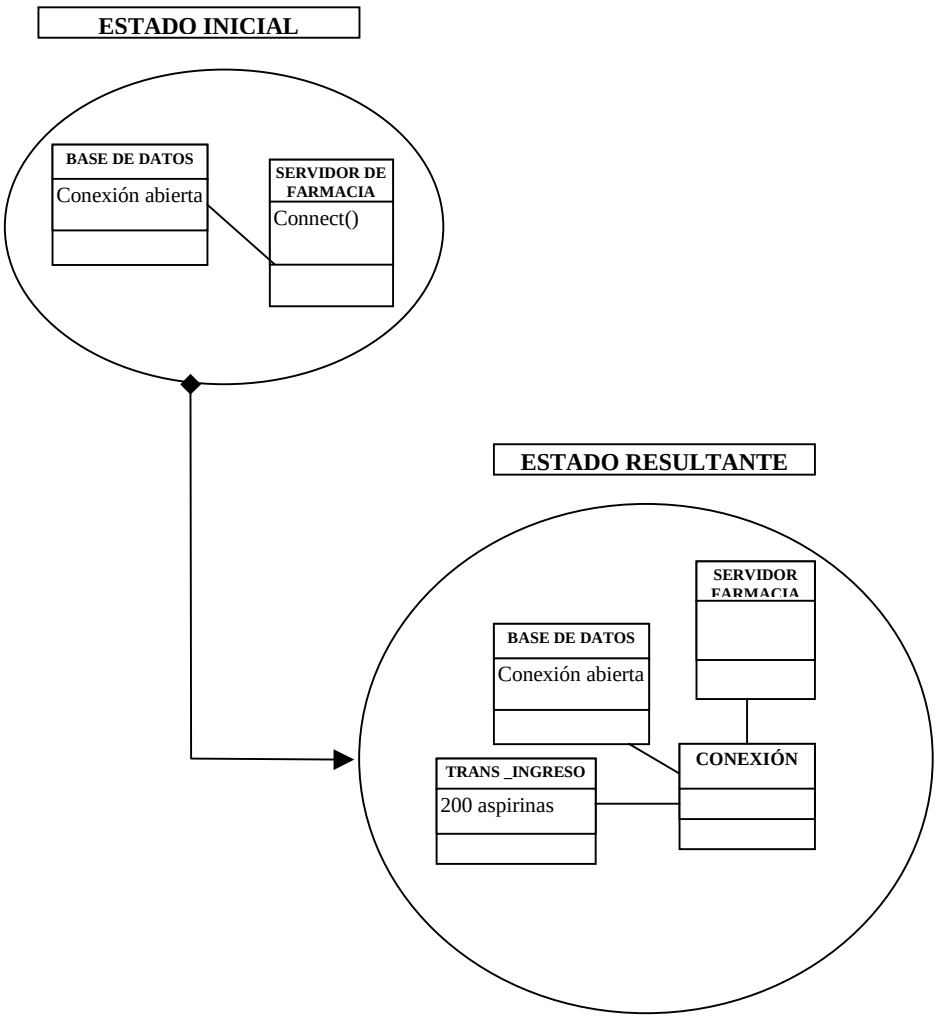

Figura 4.5 Estados del escenario 2.1

Escenario 2.2: Se desea ingresar un producto que no existe.

Asunciones:

1. El cliente se conecta desde un browser y se crea el hilo para atenderlo.
2. La conexión con la base está levantada.
3. El cliente envía el código que indica que se encuentra en una transacción de ingreso de productos (COD.002).
4. Elige el código #00009.

Resultados:

1. Se envía la lista de códigos y nombres de los productos al cliente para presentarlos por pantalla, esto se realiza en el instante que el usuario elige la opción de ingreso de productos, lo que significa que siempre tendrá disponible la lista actualizada de productos.
2. No se actualiza la tabla de inventario en la base de datos y se envía una respuesta de error al cliente.

Objetos:

- Servidor_Farmacia.
- Conexión.
- Base_Datos.

- TranIngreso.

Especificaciones del escenario 2.2.- Se desea ingresar un producto que no existe

Supuestos:

- El cliente se conectó desde un browser y se creó el hilo para atenderlo.
- La conexión con la base de datos está levantada.
- El estado de las tablas en la base son las siguientes:

PRODUCTOS:

Cod	Nombre
0001	Aspirina
0002	Ampollas
0003	Jarabe

- El cliente envía el código que indica que se encuentra en una transacción de venta (**COD.002**).
- Elige el código #**00009** perteneciente a aspirinas.
- No existe ese código en la tabla productos de la base de datos.

Resultados:

- Se envía la lista de códigos y nombres de los productos al cliente para presentarlos por pantalla, esto se realiza en el

instante que el usuario elige la opción de ventas, lo que significa que siempre tendrá disponible la lista actualizada de productos.

- No se actualiza la tabla de inventario en la base de datos y se envía respuesta de error al cliente.

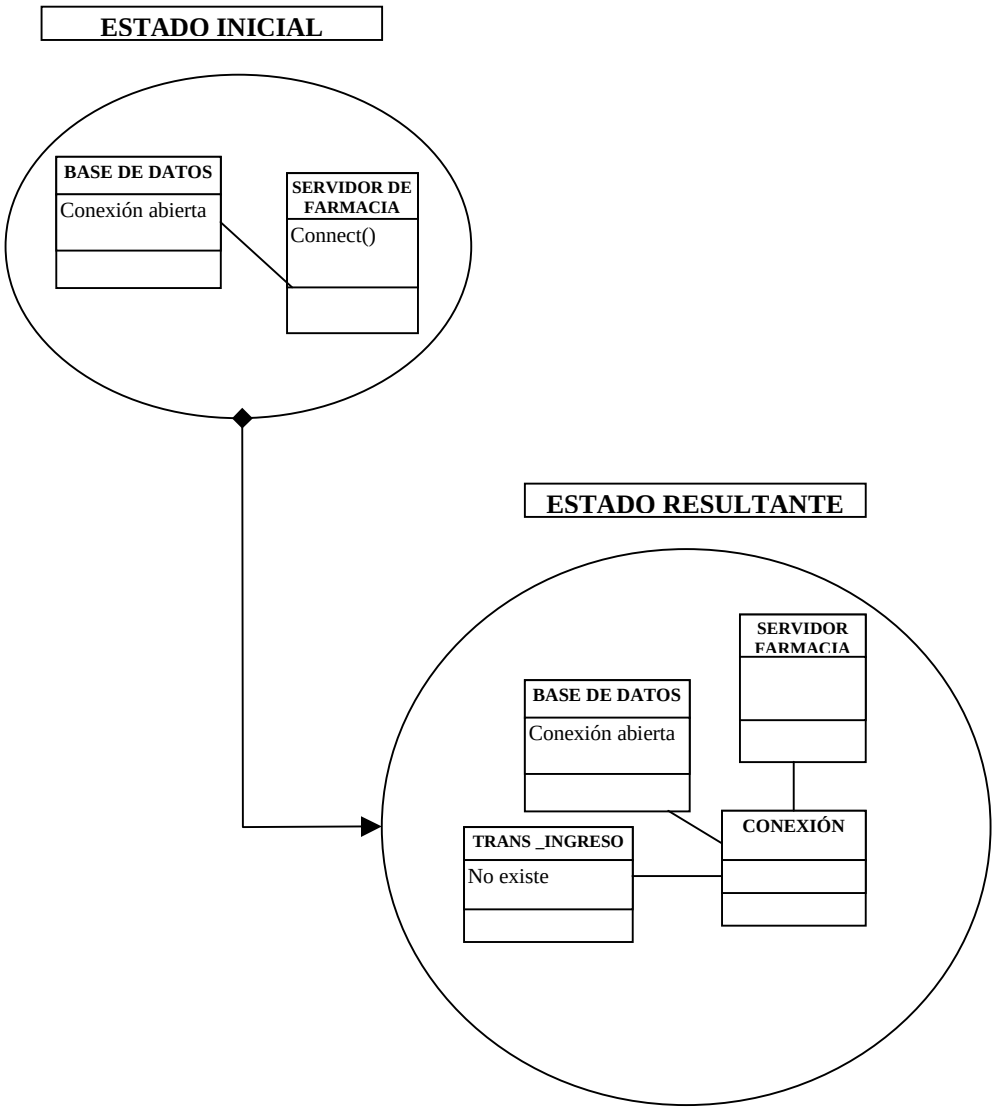

Figura 4.6 Estados del escenario 2.2

Escenario del caso de uso 3: Se realiza una consulta.

Lista de escenarios:

3.1 Se consulta una factura existente.

3.2 Se trata de consultar una factura que no existe.

Escenario 3.1: Se consulta una factura existente.

Asunciones:

1. El cliente se conecta desde un browser y se crea el hilo para atenderlo.
2. La conexión con la base de datos está levantada.
3. El cliente envía el código donde indica que se encuentra en una transacción de consulta de productos (COD.003).
4. Elige el número de factura #00003.
5. La factura existe.

Resultados:

- Se envía al cliente todos los registros pertenecientes a esa factura.

Objetos:

- Servidor_Farmacia.
- Conexión.

- Base_Datos.
- TranConsulta.

Especificaciones del escenario 3.1.- Se consulta una factura existente.

Supuestos:

- El cliente se conectó desde un browser y se creó el hilo para atenderlo.
- La conexión con la base de datos está levantada.

  El estado de las tablas en la base son las siguientes:

OPERACIONES:

# Factura	Venta Total	Fecha
00001	50000	1994-04-20
00002	10000	1994-04-20
00003	10000	1994-04-22
00004	30000	1999-04-23

DETALLE DE OPERACION:

#factura	#lote	#código	Nombre del producto	cantidad	total
00001	005001	00001	Aspirinas	10	10000
00002	005002	00003	Jarabe	1	30000
00003	005001	00001	Aspirinas	5	5000
00003	005003	00002	Ampollas	2	20000
00004	005001	00001	Aspirinas	30	30000

- El cliente envía el código que indica que se encuentra en una transacción de consulta (**COD.003**).

- Elige la factura #**00003**.

Resultados:

- Se envían los registros pertenecientes a la factura #00003

Factura: 00003

Fecha: 1994-04-23

#lote	#codigo	Nombre del producto	cantidad	total
005001	00001	Aspirinas	5	5000
005003	00002	Ampollas	2	20000

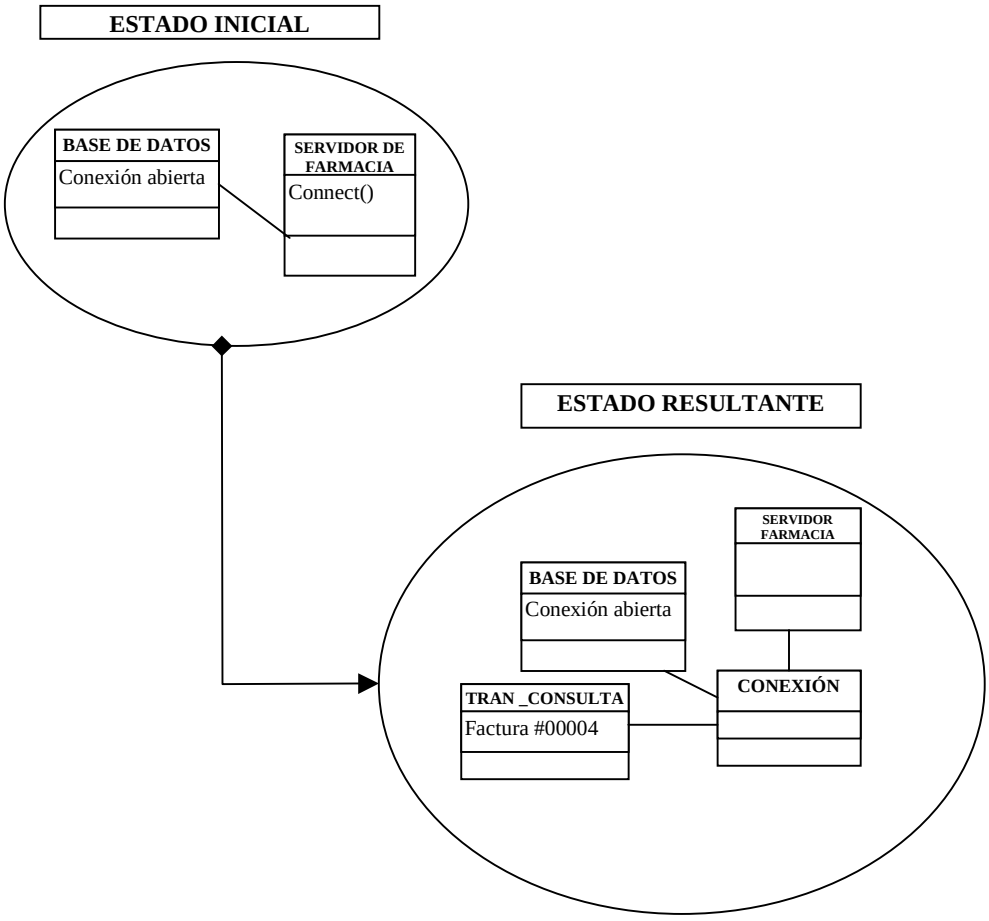

Figura 4.7 Estados del escenario 3.1

Escenario 3.2: Se trata de consultar una factura que no existe.

Asunciones:

1. El cliente se conecta desde un browser y se crea el hilo para atenderlo.
2. La conexión con la base de datos está conectada.
3. El cliente envía el código que indica que se encuentra en una transacción de ingreso de productos (COD.003)
4. Elige el número de factura #00007.
5. La factura no existe.

Resultados:

1. Se envía al cliente respuesta indicando error.

Objetos:

- Servidor_Farmacia
- Conexión
- Base_Datos
- TranConsulta

Especificaciones el escenario 3.2: Se trata de consultar una factura que no existe.

Supuestos:

- El cliente se conectó desde un browser y se creó el hilo para atenderlo.
- La conexión con la base de datos está levantada.
- El estado de las tablas en la base son las siguientes:

OPERACIONES:

# Factura	Venta Total	Fecha
00001	50000	1994-04-20
00002	10000	1994-04-20
00003	10000	1994-04-22
00004	30000	1999-04-23

DETALLE DE OPERACION:

#factura	#lote	#código	Nombre del producto	cantidad	Total
00001	005001	00001	Aspirinas	10	10000
00002	005002	00003	Jarabe	1	30000
00003	005001	00001	Aspirinas	5	5000
00003	005003	00002	Ampollas	2	20000
00004	005001	00001	Aspirinas	30	30000

- El cliente envía el código que indica que se encuentra en una transacción de consulta (**COD.003**).
- Elige la factura #**00006**.

Resultados:

- Se envía un código de error.

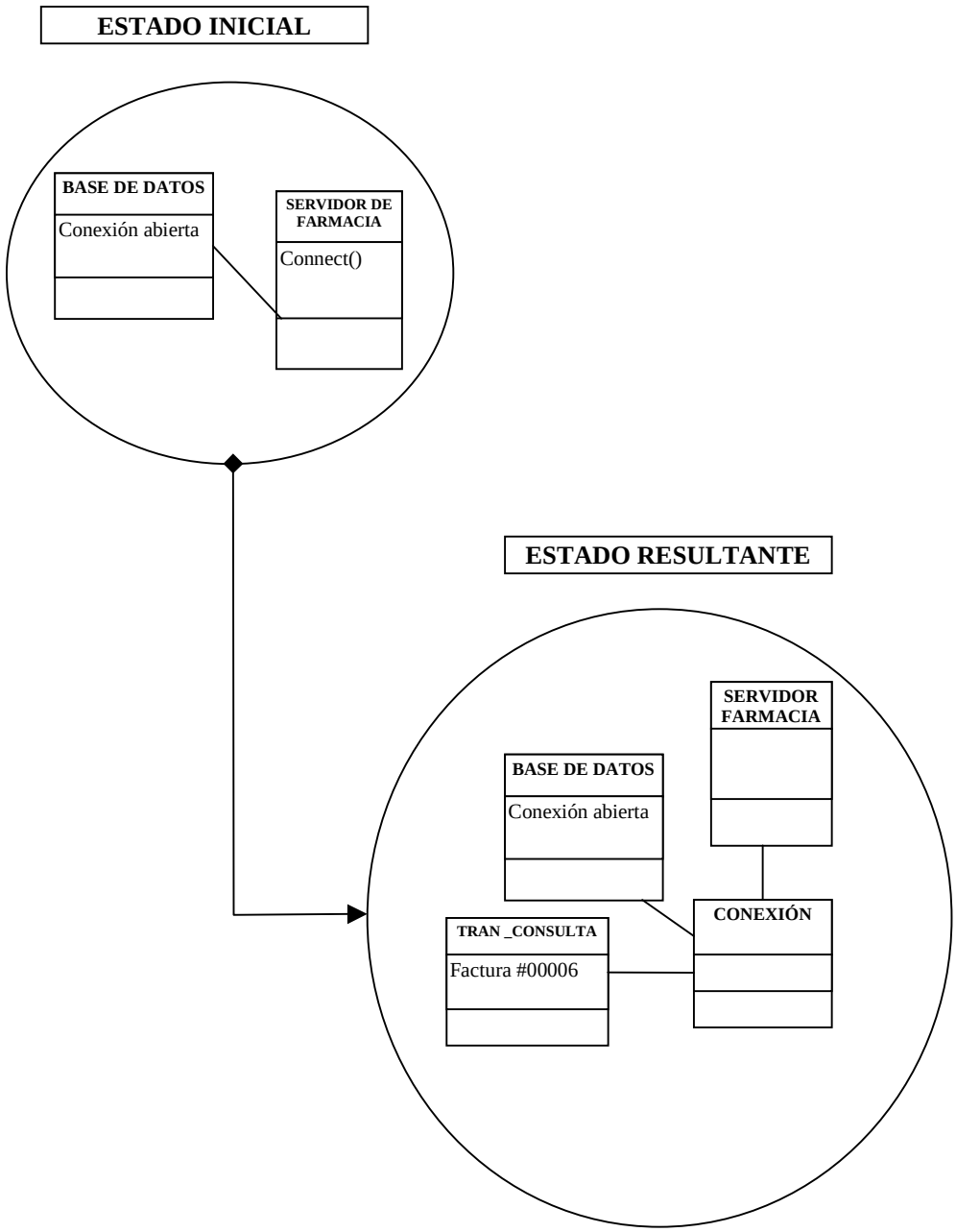

Figura 4.8 Estados del escenario 3.2

Escenario del caso de uso 4: Se da de baja un producto.

