ESCUELA POLITÉCNICA DEL EJÉRCITO

SEDE LATACUNGA

FACULTAD DE INGENIERÍA EN SISTEMAS E INFORMÁTICA

DESARROLLO DE APLICACIONES DISTRIBUIDAS BAJO INVOCACION DE METODOS REMOTOS EN ENTORNO TCP-IP

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS E INFORMÁTICA

ZAMBRANO SERRANO WILSON FRANCISCO

Latacunga, Julio de 2006

CERTIFICACION

Se certifica que el presente trabajo fue desarrollado por: Wilson Francisco Zambrano Serrano, bajo nuestra supervisión.

_____

Ing. Raúl Rosero
DIRECTOR DE PROYECTO

_____

Ing. Santiago Jácome
CODIRECTOR DE PROYECTO

AGRADECIMIENTO:

Principalmente a mis padres, Luis y Maria, por haberme dado la vida y asegurarse de que creciera con los valores que me convertirían en quien soy ahora. No conozco límite para el amor y comprensión que me han dado.

A mis 3 hermanos, por acompañarme siempre, a Dios, a la Escuela Politécnica del Ejercito sede Latacunga por el apoyo, pero sobre todo a los señores y señoras profesores (as) y compañeras y compañeros por el ánimo y apoyo que me brindaron para finalizar cada uno de los cursos

DEDICATORIA:

Este logro lo dedico a los seres más importantes y sublimes de mi vida:

A mis padres, a quienes honraré en todo momento.

A mis hermanos, para quienes serviré de ejemplo.

CAPITULO I

OBJETOS DISTRIBUIDOS

1.1. INTRODUCCIÓN

Un sistema distribuido se define como una colección de computadores autónomos conectados por una red, y con el software distribuido adecuado para que el sistema sea visto por los usuarios como una única entidad capaz de proporcionar facilidades de computación.

El desarrollo de los sistemas distribuidos vino de la mano de las redes locales de alta velocidad a principios de 1970. Más recientemente, la disponibilidad de computadoras personales de altas prestaciones, estaciones de trabajo y servidores ha resultado en un mayor desplazamiento hacia los sistemas distribuidos entre los servidores centralizados multiusuario. Esta tendencia se ha acelerado por el desarrollo de software para sistemas distribuidos, diseñado para soportar el desarrollo de aplicaciones distribuidas. Este software permite a los servidores coordinar sus actividades y compartir los recursos del sistema, hardware, software y datos.

Los sistemas distribuidos se implementan en diversas plataformas hardware, ya no estamos sujetos a las restricciones de la máquina, ahora somos capaces de utilizar los recursos de toda una red, hasta Internet, una colección de redes de área local y de área extensa interconectados, que enlazan millones de servidores.

Las aplicaciones de los sistemas distribuidos varían desde la provisión de capacidad de cómputo a grupos de usuarios, hasta sistemas bancarios, comunicaciones multimedia y abarcan prácticamente todas las aplicaciones comerciales y técnicas de los servidores. Los requisitos de dichas aplicaciones incluyen un alto nivel de fiabilidad, seguridad contra interferencias externas y privacidad de la información que el sistema mantiene. Se deben proveer accesos concurrentes a bases de datos por parte de muchos usuarios, garantizar tiempos de respuesta, proveer puntos de acceso al servicio que están distribuidos geográficamente, potencial para el crecimiento del sistema para acomodar la expansión del negocio y un marco para la integración de sistema usados por diferentes compañías y organizaciones de usuarios.

Un aspecto escasamente tenido en cuenta a la hora de desarrollar aplicaciones distribuidas es el de la fiabilidad lo cual ocasiona degradaciones de la calidad de servicio o su interrupción temporal. Ello es debido en gran parte a que los diseñadores dejan este aspecto para la fase de implementación y a la falta de soporte para la implementación de mecanismos estándares que permitan reducir la complejidad añadida para su tratamiento.

1.2. CARACTERÍSTICAS CLAVE DE LOS SISTEMAS DISTRIBUIDOS

Se establece que son seis las características principales responsables de la utilidad de los sistemas distribuidos. Se trata de compartir los recursos, apertura (openness), concurrencia, escalabilidad, tolerancia a fallos y transparencia.

- **COMPARTICIÓN DE RECURSOS**

El término 'recurso' es bastante abstracto, pero es el que mejor caracteriza el abanico de entidades que pueden compartirse en un sistema distribuido. El abanico se extiende desde componentes hardware como discos e impresoras hasta elementos software como ficheros, ventanas, bases de datos y otros objetos de datos.

La idea de compartición de recursos no es nueva ni aparece en el marco de los sistemas distribuidos. Los sistemas multiusuario clásicos desde siempre han provisto compartición de recursos entre sus usuarios. Sin embargo, los recursos de una computadora multiusuario se comparten de manera natural entre todos sus usuarios. Por el contrario, los usuarios de estaciones de trabajo monousuario o computadoras personales dentro de un sistema distribuido no obtienen automáticamente los beneficios de la compartición de recursos.

Los recursos en un sistema distribuido están físicamente encapsulados en una de las computadoras y sólo pueden ser accedidos por otras computadoras mediante las comunicaciones (la red). Para que la compartición de recursos sea efectiva, ésta debe ser manejada por un programa que ofrezca un interfaz de comunicación permitiendo que el recurso sea accedido, manipulado y actualizado de una manera fiable y consistente. Surge el término genérico de gestor de recursos.

Un gestor de recursos es un modulo software que maneja un conjunto de recursos de un tipo en particular. Cada tipo de recurso requiere algunas políticas y métodos específicos junto con requisitos comunes para todos ellos. Éstos incluyen la provisión de un esquema de nombres para cada clase de recurso, permitir que los recursos individuales sean accedidos desde cualquier localización; la traslación de nombre de recurso a direcciones de comunicación y la coordinación de los accesos concurrentes que cambian el estado de los recursos compartidos para mantener la consistencia.

Esta perspectiva nos lleva a dos modelos de sistemas distribuidos: el modelo cliente - servidor y el modelo basado en objetos.

- **APERTURA (OPENNESSS)**

Un sistema informático es abierto si el sistema puede ser extendido de diversas maneras. Un sistema puede ser abierto o cerrado con respecto a extensiones hardware (añadir periféricos, memoria o interfaces de comunicación, etc...) o con respecto a las extensiones software (añadir características al sistema operativo, protocolos de comunicación y servicios de compartición de recursos, etc...).

La apertura de los sistemas distribuidos se determina primariamente por el grado hacia el que nuevos servicios de compartición de recursos se pueden añadir sin perjudicar ni duplicar a los ya existentes.

- **CONCURRENCIA**

Cuando existen varios procesos en una única maquina decimos que se están ejecutando concurrentemente. Si el servidor esta equipado con un único procesador central, la concurrencia tiene lugar entrelazando la ejecución de los distintos procesos. Si la computadora tiene N procesadores, entonces se pueden estar ejecutando estrictamente a la vez hasta N procesos.

En los sistemas distribuidos hay muchas maquinas, cada una con uno o mas procesadores centrales. Es decir, si hay M servidores en un sistema distribuido con un procesador central cada una entonces hasta M procesos estará ejecutándose en paralelo.

En un sistema distribuido que esta basado en el modelo de compartición de recursos, la posibilidad de ejecución paralela ocurre por dos razones:

- Muchos usuarios interactúan simultáneamente con programas de aplicación.
- Muchos procesos servidores se ejecutan concurrentemente, cada uno respondiendo a diferentes peticiones de los procesos clientes.

El caso (1) es menos conflictivo, ya que normalmente las aplicaciones de interacción se ejecutan aisladamente en la estación de trabajo del usuario y no entran en conflicto con las aplicaciones ejecutadas en las estaciones de trabajo de otros usuarios.

El caso (2) surge debido a la existencia de uno o más procesos servidores para cada tipo de recurso. Estos procesos se ejecutan en distintas maquinas, de manera que se están ejecutando en paralelo diversos servidores, junto con diversos programas de aplicación. Las peticiones para acceder a los recursos de un servidor dado pueden ser encoladas en el servidor y ser procesadas secuencialmente o bien pueden ser procesadas varias concurrentemente por múltiples instancias del proceso gestor de recursos. Cuando esto ocurre los procesos servidores deben sincronizar sus acciones para asegurarse de que no existen conflictos. La sincronización debe ser cuidadosamente planeada para asegurar que no se pierden los beneficios de la concurrencia.

- **ESCALABILIDAD**

Los sistemas distribuidos operan de manera efectiva y eficiente a muchas escalas diferentes. La escala más pequeña consiste en dos estaciones de trabajo y un servidor de ficheros, mientras que un sistema distribuido construido alrededor de una red de área local simple podría contener varios cientos de estaciones de trabajo,

varios servidores de ficheros, servidores de impresión y otros servidores de propósito específico. A menudo se conectan varias redes de área local para formar internetworks, y éstas podrían contener muchos miles de servidores que forman un único sistema distribuido, permitiendo que los recursos sean compartidos entre todos ellos.

Tanto el software de sistema como el de aplicación no deberían cambiar cuando la escala del sistema se incrementa. La necesidad de escalabilidad no es solo un problema de prestaciones de red o de hardware, sino que esta íntimamente ligada con todos los aspectos del diseño de los sistemas distribuidos. El diseño del sistema debe reconocer explícitamente la necesidad de escalabilidad o de lo contrario aparecerán serias limitaciones.

La demanda de escalabilidad en los sistemas distribuidos ha conducido a una filosofía de diseño en que cualquier recurso simple - hardware o software puede extenderse para proporcionar servicio a tantos usuarios como se quiera. Esto es si la demanda de un recurso crece, debería ser posible extender el sistema para darla servicio, Por ejemplo, la frecuencia con la que se accede a los ficheros crece cuando se incrementa el numero de usuarios y estaciones de trabajo en un sistema distribuido. Entonces, debe ser posible añadir servidores para evitar el cuello de botella que se producirla si un solo servidor de ficheros tuviera que manejar todas las peticiones de acceso a los ficheros. En este caso el sistema deberá estar diseñado de manera que permita trabajar con ficheros replicados en distintos servidores, con las consideraciones de consistencias que ello conlleva.

Cuando el tamaño y complejidad de las redes de servidores crece, es un objetivo primordial diseñar software de sistema distribuido que seguirá siendo eficiente y útil con esas nuevas configuraciones de la red. Resumiendo, el trabajo necesario para procesar una petición simple para acceder a un recurso compartido debería ser prácticamente independiente del tamaño de la red. Las técnicas necesarias para

conseguir estos objetivos incluyen el uso de datos replicados, la técnica asociada de caching, y el uso de múltiples servidores para manejar ciertas tareas, aprovechando la concurrencia para permitir una mayor productividad

- **TOLERANCIA A FALLOS**

Los sistemas informáticos a veces fallan. Cuando se producen fallos en el software o en el hardware, los programas podrían producir resultados incorrectos o podrían pararse antes de terminar la computación que estaban realizando. El diseño de sistemas tolerantes a fallos se basa en dos cuestiones, complementarias entre sí: Redundancia hardware (uso de componentes redundantes) y recuperación del software (diseño de programas que sean capaces de recuperarse de los fallos).

En los sistemas distribuidos la redundancia puede plantearse en un grano mas fino que el hardware, pueden replicarse los servidores individuales que son esenciales para la operación continuada de aplicaciones críticas.

La recuperación del software tiene relación con el diseño de software que sea capaz de recuperar (rollback) el estado de los datos permanentes antes de que se produjera el fallo.

Los sistemas distribuidos también proveen un alto grado de disponibilidad en la vertiente de fallos hardware. La disponibilidad de un sistema es una medida de la proporción de tiempo que esta disponible para su uso. Un fallo simple en una maquina multiusuario que resulta en la no disponibilidad del sistema para todos los usuarios. Cuando uno de los componentes de un sistema distribuidos falla, solo se ve afectado el trabajo que estaba realizando el componente averiado. Un usuario podría desplazarse a otra estación de trabajo.

- **TRANSPARENCIA**

La transparencia se define como la ocultación al usuario y al programador de aplicaciones de la separación de los componentes de un sistema distribuido, de manera que el sistema se percibe como un todo, en vez de una colección de componentes independientes. La transparencia ejerce una gran influencia en el diseño del software de sistema.

Las transparencias definidas son:

- Transparencia de Acceso: Permite el acceso a los objetos de información remotos de la misma forma que a los objetos de información locales.
- Transparencia de Localización: Permite el acceso a los objetos de información sin conocimiento de su localización.
- Transparencia de Concurrencia: Permite que varios procesos operen concurrentemente utilizando objetos de información compartidos y de forma que no exista interferencia entre ellos.
- Transparencia de Replicación: Permite utilizar múltiples instancias de los objetos de información para incrementar la fiabilidad y las prestaciones sin que los usuarios o los programas de aplicación tengan por que conoces la existencia de las replicas.
- Transparencia de Fallos: Permite a los usuarios y programas de aplicación completar sus tareas a pesar de la ocurrencia de fallos en el hardware o en el software.
- Transparencia de Migración: Permite el movimiento de objetos de información dentro de un sistema sin afectar a los usuarios o a los programas de aplicación.
- Transparencia de Prestaciones: Permite que el sistema sea reconfigurado para mejorar las prestaciones mientras la carga varia.

- Transparencia de Escalado: Permite la expansión del sistema y de las aplicaciones sin cambiar la estructura del sistema o los algoritmos de la aplicación.

Las dos más importantes son las transparencias de acceso y de localización, su presencia o ausencia afecta fuertemente a la utilización de los recursos distribuidos. A menudo se las denomina a ambas transparencias de red. La transparencia de red provee un grado similar de anonimato en los recursos al que se encuentra en los sistemas centralizados.

En el modelo Orientado a Objetos hay una serie de objetos que solicitan servicios (clientes) a los proveedores de los servicios (servidores) a través de una interfaz de encapsulación definida. Un cliente envía un mensaje a un objeto (servidor) y éste decide qué ejecutar, RMI y CORBA son algunos de esos sistemas basados en objetos.

La programación distribuida no es fácil:

- Hay múltiples modelos, dependiendo de los sistemas a los que accedemos.
- Debemos considerar seriamente muchos aspectos de seguridad.
- Se mezclan diversas tecnologías de varios vendedores.
- Dificultar para probar y depurar.

Llegados a este punto ya podemos darnos cuenta de una de las principales dificultades de la programación distribuida ¿cómo se comunican cliente y servidor?, ¿a través de que lenguaje, entorno o herramienta?, es aquí cuando debemos hacer referencia a Java, Java es una herramienta ideal para programación distribuida debido a su portabilidad, seguridad y amplio abanico de componentes. Desarrollando en Java podemos llegar a olvidarnos de dónde vamos a ejecutar nuestro programa,

¿será en una máquina UNIX o en un servidor Windows 2000 Server?, y es más, ¿lo podemos preparar para que se ejecute en varios entornos diferentes? Con Java todo esto es posible.

La cuestión básica en la programación de un sistema distribuido es cómo comunicarse con otras máquinas a través de la red. Resumiremos aquí diversas formas de atacar la programación de aplicaciones a través de la red.

- **SOCKETS**

Los sockets son puntos finales de enlaces de comunicaciones entre procesos. Los procesos los tratan como descriptores de ficheros, de forma que se pueden intercambiar datos con otros procesos recibiendo y transmitiendo a través de sockets, el tipo de socket describe la forma en la que se transfiere la información.

Sockets proveen al usuario de un interfaz para comunicarse a través de la red con otro/ sistema. Cada dirección de identificación para sockets consiste en una dirección de Internet y en un puerto. Se suelen utilizar los conocidos protocolos TCP/IP o UDP/IP, preferentemente el primero.

Distinguimos los siguientes tipos de sockets:

- **SOCKETS STREAM**

Son un servicio orientado a la conexión, donde los datos se transfieren sin encuadrarlos en registros o bloques.

- **DATAGRAM**

Son un servicio de transporte sin conexión, son más eficientes que TCP pero su utilización no es del todo fiable. Los datos se envían y reciben en paquetes cuya entrega no está garantizada.

- **SOCKETS RAW**

Son sockets que dan acceso a protocolos de más bajo nivel, no están soportados por Java.

- **SOCKET MULTICAST**

Es un nuevo tipo de socket muy particular e importante, es un DatagramSocket con una serie de capacidades adicionales para enlazar con grupos y hosts en Internet

Una de las ventajas principales de la programación con sockets es su sencillez.

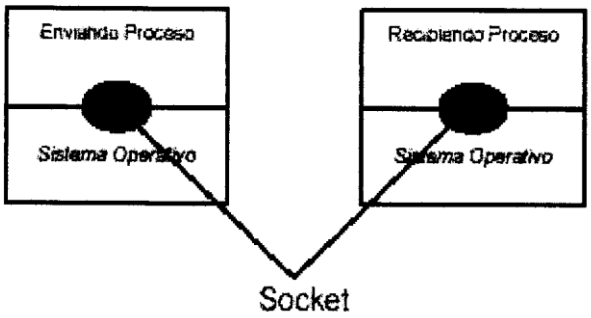

Figura 1.1 Sockets

A la hora de desarrollar una aplicación con sockets debemos tener en cuenta lo siguiente:

Es necesario tratar de manera especial los sockets para evitar problemas de seguridad, lo hacemos a través de la clase de Java SecurityManager y del paquete java security, podemos a través de estos recursos incluso generar mensajes codificados a través de algoritmos DSA.

Generalmente se conecta con servidores PROXY.

A la hora de conectar con Base de Datos, JDBC debe ser una opción a tener en cuenta, JDBC es un modelo de conexión a Base de Datos diseñado por JAVA, su utilidad radica en que es útil para acceso a bases de datos de diferentes tipos y vendedores, SQL Server, Oracle, etc.

Dentro de las diversas técnicas de programación destaca la serialización. La serialización de objetos es una técnica mediante la cual un programa puede salvar el estado de los objetos a un fichero y después recuperar los datos a la memoria y enviarlos a través de la red usando sockets de bajo nivel. Esta es una opción muy interesante que haría posible evitar lo que hoy en día se hace continuamente en proyectos comerciales, el recurrir a una base de datos para guardar el estado de un objeto, lo que repercute en el rendimiento final de la aplicación y sobre todo en su arquitectura. Java lo implementa con la interfazjava.io.Serializable.

Las interfaces remoto y home son tipos de interfaces "Java RMI Remote". La interfase java.rmi.Remote se usa con objetos distribuidos para representar el bean en un espacio de direccionamiento diferente (proceso o máquina). Un bean enterprise es un objeto distribuido, esto significa que la clase bean es ejemplarizada y vive en un contenedor pero puede ser accedida por aplicaciones que viven en otros espacios de direccionamiento.

Internamente, los EJBs pueden emplear tanto RMI como IIOP (protocolo definido por CORBA), aunque esto es totalmente transparente al usuario del objeto remoto, dado que ni siquiera se percatará de si el objeto se encuentra en su misma máquina o en otra a miles de kilómetros de distancia.

Los Objetos distribuidos amplían los sistemas orientados a objeto permitiendo que los objetos estén distribuidos a lo largo de una red heterogénea, de forma que estos objetos distribuidos trabajen como un todo unificado. Estos objetos pueden estar distribuidos en diferentes máquinas a través de la red, viviendo en su propio espacio de memoria fuera de la aplicación, aunque dando la apariencia de que están ubicados de forma local.

1.3. ELEMENTOS PRINCIPALES

- **INTERFACES REMOTAS**
    - Interfaz acordada entre servidor y cliente
    - Métodos que el cliente puede invocar
- **IMPLEMENTACIÓN DE LOS OBJETOS**
    - Implementación de los métodos de las interfaces
    - Objetos creados en el servidor
- **STUBS**
    - Actúan como referencias a objetos remotos desde el cliente
    - Retransmiten llamadas desde el cliente hacia el servidor
- **SISTEMA RUNTIME**

- Mediador entre stubs y objetos
- Acceso a los objetos (obtención de stubs)
- Servicio de nombres
- Métodos que envían objetos

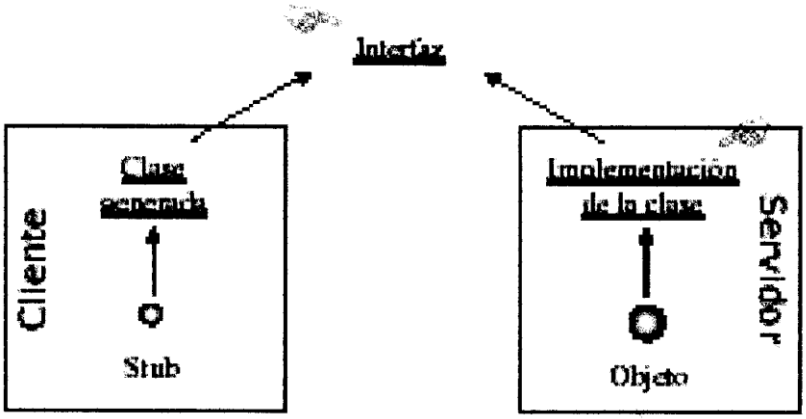

Figura 1.2 Modelo Objetos Distribuidos

Una vez compiladas las clases remotas, las clases de stubs se generan automáticamente mediante: rmic xxx.class®xxx-stub.class

1.4. COMUNICACIONES ENTRE OBJETOS DISTRIBUIDOS

1.4.1. INTERFACES

La mayoría de los lenguajes de programación modernos proporcionan medios para organizar un programa en conjuntos de módulos que puedan comunicarse unos con otros. La comunicación entre los módulos se puede realizar mediante llamadas a procedimientos entre los módulos o accediendo directamente a las variables de otro módulo. Para controlar las interacciones posibles entre los módulos, se define explícitamente una interfaz para cada módulo. Los módulos se implementan de

forma que se oculte toda la información excepto aquella que se haga disponible a través de su interfaz. De este modo, mientras la interfaz permanezca inalterada, la implementación podrá cambiar sin afectar a los usuarios del módulo.

1.4.2. LAS INTERFACES EN LOS SISTEMAS DISTRIBUIDOS

En un programa distribuido, los módulos pueden lanzarse en procesos separados. No es posible para un módulo que se ejecuta en un proceso acceder a las variables de un módulo que esta en otro proceso. Asimismo, la interfaz de un módulo escrita para RPC o RMI no puede especificar el acceso directo a variables. (Las interfaces en IDL o CORBA pueden especificar atributos, lo que parece violar esta regla. Sin embargo, los atributos no son accedidos directamente sino mediante ciertos procedimientos de escritura y lectura que se añaden automáticamente a la interfaz).

Los mecanismos de paso de parámetros, por ejemplo la llamada por valor y la llamada por referencia, utilizados en las llamadas a procedimientos locales, no son adecuados cuando el que llama y el procedimiento llamado están en procesos diferentes. La especificación de un método o procedimiento en la interfaz de un módulo en un programa distribuido describe los parámetros como entrada o salida o, a veces, ambos. Los parámetros de entrada se pasan al módulo remoto mediante el envío de los valores de los argumentos en el mensaje de petición y posteriormente se proporcionan como argumentos a la operación que se ejecutará en el servidor.

Los parámetros de salida se devuelven en el mensaje de respuesta y se sitúan como la respuesta de la llamada o reemplazando los valores de las correspondientes variables argumento en el entorno de las llamada. Cuando se proporciona un parámetro tanto como para entrada como para salida, el valor debe trasmitirse tanto en los mensajes de la petición como en los de la respuesta.

Otra diferencia entre los módulos locales y remotos es que los punteros en un proceso dejan de ser válidos en el remoto. En consecuencia, no pueden pasarse punteros como argumentos o como valores retornados como resultado de las llamadas a los módulos remotos.

1.4.3. LAS INTERFACES EN LOS SISTEMAS DISTRIBUIDOS

En el modelo cliente-servidor, cada servidor proporciona un conjunto de procedimientos disponibles para su empleo por los clientes. Por ejemplo, un servidor de archivos proporcionará procedimientos para leer y escribir archivos. El término interfaz de servicio se emplea para referirse a la especificación de los procedimientos que ofrece un servidor, y define los tipos de los argumentos de entrada y salida de cada uno de los procedimientos.

1.4.4. INTERFACES REMOTAS

En el modelo de objetos distribuidos, una interfaz remota especifica los métodos de un objeto que están disponibles para su invocación por objetos de otros procesos, y define los tipos de los argumentos de entrada y de salida de cada uno de ellos. Sin embargo, la gran diferencia es que los métodos en las interfaces remotas pueden pasar objetos como argumentos y como resultados de los métodos. Además, también se pueden pasar referencias a objetos remotos, lo cual no debe confundirse con los punteros, que se refieren a ubicaciones específicas de la memoria.

Ni las interfaces de servicio ni las interfaces remotas pueden especificar un acceso directo a variables. En este último, está prohibido el acceso directo a las variables de las instancias de un objeto.

1.4.5. LENGUAJES DE DEFINICIÓN DE INTERFACES.

Se puede integrar un mecanismo RMI con un leguaje de programación concreto si incluye una notación apropiada para definir interfaces, que permita relacionar los parámetros de entrada y de salida con el uso habitual de los parámetros en ese lenguaje. Java RMI es un ejemplo en el que se ha añadido un mecanismo RMI a un lenguaje de programación con orientación a objetos. Esta aproximación es útil cuando se puede escribir cada parte de una aplicación distribuida en el mismo lenguaje. También es conveniente dado que permite al programador emplear un solo lenguaje para las invocaciones locales y remotas.

Sin embargo muchos servicios útiles se encuentran escritos en C++ y otros lenguajes. Seria beneficioso, que pudieran acceder a ellos remotamente, muchos otros programas escritos en gran variedad de lenguajes incluyendo java. Los lenguajes de definición de interfaces (IDL) están diseñados para permitir que los objetos implementados en lenguajes diferentes se invoquen unos a otros. Un IDL proporciona una notación para definir interfaces en la cual cada uno de los parámetros de un método se podrá describir como de entrada o de salida además de su propia especificación de tipo.

La comunicación de procesos es fundamental en cualquier sistema distribuido existen diferentes posibilidades todas ellas basadas en el paso de mensajes.

- Mecanismos de bajo nivel: el programador debe preocuparse de establecer los protocolos de comunicación, representación de datos, etc.

- Mecanismo de alto nivel: *O*frecen abstracciones donde el programador no debe preocuparse de establecer protocolos.

La comunicación entre objetos de un mismo proceso es varios órdenes de magnitud más rápida que la comunicación entre objetos en diferentes máquinas. De

esto se desprende que usted no debería diseñar aplicaciones distribuidas en las cuales dos o más objetos distribuidos estén muy íntimamente relacionados. Si hay esa estrecha relación, dichos objetos deberían situarse en un mismo lugar o proceso.

Cuando dos objetos están localizados conjuntamente, pueden fallar a la vez. Si el proceso que los ejecuta falla, ambos objetos fallarán. El diseñador debe tener en cuenta esta característica: sí un objeto falla, puede saber que el otro objeto también fallará si se encuentran bajo un mismo proceso, sin embargo, si el sistema está distribuido habría que tener en cuenta que si un objeto falla, otro objeto puede que no falle. Esto nos da una tolerancia a fallos si tenemos los objetos duplicados por la red. En caso de que la red falle en un punto, los objetos que estén a ambos lados de ese punto de ruptura pueden seguir ejecutándose independientemente asumiendo que los del otro lado han fallado.

1.5. PATRONES TÍPICOS DE COMUNICACIÓN

1.5.1. COMUNICACIÓN CLIENTE - SERVIDOR

Muy utilizada en entornos distribuidos (más del 90% de los sistemas distribuidos utilizan la arquitectura cliente - servidor)

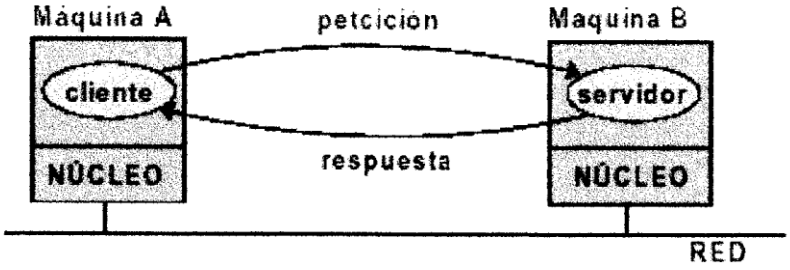

Figura 1.3 Protocolo típico: petición - respuesta

El modelo cliente - servidor de un sistema distribuido es el modelo más conocido y más ampliamente adoptado en la actualidad. Hay un conjunto de procesos

servidores, cada uno actuando como un gestor de recursos para una colección de recursos de un tipo, y una colección de procesos clientes, cada uno llevando a cabo una tarea que requiere acceso a algunos recursos hardware y software compartidos. Los gestores de recursos a su vez podrían necesitar acceder a recursos compartidos manejados por otros procesos, así que algunos procesos son ambos clientes y servidores. En el modelo, cliente - servidor, todos los recursos compartidos son mantenidos y manejados por los procesos servidores. Los procesos clientes realizan peticiones a los servidores cuando necesitan acceder a algún recurso. Si la petición es valida, entonces el servidor lleva a cabo la acción requerida y envía una respuesta al proceso cliente.

El termino proceso se usa aquí en el sentido clásico de los sistemas operativos. Un proceso es un programa en ejecución. Consiste en un entorno de ejecución con al menos un thread de control.

El modelo cliente - servidor nos da un enfoque efectivo y de propósito general para la compartición de información y de recursos en los sistemas distribuidos. El modelo puede ser implementado en una gran variedad de entornos software y hardware. Las computadoras que ejecuten los programas clientes y servidores pueden ser de muchos tipos y no existe la necesidad de distinguir entre ellas, los procesos cliente y servidor pueden incluso residir en la misma maquina.

En esta visión simple del modelo cliente - servidor, cada proceso servidor podría ser visto como un proveedor centralizado de los recursos que maneja. La provisión de recursos centralizada no es deseable en los sistemas distribuidos. Es por esta razón por lo que se hace una distinción entre los servicios proporcionados a los clientes y los servidores encargados de proveer dichos servicios. Se considera un servicio como una entidad abstracta que puede ser provista por varios procesos servidores ejecutándose en computadoras separadas y cooperando vía red.

El modelo cliente - servidor se ha extendido y utilizado en los sistemas actuales con servicios manejando muchos diferentes tipos de recursos compartidos - correo electrónico y mensajes de noticias, ficheros, sincronización de relojes, almacenamiento en disco, impresoras, comunicaciones de área extensa, e incluso las interfaces gráficas de usuario. Pero no es posible que todos los recursos que existen en un sistema distribuido sean manejados y compartidos de esta manera, algunos tipos de recursos deben permanecer locales a cada computadora de cara a una mayor eficiencia - RAM, procesador, interfaz de red local. Estos recursos clave son manejados separadamente por un sistema operativo en cada máquina, solo podrían ser compartidos entre procesos localizados en el mismo ordenador.

Aunque el modelo cliente - servidor no satisface todos los requisitos necesarios para todas las aplicaciones distribuidos, es adecuado para muchas de las aplicaciones actuales y provee una base efectiva para los sistemas operativos distribuidos de propósito general.

1.5.2. JAVA Y SU ESTILO CLIENTE/SERVIDOR

Java está revolucionando el desarrollo y la operación de las aplicaciones de red para Internet. Java es un lenguaje de programación orientado a los objetos que se dice es un subconjunto de C++ y que ha eliminado las características confusas y poco entendibles de C++. Las características omitidas consisten principalmente de la sobrecarga de operadores (aunque Java provee un método para sobrecargas), herencia múltiple y manejo de punteros. Ha agregado liberación automática de memoria (automatic garbage collection), simplificando de esta forma la programación. Las características orientadas a los objetos de Java son esencialmente las mimas de C++, con una resolución de métodos más dinámica. Java provee librerías con gran cantidad de rutinas para acoplarse a los protocolos de Internet como TCP/IP, HTTP, y FTP. Esto hace que construir aplicaciones de red sea más sencillo que en C y en C++. Las aplicaciones de Java pueden abrir y acceder

objetos en la red usando un URL (Uniform Resource Locator) con la misma facilidad con que se acceden a los sistemas de ficheros locales.

Java ha sido pensado para escribir programas que deben ser muy confiables desde muchos puntos de vista. Java hace énfasis en chequeos previos a la ejecución para evitar posibles problemas, chequeos dinámicos en tiempo de ejecución, y agregación de limitaciones para evitar situaciones malintencionadas. Java esta diseñado para el uso en ambientes distribuidos y las redes. Debido a esto es el gran énfasis dado a la seguridad. Java permite la construcción de sistemas libres de virus y muy estables. Las técnicas de autenticación se basan en claves públicas. Hay una fuerte relación entre seguridad y robustez.

Java fue diseñado para soportar aplicaciones sobre la red. En general, las redes están compuestas de una variedad de sistemas, con una variedad de arquitecturas de CPU y sistemas operativos. Para permitirle a una aplicación de Java ejecutarse en cualquiera parte de la red, el compilador genera un archivo objeto con un formato neutral a cualquier arquitectura - el código compilado es ejecutable en muchos procesadores si está presente el sistema de ejecución de Java llamado VM (Virtual Machine). Esto es útil no solo para las redes, sino además para la distribución de software en sistemas de una sola máquina. En el mercado actual de los PC, los programadores deben escribir diferentes versiones de sus aplicaciones para hacerlas compatibles con el PC y las Macintosh. Con Java, la misma versión de la aplicación funciona en todas las plataformas. El compilador de Java lo permite generando instrucciones llamadas byte - code que no están relacionadas con ninguna arquitectura particular de computadoras. En su lugar, están diseñadas para ser fácilmente interpretadas por cualquier máquina, realizando una traducción al código nativo de máquina en tiempo de ejecución.

El lenguaje Java incluye ambientes de compilación y ejecución. El usuario escribe los programas de Java en un archivo java, el cual es compilado para generar

el byte - code el cual es depositado en un archivo cuya extensión es class. Es este byte - code la fuente de alimentación de la máquina virtual de Java (VM) que se presenta como un archivo ejecutable que recibe en su línea de parámetros un archivo class.

Java es utilizado para la construcción de aplicaciones que hubiesen podido ser implementadas en otros lenguajes de igual manera. Sin embargo, es su arquitectura neutra que provee la movilidad de código, lo que ha llevado a Java a ingresar con tanta fuerza en Internet. Ha surgido la aparición de un nuevo estilo cliente servidor en Internet basado en objetos móviles - llamados applets en Java que son trozos autosuficientes de código ejecutable.

1.5.3. COMUNICACIÓN DE GRUPOS

Utiliza mensajes multicast útil para ofrecer tolerancia a fallos basado en servicios replicados.

Localizar objetos en sistemas distribuidos.

Mejor rendimiento mediante datos replicados.

Actualizaciones múltiples

Operaciones colectivas en cálculo paralelo.

1.6. CARACTERISTICAS

- Permite al diseñador del sistema retrasar las decisiones de donde y como se deben proveer los servicios.

- Es una arquitectura muy abierta de sistema que permite que nuevos recursos sean agregados cuando son requeridos.

- El sistema es flexible y escalable.

- Es posible reconfigurar el sistema dinámicamente con objetos migrando a través la red como sea requerido.

- Uno de los principales beneficios de la pasarela dinámica es que las aplicaciones no necesitan código específico para poder manejar diferentes tipos de datos definidos por el usuario. Esto significa que no se necesita recompilar cuando se producen cambios en la aplicación. Por lo tanto nos ahorramos tiempo y dinero en mantener y desarrollar aplicaciones. Se realizan los cambios en tiempo de ejecución.

1.7. VENTAJAS DE LOS OBJETOS DISTRIBUIDOS

La principal ventaja de los sistemas cliente — servidor está en la correspondencia natural de las aplicaciones en el marco cliente - servidor. Un ejemplo de esto es una agenda electrónica. Debido a que los datos son relativamente estáticos y son vistos de manera uniforme por todos los usuarios del sistema parece lógico colocarlos en un servidor que acepte peticiones sobre dichos datos. Es más, en este caso la lógica de aplicación debería estar colocada del lado del servidor, para proporcionar una mayor flexibilidad al sistema de búsquedas (cambios en los algoritmos, etc...).

Como resultado de la disponibilidad de middleware compatible para múltiples plataformas y de los avances recientes de la interoperabilidad binaria, los sistemas cliente - servidor pueden conectar clientes ejecutándose en una plataforma con servidores ejecutándose en otra plataforma completamente distinta. Las tecnologías

como Java y los ORBs (Object Request Brokers). Si las porciones de un sistema cliente - servidor encapsulan una única función y siguen un interfaz perfectamente definido, aquellas partes del sistema que proveen los servicios pueden ser intercambiadas sin afectar a otras porciones del sistema. Esto permite a los usuarios, desarrolladores y administradores adecuar el sistema con sus necesidades en cada momento.

Otra ventaja es la posibilidad de ejecutar aplicaciones que hacen uso intensivo de los recursos en plataformas hardware de bajo coste. También el sistema es más escalable, pudiéndose añadir tanto nuevo clientes como nuevos servidores.

CAPÍTULO II

INTRODUCCION A RMI

2.1. INTRODUCCIÓN

RMI o Remote Method Invocation (Invocación de Métodos Remotos) surge con la idea de poder comunicar aplicaciones de forma remota y de esta manera hacer que las aplicaciones trabajen cooperando unas con otras con la finalidad de tener un sistema distribuido potente, escalable y seguro. No obstante, no hay que pensar que RMI es la única forma de comunicar aplicaciones dentro de una red, existen otros métodos de hacerlo como es CORBA o Java IDL (o incluso DCOM si nos gusta más el mundo Microsoft) sin embargo RMI forma parte del JDK y no tiene asociado ningún coste por lo que para nosotros resulta muy interesante, además como ocurre con todo (o casi) de lo que forma parte del mundo Java el que sea gratuito no significa que sea menos potente.

La idea que subyace debajo de RMI es sencilla, lo que se pretende es llamar o ejecutar (realmente invocar) métodos de objetos remotos como si estuvieran ejecutándose en local de modo que resultan totalmente transparente las llamadas que se efectúan a los objetos, en RMI la principal característica que tenemos es que la invocación al objeto remoto se realiza sin saber en que servidor reside, para ello se hace uso de una técnica que se conoce como resolución de nombre (o naming) que posibilita invocar al objeto por su nombre y no por la máquina, en la que se encuentre, es importante indicar que gracias a esto no es necesario que los objetos que van a ser invocados estén disponibles en tiempo de compilación bastándonos con que existan en tiempo de ejecución sino queremos tener un error.

Mediante el mismo mecanismo de invocación remota, es decir, con un stub de cliente y otro de servidor, las tareas de localización del servicio remoto, empaquetamiento, encriptación e invocación son realizadas por los stubs, de forma que la clase invocantes se limita a invocar un método y recibir el resultado. De esta forma, se permite crear aplicaciones java distribuidas, que repartan la carga de procesamiento entre los distintos elementos físicos que forman el sistema distribuido. Sobre RMI si soportan las invocaciones remotas que permiten a los EJBs ejecutar métodos de objetos remotos.

Java Remote Method Invocation (RMI) Permite definir e implementar interfaces remotos en el lenguaje Java.

- Sus operaciones se pueden invocar remotamente, de la misma forma que se invocan las operaciones de interfaces locales.

- Solución de más alto nivel que el uso directo de sockets o RPCs.

- Facilidad de desarrollo.

- Apropiado para construir sistemas cliente/servidor en una Intranet, especialmente si cliente y servidor están escritos en Java

2.2. CARACTERISTICAS DE RMI

Entre las principales características de RMI podemos encontrar:

- Sencillez.

- Transparencia.

- Paso de objetos por valor (como parámetros de los métodos).

- Implementación 100% JAVA.

- Independencia del protocolo de comunicación.

En RMI, cuando los objetos remotos ya han sido referenciados, el programador los puede utilizar igual que si estuviesen en la maquina local, accediendo a sus métodos y manejándolos de forma normal. La capa RMI permanece oculta al programador proporcionando mayor claridad al código y mayor comodidad a la hora de programar la aplicación.

El paso de objetos por valor se revela como una de las características que diferencian a RMI de las otras tecnologías de programación distribuida. Esta potente posibilidad se basa en el uso del concepto de “Serialización” para el paso de objetos a través de la red. Objetos de todo tipo, tanto nativos JAVA como definidos por el usuario, puedes ser transferidos de forma totalmente transparente al programador usando esta técnica.

Por otro lado, la principal ventaja de RMI, su implementación 100% JAVA, es también su principal inconveniente. RMI no puede ser utilizado por otros lenguajes de programación, por lo que no es posible utilizarlo en aplicaciones distribuidas donde se utilizan distintos lenguajes. Por lo tanto, en la actualidad, las soluciones basadas en RMI se reducen a aquellas aplicaciones que se puedan implementar completamente en JAVA, cosa que habitualmente no suele ocurrir en el mundo real.

2.3. GESTIÓN INTERNA DE LA COMUNICACIÓN ENTRE PROCESOS

RMI se ocupa de:

- Abrir, controlar, cerrar conexiones entre cliente y servidor.
- Empaquetamiento y desempaquetamiento de datos.
- Preparar al servidor para que permanezca a la espera de llamadas a métodos desde los clientes mediante sockets.
- Atender a las llamadas transfiriéndolas a los objetos correspondientes

En definitiva, RMI permite crear e instanciar objetos en máquinas locales y al mismo tiempo crearlos en otras máquinas (máquinas remotas), de forma que la comunicación se produce como si todo estuviese en local. RMI se convierte así en una alternativa muy viable a los sockets.

2.4. SERVICIOS RMI

Como se vio anteriormente, un servicio RMI permite la invocación de métódos de objetos remotos, situados en espacios de memoria distintos y, probablemente, máquinas también distintas. Es posible entonces emplear esta solución tecnológica para unir dos capas adyacentes de una aplicación web. Para independizar a la capa superior del middleware empleado para unirla con la inferior, se encapsula toda la lógica dependiente de dicho middleware dentro de una clase, de forma que futuras sustituciones en el método de unión entre capas se limiten a la modificación de una

clase. Así, el helper se ocuparía de las tareas de configuración, localización e invocación del servicio.

Con la capa inferior, tendremos la implementación de los servicios RMI, que se limitarían a invocar a las clases de la capa que implementen los servicios solicitados desde la capa superior.

Esta solución nos permite distribuir la aplicación entre distintas máquinas, pudiendo desgranar cada capa en servicios, y colocar un servicio en cada máquina disponible.

2.5. ARQUITECTURA DE RMI

En RMI toda la información sobre los servicios del servidor se dan en una definición de interface remota.

Si comparamos RMI con sockets percibiremos que RMI es más simple. Además, esta tecnología permite despreocupamos del protocolo, en sockets depende de la aplicación, si ésta es grande puede que nos tengamos que preocupar del protocolo.

Cada capa es independiente de las otras y tiene definido su propio interfaz y protocolo, de forma que una capa puede ser cambiada sin afectar a las otras. Por ejemplo, la capa de transporte puede utilizar distintos protocolos como TCP, UDP...etc sin que el resto de las capas se vean afectadas en su funcionamiento.

La arquitectura de RMI es una arquitectura en capas cada una de las cuales presenta una serie de particularidades que vemos a continuación:

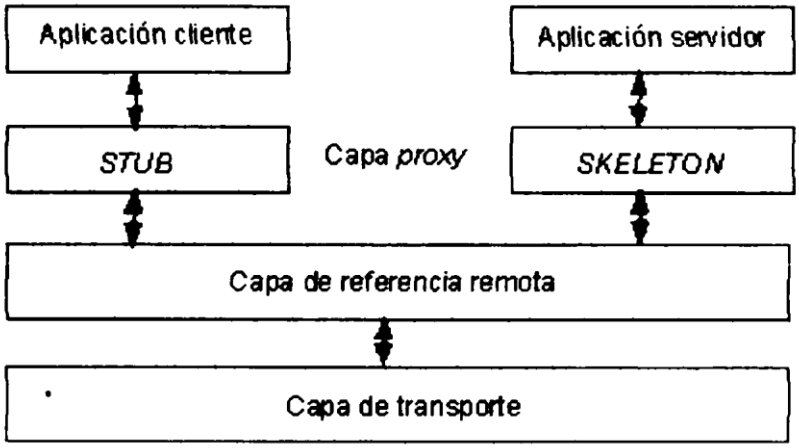

Figura 2.1 Arquitectura RMI

2.5.1. CAPA DE APLICACION

Corresponde con la implementación real de las aplicaciones cliente y servidor. Aquí tienen lugar las llamadas a alto nivel para acceder y exportar objetos remotos. Cualquier aplicación que quiera que sus métodos estén disponibles para su acceso por clientes remotos debe declarar dichos métodos en una interfaz que extienda java.rmi.Remote. Dicha interfaz se usa básicamente para "marcar" un objeto como remotamente accesible. Una vez que los métodos han sido implementados, el objeto debe ser exportado. Esto puede hacerse de forma implícita si el objeto extiende la clase UnicastRemoteObject (paquete java.rmi.server), o puede hacerse de forma explícita con una llamada al método exportObject() del mismo paquete.