Lista de escenarios:

4.1 Se da de baja productos de un lote existente.

4.2 Se trata dar de baja productos de un lote que no existe.

Escenario 4.1: Se da de baja productos de un lote existente.

Asunciones:

1. El cliente se conecta desde un browser y se crea el hilo para atenderlo.
2. La conexión con la base de datos está levantada.
3. El cliente envía el código que indica que se encuentra en una transacción de dar de baja un producto (COD.004).
4. Existen dos lotes de aspirinas:

   #005001 con 20 aspirinas.

   #005005 con 100 aspirinas.
5. Se elige el lote #005005
6. Se dan de baja 50 aspirinas del lote #005005.

Resultados:

1. Se envía la lista de lotes ingresados al cliente para presentarlos por pantalla, esto se realiza en el instante que el usuario elige la opción de dar de baja un producto, lo que significa que siempre

tendrá disponible la lista actualizada de lotes existentes en inventario.

2. Se actualiza la tabla de inventario en la base de datos.

Objetos:

- Servidor_Farmacia
- Conexión
- Base_Datos
- TranBaja

Especificaciones del escenario 4.1.- Se da de baja productos de un lote existente.

Supuestos:

- El cliente se conectó desde un browser y se creó el hilo para atenderlo.
- La conexión con la base de datos está levantada.
- El estado de las tablas en la base son las siguientes:

PRODUCTOS

Cod	Nombre
0001	Aspirina
0002	Ampollas
0003	Jarabe

INVENTARIO:

#lote	#código	Cant.	Precio	Status	Fec. Compra	Fec. vencim.
005001	**0001**	**20**	**1000**	**D**	**1999-03-20**	**1999-07-24**
005002	0003	100	30000	D	1999-03-20	1999-07-30
005003	0002	50	10000	D	1999-03-20	1999-08-24
005004	0003	50	30000	D	1999-04-14	1999-09-10
005005	**0001**	**100**	**1000**	**D**	**1999-04-20**	**1999-07-24**

- El cliente envía el código que indica que se encuentra en una transacción de venta (**COD.004**).
- Elige el lote #**005005**
- El cliente da de baja **50** aspirinas.

Resultados:

- Se envía la lista de lotes ingresados al cliente para presentarlos por pantalla, esto se realiza en el instante que el usuario elige la opción de dar de baja, lo que significa que siempre tendrá disponible la lista actualizada de productos.
- Se actualiza la tabla de inventario en la base de datos.

INVENTARIO:

#lote	#código	Cant.	Precio	Status	Fec. compra	Fec. vencim.
005001	0001	20	1000	D	1999-03-20	1999-07-24
005002	0003	100	30000	D	1999-03-20	1999-07-30
005003	0002	50	10000	D	1999-03-20	1999-08-24
005004	0003	50	30000	D	1999-04-14	1999-09-10
005005	**0001**	**50**	**1000**	**D**	**1999-04-20**	**1999-07-24**

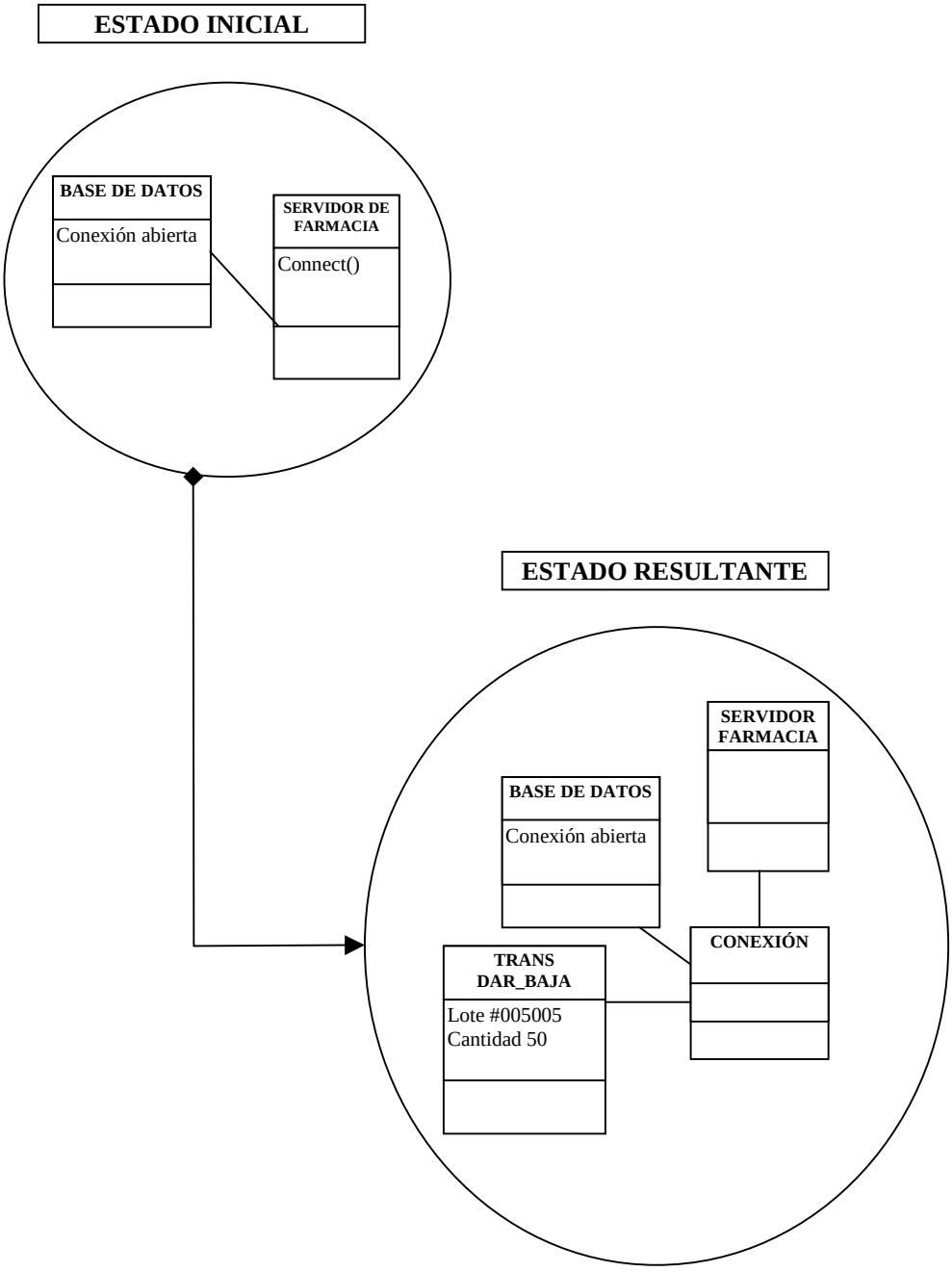

Figura 4.9 Estados del escenario 4.1

Escenario 4.2: Se trata de dar de baja un producto de un lote que no existe.

Asunciones:

1. El cliente se conecta desde un browser y se crea el hilo para atenderlo.
2. La conexión con la base de datos está levantada.
3. El cliente envía el código que indica que se encuentra en una transacción de dar de baja (COD.004).
4. Elige el lote #005009.

Resultados:

1. Se envía la lista de lotes en inventario al cliente para presentarlos por pantalla, esto se realiza en el instante que el usuario elige la opción de dar de baja, lo que significa que siempre tendrá disponible la lista actualizada de lotes existentes.
2. No se actualiza la tabla de inventario en la base de datos y se envía una respuesta de error al cliente.

Objetos:

- Servidor_Farmacia
- Conexión

- Base_Datos
- TranBaja.

Especificaciones del escenario 4.2.- Se trata dar de baja productos de un lote que no existe.

Supuestos:

- El cliente se conectó desde un browser y se creó el hilo para atenderlo.
- La conexión con la base de datos está levantada.
- El estado de las tablas en la base son las siguientes:

PRODUCTOS:

Cod.	Nombre
0001	Aspirina
0002	Ampollas
0003	Jarabe

INVENTARIO:

#lote	#codigo	Cant.	Precio	Status	Fec. compra	Fec. vencim.
005001	0001	20	1000	D	1999-03-20	1999-07-24
005002	0003	100	30000	D	1999-03-20	1999-07-30
005003	0002	50	10000	D	1999-03-20	1999-08-24
005004	0003	50	30000	D	1999-04-14	1999-09-10
005005	0001	100	1000	D	1999-04-20	1999-07-24

- El cliente envía el código que indica que se encuentra en una transacción de venta (**COD.004**).
- Elige el lote #**005009**.

- No existe ese lote en inventario.

Resultados:

- Se envía la lista de lotes existentes en inventario al cliente para presentarlos por pantalla.
- No se actualiza la tabla de inventario en la base de datos y se envía respuesta de error al cliente.

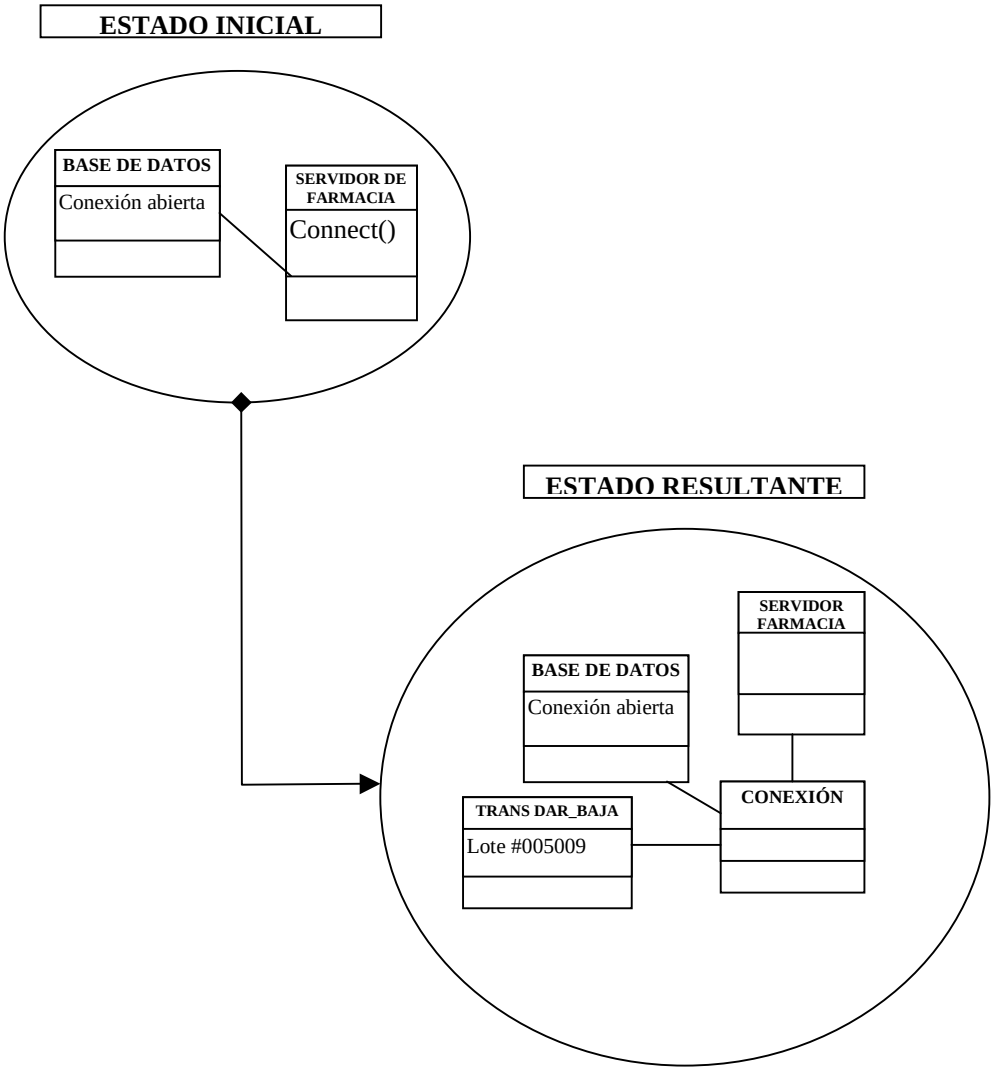

Figura 4.10 estados del escenario 4.2

Escenarios del caso de uso 5: Se crea un nuevo producto.

Lista de escenarios:

5.1 Se Crea un nuevo producto sin conflictos.

5.2 Se trata crear un nuevo producto con un código ya existente.

Escenario 5.1: Se crea un nuevo Producto sin conflictos

Asunciones:

1. El cliente se conecta desde un browser y se crea el hilo para atenderlo.
2. La conexión con la base de datos está levantada.
3. El cliente envía el código que indica que se encuentra en una transacción de creación de nuevo producto (COD.005).
4. Se elige el código #0004 y el nombre es Vitaminas.

Resultados:

1. Se envía la lista de códigos y nombres de los productos existentes en la base de datos.
2. Se inserta el nuevo registro en la tabla de productos de la base de datos.

Objetos:

- Servidor_Farmacia

- Conexión
- Base_Datos
- TranCreaNuevo.

Especificaciones del escenario 5.1.- Se Crea un nuevo producto sin conflictos.

Supuestos:

- El cliente se conectó desde un browser y se creó el hilo para atenderlo.
- La conexión con la base de datos está levantada.
- El estado de la tabla Productos en la base es la siguiente:

  PRODUCTOS:

Cod	Nombre
0001	Aspirina
0002	Ampollas
0003	Jarabe

- El cliente envía el código que indica que se encuentra en una transacción de Creación de Nuevo Producto (**COD.005**).
- Elige el código #**0004.**

Resultados:

- Se envía la lista de códigos y nombres de los productos existentes en la tabla de productos en la base de datos, con el fin de ayudar al usuario y evitar la selección de un código ya existente.
- Se actualiza la tabla Productos en la base de datos.

PRODUCTOS:

Cod	Nombre
0001	Aspirina
0002	Ampollas
0003	Jarabe
0004	Vitaminas

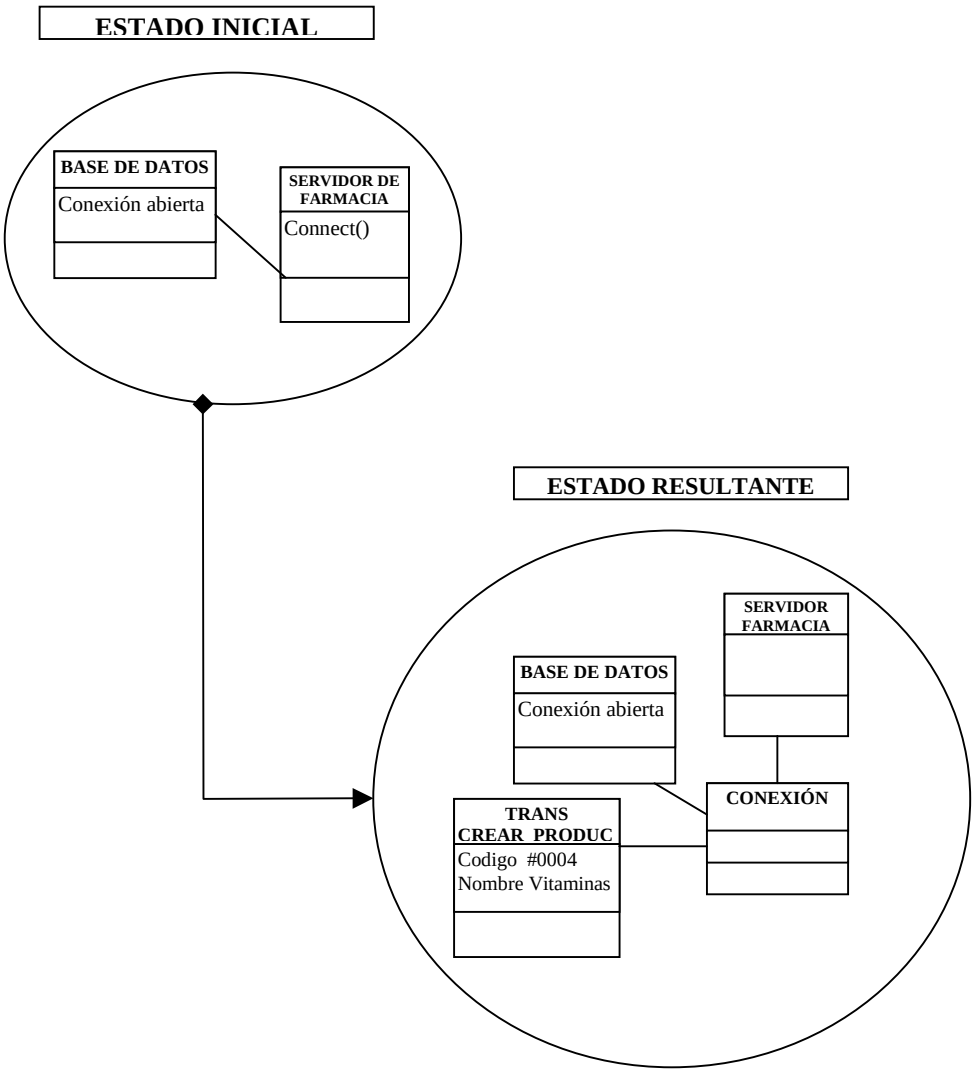

Figura 4.11 Estados del escenario 5.1

Escenario 5.2: Se trata de crear un nuevo producto con un código ya existente.

Asunciones:

1. El cliente se conecta desde un browser y se crea el hilo para atenderlo.
2. La conexión con la base de datos está levantada.
3. El cliente envía el código que indica que se encuentra en una transacción de creación de nuevo producto (COD.005).
4. Se elige el código #0002 y el nombre es Vitaminas.

Resultados:

1. Se envía la lista de códigos y nombres de los productos existentes en la base de datos.
2. Se envía mensaje de error.

Objetos:

- Servidor_Farmacia
- Conexión
- Base_Datos
- TranCreaNuevo.

Especificaciones del escenario 5.2.- Se trata crear un nuevo producto con un código ya existente.

Supuestos:

- El cliente se conectó desde un browser y se creó el hilo para atenderlo.
- La conexión con la base de datos está levantada.
- El estado de la tabla Productos en la base es la siguiente:

  PRODUCTOS:

Cod	Nombre
0001	Aspirina
0002	Ampollas
0003	Jarabe

- El cliente envía el código que indica que se encuentra en una transacción de Creación de Nuevo Producto (**COD.005**).
- Elige el código #**0002.**

Resultados:

- Se envía la lista de códigos y nombres de los productos existentes en la tabla de productos en la base de datos, con el fin de ayudar al usuario y evitar la selección de un código ya existente.
- Se envía un mensaje de error al cliente.

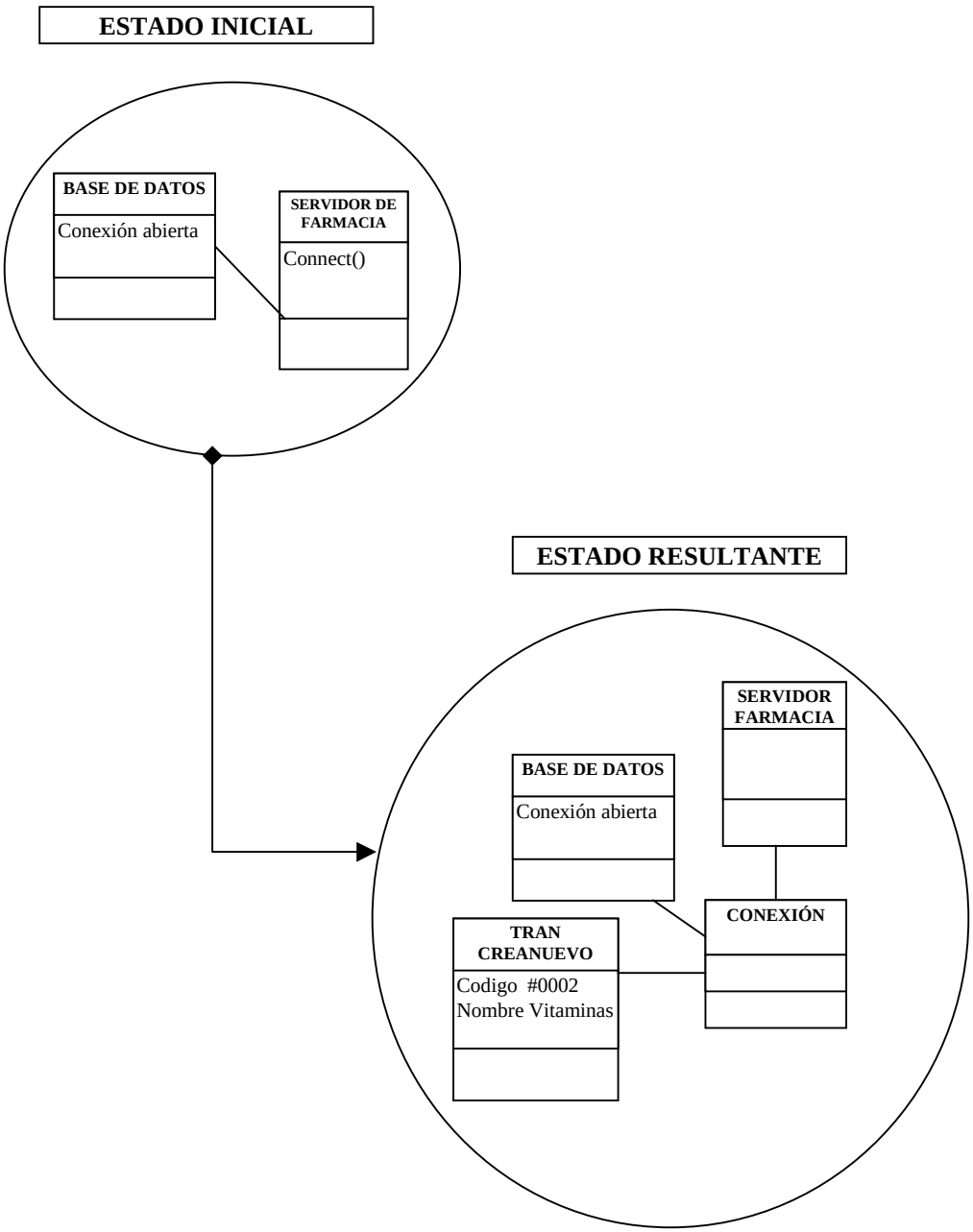

Figura 4.12 Estados del escenario 5.2

Escenario del caso de uso 6: Se elimina un producto.

Lista de escenarios:

6.1 Se elimina un producto sin conflictos.

6.2 Se trata de eliminar un producto que aún tiene una cantidad en inventario.

6.3 Se trata de eliminar un producto que no existe.

Escenario 6.1: Se elimina un producto sin conflictos

Asunciones:

1. El cliente se conecta desde un browser y se crea el hilo para atenderlo.
2. La conexión con la base de datos está levantada.
3. El cliente envía el código que indica que se encuentra en una transacción de eliminación de un producto (COD.006).
4. Se elige el código #0004 y el nombre es Vitaminas.
5. El producto a eliminar no tiene cantidades en inventario.

Resultados:

1. Se envía la lista de los códigos y nombres de los productos existentes en la base de datos que no tienen cantidades en inventario.

2. Se elimina el producto en la tabla de productos de la base de datos.

Objetos:

- Servidor_Farmacia
- Conexión
- Base_Datos
- TranEliminar.

Especificaciones del escenario 6.1.- Se elimina un producto sin conflictos.

Supuestos:

- El cliente se conectó desde un browser y se creó el hilo para atenderlo.
- La conexión con la base de datos está levantada.
- El estado de la tabla Productos en la base es la siguiente:

  PRODUCTOS:

Cod	Nombre
0001	Aspirina
0002	Ampollas
0003	Jarabe
0004	Vitaminas

INVENTARIO:

#lote	#código	Cant.	Precio	Status	Fec. compra	Fec. vencim.
005001	0001	20	1000	D	1999-03-20	1999-07-24
005002	0003	100	30000	D	1999-03-20	1999-07-30
005003	0002	50	10000	D	1999-03-20	1999-08-24
005004	0003	50	30000	D	1999-04-14	1999-09-10
005005	0001	100	1000	D	1999-04-20	1999-07-24

- El cliente envía el código que indica que se encuentra en una transacción de Eliminación de Producto (**COD.006**).
- Elige el código #**0004** perteneciente a **vitaminas.**
- **No existe cantidades de vitaminas en inventario.**

Resultados:

- Se envía la lista de los códigos y nombres de los productos existentes en la base de datos que no tienen cantidades en inventario.
- Se actualiza la tabla Productos en la base de datos.

PRODUCTOS:

Cod	Nombre
0001	Aspirina
0002	Ampollas
0003	Jarabe

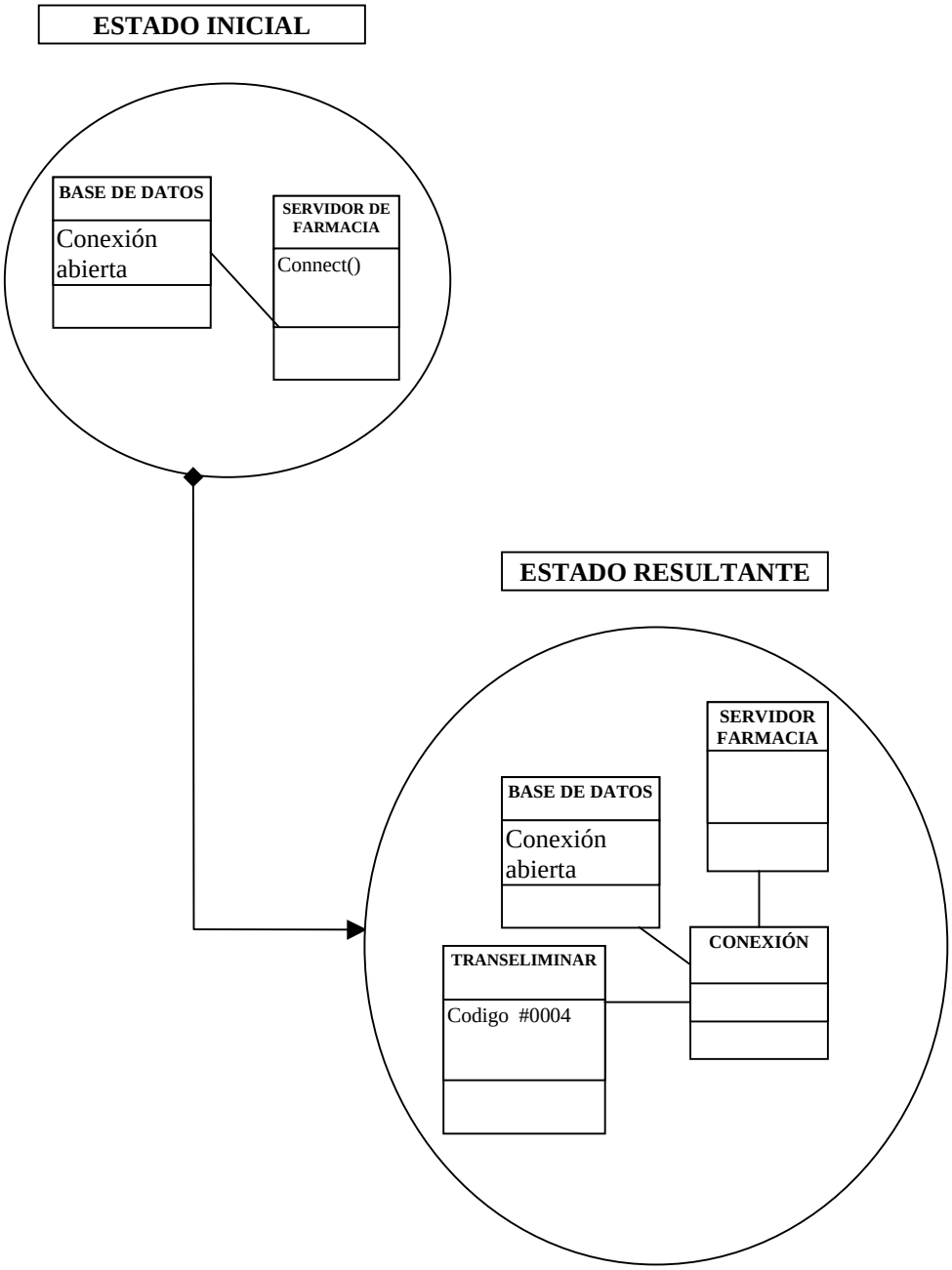

Figura 4.13 Estados del escenario 6.1

Escenario 6.2: Se trata de eliminar un producto que aún tiene una cantidad en inventario.

Asunciones:

1. El cliente se conecta desde un browser y se crea el hilo para atenderlo.
2. La conexión con la base de datos está levantada.
3. El cliente envía el código que indica que se encuentra en una transacción de eliminación de un producto (COD.006).
4. Se elige el código #0002 y el nombre es Ampollas.
5. El producto a eliminar tiene cantidades en inventario.

Resultados:

1. Se envía la lista de los códigos y nombres de los productos existentes en la base de datos que no tienen cantidades en inventario.
2. Se envía un mensaje de error.

Objetos:

- Servidor_Farmacia
- Conexión
- Base_Datos
- TranEliminar.

Especificaciones del escenario 6.2.- Se trata de eliminar un producto que aún tiene una cantidad en inventario.

Supuestos:

- El cliente se conectó desde un browser y se creó el hilo para atenderlo.
- La conexión con la base de datos está levantada.
- El estado de la tabla Productos en la base es la siguiente:

PRODUCTOS:

Cod	Nombre
0001	Aspirina
0002	Ampollas
0003	Jarabe
0004	Vitaminas

INVENTARIO:

#lote	#código	Cant.	Precio	Status	Fec. compra	Fec. vencim.
005001	0001	20	1000	D	1999-03-20	1999-07-24
005002	0003	100	30000	D	1999-03-20	1999-07-30
005003	**0002**	**50**	**10000**	**D**	**1999-03-20**	**1999-08-24**
005004	0003	50	30000	D	1999-04-14	1999-09-10
005005	0001	100	1000	D	1999-04-20	1999-07-24

- El cliente envía el código que indica que se encuentra en una transacción de Eliminación de Producto (**COD.006**).
- Elige el código #**0002** perteneciente a **ampollas.**
- **No existen cantidades de vitaminas en inventario.**

Resultados:

- Se envía la lista de los códigos y nombres de los productos existentes en la base de datos que no tienen cantidades en inventario
- Se envía un mensaje de error.

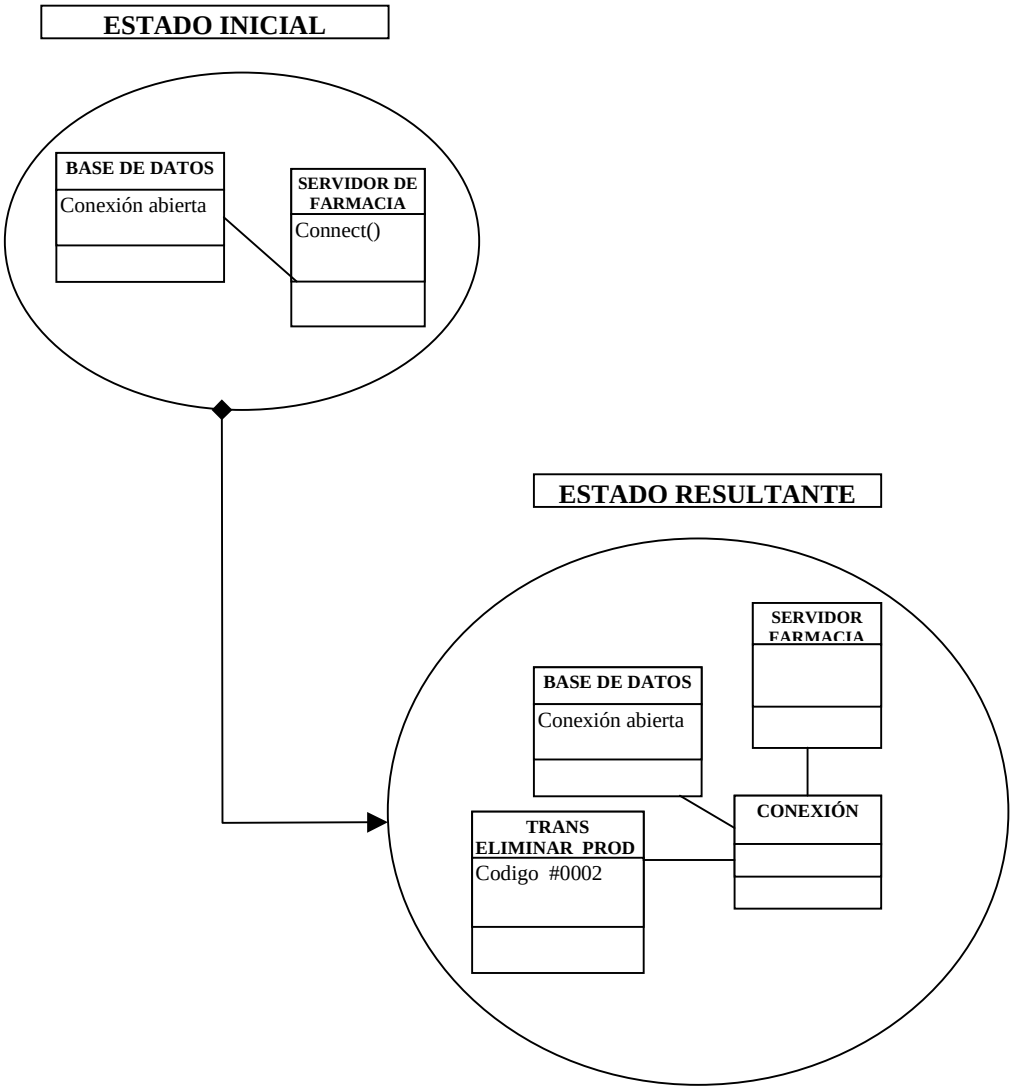

Figura 4.14 Estados del escenario 6.2

Escenario 6.3: Se trata de eliminar un producto que no existe.

Asunciones:

1. El cliente se conecta desde un browser y se crea el hilo para atenderlo.
2. La conexión con la base de datos está levantada.
3. El cliente envía el código que indica que se encuentra en una transacción de eliminación de un producto (COD.006).
4. Se elige el código #0007.
5. Este código de producto no existe en inventario.