2.5.2. CAPA PROXY

Capa stub-skeleton, la capa de enlace del cliente se le conoce como Stub y a la del servidor como Skeleton. Esta capa es la que interactúa directamente con la capa de aplicación. Todas las llamadas a objetos remotos y acciones junto con sus parámetros y retorno de objetos tienen lugar en esta capa.

2.5.3. CAPA DE REFERENCIA REMOTA

Es responsable del manejo de la parte semántica de las invocaciones remotas. También es responsable de la gestión de la replicación de objetos y realización de tareas específicas de la implementación con los objetos remotos, como el establecimiento de las persistencias semánticas y estrategias adecuadas para la recuperación de conexiones perdidas. En esta capa se espera una conexión de tipo stream (stream-oriented connection) desde la capa de transporte y de crear independencia de los protocolos.

2.5.4. CAPA DE TRANSPORTE

Es la responsable de mantener la conexión entre el cliente y el servidor y por tanto entre las máquinas virtuales que cada uno utilice. La tabla con los diferentes objetos remotos disponibles esta situada en esta capa y siempre que se crea un nuevo objeto remoto se añade una entrada en la misma para que puedan crearse objetos de esta clase desde la capa de referencia. El protocolo de transporte subyacente para RMI es JRMP (Java Remote Method Protocol), que solamente es "comprendido" por programas Java.

2.6. STUBS Y SKELETONS

2.6.1. STUB

Clase usada por el cliente en sustitución de la remota Implementa el interfaz remoto.

La implementación de cada operación envía un mensaje a la máquina virtual que ejecuta el objeto remoto y recibe el resultado.

Los parámetros y el valor de retorno se envían serializados.

El Stub del objeto remoto es el proxy del cliente para el objeto remoto. Es responsable de:

- Inicializar la llamada al objeto remoto.
- Hacer el marshaling de argumentos a un marshal stream, obtenido de la capa de referencia remota.
- Informa a la capa de referencia remota que debe invocarse la llamada.
- Hacer unmarshaling al valor de retorno o excepción.
- Informar al nivel de referencia remota que la llamada está completa.

2.6.2. SKELETON

Un skeleton es una clase auxiliar generada por RMI, se entiende cómo comunicarse con el stub a través del enlace RMI. Y mantiene una conversación con el stub, lee los parámetros de la llamada, efectúa la invocación al servicio remoto (to the remote service implementation object), acepta el valor de retorno y lo retorna al stub. En la implementación de RMI de Java 2 SDK la clase skeleton está obsoleta debido al protocolo new wire.

Es una clase usada por el servidor que recibe los mensajes, invoca la operación del objeto que implementa el interfaz remoto y envía el resultado al llamador.

Es responsable de:

- Hacer unmarshaling a los argumentos desde el marshal stream.

- Hacer la llamada a la implementación del objeto remoto.

- Hacer marshaling al valor de retorno de la llamada o una excepción.

En términos de patrones, un skeleton es un Adapter

Los Stubs y Skeletons son transparentes al código, cuando un cliente invoca una operación remota que devuelve una referencia a un objeto remoto, obtiene una instancia del stub correspondiente.

El Stub y el Skeleton se generan con el compilador rmic

2.7. SERVICIOS DE NOMBRES

Permite asociar nombres simples a objetos del lado del cliente, el RMI Registry se accede a través de la clase estática Naming. Esta provee el método lookup() que acepta un URL donde se especifica el nombre del host del servidor y el nombre del servicio. El método retorna una referencia remota al objeto que ofrece el servicio.

RMI puede usar diferentes tipos de servicios de directorio, incluyendo JNDI (the Java Naming and Directory Interface), incluyendo un servicio muy simple llamado: RMI Registry, rmiregistry este corre en cada maquina que posee objetos de servicio remoto y acepta preguntas sobre el servicio.

En un host, un servidor crea un servicio remoto, construyendo primero un objeto local que implementa el servicio, luego se exporta el objeto a RMI. Cuando el objeto es exportado RMI crea un servicio que espera que los clientes se conecten para solicitar

el servicio (listening service). Después de exportar el objeto, el servidor lo registra en el RMI registry con un nombre público.

Un registro existe en cada nodo que permita conexiones del RMI a los servidores en ese nodo. El registro en un nodo particular contiene una base de datos transitoria. Cuando los cargadores del nodo, la base de datos del registro son vacíos. Los nombres almacenados en el registro son puros y no se analizan. Un servicio que se almacena en el registro puede desear prefijar su nombre del servicio por un nombre del paquete (aunque no está requerido), para reducir las colisiones conocidas en el registro.

En el entorno Java RMI el servicio de nombres básico conoce como Registry. Este servicio mantiene tuplas del tipo (nombre del servicio, stub del objeto remoto), donde nombre del servicio es una clave a través de la cual se determina un único stub de un objeto remoto. Para manipular estas tuplas Registry ofrece las operaciones: bind, unbind, rebind, lookup y list.

Es decir, este servicio permite que los servidores den de alta (bind o rebind) o baja (unbind) servicios y que los clientes, que pregunten por un determinado servicio (lookup), obtengan el identificador de comunicación de éste para poder solicitar posteriormente operaciones sobre ese servicio.

Por otra parte, el servicio de nombres Registry es también un servicio, por lo que tiene un identificador de comunicación que debe ser conocido de antemano por sus clientes para poder acceder a él.

En general, hay varias alternativas, dependiendo de si el servicio de nombres puede cambiar o no de ubicación, en tiempo de ejecución, sin afectar al funcionamiento de los clientes (reubicación dinámica o estática).

En Java RMI se usa la reubicación estática del servicio de nombres, ya que Registry se ejecuta siempre en una determinada máquina y puerto fijo y su reubicación supondría tener que modificar ciertas partes del sistema RMI. Finalmente, si la ubicación del servicio de nombres Registry debe ser de dominio publico ¿Qué método se utiliza en Java para hacer publica dicha ubicación?, para hacer que dicha información sea de dominio publico (well known) se ofrece una clase LocateRegistry que contiene los métodos necesarios para obtener el identificador de un servicio de nombres, así como para crear un servicio de nombres en una máquina y puerto concretos.

Por tanto, un cliente o servidor realizarán siempre los dos pasos siguientes para localizar un servicio o darlo de alta:

- Obtener de la clase LocateRegistry el identificador de comunicación del servicio de nombres Registry de la máquina local o remota.

- Acceder a la información que guarda el servicio de nombres Registry para dar de alta (bind o rebind) un servicio, para obtener su identificador de comunicación (lookup) o para consultar los servicios que tiene dados de alta (list).

Por ejemplo, sea un cliente que quiere obtener el identificador de comunicación de un servicio Calculador que ofrece operaciones matemáticas. Sabe que dicho servicio está dado de alta en el servicio de nombres de la máquina 2, pero no conoce el identificador de Registry de dicha máquina, para poder solicitarle el identificador de Calculador.

Veamos gráficamente los pasos que sigue el cliente para obtener dicho identificador:

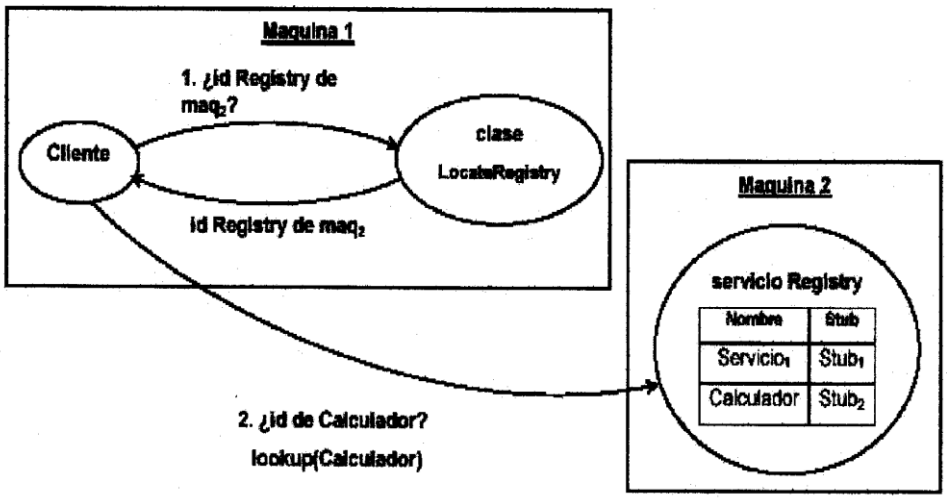

Figura 2.2 Llamada de un objeto RMI

En el paso 1,el cliente le pide a la clase LocateRegistry local el identificador del servicio de nombres Registry de la maquina 2, una vez obtenido, en el paso 2 ya puede preguntar al Registry remoto el nombre del servicio Calculador con el que desea contactar.

En general, para cualquier operación que quieran hacer los clientes o servidores sobre un servicio de nombres, siempre es necesario que estos suministren la siguiente terna de información:

rmi://<host_name>[:<name_service_port>]/<service_name>

El name_service_port sólo debe especificarse si el servicio deseado corre en un puerto diferente al 1099.

Donde los dos primeros elementos hacen referencia a la máquina y al puerto del servicio de nombres Registry a consultar y el último, al servicio que mantiene el servicio de nombres.

2.8. PASO DE PARAMETROS

Hay 2 Mecanismos básicos:

2.8.1. TIPOS PRIMITIVOS

Se pasan por valor. Todos son serializables, es la semántica normal para el pase de parámetros en Java, los valores obtenidos de la ejecución de los métodos también son copiados.

2.8.2. OBJETOS REMOTOS

RMI emplea serialización para transformar los objetos a un formato que pueda ser enviado por cables de red. Los objetos remotos no viajan, en cambio se envían referencias (que permiten crear talones o sustitutos) a quienes los reciben, ya sea como argumentos de un método o como resultados de un método. Un talón se debe convertir al tipo (cast) de las interfaces remotas que implemente la clase del objeto remoto al que corresponde. El tipo del talón determina que métodos remotos se pueden invocar en el objeto remoto (aquellos que se declaran en la interfaz correspondiente), si un objeto remoto implementa varias interfaces remotas un cliente solo puede convertir el talón a una de ellas (y por lo tanto solo podrá invocar los métodos declarados en esa interfaz).

2.9. SERIALIZACION DE OBJETOS

Serialización de objetos consiste en convertir un objeto en un stream de datos que puede enviarse a cualquier lugar. Para reconstituir el objeto se desserealiza el stream de datos y se convierte en un objeto. Para indicar que un objeto es serealizable, el objeto debe implementar la interface lava.lang.Serealizable, no se

necesita añadir nueva funcionalidad la funcionalidad de serialización es genérico y está contenida en las clases.

Manipulación de objetos por valor en el programa de llegada para que tenga sentido enviar objetos por valor, el programa al que llegan debe poder manipularlos, e invocar sus métodos, esto significa que el programa receptor incluye (de antemano) código en que se invocan métodos de los objetos pasados por valor.

El receptor puede tener el código que implementa los métodos o bien cargar la clase dinámicamente (la obtiene del host de salida), si el receptor no tiene las clases es necesario definir una interfaz para estos objetos, compartida por ambos programas, emisor y receptor. Así un cliente puede transferir al servidor la ejecución de programas arbitrarios definidos en el cliente siempre que se ajusten a una interfaz.

2.10. CREAR UN SERVIDOR RMI

Un servidor RMI consiste en definir un objeto remoto que va a ser utilizado por los clientes. Para crear un objeto remoto, se define una interfaz, y el objeto remoto será una clase que implemente dicha interfaz. Veamos como crear un servidor de ejemplo mediante 3 pasos:

2.10.1. DEFINIR INTERFAZ REMOTO

Cuando se crea un interfaz remoto:

El interfaz debe ser público.

Debe extender (heredar de) el interfaz java.rmi.Remote, para indicar que puede llamarse desde cualquier máquina virtual Java.

Cada método remoto debe lanzar la excepción Java.rmi.RemoteException en su cláusula throws, además de las excepciones que pueda manejar.

Veamos un ejemplo de interfaz remoto:

```
public interface MiInterfazRemoto extends java.rmi.Remote
public void miMetodo1() throws java.rmi.RemoteExcention;
public int miMetodo2() throws java.rmi.RemoteException;
```

2.10.2. IMPLEMENTAR EL INTERFAZ REMOTO

```
public class MiClaseRemota
extends java.rmi.server.UnicastRemoteObject
implements MiInterfazRemoto
{       public MiClaseRemotao throws
Java.rmi.RemoteException
{
  //Código del constructor
}
                        public void miMetodol1() throws
                        java.rmi.RemoteException
{
                          //Aquí ponemos el código que queramos
System.out.print1n("Estoy en miMetodol1()");
}
public int miMetodo2() throws
java.rmi.RemoteException
{
return 5; // Aquí ponemos el código
}
```

```
public void otroMetodo()
{
//Si definimos otro método, éste no podría
//llamarse remotamente al no ser del //interfaz remoto
}
public static void main(String[] args)
{
  Try
   {
MiInterfazRemoto mir = new MiClaseRemota(); java.rmi.Naming.rebind("//" +
java.net.InetAddress.getLocalHost().getHostAddres() + ":" + args[0] + "/PruebaRMI",
mir);
}
catch (Exception e){}
}
```

Como se puede observar, la clase MiClaseRemota implementa el interfaz MiInterfazRemoto que hemos definido previamente. Además, hereda de UnícastRemoteObject,que es una clase de Java que podemos utilizar como superclase para implementar objetos remotos.

Luego, dentro de la clase, definimos un constructor (que lanza la excepción RemoteException porque también la lanza la superclase UnicastRemoteObject), y los métodos de la interfaz que implemente.

Finalmente, en el método main, definimos el código para crear el objeto remoto que se quiere compartir y hacer el objeto remoto visible para los clientes, mediante la clase Naming y su método rebind(...).

2.10.3. COMPILAR Y EJECUTAR EL SERVIDOR

Ya tenemos definido el servidor. Ahora tenemos que compilar sus clases mediante los siguientes pasos:

Compilamos el interfaz remoto. Además lo agrupamos en un fichero JAR para tenerlo presente tanto en el cliente como en el servidor:

javac MiInterfazRemoto.java

jar cvf objRemotos.jar MiInterfazRemoto.class

Luego, compilamos las clases que implementen los interfaces. Y para cada una de ellas generamos los ficheros Stub y Skeleton para mantener la referencia con el objeto remoto, mediante el comando rmic:

set CLASSPATH=%CLASSPATH%;.\objRemotos.jar;. javac MiClaseRemota.java
rmic -d . MiClaseRemota

Observamos en nuestro directorio de trabajo que se han generado automáticamente dos ficheros .class (MiClaseRemota-Skel.class y MiClaseRemota-Stub.class) correspondientes a la capa stub-skeleton de la arquitectura RMI.

Para ejecutar el servidor, seguimos los siguientes pasos:

Se arranca el registro de RMI para permitir registrar y buscar objetos remotos. El registro se encarga de gestionar un conjunto de objetos remotos a compartir, y buscarlos ante las peticiones de los clientes. Se ejecuta con la aplicación rmiregistry distribuida con Java, a la que podemos pasarle opcionalmente el puerto por el que conectar (por defecto, el 1099):

start rmiregistry 1234

Figura 2.3 Monitoreo del nombre de registros

Por ultimo, se lanza el servidor:

java -Djava.rmi.server.hostname=127.0.0.1 MiClaseRemota 1234

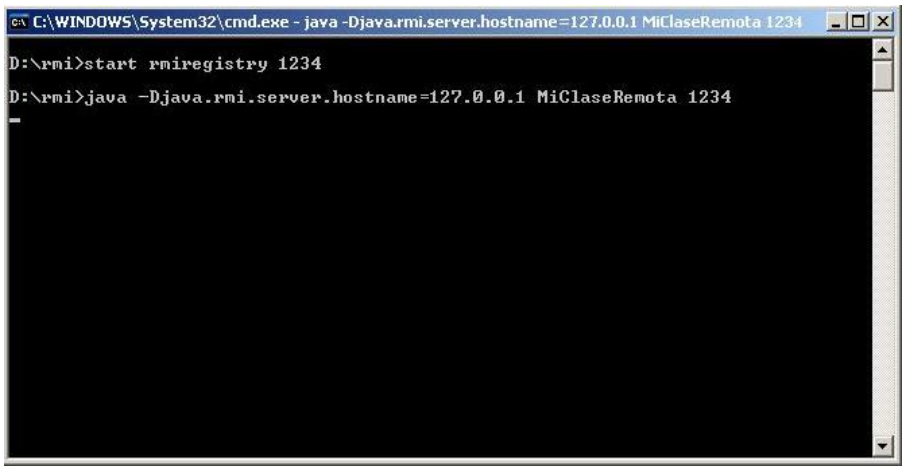

Figura 2.4 Ejecución del Servidor RMI

2.11. CREAR UN CLIENTE RMI

Vamos ahora a definir un cliente que accederá al objeto remoto que creemos. Para ello seguimos los siguientes pasos:

2.11.1. DEFINIR LA CLASE PARA OBTENER LOS OBJETOS REMOTOS NECESARIOS

La siguiente clase obtiene un objeto de tipo MiInterfazRemoto, implementado en nuestro servidor:

```
public class MiClienteRMI
{
  public static void main(String[] args)
{
          Try
{
MiTnterfazRemoto mir=
(MiInterfazRemoto)java.rmi.Namíng.lookup("//
" + args[0] + :" + args[1] + "/PruebaRMI");
//Imprimimos miMetodo1() tantas veces como //devuelva míMetodo2()
for (int i=l;i<=mir.miMetodo2();i++){ mir.miMetodo1();}
}
catch (Exception e)
{
e.printStackTraceo;
}
}
}
}
```

Como se puede observar, simplemente consiste en buscar el objeto remoto en el registro RMI de la máquina remota. Para ello usamos la clase Naming y su método lookup(…).

2.11.2. COMPILAR Y EJECUTAR EL CLIENTE

Una vez que ya tenemos definido el cliente, para compilarlo hacemos:

set CLASSPATH=%CLASSPATH%;.\objRemotos.jar;. javac MiClienteRMI.java

Luego, para ejecutar el cliente hacemos:

java MiClienteRMI 127.0.0.1 1234

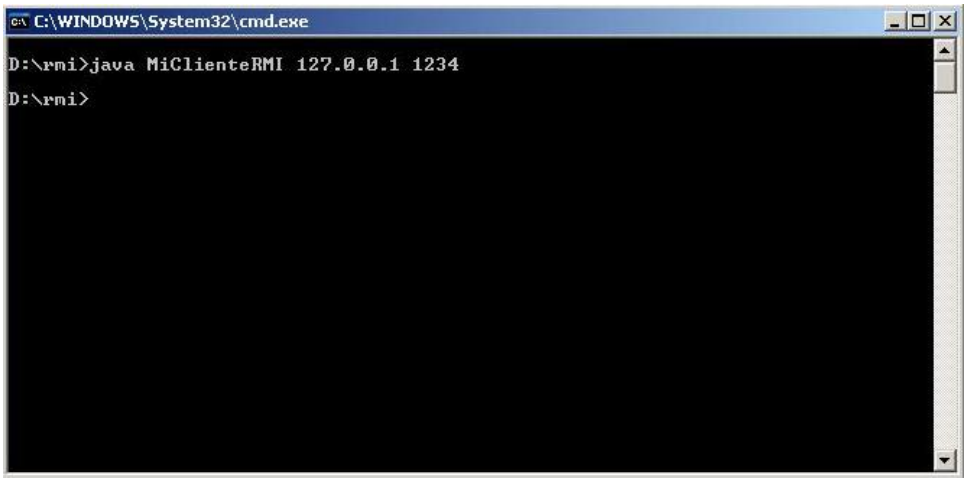

Figura 2.5 Ejecución del Cliente RMI

Se debe poder acceder al fichero Stub de la clase remota. Para ello, o bien lo copiamos al cliente y lo incluimos en su CLASSPATH, o lo eliminamos del CLASSPATH del servidor e incluimos su ruta en el java.rmi.codebase del servidor (si no se elimina del CLASSPATH del servidor, se ignorará la opción java.rmi.codebase, y el cliente no podrá acceder al Stub).
Si echamos un vistazo a la ventana donde está ejecutándose el servidor RMI, veremos como se ha encontrado el objeto remoto y ejecutado sus métodos:

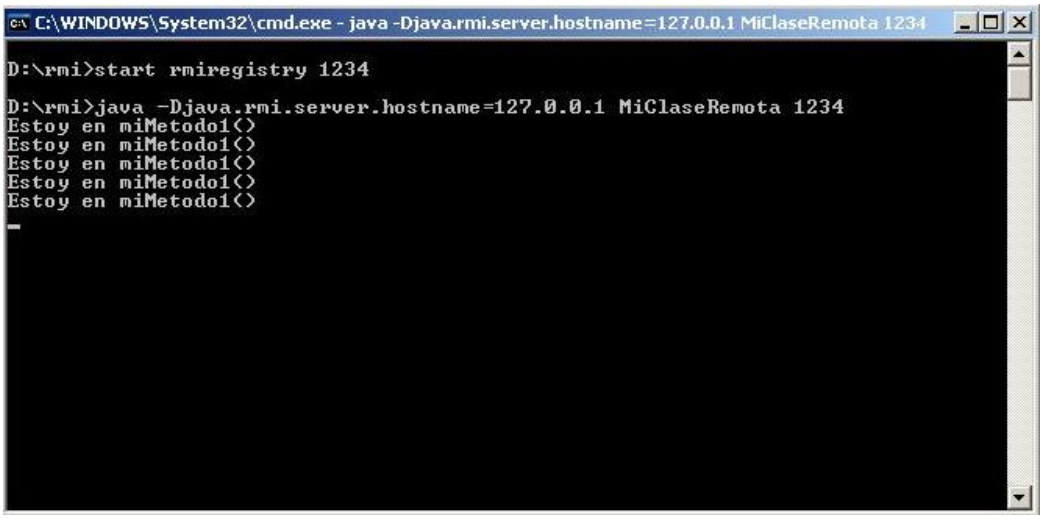

Figura 2.6 Registro del Servidor RMI

2.12. ANALISIS DE LAS PROPIEDADES DE SEGURIDAD

RMI extiende la filosofía de seguridad de JAVA a la hora de controlar la ejecución de las clases. El concepto de SandBox o caja de arena, se extiende a la programación distribuida de la mano del Security Manager que controla la descarga y funcionamiento de las clases provenientes de la red. Es obligatorio en cualquier implementación RMI crear una instancia del SecurityManager que nos va a garantizar un nivel de seguridad similar al que tienen los applets.