Resultados:

1. Se envía la lista de los códigos y nombres de los productos existentes en la base de datos que no tienen cantidades en inventario.
2. Se envía un mensaje de error.

Objetos:

- Servidor_Farmacia
- Conexión
- Base_Datos
- TranEliminar.

Especificaciones del escenario 6.3.- Se trata de eliminar un producto que no existe.

Supuestos:

- El cliente se conectó desde un browser y se creó el hilo para atenderlo.
- La conexión con la base de datos está levantada.
- El estado de la tabla Productos en la base es la siguiente:

  PRODUCTOS:

Cod	Nombre
0001	Aspirina
0002	Ampollas
0003	Jarabe
0004	Vitaminas

- El cliente envía el código que indica que se encuentra en una transacción de Eliminación de Producto (**COD.006**).
- Elige el código #**0007.**
- **No existe este código en la tabla PRODUCTOS.**

Resultados:

- Se envía la lista de los códigos y nombres de los productos existentes en la base de datos que no tienen cantidades en inventario
- Se envía un mensaje de error.

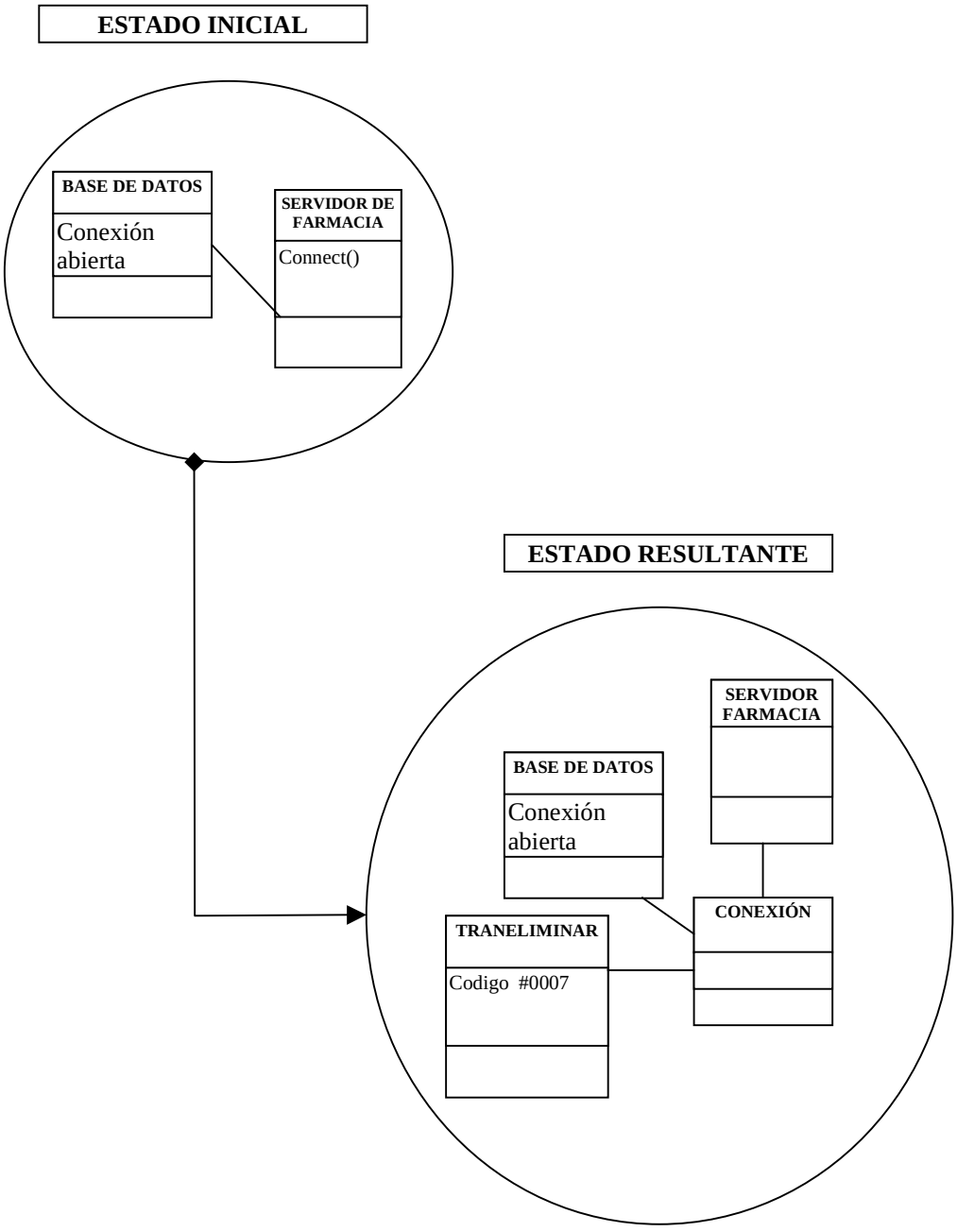

Figura 4.15 Estados del escenario 6.3

Escenario del caso de uso 7: Se arranca proceso de caducidad.

Lista de escenarios:

7.1 Se ejecuta el proceso sin conflictos.

ESCENARIO 7.1: Se ejecuta el proceso sin conflictos

Asunciones:

1. El cliente se conecta desde un browser y se crea el hilo para atenderlo.
2. La conexión con la base de datos está levantada.
3. El cliente envía el código que indica que se encuentra en una transacción de arrancar proceso de expiración de productos (COD.007).
4. La fecha actual es 1999-04-29.

Resultados:

1. Se envía la fecha de la última vez que se ejecutó el proceso.
2. Se actualiza la tabla INVENTARIO cambiando el campo status de los registros cuyas fechas sean menores a la fecha del sistema.
3. Se envía un mensaje de reconocimiento de l proceso ejecutado.

Objetos:

- Servidor_Farmacia
- Conexión
- Base_Datos
- TranCaduca.

Especificaciones del escenario 7.1.- Se ejecuta el proceso sin conflictos.

Supuestos:

- El cliente se conectó desde un browser y se creó el hilo para atenderlo.
- La conexión con la base de datos está levantada.
- El estado de la tabla Productos en la base es la siguiente:

INVENTARIO:

#lote	#código	Cant.	Precio	Status	Fec. Compra	Fec. vencim.
005001	**0001**	**20**	**1000**	**D**	**1999-01-20**	**1999-04-29**
005002	0003	100	30000	D	1999-03-20	1999-07-30
005003	0002	50	10000	D	1999-03-20	1999-08-24
005004	0003	50	30000	D	1999-04-14	1999-09-10
005005	0001	100	1000	D	1999-04-20	1999-07-24

- El cliente envía el código que indica que se encuentra en una transacción de Expiración (**COD.007**).
- La fecha actual es 1999-04-29**.**
- **Existe un lote de aspirinas que caduca en 1999-04-29.**

Resultados:

- Se envía la fecha de la última vez que se ejecutó el proceso.
- Se actualiza la tabla INVENTARIO en la base de datos cambiando el campo status del lote vencido.

INVENTARIO:

#lote	#código	Cant.	Precio	Status	Fec. Compra	Fec. vencim.
005001	**0001**	**20**	**1000**	**V**	**1999-03-20**	**1999-04-29**
005002	0003	100	30000	D	1999-03-20	1999-07-30
005003	0002	50	10000	D	1999-03-20	1999-08-24
005004	0003	50	30000	D	1999-04-14	1999-09-10
005005	0001	100	1000	D	1999-04-20	1999-07-24

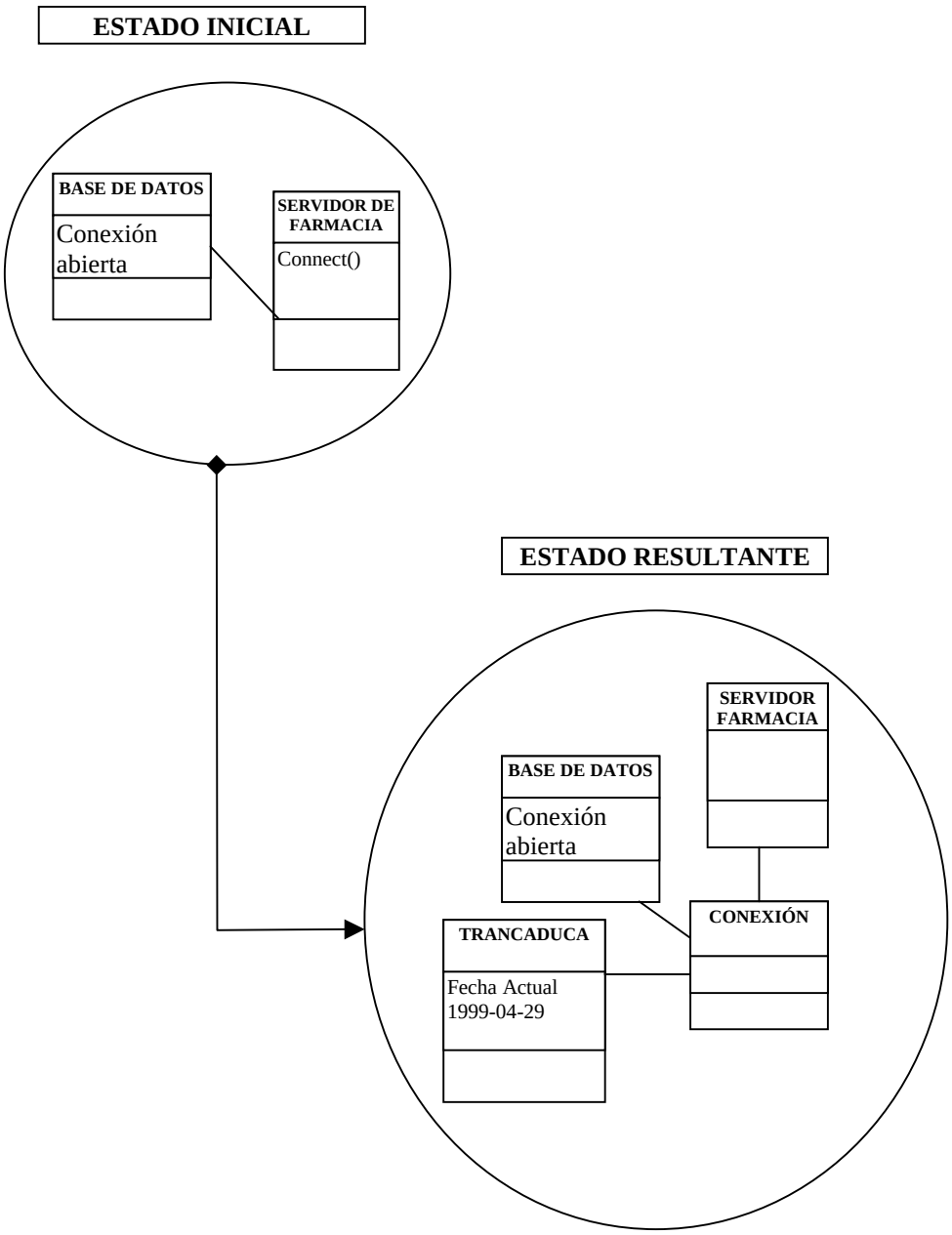

Figura 4.16 Estados del escenario 7.1

4.2.4 Diagramas de Interacción de Objetos

Los diagramas de interacción de objetos definen como los objetos en el sistema colaboran para producir el resultado de los escenarios. Cada diagrama de interacción de objetos describe las responsabilidades de los objetos participantes y las interacciones entre ellos.

1.1. Venta de un producto sin conflictos

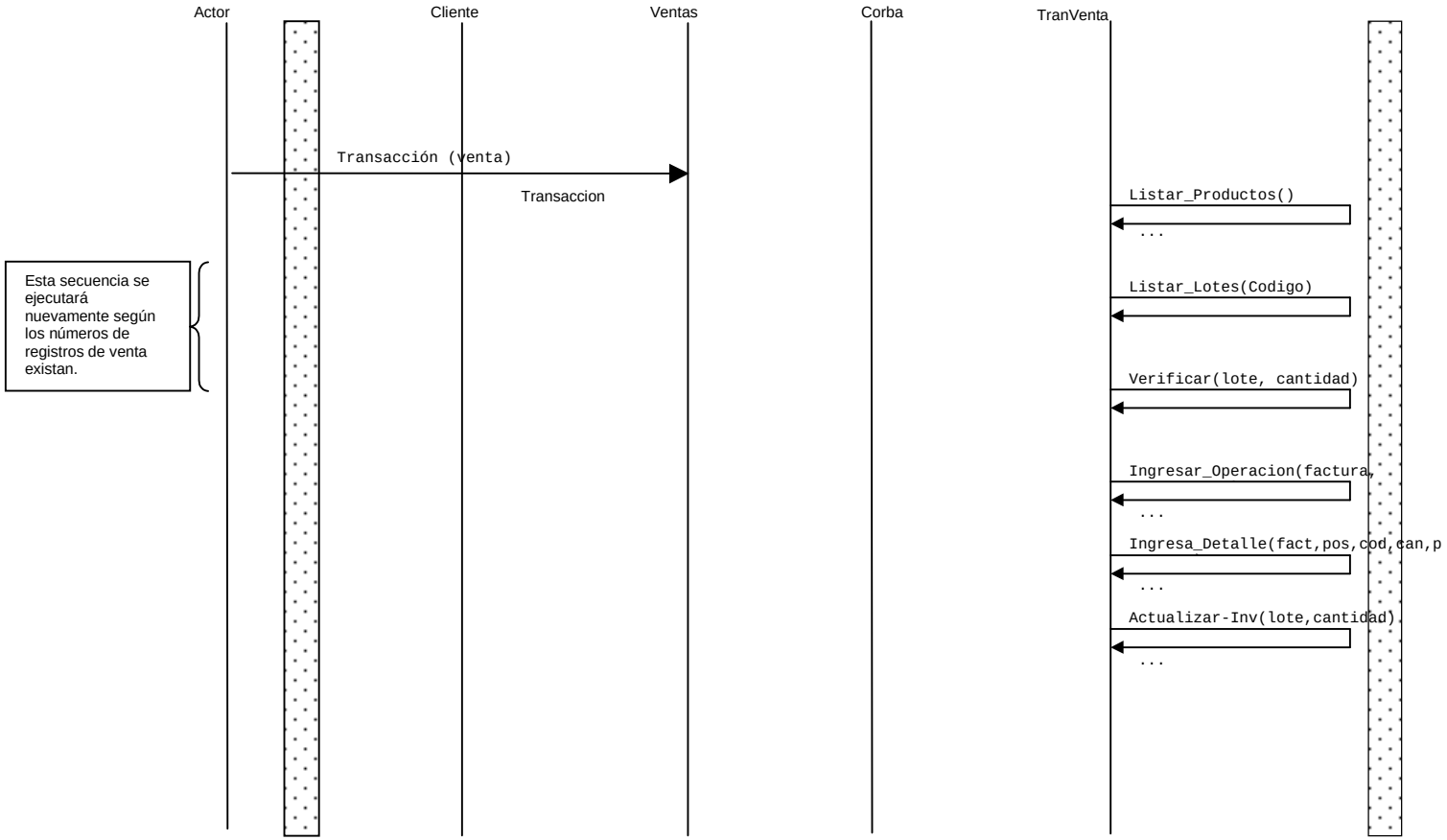

Figura 4.17 Diagrama de interacción de objetos: escenario 1.1

1.2. Venta de un producto usando dos lotes.

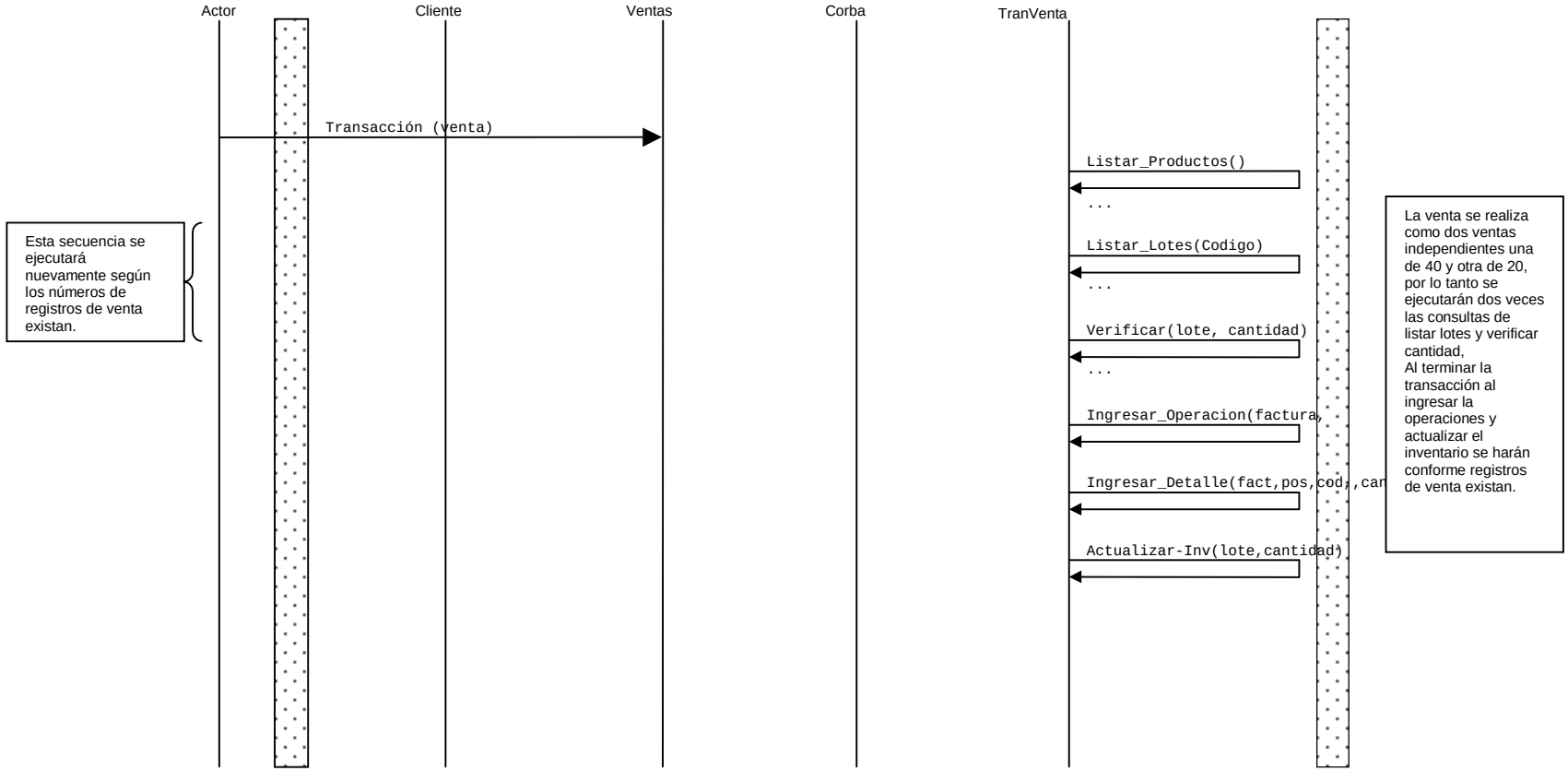

Figura 4.18 Diagrama de interacción de objetos: escenario 1.2

1.3. Trata de vender una cantidad no disponible.

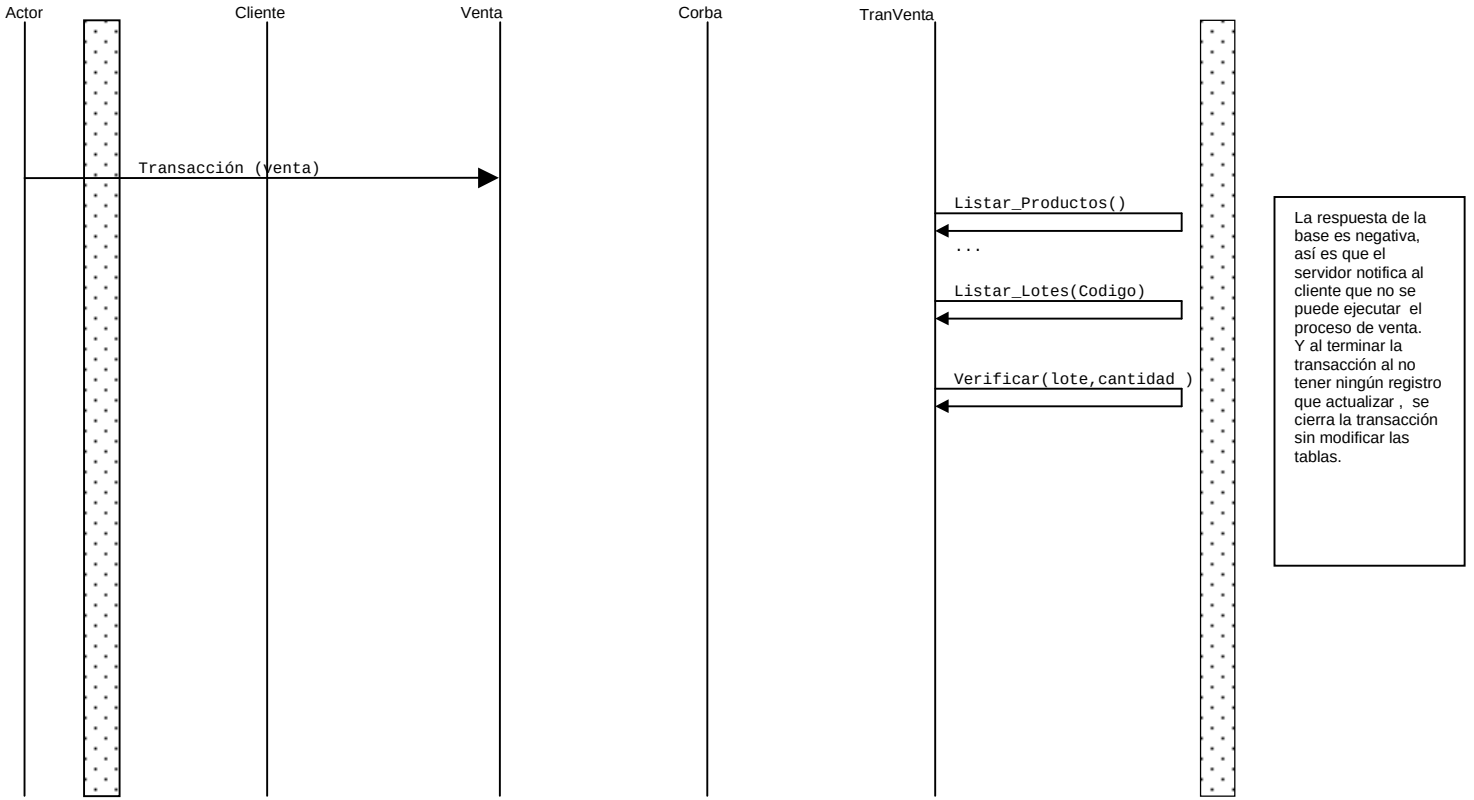

Figura 4.19 Diagrama de interacción de objetos: escenario 1.3

2.1. Ingreso de productos sin conflictos

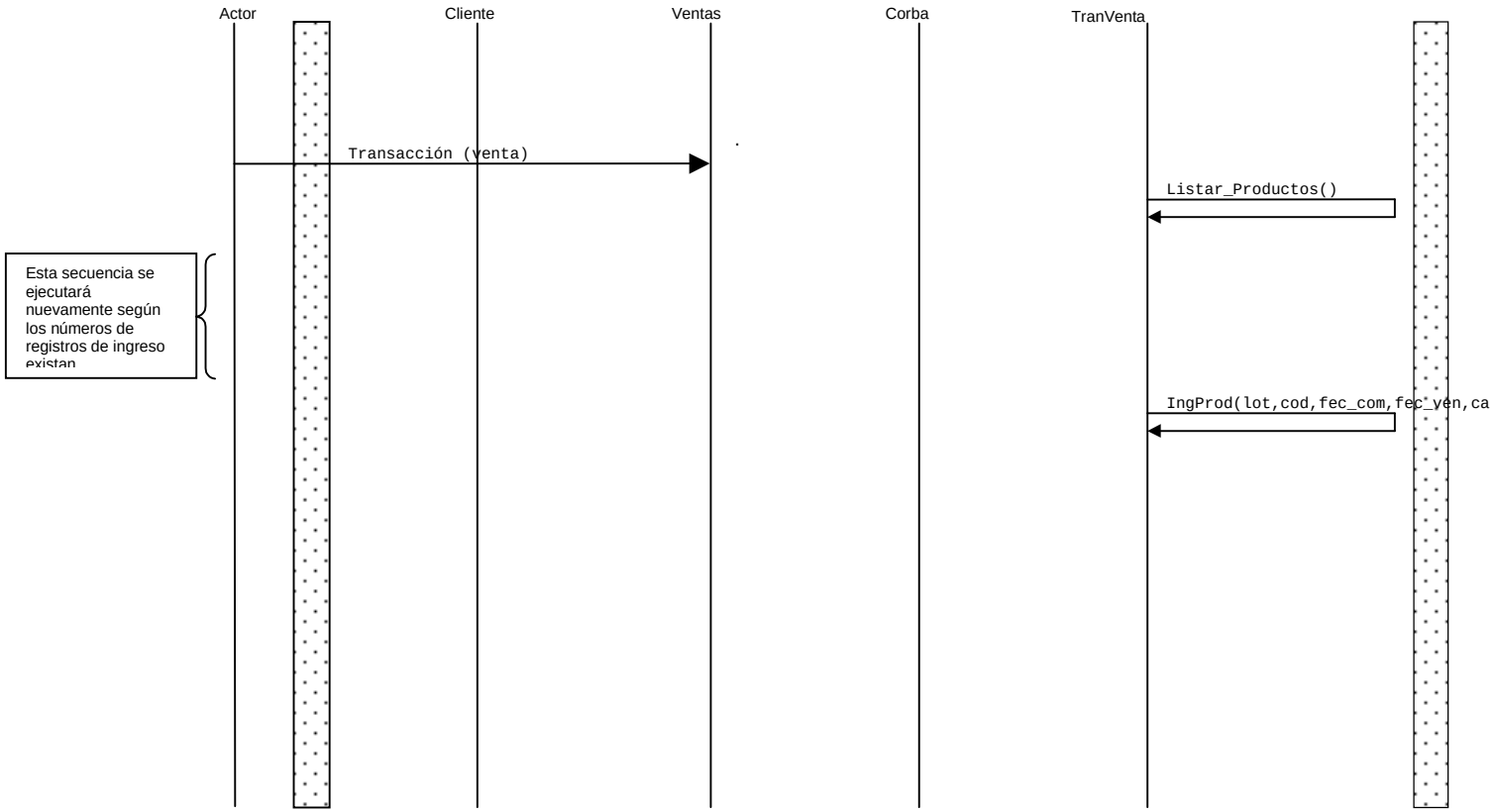

Figura 4.20 Diagrama de interacción de objetos: escenario 2.1

2.2. Se desea ingresar un producto que no existe

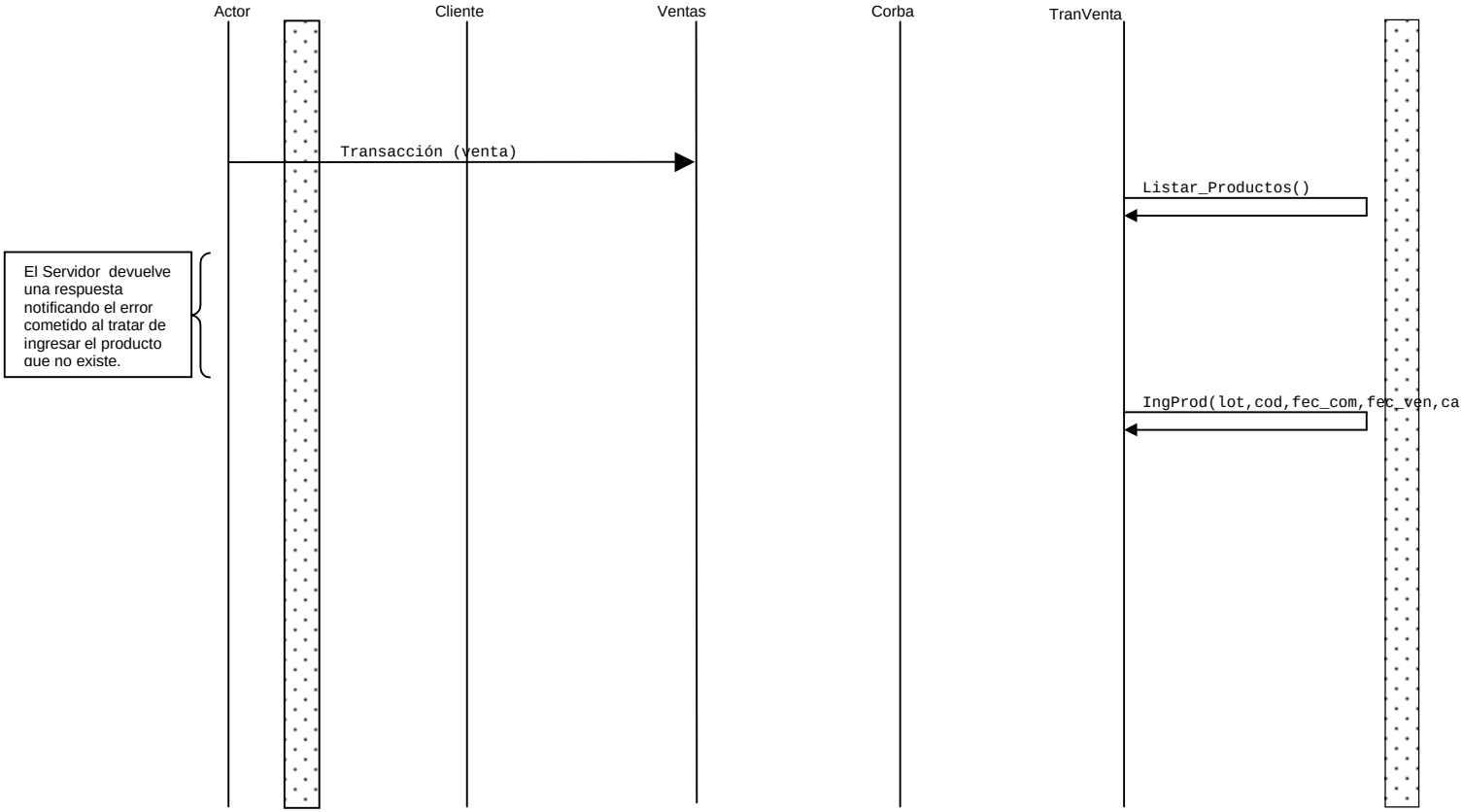

Figura 4.21 Diagrama de interacción de objetos: escenario 2.2

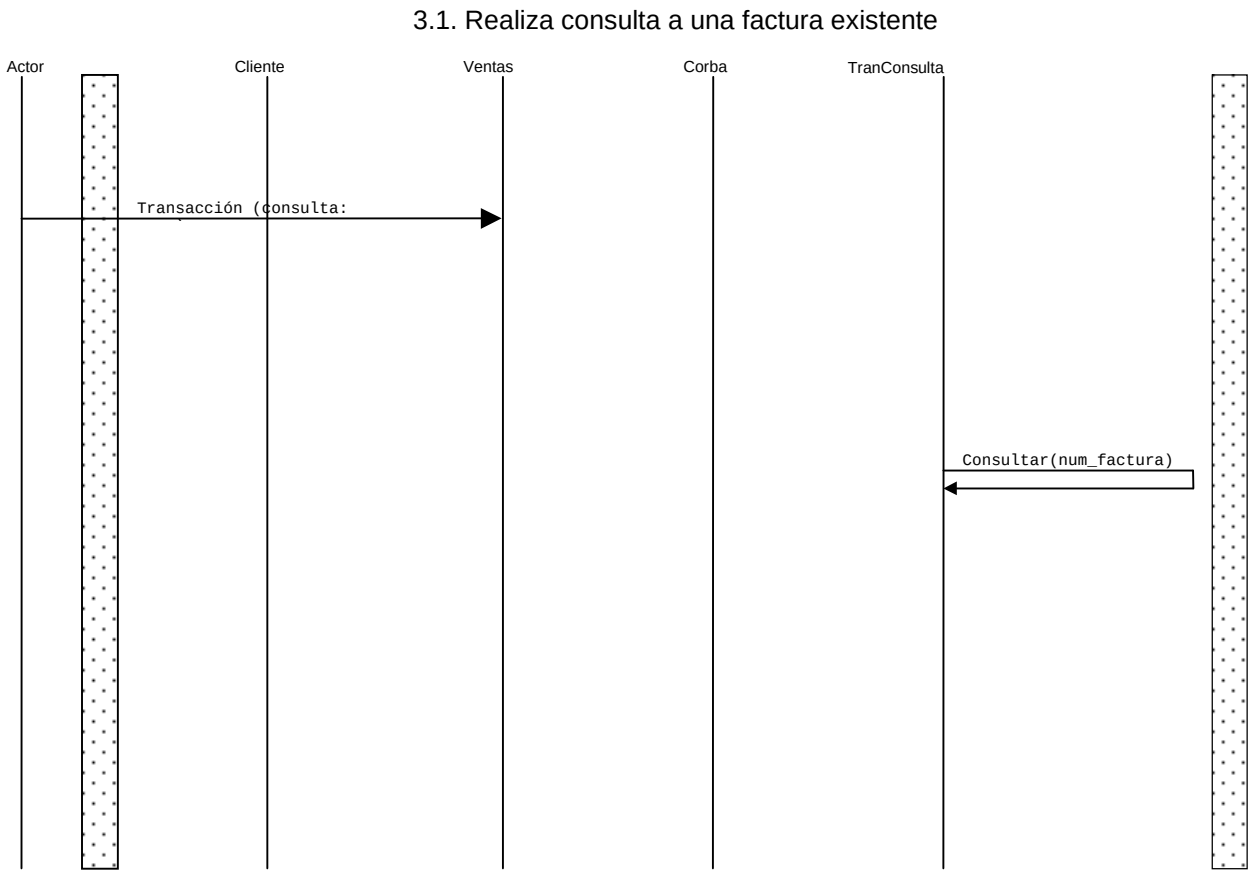

Figura 4.22 Diagrama de interacción de objetos: escenario 3.1

3.2. Se trata de consultar una factura que no existe

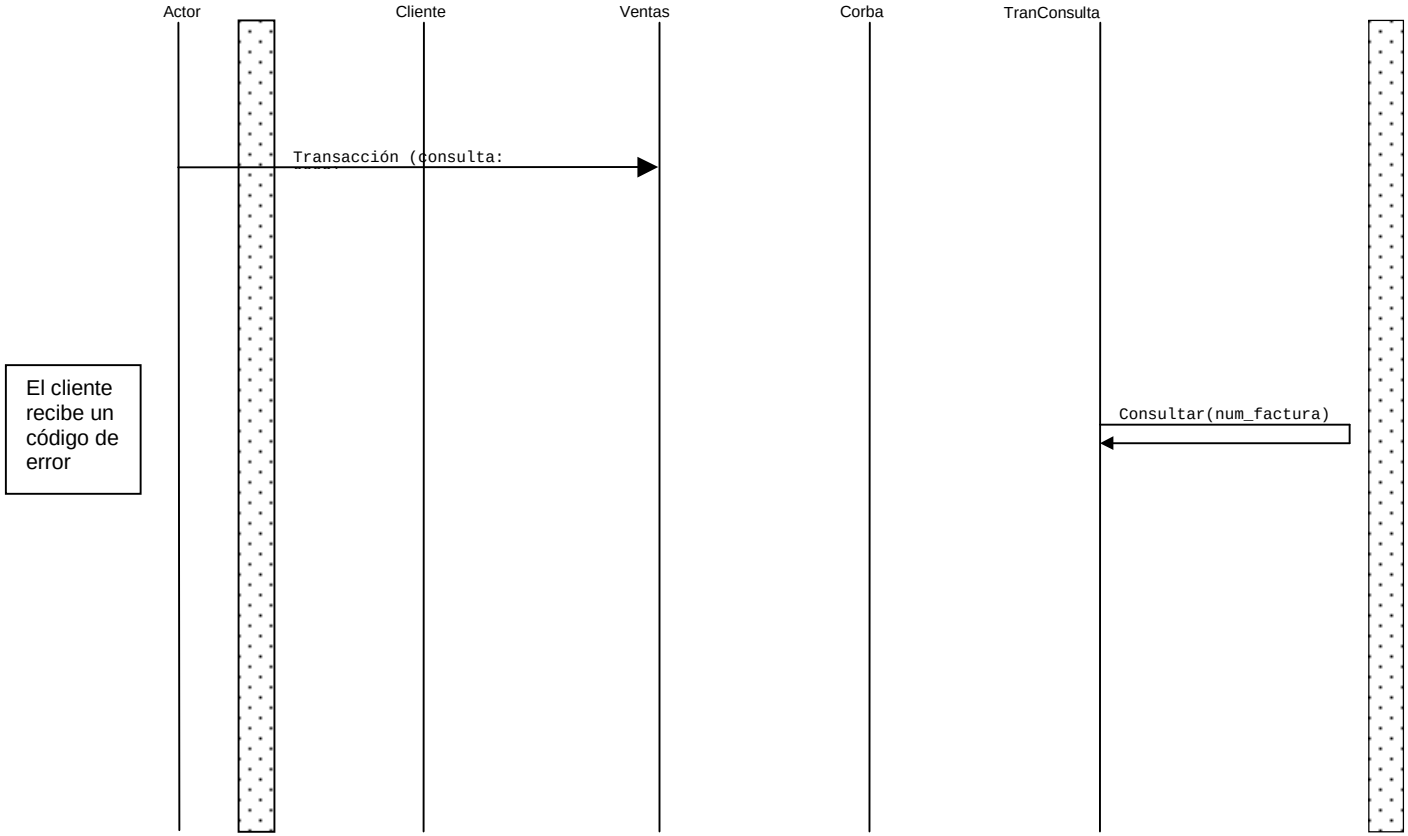

Figura 4.23 Diagrama de interacción de objetos: escenario 3.2

4.1. Se da de baja productos de un lote existente

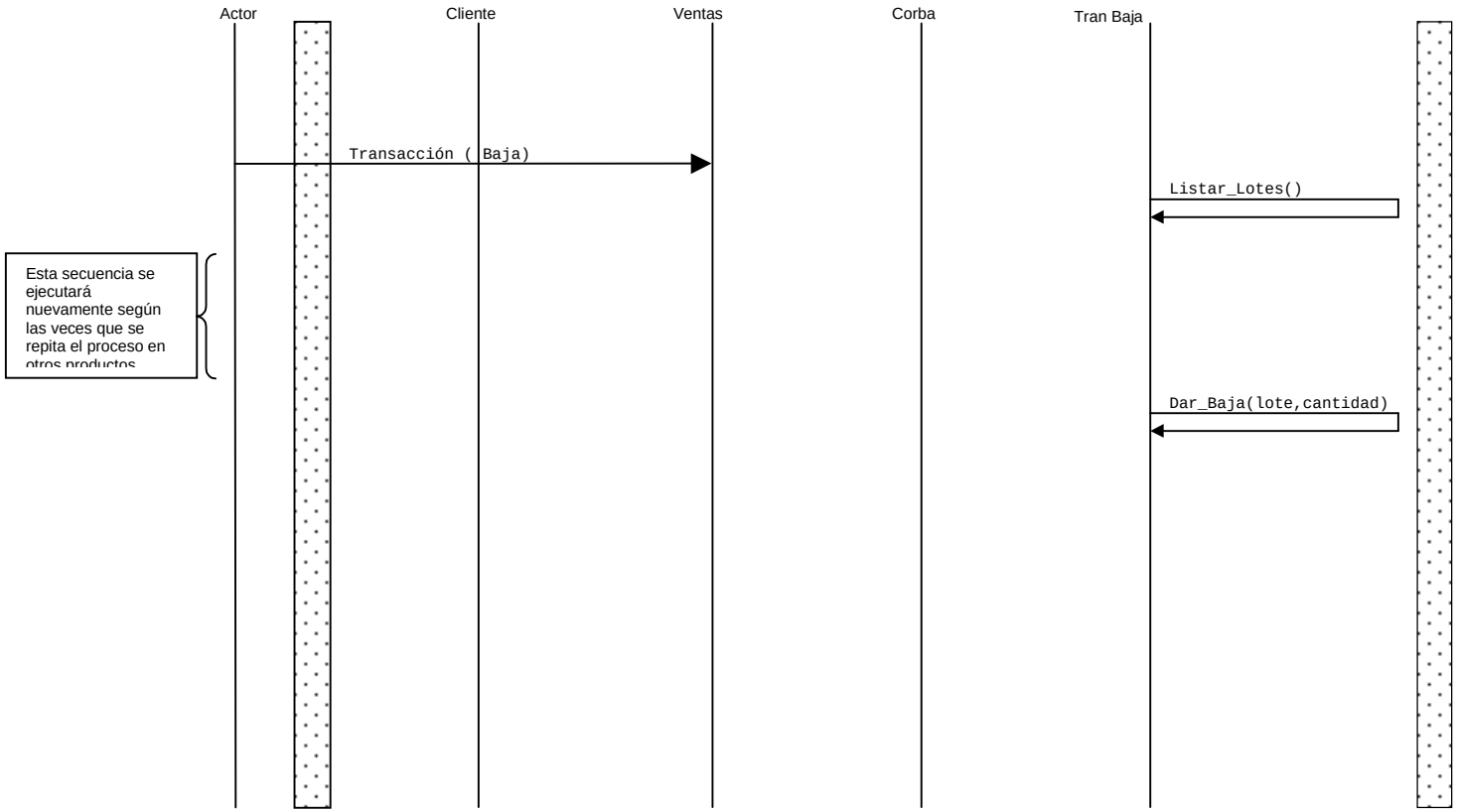

Figura 4.24 Diagrama de interacción de objetos: escenario 4.1

4.2. Se trata dar de baja productos de un lote que no existe.

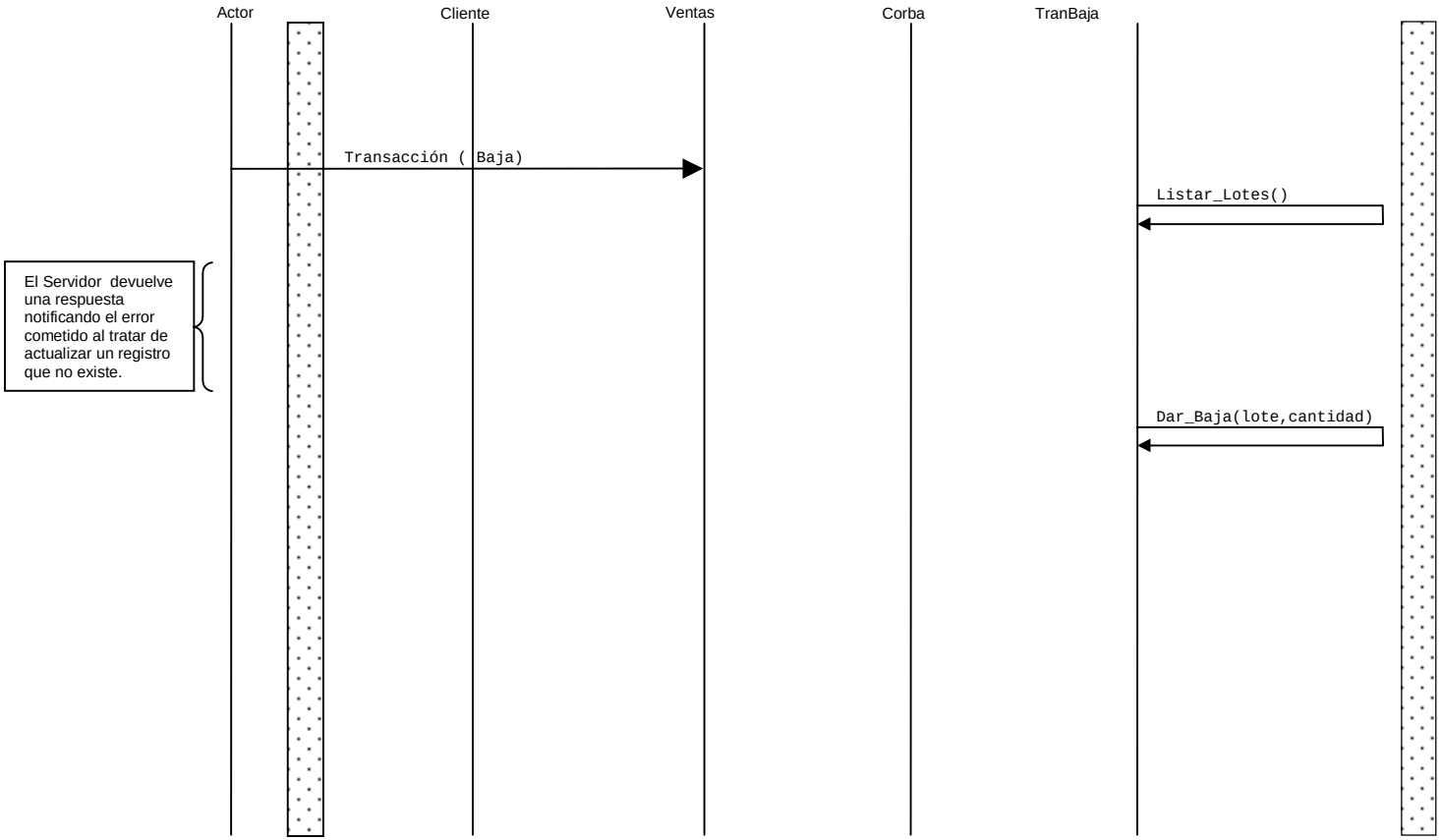

Figura 4.25 Diagrama de interacción de objetos: escenario 4.2

5.1. Se crea un nuevo producto sin conflictos.

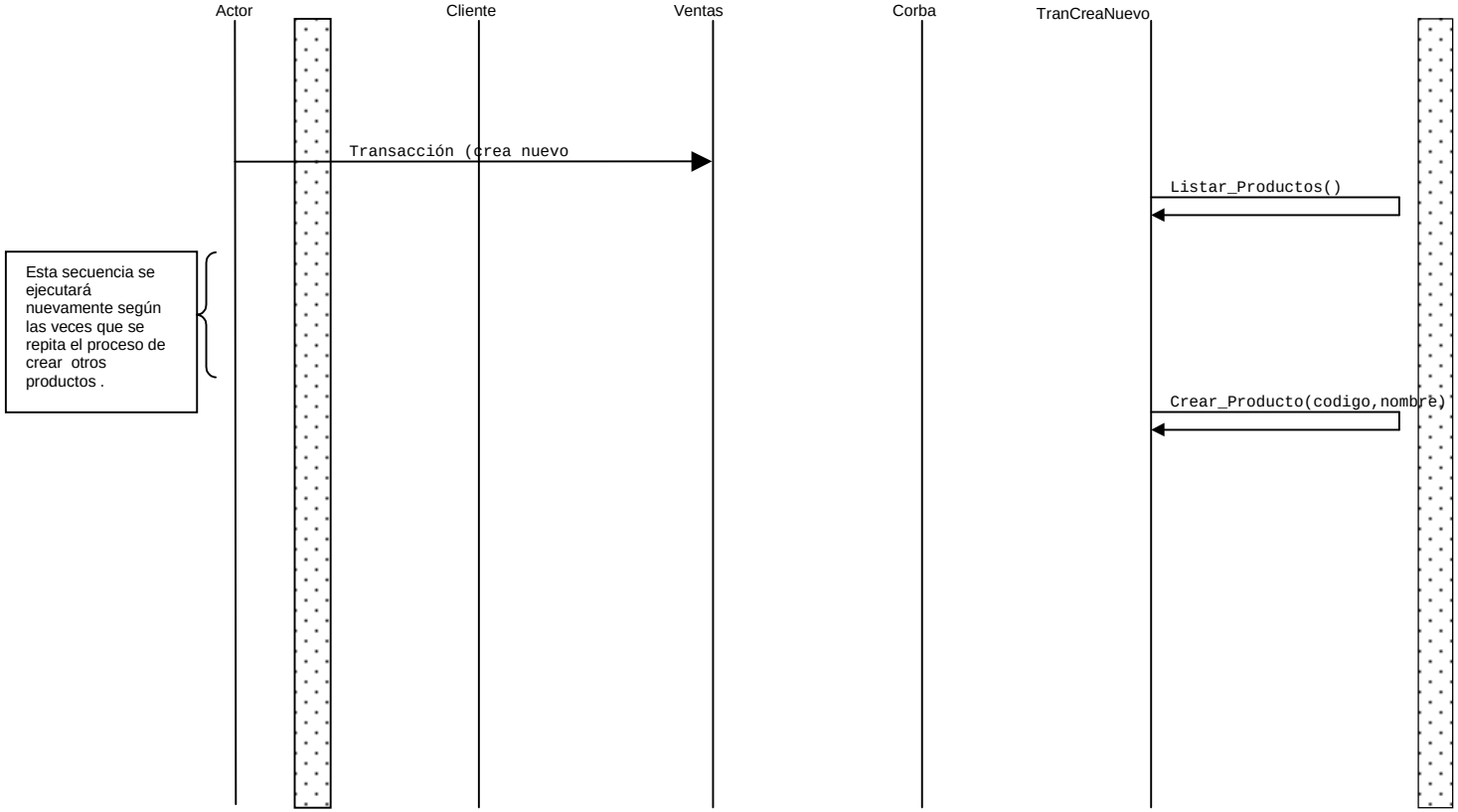

Figura 4.26 Diagrama de interacción de objetos: escenario 5.1

5.2. Se trata crear un nuevo producto con un código ya existente.

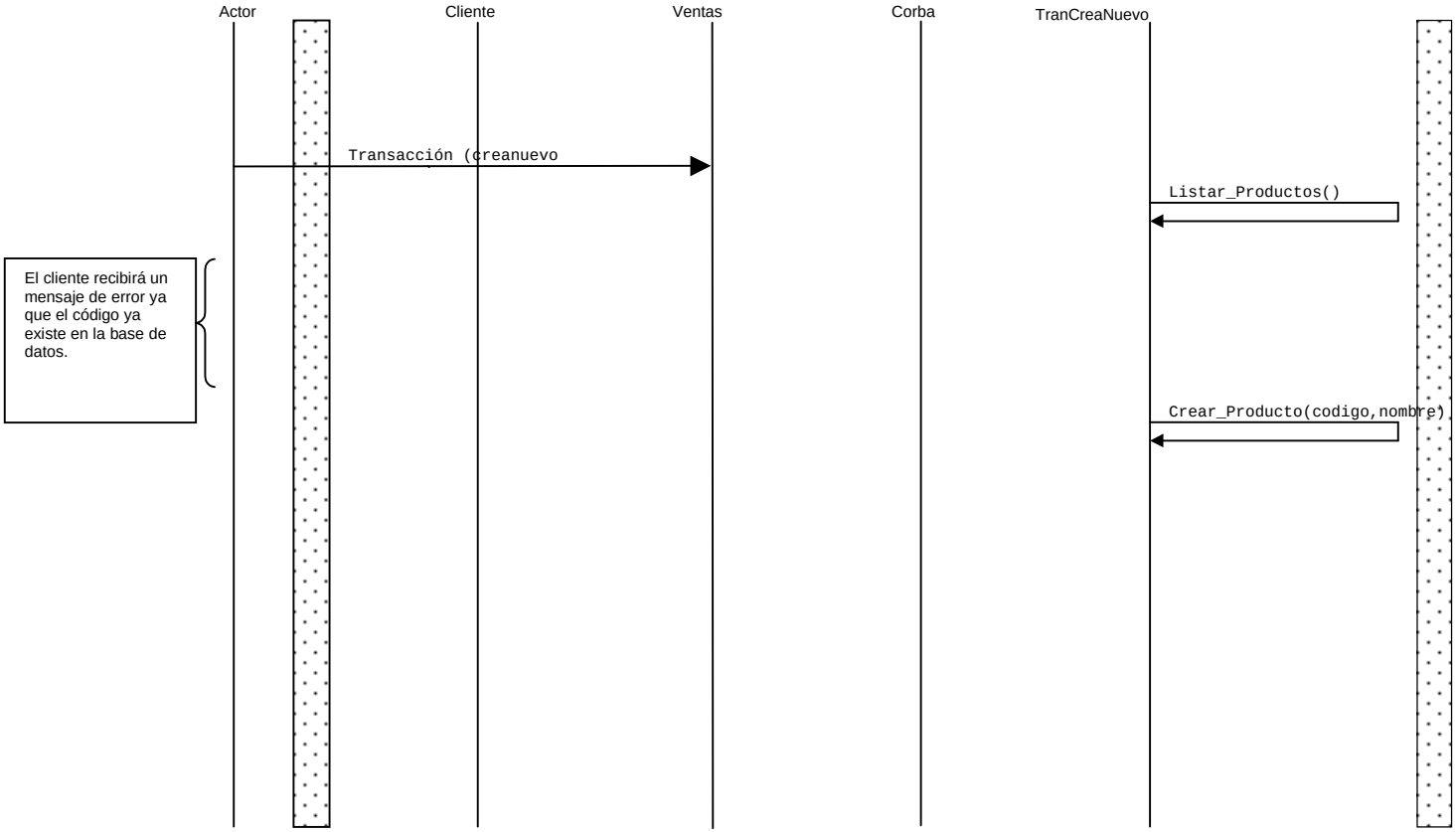

Figura 4.27 Diagrama de interacción de objetos: escenario 5.2

6.1. Se elimina un producto sin conflictos.

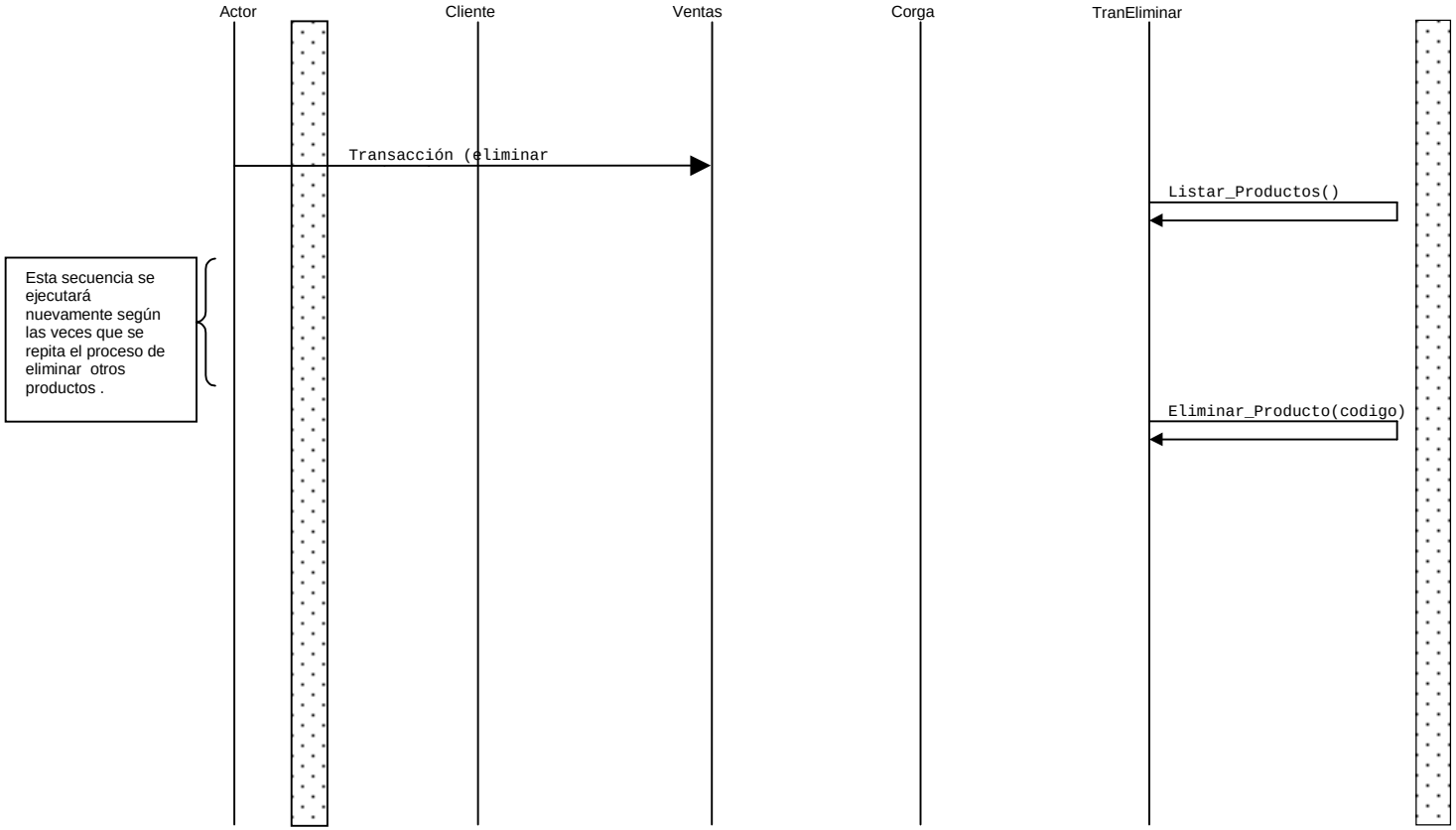

Figura 4.28 Diagrama de interacción de objetos: escenario 6.1

6.2. Se trata de eliminar un producto que aún tiene una cantidad en inventario.

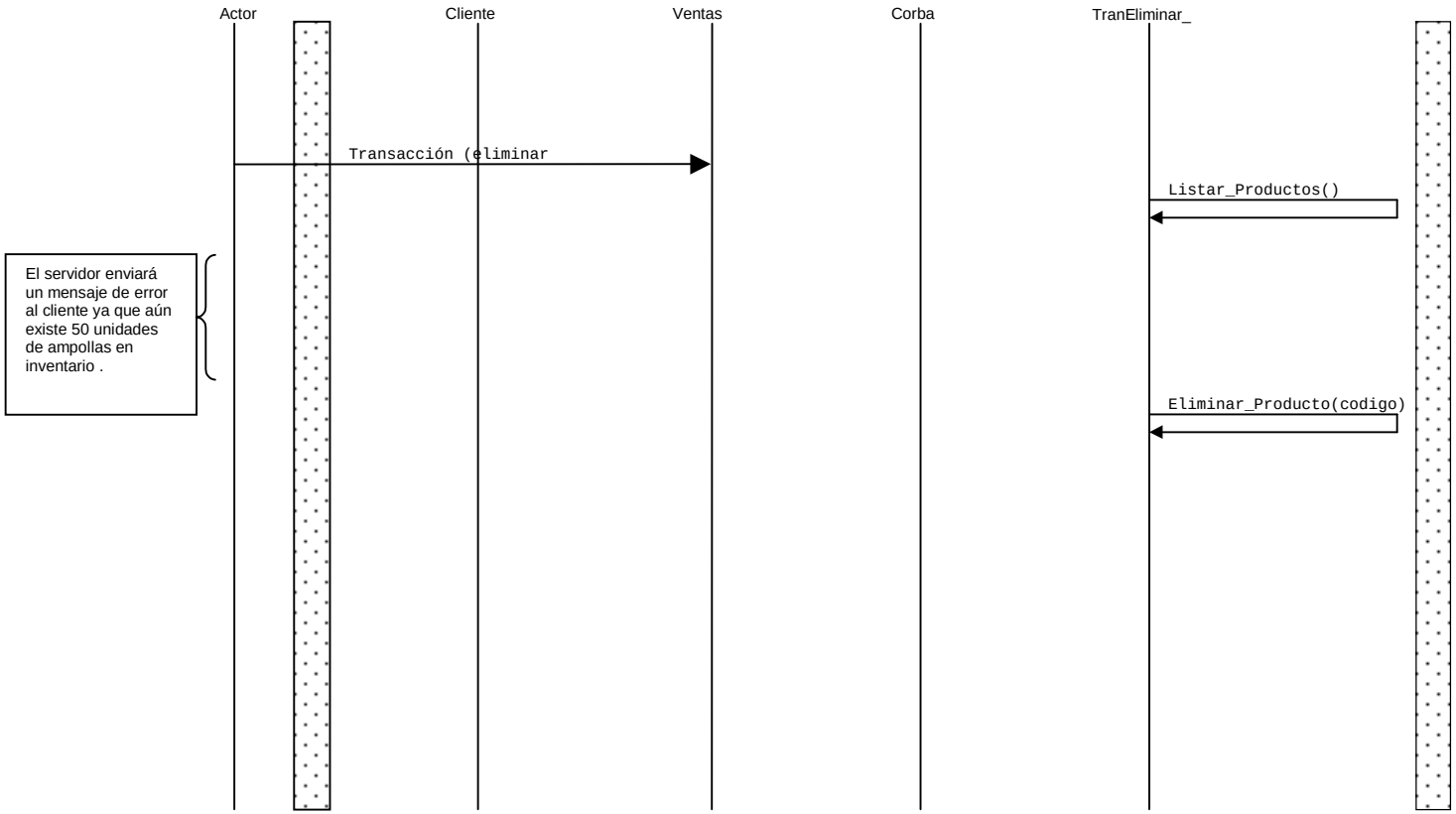

Figura 4.29 Diagrama de interacción de objetos: escenario 6.2

6.3. Se trata de eliminar un producto que no existe.