RMI no implementa ninguna política de seguridad en la capa de transporte. Las comunicaciones se realizan en "texto plano", por lo que, conceptos básicos de seguridad como: autentificación, confidencialidad...,etc. no se tienen en cuenta. El servidor RMI no autentifica las peticiones de acceso sobre sus objetos, de forma que un posible cliente ilícito podría tener acceso a los objetos. Esto es un grave inconveniente en aplicaciones que funcionen sobre redes de comunicación inseguras, como puede ser Internet.

Este problema puede ser resuelto mediante el uso de las Custom Socket Factories, que como ya hemos dicho en el punto anterior, permiten al programador crear su propio protocolo de encriptación e implementarlo en sus propios sockets. Pero pocos pueden aspirar a programar un protocolo de encriptación seguro en el poco tiempo que se suele tener para desarrollar una aplicación. La utilización de implementaciones existentes y probadas suelen ser una garantía de calidad y suelen evitarnos rompederos de cabeza. La propuesta de SUN para encriptar las comunicaciones RMI es SSL (Secure Socket Layer).

2.13. UNA COMPARACION DETALLADA DE CORBA, DCOM Y JAVA/RMI

El computar distribuido del objeto extiende un sistema de programación orientado al objeto permitiendo que los objetos sean distribuidos a través de un heterogeneous network, de modo que cada uno de éstos distribuyera el interoperate de los componentes del objeto como entero unificada. Estos objetos se pueden distribuir en diversas computadoras a través de una red, viviendo dentro de su propio espacio de dirección fuera de un uso, pero aparecen como si eran locales a un uso.

Tres de los paradigmas distribuidos más populares del objeto son el modelo componente distribuido del objeto de Microsoft (DCOM) , la arquitectura común del corredor de la petición del objeto de OMG (CORBA) y la invocación del método de Java/Remote de JavaSoft (Java/RMI).

Las diferencias entre estos tres modelos del punto de vista de un programador y de un punto de vista arquitectónico.

2.13.1. CORBA

Confía en un protocolo llamado el Inter-ORB protocolo del Internet (IIOP) para los objetos remoting.

Todo en la arquitectura de CORBA depende de un corredor de la petición del objeto (ORB). El ORB actúa como un excedente central del autobús del objeto que cada objeto de CORBA obra recíprocamente transparente con otros objetos de CORBA localizó localmente o remotamente. Cada objeto del servidor de CORBA tiene un interfaz y expone un sistema de métodos. Para solicitar un servicio, un cliente de CORBA adquiere una referencia del objeto a un objeto del servidor de CORBA. El cliente puede ahora hacer método invita la referencia del objeto como si el objeto del servidor de CORBA residiera en el espacio de dirección del cliente. El ORB es responsable de encontrar la puesta en práctica de un objeto de CORBA, preparándola para recibir peticiones, para comunicar peticiones a él y para llevar la contestación de nuevo al cliente. Un objeto de CORBA obra recíprocamente con el ORB a través del interfaz de ORB o a través de un adaptador del objeto - un adaptador básico del objeto (BOA) o un adaptador portable del objeto (POA). Puesto que CORBA es justo una especificación, puede ser utilizado en plataformas diversas del sistema operativo de los chasis a las cajas de UNIX a las máquinas de Windows a los dispositivos del handheld tan largos como hay una puesta en práctica de ORB para esa plataforma. Los vendedores importantes de ORB como Inprise tienen puestas en práctica de CORBA ORB a través de su producto de VisiBroker para Windows, UNIX y las plataformas e Iona del chasis a través de su producto de Orbix.
CORBA - IDL

módulo SimpleStocks{
interfaz StockMarket{
get-price del flotador (en símbolo de la secuencia);
};};
File : StockMarket.id1

2.13.2. DCOM

Que a menudo se llama COM en el alambre, apoya objetos remoting funcionando en un protocolo llamado el Remote Procedure Call del objeto (ORPC). Esta capa de ORPC se construye encima del RPC de DCE's y obra recíprocamente con servicios del tiempo de pasada de COM's. Un servidor de DCOM es un cuerpo del código que es capaz de servir encima de objetos de un tipo particular en el tiempo de pasada.

Cada objeto del servidor de DCOM puede apoyar las interfaces cada uno del múltiplo que representa un diverso comportamiento del objeto. Un cliente de DCOM llama en los métodos expuestos de un servidor de DCOM adquiriendo un indicador a uno de los interfaces del objeto del servidor.

El comienzo del objeto del cliente entonces que llama los métodos expuestos del objeto del servidor a través del indicador adquirido del interfaz como si el objeto del servidor residiera en el espacio de dirección del cliente. Según lo especificado por COM, la disposición de la memoria de un objeto del servidor se conforma con la disposición vitable de C++. Puesto que la especificación de COM está en el nivel binario permite que los componentes del servidor de DCOM sean escritos en lenguajes de programaciones diversas como C++, Java, el PASCAL del objeto (Delphi), básicas visuales e incluso COBOL. Mientras una plataforma apoya servicios de COM, DCOM se puede utilizar en esa plataforma. DCOM es muy usado ahora en la plataforma de Windows. Las compañías como software AG proporcionan puestas en práctica del servicio de COM a través de su producto de EntireX para UNIX, Linux y las plataformas del chasis Digital para la plataforma abierta y Microsoft de VMS para las plataformas de Windows y de Solaris.
DCOM - IDL

[
uuid(737la240-é5l-lldO-b4cl-444553540000),

```
version(1.0)
]
biblioteca SimpleStocks
{
importlib("stdole32.t1b");
[ uuíd(BC4COABO-A45-lld2-99C5-00AO2414C655), dual ]
interfaz TStockMarket :      IDispatch
{
Get-price de HRESULT([adentro] BSTR pl, [fuera de, retval]
flotador * rtn);
}
[uuid(BC4COAB3-A45-lld2-99C5-00AO2414C655) ]
coclass StockMarket
{
interfaz IStockMarket;
};};
File : StockMarketLib.id1
```

2.13.3. RMI

Confía en un protocolo llamado el protocolo alejado del método de Java (JRMP). Java confía pesadamente en la serialización del objeto de Java, que permite que transmiten los objetos sean formados (o) como corriente. Puesto que la serialización del objeto de Java es específica a Java, el objeto del servidor de Java/RMI y el objeto del cliente tienen que ser escritos en Java. Cada objeto del servidor de Java/RMI define un interfaz que se pueda utilizar para tener acceso al objeto del servidor fuera de la Java actual Machine(JVM) virtual y en JVM de otra máquina. El interfaz expone un sistema de los métodos que son indicativos de los servicios ofrecidos por el objeto del servidor. Para que un cliente localice un servidor opóngase para la primera vez, el RMI depende de un mecanismo de nombramiento llamado un RMIRegistry que los

funcionamientos en el servidor trabajen a máquina y lleva a cabo la información sobre objetos disponibles del servidor.

Un cliente de Java/RMI adquiere una referencia del objeto a un objeto del servidor de Java/RMI haciendo operaciones de búsqueda para una referencia del objeto del servidor e invoca métodos en el objeto del servidor como si el objeto del servidor de Java/RMI residiera en el espacio de dirección del cliente. Se nombran los objetos del servidor de Java/RMI usando URLs y para un cliente para adquirir una referencia del objeto del servidor, debe especificar el URL del objeto del servidor mientras que usted con el URL a un HTML PAGE. Puesto que Java/RMI confía en Java, puede ser utilizado en plataformas diversas del sistema operativo de los chasis a las cajas de UNIX a las máquinas de Windows a los dispositivos del handheld tan largos como hay una puesta en práctica virtual de la máquina de Java (JVM) para esa plataforma. Además Javasoft y Microsoft, los muchos de otras compañías han anunciado puertos virtuales de la máquina de Java.
Java/RMI - definición de interfaz

```
Paquete SimpleStocks;
importación java.rmi.*;
importación java.util.*;

public interface StockMarket extends java.rmi.Remote
{
float get-price( String symbol ) throws RemoteException;
}
```

File : StockMarket.java

CAPITULO III

ESTABLECER FUNCIONAMIENTO DE LA EMPRESA DE IMPLEMENTOS DEPORTIVOS

3.1. OBJETIVO GENERAL

Mejorar la calidad de vida de los habitantes y obtener una utilidad que justifique los costos, gastos y mano de obra incurridos.

3.2. OBJETIVOS ESPECIFICOS

Aprovechar mejor los recursos de la zona (materia prima, recurso humano).

Satisfacer el mercado regional y local.

3.3. OBJETIVOS POR DEPARTAMENTOS

3.3.1. ADMINISTRACION

Administrar, para tener una empresa competitiva y de esta manera mejorar el nivel de vida de los empleados y contribuir al desarrollo de país.

3.3.2. PRODUCCION

La finalidad de este departamento es que la materia prima, maquinaria y recurso humano sea óptimamente utilizado evitando así presencia de mermas y desperdicios contribuyendo con la productividad de la empresa.

3.3.3. COMERCIALIZACION

Incurrir en el mercado regional y local implementando estrategias de venta. Además seguir un cronograma de distribución por áreas y sectores, considerando los feriados, pedidos fijos, imprevistos y demanda generalizada.

3.3.4. FINANZAS

Generar una ganancia que justifique los gastos incurridos en la comercialización. Realizar un estudio de proyección de ventas para determinar si necesitamos financiamiento interno y/o externo, buscando una fuente de financiamiento adecuada.

3.3.5. RECURSOS HUMANOS

Contratación de personas responsables en la comercialización del producto, el cual se sienta motivado y satisfecho trabajando en forma eficaz y eficiente.

3.4. VISION

Ser una corporación líder en el mercado regional y local, en la comercialización de implementos deportivos que superen las expectativas de nuestros clientes, ofertando un producto de calidad, inmerso en el proceso de mejoramiento continuo manteniendo y respetando el medio ambiente.

3.5. MISION

Comercializar implementos deportivos competitivos y confiables que cumplan con los estándares de calidad nacionales así como proporcionar asistencia técnica especializada y oportuna.

Comercializar un producto a precios competitivos para satisfacer las necesidades y expectativas del consumidor.

3.6. ORGANIGRAMA

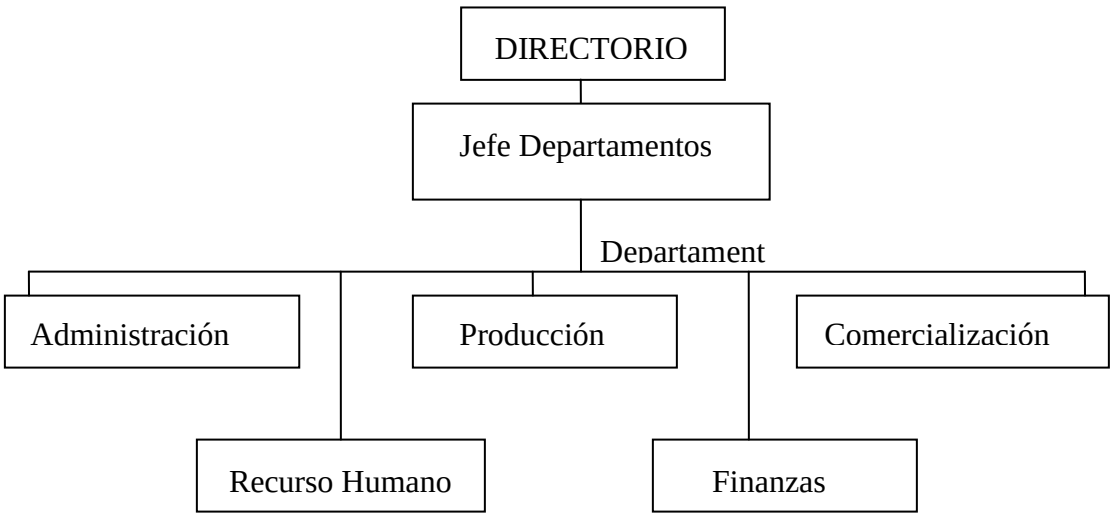

Figura 3.1 Organigrama general de empresas

El Directorio esta compuesto por el Presidente y Directores

3.7. ORIGENES DE LAS NECESIDADES

A continuación se examinan los principales aspectos y consideraciones que influyen en los procesos de definición de necesidades. Es importante considerarlos solo como antecedentes de referencia general, otros aspectos que influyen en la definición de necesidades.

3.8. PROCESO DE PRODUCCION

En empresas que tiene procesos de producción cortos o ninguna operación básica, suele darse el caso que la función comercial sea quien tenga la responsabilidad de definir la necesidad a cubrir por compras. La previsión de ventas, la acumulación de los pedidos recibidos o la llegada de un pedido importante, son prácticamente los programas de compra que especifican productos, calidades, cantidades e incluso, en ocasiones, los precios a conseguir en la compra. En empresas con largos procesos de producción, es frecuente que sea la responsabilidad de la misma función de producción la definición de la necesidad. Pero se da también casos en los que no es la función de producción quien define la necesidad, porque algún elemento del producto comprado, por ejemplo la calidad o el precio, influye tanto en la calidad y precio del producto terminado; casos en que la función comercial es la responsable de definir qué se debe comprar.

3.9. PRODUCTO QUE LA EMPRESA COMPRADORA VENDE

Existen compañías (como las cadenas que venden implementos deportivos) que practicante deben “tener todo en su ramo”, por ser ese uno de los fines para los que fueron constituidas. En este caso, el conocimiento de los elementos de la necesidad (producto, calidad y a veces precio) es indicado por las empresas matriz, no ocurriendo lo mismo con los elementos, cantidad y plazo que por supuesto han de ser especificados por la compañía compradora.

Por otro lado, es importante tener en cuenta que trabajar con licencias sobre productos representa con frecuencia trabajar con planos y especificaciones del concedente de la licencia, los que especifican sus componentes y calidades en forma adecuada al país de origen, pero a veces inadecuada para el mercado en el que la función de aprovisionamiento tiene que comprar.

3.10. PRODUCTO QUE LA EMPRESA COMPRA

Podemos observar que en razón de la posición en que el producto comprado se halle en su propio ciclo de vida, nos vamos encontrando con diversas formas de definir la necesidad. En productos maduros, el elemento calidad y el elemento precio presentan pocas dificultades en su definición dada la existencia de estándares de calidad y de tarifas que son comúnmente empleados, sólo es preciso por tanto fijar los elementos de cantidad y plazo.

Por otro lado, en productos que están en periodos de desarrollo, al final de su etapa de introducción o en su etapa de crecimiento muchas veces se tendrá que definir lo que se compra fijando las características de su funcionamiento, en vez de definir un proceso de elaboración a través de las especificaciones técnicas de fabricación.

Hay productos cuyo aprovisionamiento se logra a través de un mercado proveedor muy estructurado, como puede ser el caso de materias primas. La compra de este tipo de materias reviste de una particular importancia por diversas razones la mas importante se debe a que al tratarse de productos de uso continuado, hay que llegar a acuerdos de suministro que cubran extensos periodos de tiempo, y con frecuencia, ocurren oscilaciones notables en los precios de estas materias primas por lo tanto hay que prever el riesgo que esas oscilaciones se presenten, sus repercusiones y los modos de protegerse de ellas o suavizar su efecto.

3.11. LA PRESUPUESTACION

El proceso presupuestario no implica solo el cálculo de cifras, podríamos decir que es un sistema integral donde se presta particular atención a la fijación de unos objetivos con base en los cuales se planifican y controlan las diferentes actividades que puede realizar la empresa.

La presupuestación logística consiste en la estimación y el análisis de la demanda futura para un producto en particular, sus componentes y/o servicios, utilizando inputs como ratios históricos de ventas, estimaciones de marketing e información promocional, a través de diferentes técnicas de previsión.

En este sentido, la presupuestación logística abarca la predicción de la demanda con el objetivo de mejorar el flujo de información en la cadena de suministro de la empresa y por tanto preparar a la organización en el sentido de medios técnicos, humanos y financieros para soportar las operaciones futuras de la empresa. Es decir, estimación de compras, producción, necesidades de almacenaje, transportes, etc.

3.11.1. IMPORTANCIA DE LA PRESUPUESTACION

En la actualidad, la disposición de presupuestos o previsiones de demanda, constituye una parte fundamental de la logística por las implicaciones que una variación en ésta supone en los principales procesos de la cadena de suministro (gestión de stocks, aprovisionamiento, transporte, fabricación, nivel de servicio, etc.) y por los beneficios que proporciona su correcta estimación y control.

3.11.2. TECNICAS DE ELABORACION DE PRESUPUESTOS

Existen diversas técnicas y métodos utilizados para predecir el comportamiento de la demanda, desde la simple recogida de información de la red de ventas y su posterior análisis y extrapolación, hasta métodos complejos basados en modelos econométricos y estadísticos.

Lo primero a decidir en una empresa es si el presupuesto va a ser top-down o bottom-up, con independencia de que a posterior se revise en sentido contrario. Asimismo debe establecerse el horizonte temporal del mismo mensual, trimestral o anual, con independencia del grado de actualizaciones a realizar cada 3 días, semanalmente, etc.

La compañía utiliza el método que mejor se adapte a sus procesos y sistemática. No obstante, el método utilizado para calcular y elaborar el presupuesto, no debe basarse únicamente en el instinto, el conocimiento del mercado y la experiencia de individuos, sino en la sistematización del tratamiento de todas las variables bajo las cuales se ven afectados macroeconómicas, sectoriales, marketing, comerciales (por producto, por canal, por marca), del equipo comercial, estacionalidad, incidencias, etc.

3.11.3. LA RELACION A LA ESTRATEGIA

El proceso presupuestario no implica solo el cálculo de cifras. Podríamos decir que es un sistema integral donde se presta particular atención a la fijación de unos objetivos con base en los cuales se planifican y controlan las diferentes actividades que puede realizar una administración.

Los presupuestos forman una estructura de información que atraviesa por las diferencies funciones y/o procesos empresariales, con el fin de guiar y apoyar el

proceso de dirección. Permitiendo una primera e inmediata herramienta de control acerca del cumplimento de los planes estratégicos.

3.11.4. PRINCIPALES VENTAJAS DE TRABAJAR CON PRESUPUESTOS

- Es un estudio anticipado de las cosas y de las posibilidades de logro.
- Coordina a los miembros de una organización hacia un determinado plan global.
- Formaliza las responsabilidades de la Planificación.
- Suministra expectativas definidas acerca de los resultados de las decisiones.
- Permite juzgar la acción mediante confrontaciones entre objetivos y resultados.
- El control presupuestario es el medio de mantener el plan de operaciones dentro de los límites razonables.
- Mediante él se comparan unos resultados reales frente a los presupuestos, se determinan variaciones y en ocasiones da a la administración la posibilidad de tomar medidas correctivas.

Los presupuestos representan proyecciones de ingresos y costos, que normalmente cubren uno o más años. El presupuesto maestro de una firma incluye todas aquellas actividades cuya fiscalización es considerada como de importancia para el sano desarrollo de los negocios de la firma.

Por otra parte, el proceso de control se centra en el análisis de los resultados de las actividades planificadas, evaluando su desempeño, diagnosticando sus méritos y llevando a cabo acciones correctivas si fuese necesario, lo cual equivale a una redefinición de planes.

3.12. GESTION DE COMPRAS

La Gestión de Compras es crucial para alcanzar el éxito en la reducción de costos de la Cadena de Suministro. A través de métodos inteligentes y de una cuidadosa elección e integración de los proveedores una compañía puede mejorar la calidad así como reducir el costo de las mercancías o servicios. La función del Departamento de Compras, o de aquel componente de la organización que tomase su lugar, está definida por el conjunto de actividades que se desarrollan en la empresa para cubrir las necesidades de materiales y servicios provenientes del exterior. Los aspectos cuantitativos de estas necesidades corresponden a otros departamentos de la dirección de aprovisionamientos o de producción.

El concepto de materiales abarca, en un sentido amplio, las materias primas, los productos energéticos, los productos semielaborados, los envases y embalajes, los accesorios, los repuestos, los bienes de consumo, la maquinaria, etc., así como cierta clase de servicios.

En general, la gestión de compras incluye la selección y gestión de proveedores de mercancías y servicios, la negociación de precios y términos de compra, y la adquisición de mercancías y servicios según el nivel de calidad definido. Entonces, el principal objetivo del departamento de compras es realizar la compra del material necesario, en el tiempo que se precisa y al mejor precio.

En la función de compras se pueden distinguir cuatro subfunciones:

- La expresión de la necesidad (cualitativamente).
- La adquisición propiamente dicha.
- La puesta a disposición del usuario "interno".
- El control cuantitativo y cualitativo de lo adquirido.

Tanto la primera como la última de estas subfunciones, son compartidas con otros departamentos, en concreto con producción o con distribución o comercial, según los casos.

Los otros objetivos generales que persigue la gestión del aprovisionamiento son los siguientes:

- Apoyar en la consecución de los objetivos estratégicos, tales como el plan de negocio, estrategia de producción, creación de valor para el accionista, etc.
- Minimizar el costo total de la gestión de compras de acuerdo al plan de negocio de la compañía.
- Mejorar la calidad y el flujo de los bienes y servicio.
- Encontrar una fuente de suministro competente y fiable.
- Soportar y responder a los cambios de la demanda de los clientes.
- Minimizar el riesgo de aprovisionamiento o fluctuación de los precios.

- Gestionar a los proveedores para reducir el gasto y mejorar la calidad de las mercancías y servicios.

- Optimizar el coste del aprovisionamiento en base al riesgo a asumir.

3.12.1. IMPORTANCIA DE LA GESTION DE COMPRAS

La gestión de compras es uno de los puntos más significativos en la cadena de suministro. Está recogida en la norma de calidad ISO, donde están incluidos los diversos apartados de aseguramiento de la calidad:

- Proceso de compras.

- Información de las compras.

- Verificación de los productos comprados.