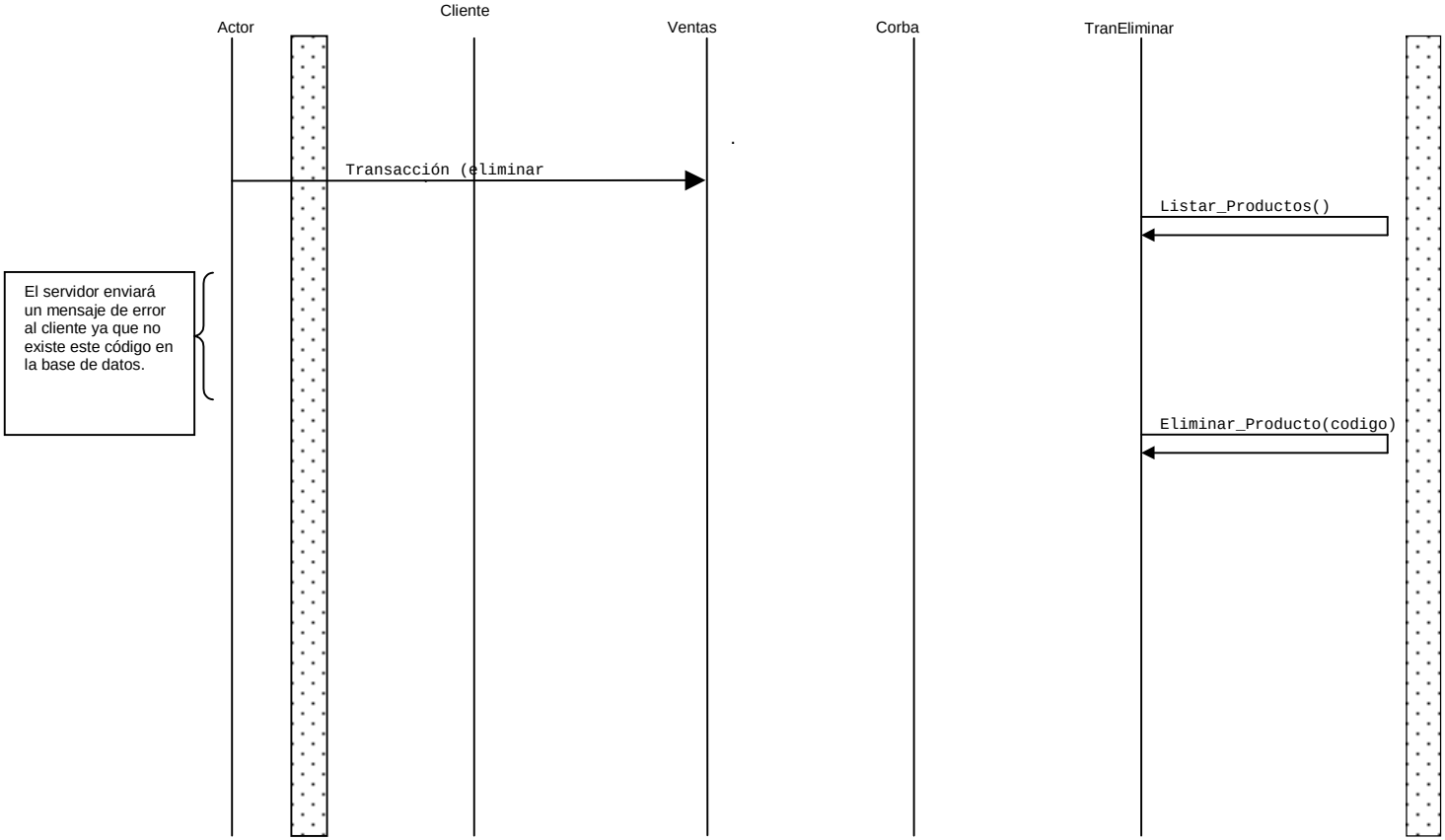

Figura 4.30 Diagrama de interacción de objetos: escenario 6.3

7.1. Se ejecuta el proceso sin conflictos.

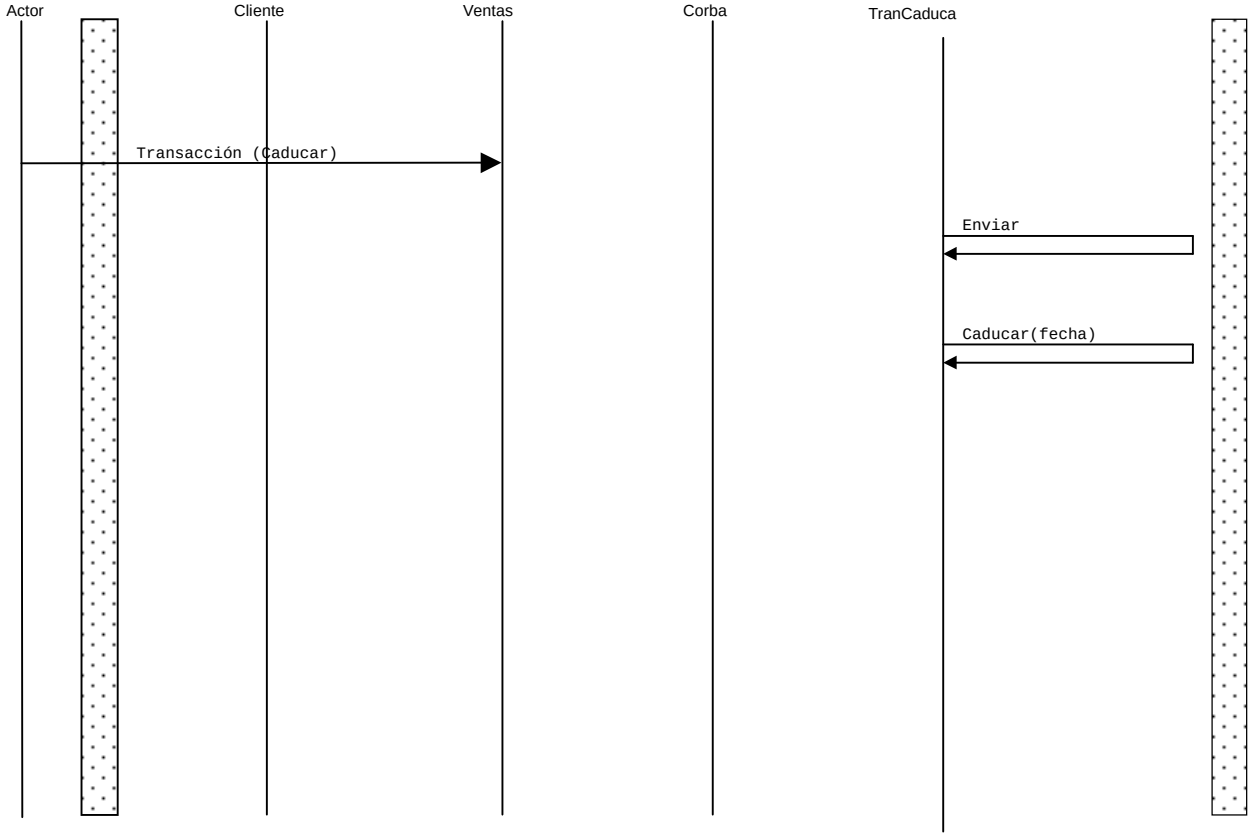

Figura 4.31 Diagrama de interacción de objetos: escenario 7.1

4.2.5 Flujo de layouts del applet cliente.

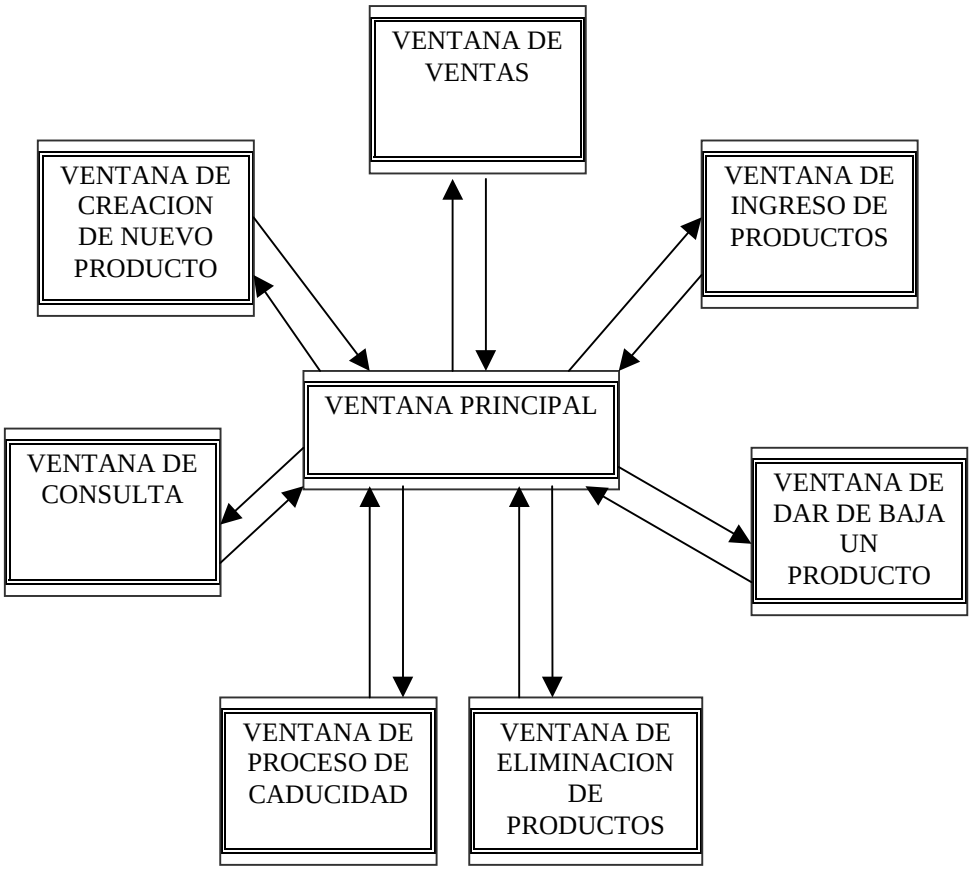

Figura 4.32 Flujo de layouts

Figura 4.33 Pantalla principal

Figura 4.34 Pantalla de ventas

Figura 4.35 Pantalla de consultas

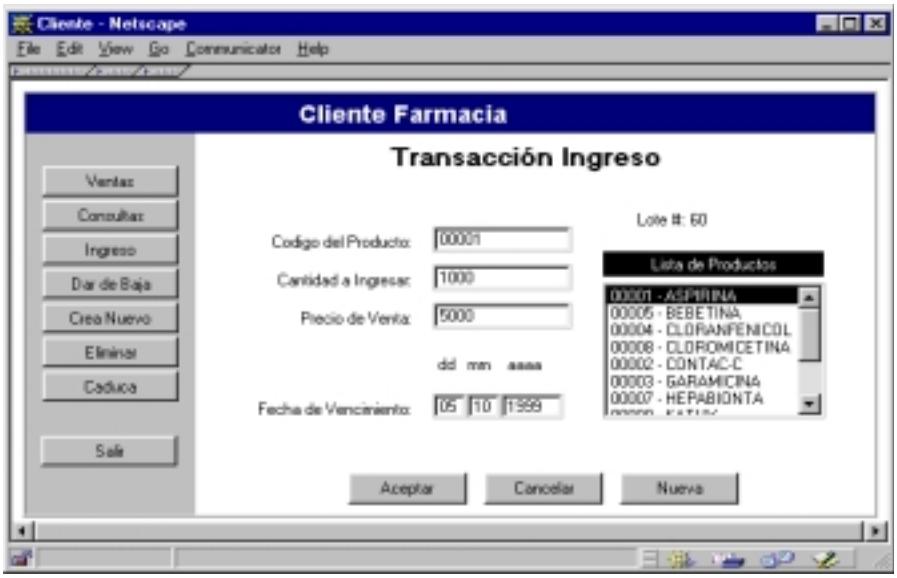

Figura 4.36 Pantalla de ingreso de productos

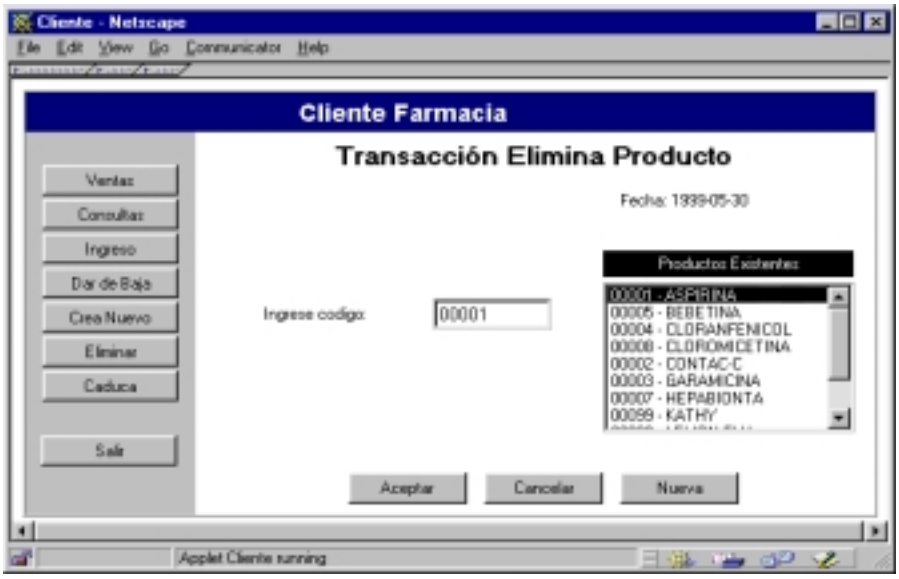

Figura 4.37 Pantalla eliminar productos

Figura 4.38 Pantalla dar de baja productos

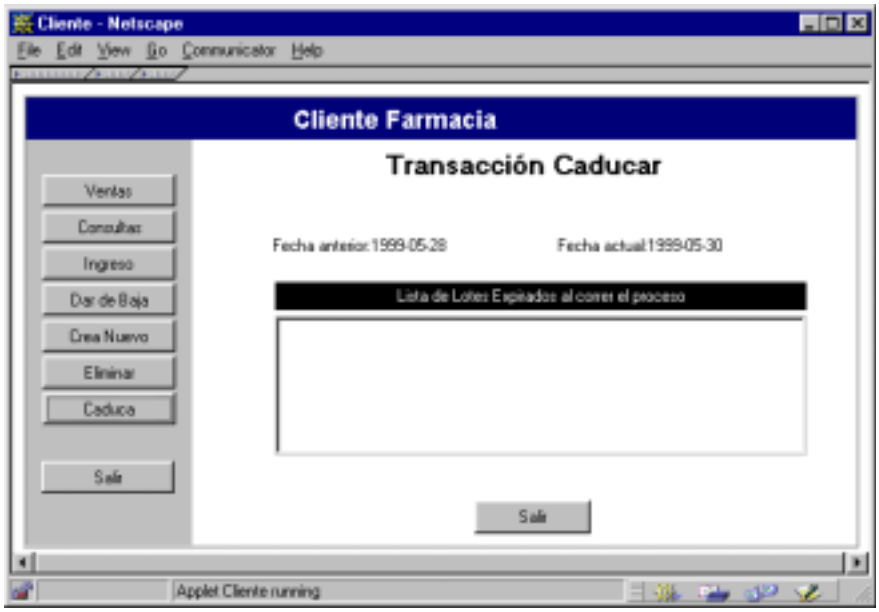

Figura 4.39 Pantalla de caducar productos

4.3 Diseño del servidor.

Para el desarrollo del servidor fue necesario crear los siguientes objetos.

Servidor_Farmacia.- Este objeto se responsabiliza del inicio de la aplicación y creación de los demás objetos que le dan funcionalidad al servidor.

Base_Datos.- Este objeto se encarga de establecer la conexión con la Base de Datos y manejar todas las operaciones referente a la base de datos.

TranVenta.- Este objeto tiene como responsabilidad controlar todo el proceso de venta de uno o varios productos.

TranConsulta.- Este objeto tiene como responsabilidad controlar todo el proceso de consulta de una venta ya realizada.

TranIngresar.- Este objeto tiene como responsabilidad controlar todo el proceso de ingreso de productos en inventario.

TranBaja.- Este objeto tiene como responsabilidad controlar todo el proceso de dar de baja, total o parcialmente el stock de productos en inventario.

TranEliminar.- Este objeto tiene como responsabilidad controlar todo el proceso de eliminación de un producto de la tabla de productos en la base.

TranCreaNuevo.- Este objeto tiene como responsabilidad controlar todo el proceso de creación de un nuevo producto en la tabla productos en la base.