Por otro lado, toda actividad que desarrolla la empresa genera una utilidad y la utilidad es susceptible de ser medida en términos monetarios; generalmente se identifica el beneficio como medida de la utilidad producida por la empresa. Puede afirmarse que la compra o la gestión de compras también reporta beneficio, aunque sea un beneficio indirecto, ya que sólo se hace efectivo una vez realizada la venta.

La utilidad de una buena gestión de compras está, básicamente, en adquirir lo que se quiere sin pagar más de lo necesario. Si dado un producto que se produce y se vende con cierto margen, somos capaces de limar los costes de los componentes que a él se incorporan, habremos aumentado, sin duda, el beneficio.

A través de la reducción de los costes asociados a las transacciones de las compras en base a una gestión de aprovisionamientos efectiva, una compañía puede mejorar directamente su margen de beneficios, trasladar los ahorros al cliente o conseguir una combinación de los dos sin afectar a la calidad de sus productos.

3.12.2. LA GESTION DE COMPRAS Y LA ESTRATEGIA CORPORATIVA

Con el rápido cambio del entorno en el que deben operar hoy en día las empresas, éstas necesitan ser cada día más competitivas. Para conseguir esta alta competitividad, y entre otras cuestiones, es evidente que una gestión muy eficaz de compras se convierte en algo absolutamente primordial. Desempeña un papel crítico en la reestructuración de las empresas que intentan ganar, o recuperar una posición. Además, el concepto de mercado y economía nacionales, ha cedido el paso a mercados de una economía global.

Hoy son considerados como temas estratégicamente importantes; las fuentes alternativas de suministro en todo el mundo, cambios legislativos, formación de carteles de proveedores, solidez general y nivel competitivo de proveedores clave, cambios tecnológicos que pueden afectar el abastecimiento, modelos de distribución e innovaciones en la relación con proveedores en la gestión de materiales.

3.12.3. POLITICAS DE COMPRAS

Toda compañía debe disponer de una estrategia de compras y aprovisionamiento que debe estar en línea con su estrategia global. Asimismo, es recomendable que la compañía disponga de una Política de compras, que debe incluir los siguientes aspectos:

- Los parámetros de decisión: precio, plazo, calidad, servicio requerido, etc.

- La búsqueda de nuevas fuentes de suministro y la selección de proveedores: características de los productos, el modo de identificación de suministradores potenciales, la petición de ofertas, el número de proveedores a seleccionar, etc.

- La negociación: tipo de información, tiempo de espera, reglas básicas etc.

- La relación con las áreas funcionales de la compañía.

- Los criterios de ética profesional.

3.13. CONOCER LAS NECESIDADES DE LA EMPRESA

Antes de cada proceso de compras es preciso conocer perfectamente el producto cuya compra se va a realizar, así como las necesidades de su empresa relativas a este producto. Puesto que un comprador no solo adquiere productos, también se deben negociar actividades y servicios.

3.14. EL ESTUDIO DEL MERCADO DE PROVEEDORES

La compra supone decidir qué se compra y a quién: los estudios previos de la preparación se basan en lo potencial: si antes tratábamos de responder qué se puede comprar, ahora nos plantearemos. ¿a quién se puede comprar?. Debemos conocer todos los proveedores que pueden suministrar un producto de nuestro consumo, para ello contamos con fuentes de información.

3.15. INVENTARIOS

El inventario incluye todos aquellos bienes y materiales que se utilizan en los procesos de fabricación y distribución Las materias primas, las partes componentes y los productos terminados son parte del inventario, así como los diversos abastecimientos requeridos en el proceso de producción y de distribución.

El inventario compensa una administración poco consistente e ineficiente, incluyendo malos pronósticos, programación fortuita y atención inadecuada a los procesos de preparación y de generación de órdenes. En otras palabras, el inventario puede encubrir irregularidades y es una manera de que la administración las pase por alto. En estos casos, el inventario incrementa los costos y la productividad, sin reforzar los ingresos netos. Es "pasivo" sin importar en qué parte de la organización se prepare la hoja de estado de posición financiera (balance). Además, la situación empeora si una organización tiene artículos equivocados en su inventario o aun peor la falta de un inventario adecuado puede interrumpir el proceso de producción.

En la administración del inventario, los objetivos, las políticas y las decisiones que se tomen deben ser congruentes con los objetivos generales de la empresa, así como con los objetivos de mercadotecnia, financieros y de fabricación.

En todo momento, las decisiones referentes al inventario están entrelazadas con las decisiones de planeación de capacidad, con las estructuras de planeación a largo, mediano o corto plazo, así como en las fases de ejecución y control de la administración de las operaciones.

3.15.1. CLASIFICACIONES FUNCIONALES

La función principal del inventario es de amortiguamiento y desacoplamiento, pues funciona como amortiguador de golpes las cuales se despliegan las siguientes:

- Entre las demandas de los clientes y la capacidad de producción del fabricante.

- Entre los requerimientos de ensambles finales y la disponibilidad de los componentes.

- Entre los materiales que ingresan para una operación y los resultados de la operación precedente, y

- Entre los procesos de fabricación y la oferta de materias primas.

3.15.2. INVENTARIOS DE ANTICIPACION

Cierres por vacaciones, los periodos altos por ventas, las promociones de ventas y las posibles huelgas son situaciones que pueden conducir a una empresa a que produzca o compre artículos terminados, componentes, materiales o suministros adicionales. Los inventarios de anticipación permiten a una organización hacer frente, por adelantados, a una emergencia en la demanda o a una oferta insuficiente.

3.15.3. INVENTARIOS DE FLUCTUACION

Existe fluctuación tanto en la demanda como en la oferta. Cuando es económico, los inventarios se tienen para satisfacer la fluctuación más alta en la demanda de

artículos terminados, del mismo modo que se tienen para absorber las variaciones en los requerimientos de materias primas, componentes, suministros de producción y artículos de oficina, también se los conocen como inventario de seguridad.

3.16. PROMOCIONES

En los mercados tan altamente competidos que se están dando actualmente se hace necesario que el comercializador promocione sus productos más frecuentemente, con promociones mejor diseñadas y pensando en el cliente. Las promociones casi siempre han estado relacionadas con «regalar algo»; cuando a un productor o un comerciante le hablan de promoción por lo general piensa en «pague 2 y lleve 3», o en un porcentaje determinado de descuento por compra, la realidad es que la promoción es mucho más que un simple descuento.

3.16.1. ¿QUÉ ES PROMOCION?

Son una serie de actividades cuyo objetivo general es estimular la compra. Para estimular la compra (hacer promociones efectivas) es necesario conocer muy bien al cliente, saber qué productos quiere, cómo los quiere, qué espera de los bienes o servicios que desea comprar, no se pueden sacar promociones para resolver los problemas de la empresa.

En otras palabras, para hacer promociones realmente efectivas se requiere tener bien definido el target al cual se va a dirigir la actividad promocional, estar enfocados, no se puede sacar una promoción para todos. Si sé qué busca mi prospecto (producto básico) puedo diseñar una promoción que le llegue fácilmente, que cubra sus expectativas. Por ejemplo, si sé que mi cliente espera seguridad de la categoría de producto que le voy a ofrecer, la estrategia promocional deberá estar enfocada a comunicar seguridad. Si busca belleza, pues mi estrategia de comunicación se enfocará a comunicar sentimientos de belleza en mi cliente.

Antes de continuar con las estrategias promociónales es importante definir la comunicación y la importancia que cada uno de sus componentes tiene para la promoción.

Como se puede ver en el esquema de comunicación, para que esta se produzca es necesario que haya una respuesta y ésta debe ser el objetivo de la estrategia promocional: hacer que el cliente potencial compre el producto o servicio.

El mensaje promocional debe estar relacionado con los beneficios del producto o servicio, es decir, con lo que el producto hace por el cliente, es importante no tratar de enumerar o convencer al cliente de los múltiples beneficios del producto, sino más bien centrarse en uno o dos (los más importantes para el cliente).

Al diseñar el mensaje promocional se debe pensar en el cliente, en qué busca de mis productos o servicios, es decir, pensar en el producto básico, en que el lenguaje y la simbología a utilizar sean claros para el cliente.

3.16.2. OBJETIVOS DE LAS PROMOCIONES

Los comerciantes al realizar las promociones pueden buscar uno o varios objetivos así: Cambiar actitudes, reducir inventarios, reposicionar el producto, cambiar comportamientos de compra, frenar la competencia, destacar el producto, ganar participación en el mercado, generar la recompra. Desafortunadamente la mayoría de los comerciantes al diseñar las promociones no piensan sino en reducir inventarios, perdiendo oportunidades de promocionar sin muchos costos sus productos.

3.16.3. CLASE DE PROMOCIONES

La empresa que venden los implementos deportivos utilizan con más frecuencia las siguientes clases de promociones:

Publicidad: Para que la publicidad sea efectiva es necesario conocer muy bien al cliente, sus hábitos de consumo, su estilo de vida, etc. Diseñar los mensajes pensando en los beneficios que busca el posible cliente de los productos o servicios, utilizar los canales adecuados (prensa, radio, televisión, etc.). Es muy importante además, hacer un buen plan de medios, que optimice los recursos. La eficacia del plan de medios depende de: los objetivos de la promoción, de los mercados meta a los cuales se pretenda llegar, de los fondos disponibles y de la naturaleza de los medios (alcance, frecuencia, impacto, costo).

Concursos a sus clientes potenciales: Su objetivo es motivar la recompra.

Patrocinio de eventos: Esta técnica se ha convertido en una de las formas preferidas por fabricantes y comercializadores para promocionar sus productos o servicios, se patrocinan preferiblemente eventos deportivos. El comercializador debe recordar que la promoción no es sólo regalar, debe atreverse a crear promociones diferentes pero cuidándose de no dejar permanentemente una misma promoción, pues esta con el tiempo pierde vigencia.

CAPÍTULO IV

APLICACION DE LA METODOLOGIA UML EN EL DESARROLLO DEL SOFTWARE

4.1. PLAN DE DESARROLLO DE SOFTWARE

4.1.1. INTRODUCCIÓN

UML es una especificación de notación orientada a objetos. Se basa en las anteriores especificaciones BOOCH, RUMBAUGH y COAD-YOURDON. Divide cada proyecto en un número de diagramas que representan las diferentes vistas del proyecto. Estos diagramas juntos son los que representa la arquitectura del proyecto.

Con UML nos debemos olvidar del protagonismo excesivo que se le da al diagrama de clases, este representa una parte importante del sistema, pero solo representa una vista estática, es decir muestra al sistema parado. Sabemos su estructura pero no sabemos que le sucede a sus diferentes partes cuando el sistema empieza a funcionar. UML introduce nuevos diagramas que representa una visión dinámica del sistema. El diagrama de clases continua siendo muy importante, pero se debe tener en cuenta que su representación es limitada, y que ayuda a diseñar un sistema robusto con partes reutilizables, pero no a solucionar problemas de propagación de mensajes ni de sincronización o recuperación ante estados de error. En resumen, un sistema debe estar bien diseñado, pero también debe funcionar bien.

UML también intenta solucionar el problema de propiedad de código que se da con los desarrolladores, al implementar un lenguaje de modelado común para todos los

desarrollos se crea una documentación también común, que cualquier desarrollador con conocimientos de UML será capaz de entender, independientemente del lenguaje utilizado para el desarrollo.

UML es ahora un estándar, no existe otra especificación de diseño orientado a objetos, ya que es el resultado de las tres opciones existentes en el mercado. Su utilización es independiente del lenguaje de programación y de las características de los proyectos, ya que UML ha sido diseñado para modelar cualquier tipo de proyectos, tanto informáticos como de arquitectura, o de cualquier otro ramo.

4.1.2. PROPOSITO

El Plan de Acción del software, además de ser una herramienta documentaria y de contener la descripción del enfoque de desarrollo del software, su gran utilidad radica en la importancia que toma respecto al aporte consistente de la información relacionada al proyecto, partiendo desde la óptica de requerimientos que conlleva al lineamiento y arquitectura del sistema, hasta la concepción administrativa en su operación, control y mantenimiento.

Este producto documentario podrá ser utilizado por cualquier persona de la institución cuyo interés particular esté orientado a su conocimiento, enriquecimiento y en fin de hacer de él un instrumento de trabajo particular o general en el desenvolvimiento de sus actividades, obteniendo claridad sobre lo que se debe hacer, cuándo y cómo se debe hacer, que exigencias previas se requieren y que otras tareas se derivan de su quehacer.

4.1.3. ALCANCE

El Plan de Desarrollo del Software esta demarcado por la descripción del plan global utilizado para el desarrollo del Sistema de Compra y Venta con RMI dentro del

ambiente de la tecnología y de las comunicaciones, además durante el desarrollo se definen las características del producto a desarrollar Para el Plan de Desarrollo del Software.

4.2. VISION GENERAL DEL PROYECTO

4.2.1. GENERAL

Las Empresas que venden implementos deportivo siempre desean ser mejores que las competencias y tener sus productos a tiempo y satisfacer a sus clientes, además que su proceder empresarial es el mejoramiento del sistema es precisamente abordar la preconstrucción temporal anhelada, asumiendo el desarrollo de una herramienta técnica que mediante el compromiso participativo de sus entes administrativos, operativos e institucionales, logrará configurarla como instrumento satisfactor de necesidades manifestadas por la misma.

4.2.2. ALCANCE, LIMITACIONES Y SUPUESTOS

La conformación, descripción y desarrollo de todos los casos de uso manifestados durante las fases correspondientes, en especial en las de identificación de requerimientos y la de pruebas, permitirán aportar de manera detallada y precisa el espacio característico para su operación y delimitación, razón por la cual este literal será de revisión permanente, pero el alcance proyectado de manera global cubrirá los siguientes:

El manejo de las gestiones de Agencias, Proveedores, Clientes, Tipos de productos, Productos, Pedidos, Ventas y Usuarios.

El sistema lleva y elabora información fiable, pero esto no significa que no deba existir protección de información o mal manejo del sistema por parte de los usuarios, el sistema estará implementado hacia el ambiente intranet.

4.2.3. ENTREGABLES DEL PROYECTO

A continuación se indican y describen cada uno de los son objeto de modificaciones a lo largo del proceso de desarrollo que serán generados y utilizados por el proyecto y que constituyen los entregables, solo al término del proceso podríamos tener una versión definitiva y completa de cada uno de ellos.

4.2.3.1. PLAN DE DESARROLLO DEL SOFTWARE

Es le presente documento

4.2.3.2. MODELO DE CASOS DE USO DEL NEGOCIO

Es un modelo de las funciones del negocio vistas desde la perspectiva de los actores externos (solicitantes finales, otros sistemas etc.). Permite situar al sistema en el contexto organizacional haciendo énfasis en los objetivos en este ámbito. Este modelo se representa con un Diagrama de Casos de Uso.

4.2.3.3. MODELO DE OBJETOS DEL NEGOCIO

Es un modelo que describe la realización de cada caso de uso del negocio, estableciendo los actores internos, la información que en términos generales manipulan y los flujos de trabajo asociados al caso de uso del negocio.

4.2.3.4. GLOSARIO

Es un documento que define los principales términos usados en el proyecto. Permite establecer una terminología consensuada

4.2.3.5. MODELO DE CASOS DE USO

El modelo de Casos de Uso representa la funcionalidad completa de una sistema (o una clase), mostrando su interacción con los agentes externos. Esta representación se hace a través de las relaciones entre los actores (agentes externos) y los casos de uso (acciones) dentro del sistema.

4.2.3.6. VISION

Este documento define la visión del producto desde la perspectiva del cliente, especificando las necesidades y características del producto.

4.2.3.7. ESPECIFICACION DE CASOS DE USO

Para los casos de uso que lo requieran (cuya funcionalidad no sea evidente o que no baste con una simple descripción narrativa) se realiza una descripción detallada utilizando una plantilla de documento, donde se incluyen: precondiciones, post-condiciones, flujo de eventos.

4.2.3.8. MODELO DE ANALISIS Y DISEÑO

Este modelo establece la realización de los casos de uso en clases, pasando desde una representación en términos de análisis (sin incluir aspectos de implementación) hacia una de diseño, de acuerdo al avance del proyecto.

4.2.3.9. MODELO DE DATOS

Previendo que la información del sistema será soportada por una base de datos relacional, este modelo describe la representación lógica de los datos, de acuerdo con el enfoque para modelado relacional de datos.

4.2.3.10. MODELO DE DESPLIEGUE

Este modelo muestra el despliegue, la configuración de tipos de nodos del sistema, en los cuales se hará el despliegue de los componentes.

4.2.3.11. CASOS DE PRUEBA

Cada prueba es especificada mediante un documento que establece las condiciones de ejecución, las entradas de la prueba, y los resultados esperados. Estos casos de prueba son aplicados como pruebas de regresión en cada iteración. Cada caso de prueba llevará asociado un procedimiento de prueba con las instrucciones para realizar la prueba.

4.2.4. ORGANIZACIÓN DEL PROYECTO

4.2.4.1. PARTICIPANTES EN EL PROYECTO

De momento no se incluye ningún personal de ninguna empresa para proporcionar los requisitos y validar el sistema.

El personal del proyecto, considerando las fases de Inicio, Elaboración y Construcción, estará formado por los siguientes puestos de trabajo y personal asociado:

Jefe de Proyecto. Labor de Wilson Zambrano, con poca experiencia en metodologías de desarrollo, herramientas CASE y notaciones, en particular la notación UML.

Analista de Sistemas. El perfil establecido es: Ingeniero en Informática con conocimientos de UML, uno de ellos al menos con experiencia en sistemas afines a la línea del proyecto, labor que llevará a cabo Wilson Zambrano.

Analista - Programador. Con experiencia en el entorno de desarrollo del proyecto, con el fin de que los prototipos puedan ser lo más cercanos posibles al producto final. Este trabajo lo llevará a cabo Wilson Zambrano.

Ingeniero de Software. El perfil establecido es: Ingeniero en Informática recién titulado que participará como becario en el convenio universidad-empresa, realizando labores de gestión de requisitos, gestión de configuración, documentación y diseño de datos. Encargada de las pruebas funcionales del sistema, lo llevara a cabo Wilson Zambrano.

4.2.4.2. ROLE Y RESPONSABILIDADES

A continuación se describen las principales responsabilidades de cada uno de los roles en el equipo de desarrollo durante las fases de Inicio y Elaboración, de acuerdo con los roles que desempeñan.

Puesto	Responsabilidad
Jefe de Proyecto	El jefe de proyecto asigna los recursos, gestiona las prioridades, coordina las interacciones con los clientes y usuarios, y mantiene al equipo del proyecto enfocado en los objetivos. El jefe de proyecto también establece un conjunto de

	prácticas que aseguran la integridad y calidad de los artefactos del proyecto. Además, el jefe de proyecto se encargará de supervisar el establecimiento de la arquitectura del sistema.
Analista de Sistemas	Captura, especificación y validación de requisitos, interactuando con el cliente y los usuarios mediante entrevistas. Elaboración del Modelo de Análisis y Diseño. Colaboración en la elaboración de las pruebas funcionales y el modelo de datos.
Programador	Construcción de prototipos. Colaboración en la elaboración de las pruebas funcionales, modelo de datos y en las validaciones con el usuario
Ingeniero de Software	Gestión de requisitos, gestión de configuración y cambios, elaboración del modelo de datos, preparación de las pruebas funcionales, elaboración de la documentación. Elaborar modelos de implementación y despliegue.

Tabla 4.1 Roles y Responsabilidades del Equipo de Desarrollo del Proyecto

4.2.5. GESTION DEL PROYECTO

4.2.5.1. ESTIMACIONES DEL PROYECTO

A continuación se muestran los recursos y costos directos e indirectos que se han generado:

Recurso Humano

Nombre	Disponibilidad para el trabajo	Tiempo	Conocimiento de las herramientas	Remuneración
Wilson Zambrano	25%	18 meses	15%	$120
TOTAL:				$ 2160

Tabla 4.2 Costos de Recurso Humano.

Entorno de Ingeniería de Software

Descripción	Precio	Cantidad	Tiempo	Costo
Base de Datos Sql	$0	1	18 meses	$0
Jdeveloper	$0	1	18 meses	$0
TOTAL:				$0

Tabla 4.3 Costos de Software

Descripción	Precio	Cantidad	Tiempo	Costo
Computadora	1800	1	18 meses	$ 1800
Impresora	160	1	18 meses	$ 160
Cartuchos	50	1	18 meses	$ 50
TOTAL:				$ 2010

Tabla 4.4 Costos de Hardware

Costo Total del Proyecto

Descripción	Costo
Recurso Humano	$ 2160
Software	$ 0
Hardware	$ 2010
TOTAL:	$4170

Tabla 4.5 Costo Total del Proyecto

4.2.6. PLAN DEL PROYECTO

En esta sección se presenta la organización y el calendario del proyecto.

4.2.6.1. PLAN DE LAS FASES

El desarrollo se llevará a cabo en base a fases. La siguiente tabla muestra la distribución de tiempos de cada fase.

Fase	Nro. Iteraciones	Duración
Inicio	1	5 semanas
Elaboración	1	10 semanas
Construcción	2	18 semanas
Transición	-	-

Tabla 4.6 Distribución de tiempos para cada Fase

Los hitos que marcan el final de cada fase se describen en la siguiente tabla.

Descripción	Hito
Fase de Inicio	En esta fase desarrollará los requisitos del producto desde la perspectiva del usuario. Los principales casos de uso serán identificados y se hará un refinamiento del Plan de Desarrollo del Proyecto.
Fase de Elaboración	En esta fase se analizan los requisitos y se desarrolla un prototipo de arquitectura (incluyendo las partes más relevantes y / o críticas del sistema). Al final de esta fase, todos los casos de uso correspondientes a requisitos que serán implementados en la primera release de la fase de Construcción deben estar

	analizados y diseñados (en el Modelo de Análisis / Diseño). La revisión y aceptación del prototipo de la arquitectura del sistema marca el final de esta fase. En nuestro caso particular, por no incluirse las fases siguientes, la revisión y entrega de todos los artefactos hasta este punto de desarrollo también se incluye como hito. La primera iteración tendrá como objetivo la identificación y especificación de los principales casos de uso, así como su realización preliminar en el Modelo de Análisis / Diseño, también permitirá hacer una revisión general del estado de los artefactos hasta este punto y ajustar si es necesario la planificación para asegurar el cumplimiento de los objetivos. Ambas iteraciones tendrán una duración de una semana.

Fase de Construcción	Durante la fase de construcción se terminan de analizar y diseñar todos los casos de uso, refinando el Modelo de Análisis / Diseño. Se comienza la elaboración de material de apoyo al usuario.
Fase de Transición	En esta fase se prepararán dos releases para distribución, asegurando una implantación y cambio del sistema previo de manera adecuada, incluyendo el entrenamiento de los usuarios. El hito que marca el fin de esta fase incluye, la entrega de toda la documentación del proyecto con los manuales de instalación y todo el material de apoyo al usuario, la finalización del entrenamiento de los usuarios y el empaquetamiento del producto.

Tabla 4.7 Distribución de tiempos para cada Fase

4.2.6.2. CALENDARIO DEL PROYECTO

Para este proyecto se ha establecido el siguiente calendario:

Descripción	Comienzo
Modelado del Negocio	
Modelo de Casos de Uso del Negocio	Semana 1 18/11/2005
Requisitos	
Glosario	Semana 1 01/12/2005
Visión	Semana 2 08/12/2005
Modelo de Casos de Uso	Semana 3 15/12/2005
Especificación de Casos de Uso	Semana 3 15/12/2005
Análisis / Diseño	
Modelo de Análisis / Diseño	Semana 9 06/01/2006
Modelo de Datos	Semana 2 06/01/2006
Pruebas	
Pruebas de funcionlidad	Semana 3 05/03/2006
Gestión del proyecto	
Plan de Desarrollo del Software	Semana 1 05/04/2006

Tabla 4.8 Calendario del Proyecto

4.2.7. SEGUIMIENTO Y CONTROL DEL PROYECTO

4.2.7.1. MODELADO DEL NEGOCIO

Las empresa de compra y venta tiene un modulo en general para la adquisición y venta de los productos el siguiente grafico describe la funcionalidad del modulo.

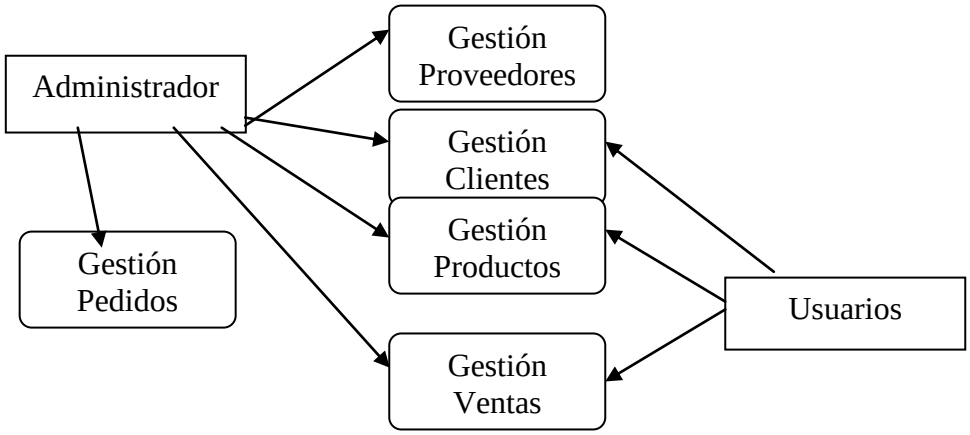

Tabla 4.9 Modelado del Negocio

4.2.7.2. MODELADO DE CASOS DE USO DEL NEGOCIO

Las empresas interactúan con distintos elementos externos, entre los que se identifican:

Cliente.- Persona que se realiza las compras en la empresa.
Proveedor.- Persona o empresa que suministra los productos
Usuario.- Persona que administra el sistema.

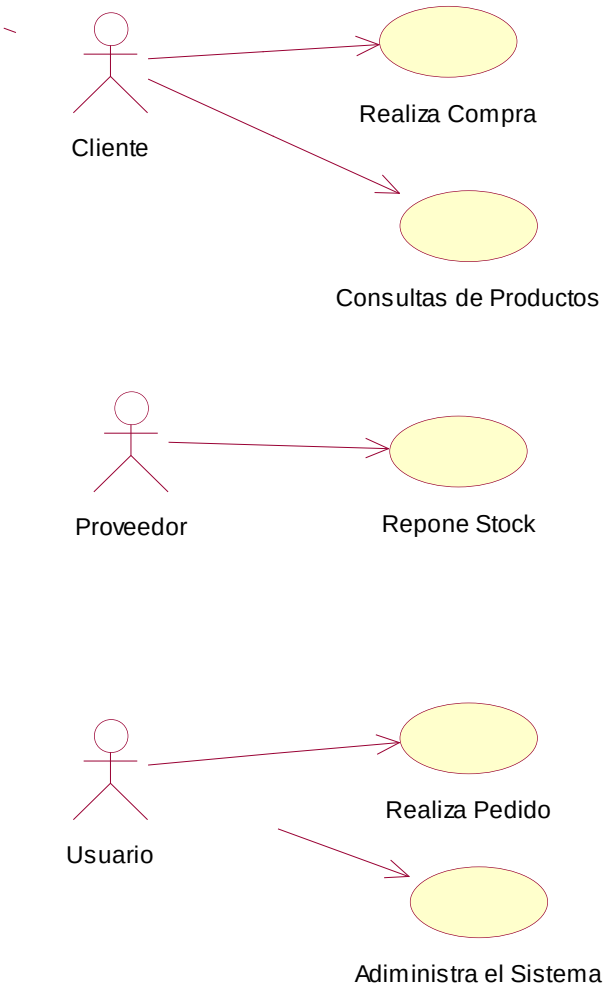

Figura 4.10 Modelo Caso de Uso del Negocio

4.3. REQUISITOS

4.3.1. GLOSARIO

4.3.1.1. INTRODUCCION

El Glosario recoge todos y cada uno de los términos manejados a lo largo de todo el proyecto. Se trata de un diccionario informal de datos y definiciones de la nomenclatura que se maneja.

4.3.1.2. PROPOSITO

El propósito de este glosario es definir con exactitud la terminología manejada en el proyecto.

4.3.1.3. ALCANCE

El alcance del presente documento pretende que todos los usuarios manejen la misma terminología con claridad.

4.3.1.4. DEFINICIONES

A continuación se presentan todos los términos manejados en el proyecto.

4.3.1.4.1. ADMINISTRADOR

El Administrador es la persona encargada de ingresar todos los datos para que los usuarios puedan trabajar correctamente en el sistema.

4.3.1.4.2. USUARIO

Se refiere a las personas pertenecientes a la empresa que hacen uso del sistema.

4.3.1.4.3. GESTION AGENCIAS

La gestión de Agencias es el ingreso de cada Agencia o Sucursal del la empresa en el sistema. Este caso de uso se puede invocar a través de la interfaz de Agencias.

4.3.1.4.4. GESTION PROVEEDORES

La gestión de Proveedores es el ingreso, modificación y eliminación del proveedor para la empresa. Este caso de uso se puede invocar a través de la interfaz de Proveedores.

4.3.1.4.5. GESTION PRODUCTOS

La gestión de Producto es el ingreso, modificación y eliminación del producto para la venta. Este caso de uso se puede invocar a través de la interfaz de Productos.

4.3.1.4.6. GESTION CLIENTES

La gestión de Clientes es el ingreso, modificación y eliminación del cliente en la empresa. Este caso de uso se puede invocar a través de la interfaz de Clientes.

4.3.1.4.7. GESTION USUARIOS

La gestión de Usuario es el ingreso, modificación y eliminación del usuario en la empresa. Este caso de uso se puede invocar a través de la interfaz de Usuarios.

4.3.1.4.8. GESTION DE PEDIDOS O AQUISICION DE PRODUCTOS

La gestión de Pedido es el ingreso, modificación y eliminación del pedido a los proveedores registrados para la empresa. Este caso de uso se puede invocar a través de la interfaz de Pedidos.

4.3.1.4.9. GESTION VENTAS

La gestión de Venta es el ingreso, anulación de la venta en la empresa. Este caso de uso se puede invocar a través de la interfaz de Ventas.

4.3.1.5. VISION

4.3.1.5.1. PROPOSITO

El propósito de éste documento es recoger, analizar y definir las necesidades de alto nivel y las características del sistema de Compra y Venta, el presente documento se basa en los requerimientos de los usuarios.

4.3.1.5.2. ALCANCE

El documento Visión se refiere, como ya se ha indicado, al sistema de Compra y Venta (El sistema permitirá atender con rapidez y eficacia al cliente).

4.3.1.5.3. SUPOSICIONES Y DEPENDENCIAS

La versión del sistema operativo puede variar de acuerdo al hardware designado para el producto del software.

Los Navegadores de Internet pueden ser: Microsoft Internet Explorer 5 o superior.

4.3.1.5.4. RESTRICCIONES

Debe contemplarse las siguientes restricciones:

- El sistema tiene opciones básicas para la adquisición y Venta de productos.
- El sistema esta creado en entorno Windows, no se ah probado en ambiente Linux.
- El sistema estará implementado hacia el ambiente de intranet.

4.3.1.5.5. OTROS REQUISITOS DEL PRODUCTO

Para la correcta operación del Sistema es fundamental conocer los requisitos que el sistema puede utilizar:

4.3.1.6. ESTANDARES APLICABLES

Normas de comunicación: TCP/IP
Plataforma: Windows Xp

4.3.1.7. REQUISITOS DEL SISTEMA

Requisitos de Hardware:
Un servidor WEB, la configuración de hardware recomendada para el servidor mencionado es la siguiente:
CD-ROM drive
512MB – 1 GB de memoria
1 GB de espacio libre en disco para instalación
1 tarjetas de red Ethernet y soporte TCP/IP instalado

Requerimientos de Software:

Instalar J2SDK versión 1.4.2 o superior

Instalar el servidor Tomcat.

Navegadores de Internet: (Microsoft Internet Explorer 5 o superior).

4.4. MODELO DE CASOS DE USO

Casos de Uso

A continuación se presentan los diagramas de casos de uso planteados.

4.4.1. ESPECIFICACION DE CASOS DE USO

4.4.1.1. GESTION DE AGENCIAS

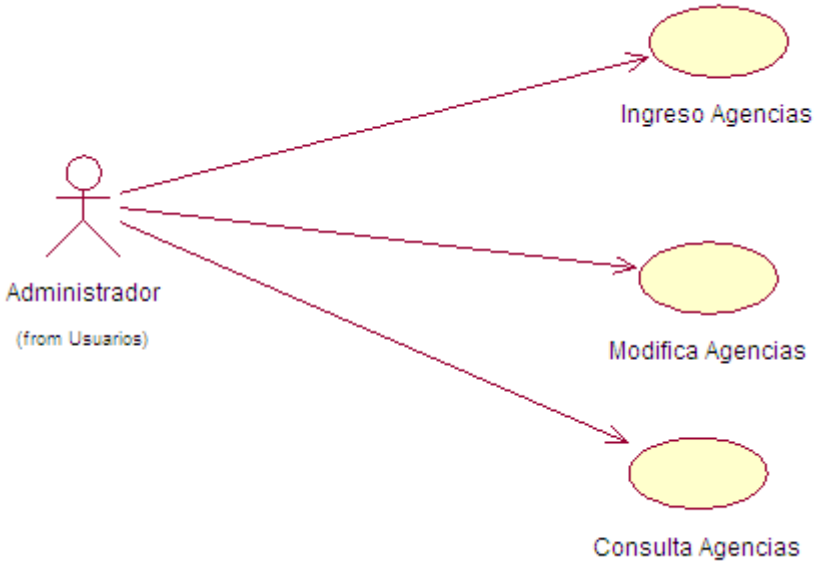

Figura 4.11 Caso de Uso del Negocio Gestión Agencias

4.4.1.2. GESTION DE PROVEEDORES

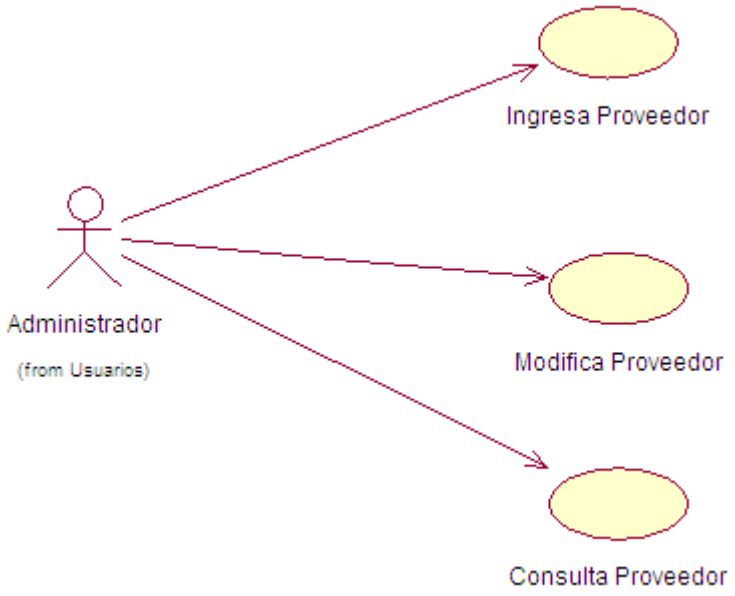

Figura 4.12 Caso de Uso del Negocio Gestión Proveedores

4.4.1.3. GESTION DE PRODUCTOS

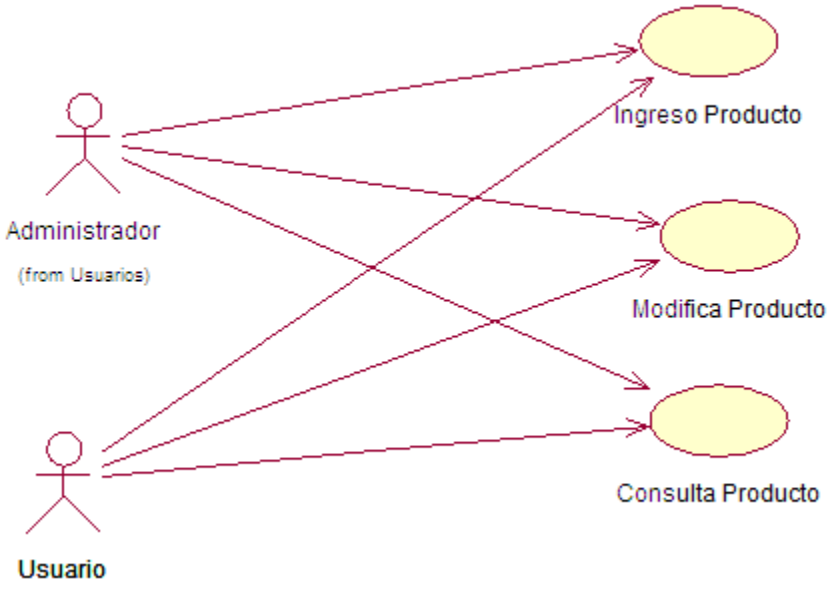

Figura 4.13 Caso de Uso del Negocio Gestión Productos

4.4.1.4. GESTION DE CLIENTES

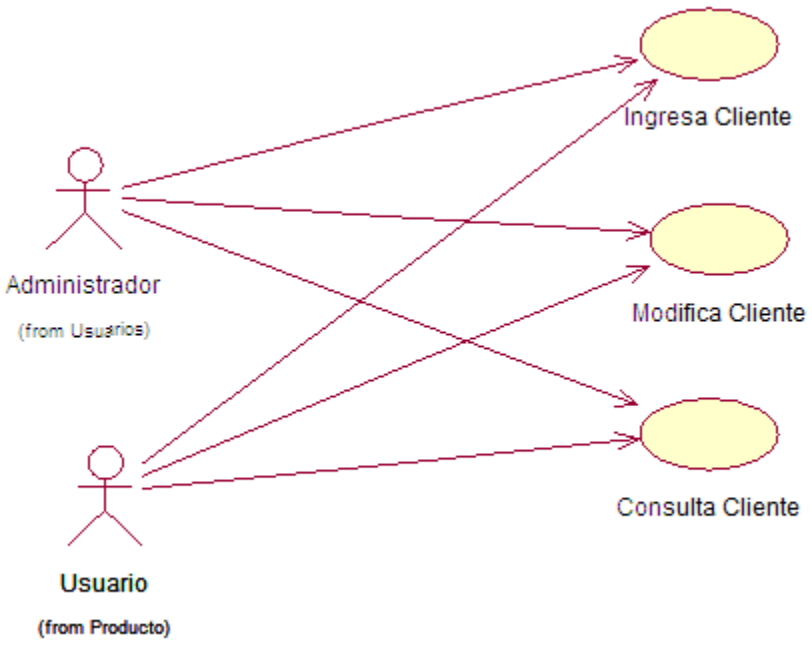

Figura 4.14 Caso de Uso del Negocio Gestión Clientes

4.4.1.5. GESTION DE USUARIOS

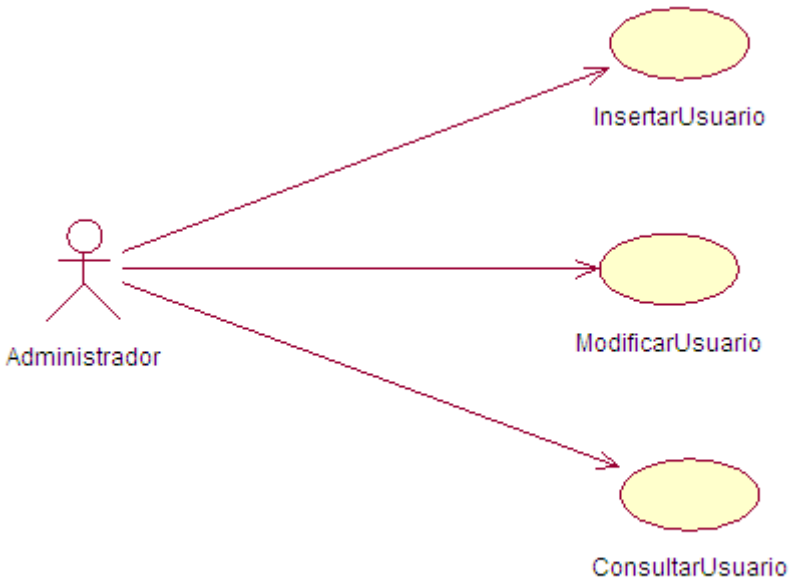

Figura 4.15 Caso de Uso del Negocio Gestión Usuarios

4.4.1.6. GESTION DE ADQUISICIONES

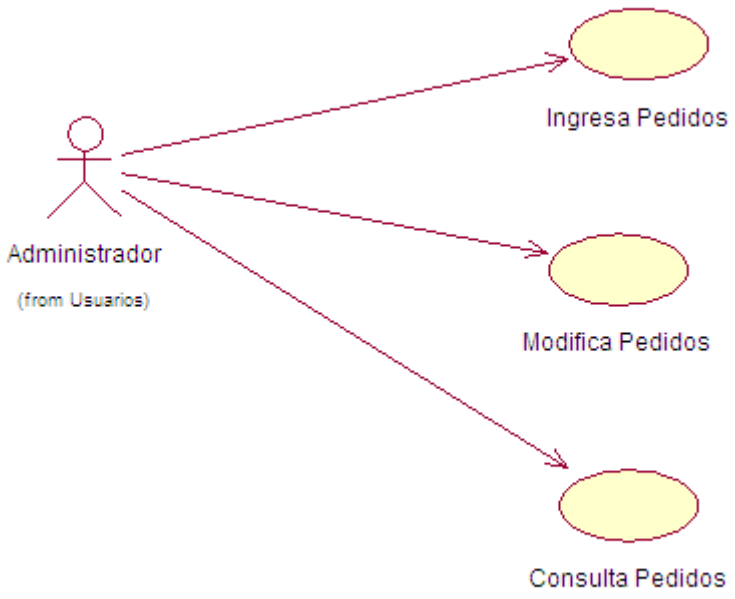

Figura 4.16 Caso de Uso del Negocio Gestión Adquisiciones

4.4.1.7. GESTION DE VENTAS

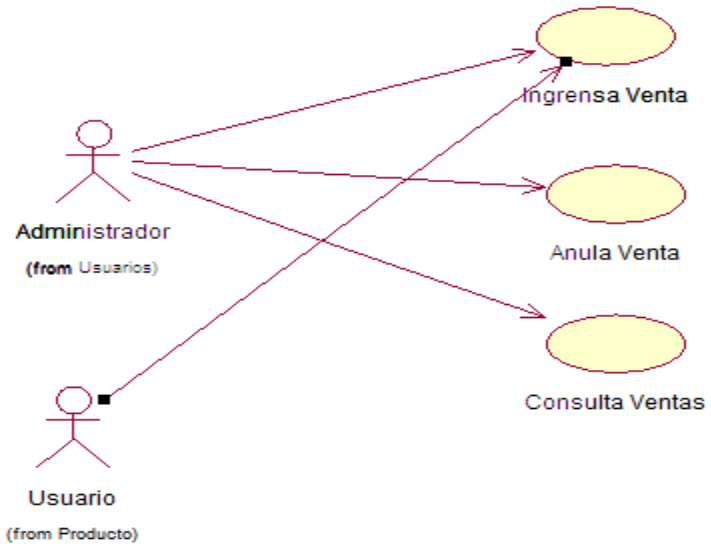

Figura 4.17 Caso de Uso del Negocio Gestión de Ventas

4.4.2. DESCRIPCION CASOS DE USO

A continuación muestra una descripción para ayudar a comprender los Casos de Uso.