TranCaduca.- Este objeto tiene como responsabilidad controlar todo el proceso de expiración de uno o varios productos en inventario, basándose en la fecha de expiración de cada lote ingresado a inventario.

4.3.1 Diagrama de Clases del Servidor

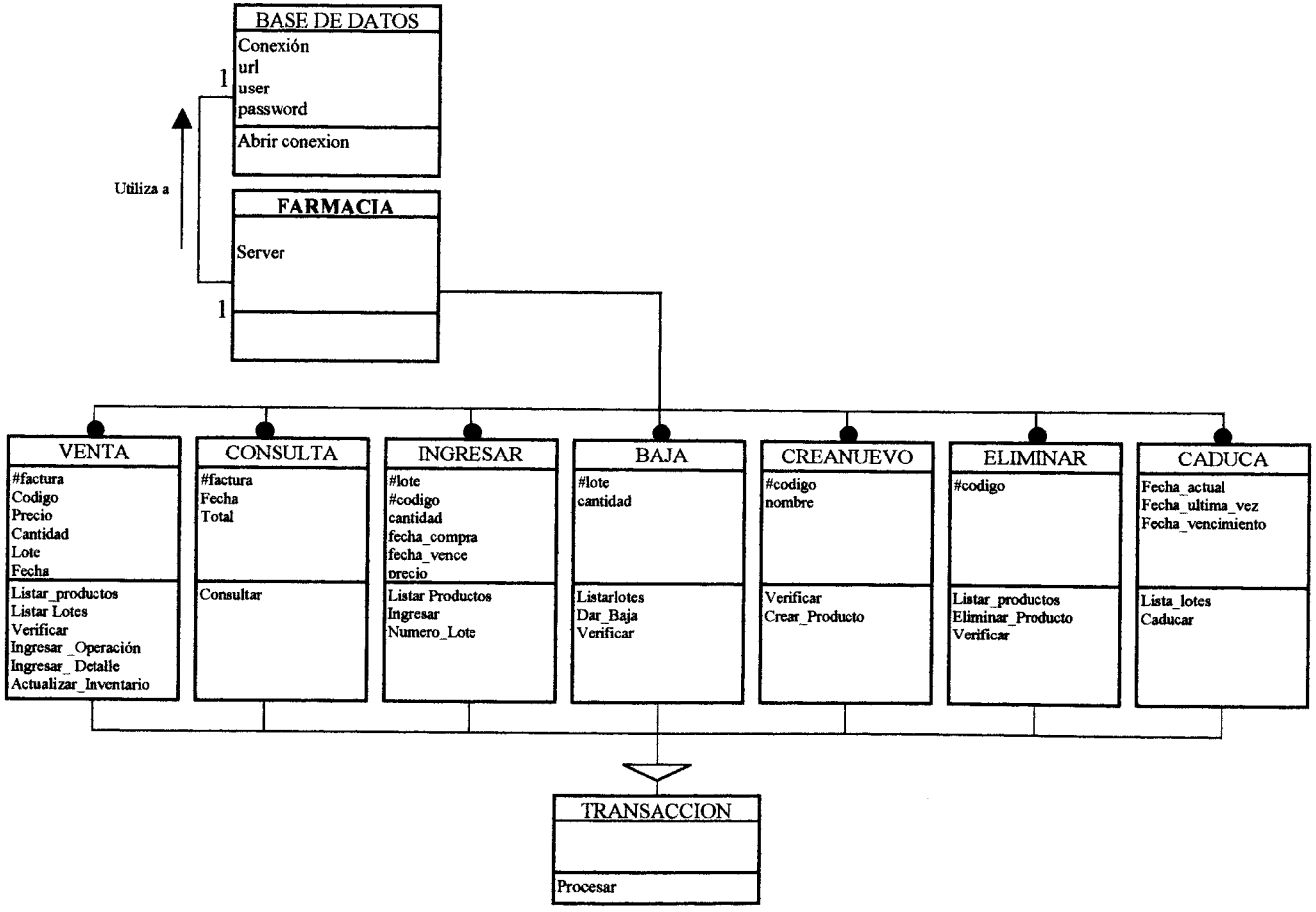

Figura 4.40 Diagramas de clase del servidor

4.3.2 Diseño de los Datos manejados por el servidor.

Para el diseño de los datos, se utilizó la herramienta SQL Server. Se crearon las siguientes tablas: OPERACIONES, DETALLE DE OPERACIÓN, INVENTARIO, PRODUCTOS, SECUENCIALES.

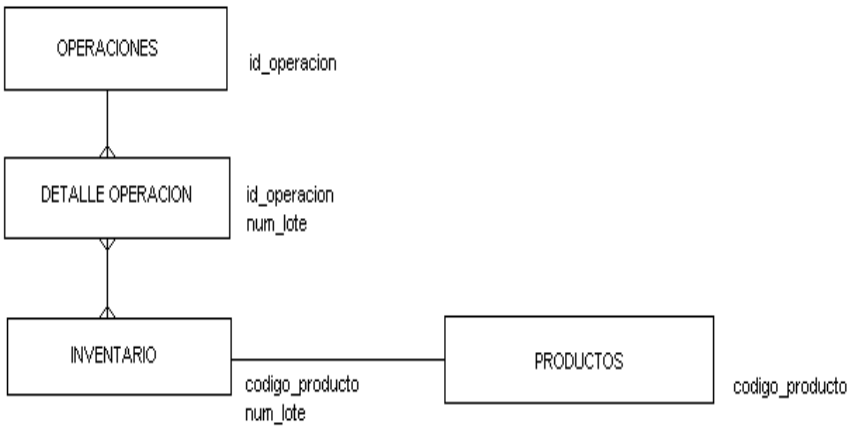

Figura 4.41 Relaciones de las tablas de la base de datos.

4.3.2.1 Diccionario de la base de datos.

OPERACIONES:

ID_FACTURA: Código de la factura. Tipo de dato: numérico. Clave primaria.

FECHA_FACTURA: Fecha de la factura. Tipo de dato: datetime.

VALOR_TOTAL: Valor total a pagarse impreso en la factura. Tipo de dato: numérico.

DETALLE DE OPERACIÓN

ID_FACTURA: Código de la factura. Tipo de dato: numérico. Clave primaria.

NUM_LINEA: Número de línea de la factura. Tipo de dato: numérico.

CODIGO_PRODUCTO: Código del producto. Tipo de dato: Varchar.

PRECIO: Precio del producto. Tipo de dato: numérico.

CANTIDAD: Cantidad del producto. Tipo de dato: numérico.

NUM_LOTE: Número del lote. Tipo de dato: numérico.

INVENTARIO

CODIGO_PRODUCTO: Código del producto. Tipo de dato: char. Clave primaria.

CANTIDAD: Indica la cantidad del producto. Tipo de dato: numérico.

FECHA_COMPRA: Indica fecha de compra del producto. Tipo de dato: datetime.

FECHA_VENCIMIENTO: Fecha de vencimiento del producto. Tipo de dato: datetime.

ESTATUS: Indica el estatus del producto, D de disponible y V de vencido. Tipo de dato: char.

PRECIO: Precio del producto. Tipo de dato: numérico.

NUM_LOTE: Número del lote de productos. Tipo: numérico. Clave primaria.

PRODUCTOS

CODIGO_PRODUCTO: Código del producto. Tipo de dato: char. Clave primaria.

NOMBRE_PRODUCTO: Nombre del producto. Tipo de dato: char.

SECUENCIALES

SEC_OPERACIONES: Indica las diferentes secuencias de operaciones realizadas. Tipo de dato: numérico.

SEC_LOTE: Indica la secuencia del lote de cada producto. Tipo de dato: numérico.