Nombre	Gestión de Agencias
Alias	
Actores	Responsable
Función	Permitir el mantenimiento de las Agencias
Descripción	El Responsable puede consultar todas las agencias existentes, además puede registrar Agencias nuevas, ingresasando todos sus datos correspondientes. También es posible modificar algunos campos de las agencias ya existentes

Tabla 4.18 Descripción de Caso de Uso Gestión Agencias

Nombre	Gestión de Proveedores
Alias	
Actores	Responsable
Función	Permitir el mantenimiento de los Proveedores
Descripción	El Responsable puede consultar todas los proveedores existentes, además puede registrar nuevos proveedores, ingresasando todos sus datos correspondientes. También es posible modificar algunos campos de los proveedores ya existentes

Tabla 4.19 Descripción de Caso de Uso Gestión Proveedores

Nombre	Gestión de Productos
Alias	
Actores	Responsable
Función	Permitir el mantenimiento de los Productos
Descripción	El Responsable puede consultar todos los productos existentes, además puede registrar nuevos productos ingresasando todos sus datos correspondientes. También es posible modificar algunos campos de los productos ya existentes

Tabla 4.20 Descripción de Caso de Uso Gestión Productos

Nombre	Gestión de Clientes
Alias	
Actores	Responsable
Función	Permitir el mantenimiento de los Clientes

Descripción	El Responsable puede consultar todos los clientes existentes, además puede registrar nuevos clientes ingresasando todos sus datos correspondientes. También es posible modificar algunos campos de los clientes ya existentes

Tabla 4.21 Descripción de Caso de Uso Gestión Cliente

Nombre	Gestión de Usuarios
Alias	
Actores	Responsable
Función	Permitir el mantenimiento de los Usuarios
Descripción	El Responsable puede consultar todas los Usuarios existentes, además puede registrar nuevos usuarios ingresasando todos sus datos correspondientes. También es posible modificar algunos campos de los usuarios ya existentes

Tabla 4.22 Descripción de Caso de Uso Gestión Usuarios

Nombre	Gestión de Adquisición de Productos
Alias	
Actores	Responsable
Función	Permitir el mantenimiento de las Adquisiciones
Descripción	El Responsable puede consultar todas las adquisiciones existentes, además puede registrar nuevas adquisiciones. También es posible modificar las adquisiciones antes enviadas

Tabla 4.23 Descripción de Caso de Uso Gestión Adquisición

Nombre	Gestión de Ventas
Alias	
Actores	Responsable
Función	Permitir el mantenimiento de las Ventas
Descripción	El Responsable puede consultar todas las ventas existentes, además puede registrar nuevas ventas También es posible anular las ventas

Tabla 4.24 Descripción de Caso de Uso Gestión Ventas

4.4.3. EVENTOS

En este formato se establecen los eventos que pueden ser generados por el actor y van a ser atendidos por cada Caso de Uso. Por evento entendemos la interacción que tiene un actor con la aplicación a través de la interfaz gráfica, como el clic de un ratón.

FORMATOS DE EVENTOS	
Nombre Caso de Uso	Consulta de Agencias
Alias	
Actores	Responsable
Evento	Respuesta del Sistema
1. Selecciona opción Agencias	2. Muestra la pantalla de las agencias

Tabla 4.25 Eventos de Caso de Uso Consulta Agencias

FORMATOS DE EVENTOS	
Nombre Caso de Uso	Ingreso de Agencias
Alias	
Actores	Responsable

Evento	Respuesta del Sistema
1. Selecciona opción Agencias	2. Muestra la pantalla de las agencias
3. Selecciona ingreso nuevo	4. Muestra la pantalla de ingreso de agencias
5. Digitar datos nuevos	
6. Clic en grabar datos	
	7 Capturar datos nuevos.
	8. Graba datos en el sistema

Tabla 4.26 Eventos de Caso de Uso Ingresa Agencia

FORMATOS DE EVENTOS	
Nombre Caso de Uso	Modificación de Agencias
Alias	
Actores	Responsable
Evento	Respuesta del Sistema
1. Selecciona opción Agencias	2. Muestra la pantalla de las agencias
3. Selecciona la agencia que se va a modificar	4. Muestra la pantalla de modificación con los respectivos datos
5. Digitar datos nuevos	
6. Clic en grabar datos	
	7. Captura datos nuevos.
	8. Graba datos en el sistema

Tabla 4.27 Eventos de Caso de Uso Modifica Agencia

FORMATOS DE EVENTOS	
Nombre Caso de Uso	Consulta de Proveedores
Alias	
Actores	Responsable
Evento	Respuesta del Sistema
1. Selecciona opción Proveedores	2. Muestra la pantalla de los proveedores

Tabla 4.28 Eventos de Caso de Uso Consulta Proveedores

FORMATOS DE EVENTOS	
Nombre Caso de Uso	Ingreso de Proveedor
Alias	
Actores	Responsable
Evento	Respuesta del Sistema
1. Selecciona opción Proveedor	2. Muestra la pantalla de los proveedores
3. Selecciona ingreso nuevo 5. Digitar datos nuevos 6. Clic en grabar datos	4. Muestra la pantalla de ingreso de proveedores 7 Capturar datos nuevos. 8. Graba datos en el sistema

Tabla 4.29 Eventos de Caso de Uso Ingresa Proveedor

FORMATOS DE EVENTOS	
Nombre Caso de Uso	Modificación de Proveedores

Alias	
Actores	Responsable
Evento	Respuesta del Sistema
1. Selecciona opción proveedores 3. Selecciona el proveedor que se va a modificar 5. Digitar datos nuevos 6. Clic en grabar datos	2. Muestra la pantalla de los proveedores 4. Muestra la pantalla de modificación con los respectivos datos 7. Captura datos nuevos. 8. Graba datos en el sistema

Tabla 4.30 Eventos de Caso de Uso Modifica Proveedores

	FORMATOS DE EVENTOS
Nombre Caso de Uso	Consulta de Productos
Alias	
Actores	Responsable
Evento	Respuesta del Sistema
1. Selecciona opción Productos	2. Muestra la pantalla de los productos

Tabla 4.31 Eventos de Caso de Uso Consulta Productos

	FORMATOS DE EVENTOS
Nombre Caso de Uso	Ingreso de Productos
Alias	
Actores	Responsable
Evento	Respuesta del Sistema

1. Selecciona opción Productos	2. Muestra la pantalla de los productos
3. Selecciona ingreso nuevo	4. Muestra la pantalla de ingreso de productos
5. Digitar datos nuevos	
6. Clic en grabar datos	
	7 Capturar datos nuevos.
	8. Graba datos en el sistema

Tabla 4.32 Eventos de Caso de Uso Ingresa Producto

FORMATOS DE EVENTOS	
Nombre Caso de Uso	Modificación de Productos
Alias	
Actores	Responsable
Evento	Respuesta del Sistema
1. Selecciona opción Productos	2. Muestra la pantalla de los productos
3. Selecciona el productos que se va a modificar	4. Muestra la pantalla de modificación con los respectivos datos
5. Digitar datos nuevos	
6. Clic en grabar datos	
	7. Captura datos nuevos.
	8. Graba datos en el sistema

Tabla 4.33 Eventos de Caso de Uso Modifica Producto

FORMATOS DE EVENTOS	
Nombre Caso de Uso	Consulta de Clientes

Alias	
Actores	Responsable
Evento	Respuesta del Sistema
1. Selecciona opción Clientes	2. Muestra la pantalla de los clientes

Tabla 4.34 Eventos de Caso de Uso Consulta Clientes

FORMATOS DE EVENTOS	
Nombre Caso de Uso	Ingreso de Clientes
Alias	
Actores	Responsable
Evento	Respuesta del Sistema
1. Selecciona opción Clientes 3. Selecciona ingreso nuevo 5. Digitar datos nuevos 6. Clic en grabar datos	2. Muestra la pantalla de los clientes 4. Muestra la pantalla de ingreso de clientes 7 Capturar datos nuevos. 8. Graba datos en el sistema

Tabla 4.35 Eventos de Caso de Uso Ingresa Cliente

FORMATOS DE EVENTOS	
Nombre Caso de Uso	Modificación de Clientes
Alias	
Actores	Responsable
Evento	Respuesta del Sistema

1. Selecciona opción clientes	2. Muestra la pantalla de los clientes
3. Selecciona el cliente que se va a	4. Muestra la pantalla de modificación con los respectivos
modificar 5. Digitar datos nuevos 6. Clic en grabar datos	Datos 7. Captura datos nuevos. 8. Graba datos en el sistema

Tabla 4.36 Eventos de Caso de Uso Modifica Cliente

FORMATOS DE EVENTOS	
Nombre Caso de Uso	Consulta de Usuarios
Alias	
Actores	Responsable
Evento	Respuesta del Sistema
1. Selecciona opción Usuarios	2. Muestra la pantalla de los usuarios

Tabla 4.37 Eventos de Caso de Uso Consulta Usuarios

FORMATOS DE EVENTOS	
Nombre Caso de Uso	Ingreso de Usuarios
Alias	
Actores	Responsable
Evento	Respuesta del Sistema
1. Selecciona opción Usuarios	2. Muestra la pantalla de los usuarios

3. Selecciona ingreso nuevo 5. Digitar datos nuevos 6. Clic en grabar datos	4. Muestra la pantalla de ingreso de usuario 7 Capturar datos nuevos. 8. Graba datos en el sistema

Tabla 4.38 Eventos de Caso de Uso Ingresa Usuario

FORMATOS DE EVENTOS	
Nombre Caso de Uso	Modificación de Usuarios
Alias	
Actores	Responsable
Evento	Respuesta del Sistema
1. Selecciona opción Usuarios 3. Selecciona el usuario que se va a modificar 5. Digitar datos nuevos 6. Clic en grabar datos	2. Muestra la pantalla de los usuarios 4. Muestra la pantalla de modificación con los respectivos datos 7. Captura datos nuevos. 8. Graba datos en el sistema

Tabla 4.39 Eventos de Caso de Uso Modifica Usuario

FORMATOS DE EVENTOS	
Nombre Caso de Uso	Consulta de Adquisición de Productos
Alias	
Actores	Responsable
Evento	Respuesta del Sistema

1. Selecciona opción Compras	2. Muestra la pantalla de las compras

Tabla 4.40 Eventos de Caso de Uso Consulta Adquisiciones

FORMATOS DE EVENTOS	
Nombre Caso de Uso	Ingreso de Adquisición de productos
Alias	
Actores	Responsable
Evento	Respuesta del Sistema
1. Selecciona opción Compras 3. Selecciona ingreso nuevo 5. Digitar datos nuevos 6. Clic en grabar datos	2. Muestra la pantalla de las compras 4. Muestra la pantalla de ingreso compras o pedidos 7 Capturar datos nuevos. 8. Graba datos en el sistema

Tabla 4.41 Eventos de Caso de Uso Ingresa Adquisicion

FORMATOS DE EVENTOS	
Nombre Caso de Uso	Modificación de Adquisiciones de compras
Alias	
Actores	Responsable
Evento	Respuesta del Sistema
1. Selecciona opción Compras 3. Selecciona la compra que	2. Muestra la pantalla de las compras 4. Muestra la pantalla de modificación con los

se va a modificar 5. Digitar datos nuevos 6. Clic en grabar datos	respectivos datos 7. Captura datos nuevos. 8. Graba datos en el sistema

Tabla 4.42 Eventos de Caso de Uso Modifica Adquisición

FORMATOS DE EVENTOS	
Nombre Caso de Uso	Consulta de Ventas
Alias	
Actores	Responsable
Evento	Respuesta del Sistema
1. Selecciona opción Ventas	2. Muestra la pantalla de las ventas

Tabla 4.43 Eventos de Caso de Uso Consulta Ventas

FORMATOS DE EVENTOS	
Nombre Caso de Uso	Ingreso de Ventas
Alias	
Actores	Responsable
Evento	Respuesta del Sistema
1. Selecciona opción Ventas 3. Selecciona ingreso nuevo 5. Digitar datos nuevos 6. Clic en grabar datos	2. Muestra la pantalla de las ventas 4. Muestra la pantalla de ingreso de ventas 7 Capturar datos nuevos.

	8. Graba datos en el sistema

Tabla 4.44 Eventos de Caso de Uso Ingresa Venta

FORMATOS DE EVENTOS	
Nombre Caso de Uso	Modificación de ventas
Alias	
Actores	Responsable
Evento	Respuesta del Sistema
1. Selecciona opción ventas 3. Selecciona la venta que se va a	2. Muestra la pantalla de las ventas 4. Muestra la pantalla de modificación con los respectivos
modificar 5. Digitar datos nuevos 6. Clic en grabar datos	Datos 7. Captura datos nuevos 8. Graba datos en el sistema

Tabla 4.45 Eventos de Caso de Uso Modifica Venta

4.4.4. DIAGRAMA DE ESTRUCTURA ESTATICA (DE CLASES)

Nos muestra una vista de la aplicación del sistema. Las clases son la plantilla de los objetos, y aquí podemos ver representados estos con sus atributos o características y su comportamiento o métodos, así como la relación entre ellas.

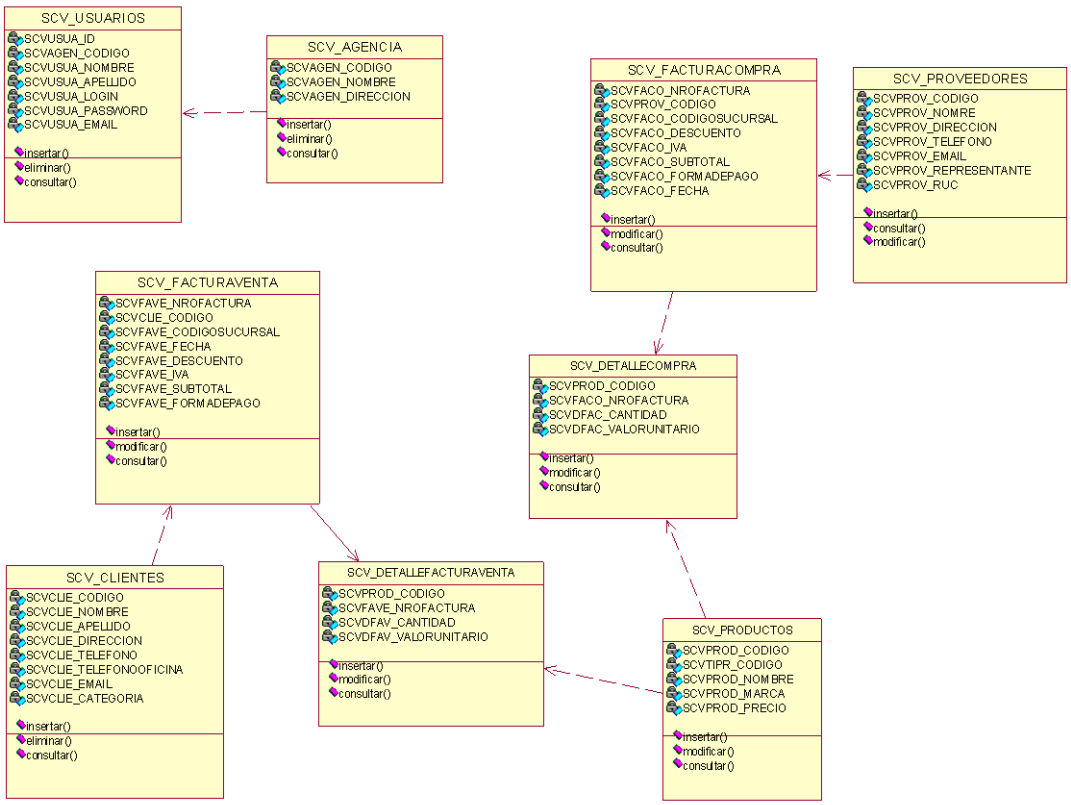

Figura 4.46 Diagrama de Clases

4.4.5. DIAGRAMA DE INTERACCION

Son aquellos que muestran las interacciones de un usuario con el sistema. Interacción es una cadena de mensajes enviados entre los objetos en respuesta a un evento generado por el usuario sobre la aplicación.

Los diagramas de interacción pueden ser Diagramas de Secuencia y Diagramas de Colaboración. Estos diagramas conforman la etapa del diseño de la aplicación, y se crean a partir de los diagramas de Casos de Uso y el Conceptual.

Los Diagramas de Secuencia representan una interacción entre objetos de manera secuencial en el tiempo. Muestra la participación de objetos en la interacción entre sus "líneas de vida", (desde que se instancias) y los mensajes que ellos organizadamente intercambian en el tiempo. El responsable o ACTOR es quien inicia el ciclo interactuando inicialmente con la interfaz de usuario.