CONCLUSIONES Y RECOMENDACIONES

1 El sistema Control de Ventas e Inventario para ser utilizado en Farmacias, ha sido desarrollado utilizando las más modernas técnicas y herramientas informáticas tales como; diseño orientado a objetos que permite que el código del sistema sea reutilizado y de fácil mantenimiento. Por otro lado, el modelo distribuido Cliente/Servidor implementado usando tecnología Corba hace que sea transparente para la aplicación la ubicación del servidor, protocolos de comunicación, plataforma de operación, lenguaje de implementación, etc., características que son manejadas por el componente central de Corba ORB (Object Request Broker). Para el desarrollo del sistema se utilizó el lenguaje Java.

Por lo antes expuesto, el sistema Control de Ventas e Inventario para una Farmacia al utilizar la tecnología antes descrita permite que este diseño sea puesto en internet para que clientes remotos puedan realizar transacciones , tales como: venta, consulta de productos, etc.

2 Dados los grandes beneficios que proporciona esta nueva tecnología se recomienda que las futuras aplicaciones que se desarrollen en nuestro país deben seguir esta tendencia informática, aportando al sector empresarial un mejor nivel competitivo.

BIBLIOGRAFIA

1. ORFALI R., HARKEY D. & EDWARDS J., Essential Client/Server Survival Guide. Second Edition, New York: John Wiley & Sons, Inc, 1996.

2. ORFALI R. & HARKEY D., Client / Server Programming with Java and Corba. Second Edition, New York: John Wiley & Sons, Inc, 1998.

3. LEMAY LAURA Y PERKINS CHARLES. Aprendiendo Java 1.1 en 21 días. Segunda edición, Prentice Hall, 1998.

4. PELAEZ E. PhD. Curso de Análisis y Diseño Orientado a Objetos. FIEC, ESPOL, 1996.

5. BOOCH GRADY. Análisis y Diseño Orientado a Objetos con Aplicaciones. Segunda edición, Addison-Wesley/Díaz de Santos, 1996.

6. CABRERA I., ECHEVERRÍA F., Simulación del Proceso de Comercialización Agrícola (Tópico de Graduación: Tecnología Cliente/Servidor con Arquitectura Corba, FIEC, ESPOL, 1998).