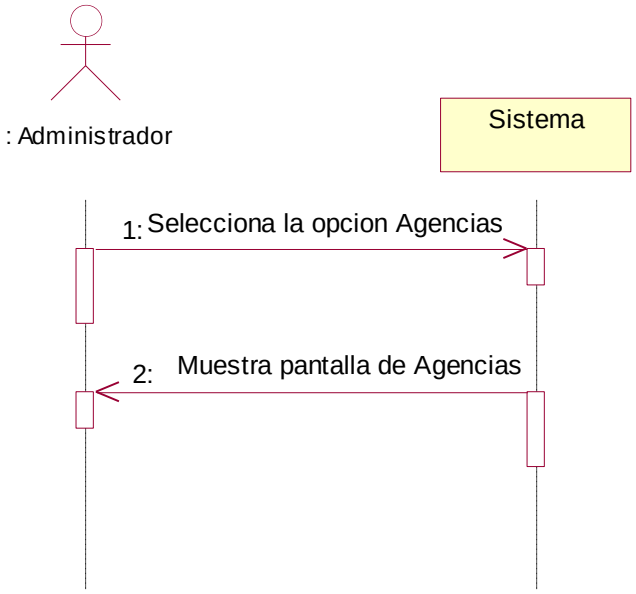

Figura 4. 47 Diagrama de Secuencia Consulta de Agencias

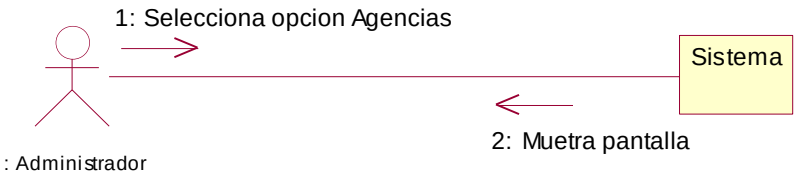

Figura 4.48 Diagrama de Colaboración Consulta de Agencias

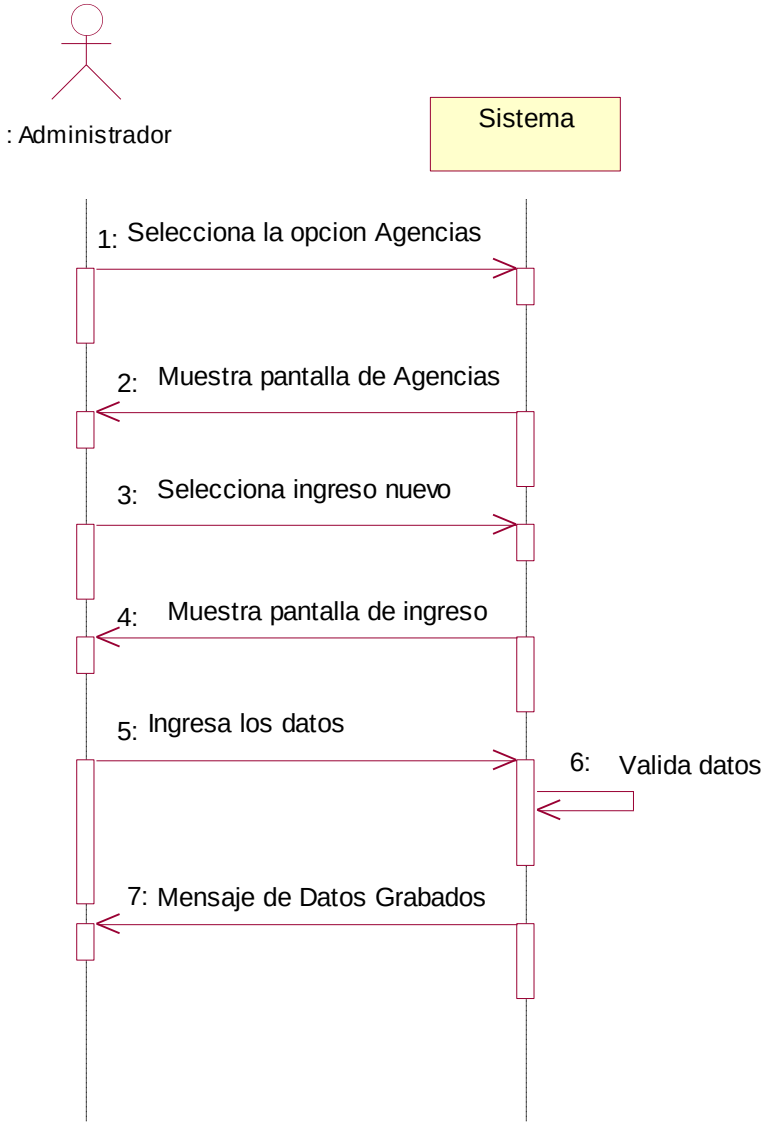

Figura 4.49 Diagrama de Secuencia Insertar Agencias

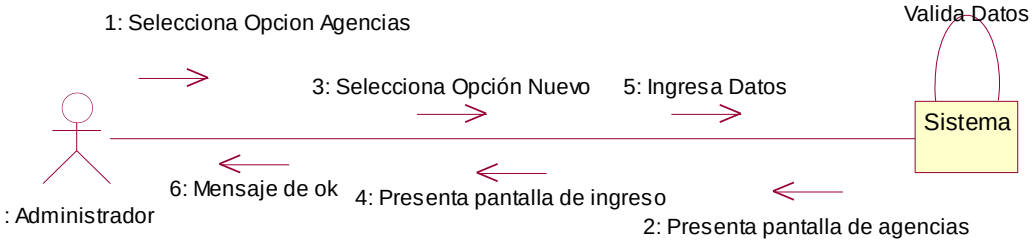

Figura 4.50 Diagrama de Colaboración Inserta Agencias

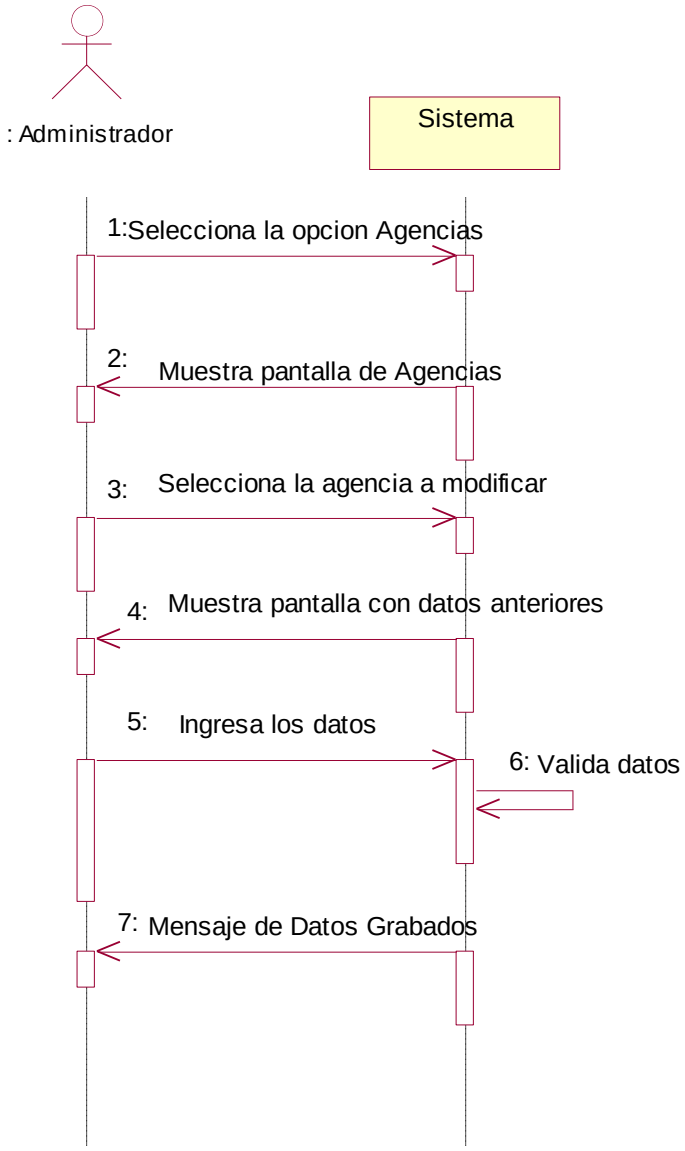

Figura 4.51 Diagrama de Secuencia Modifica Agencia

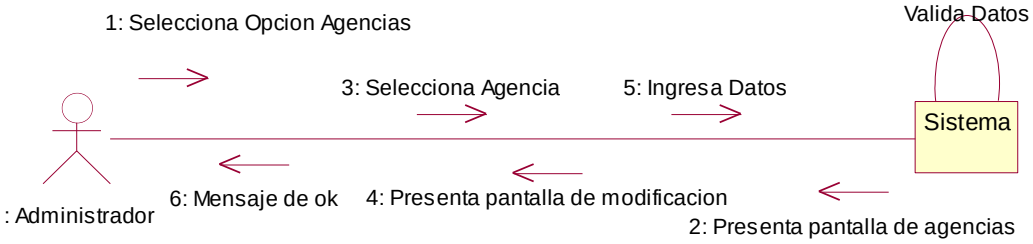

Figura 4.52 Diagrama de Colaboración Modifica Agencias

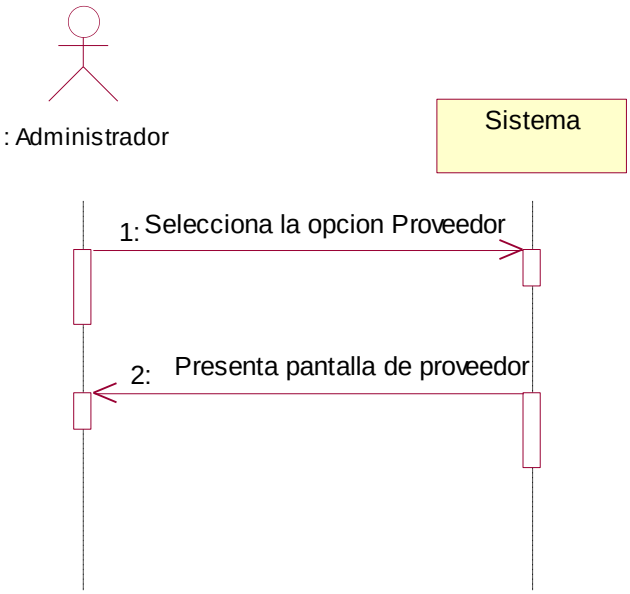

Figura 4.53 Diagrama de Secuencia Consulta de Proveedores

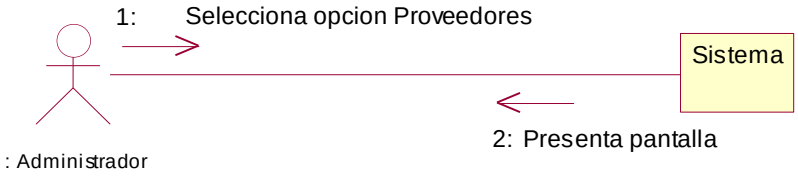

Figura 4.54 Diagrama de Colaboración Consulta de Proveedores

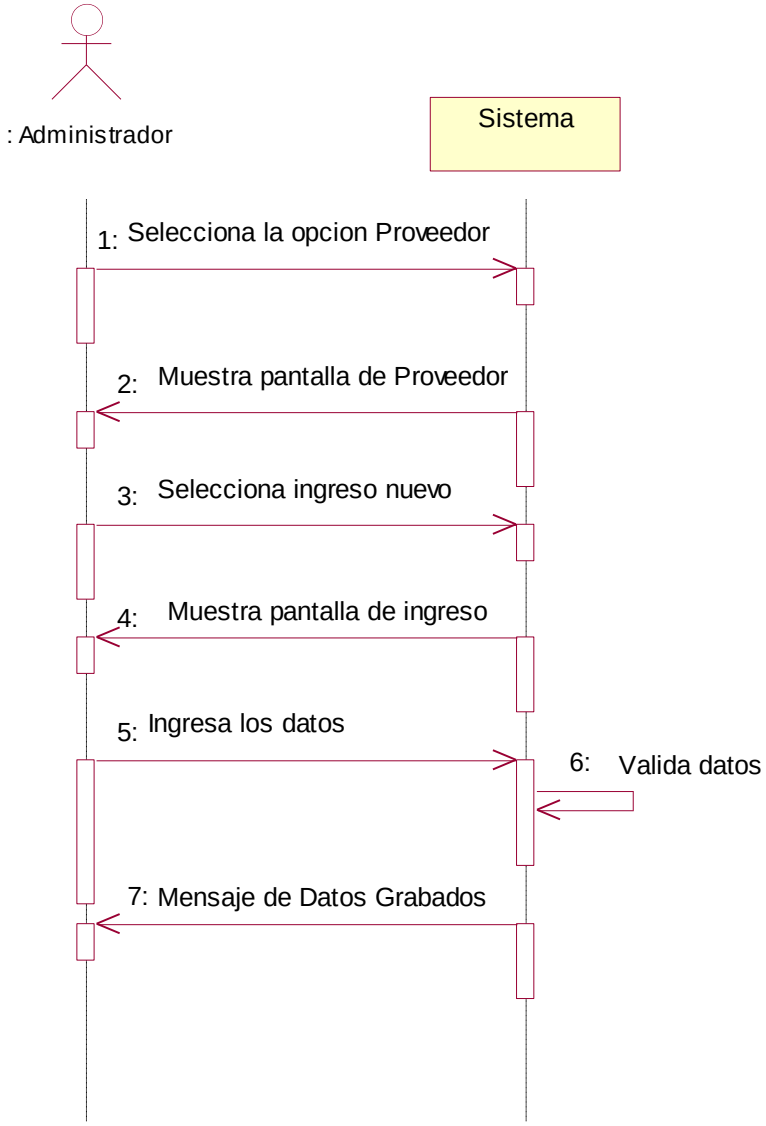

Figura 4.55 Diagrama de Secuencia Insertar Proveedor

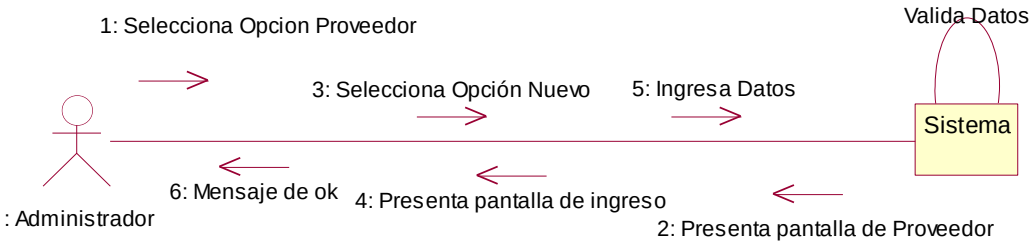

Figura 4.56 Diagrama de Colaboración Inserta Proveedor

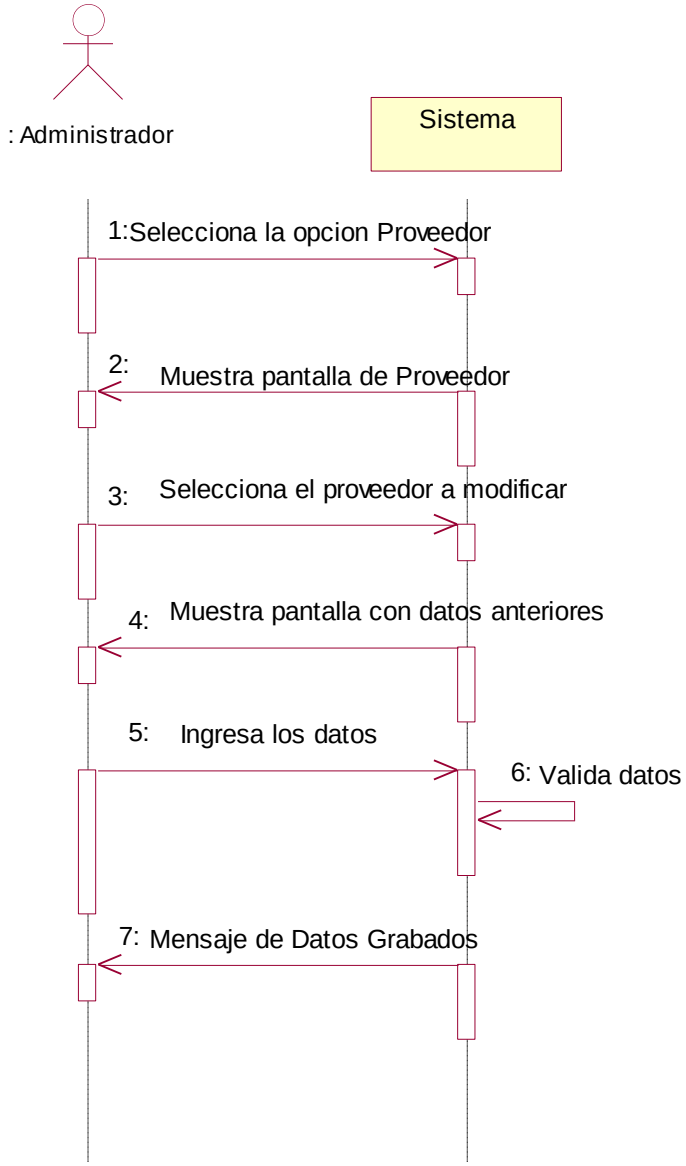

Figura 4.57 Diagrama de Secuencia Modifica Proveedor

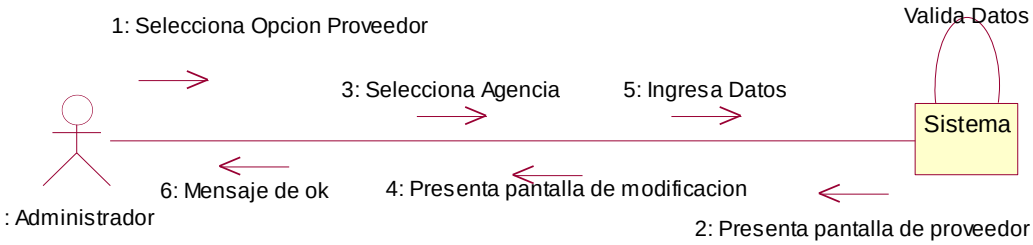

Figura 4.58 Diagrama de Colaboración Modifica Proveedor

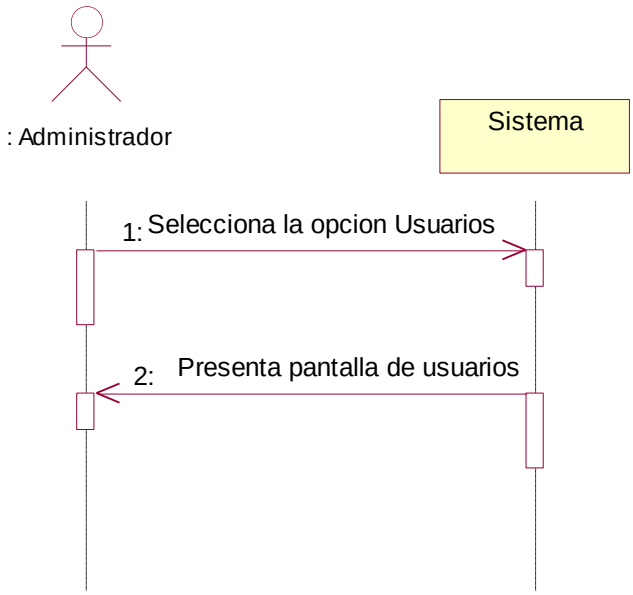

Figura 4.59 Diagrama de Secuencia Consulta de Usuarios

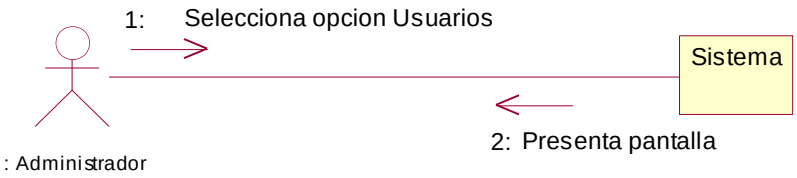

Figura 4.60 Diagrama de Colaboración Consulta de Usuarios

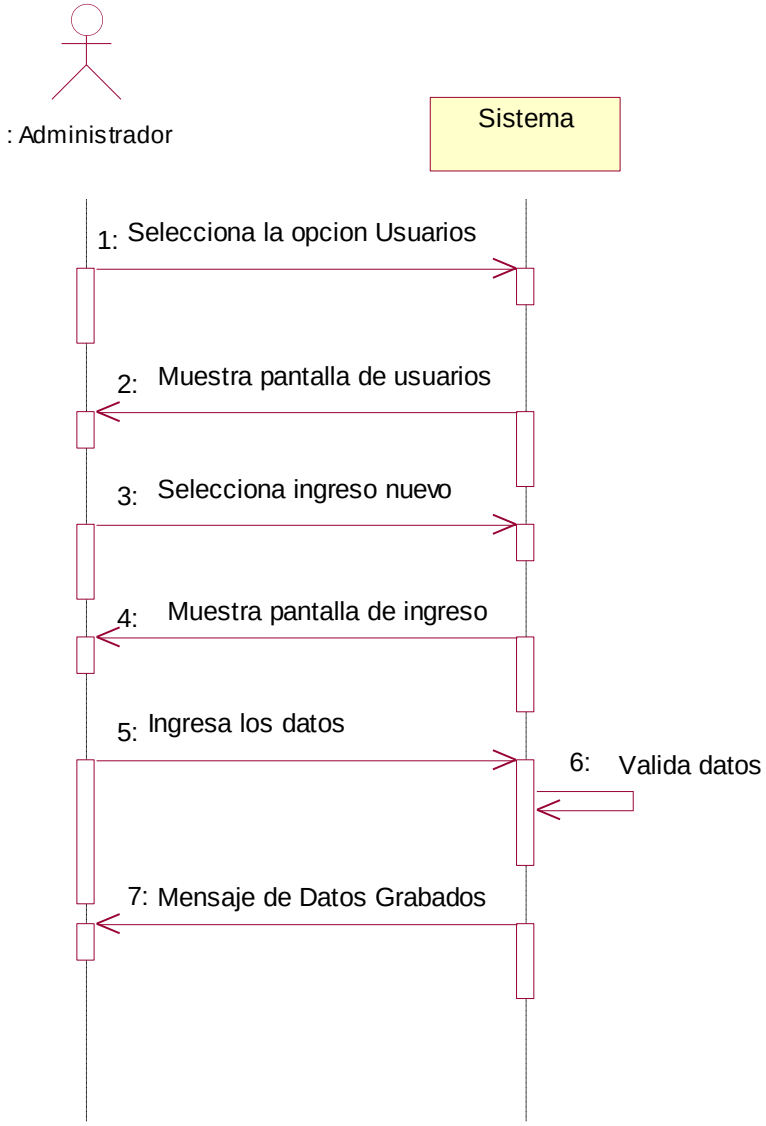

Figura 4.61 Diagrama de Secuencia Insertar Usuarios

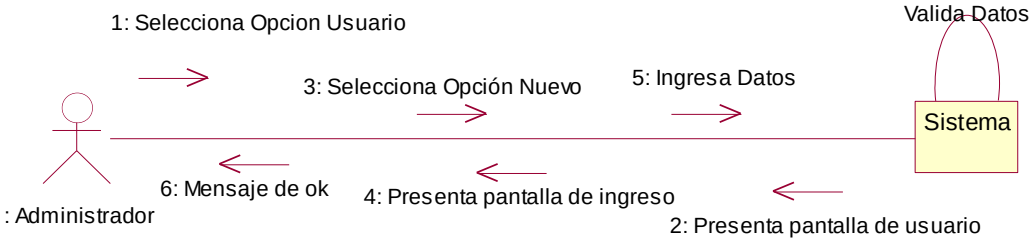

Figura 4.62 Diagrama de Colaboración Inserta Usuarios

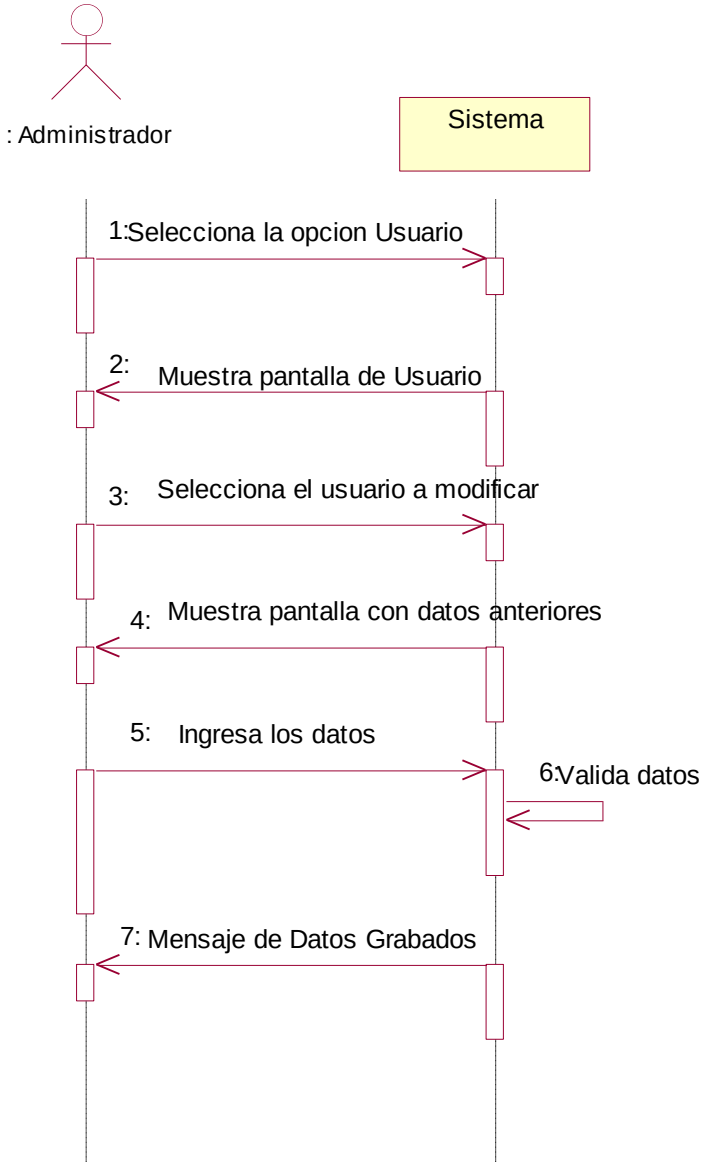

Figura 4.63 Diagrama de Secuencia Modifica Usuario

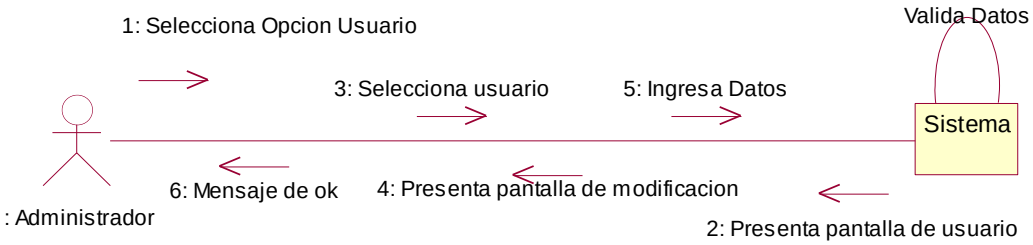

Figura 4.64 Diagrama de Colaboración Modifica Usuario

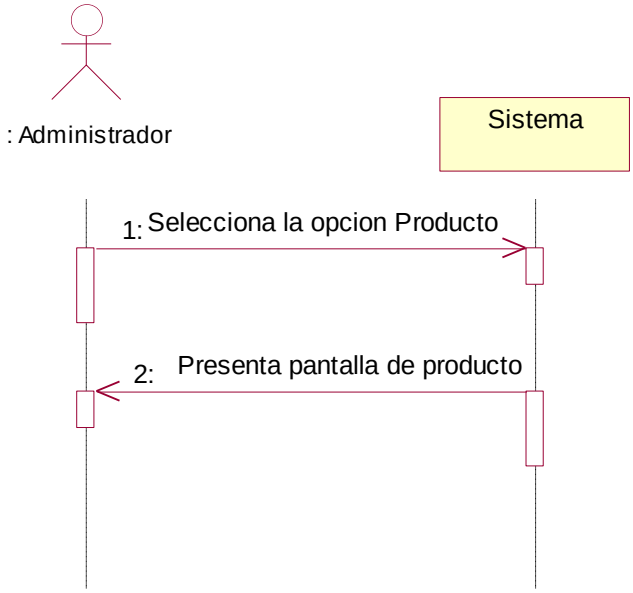

Figura 4.65 Diagrama de Secuencia Consulta de Productos

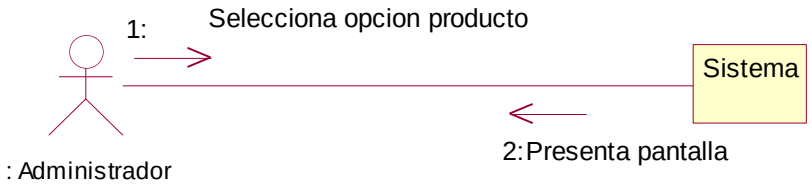

Figura 4.66 Diagrama de Colaboración Consulta de Productos

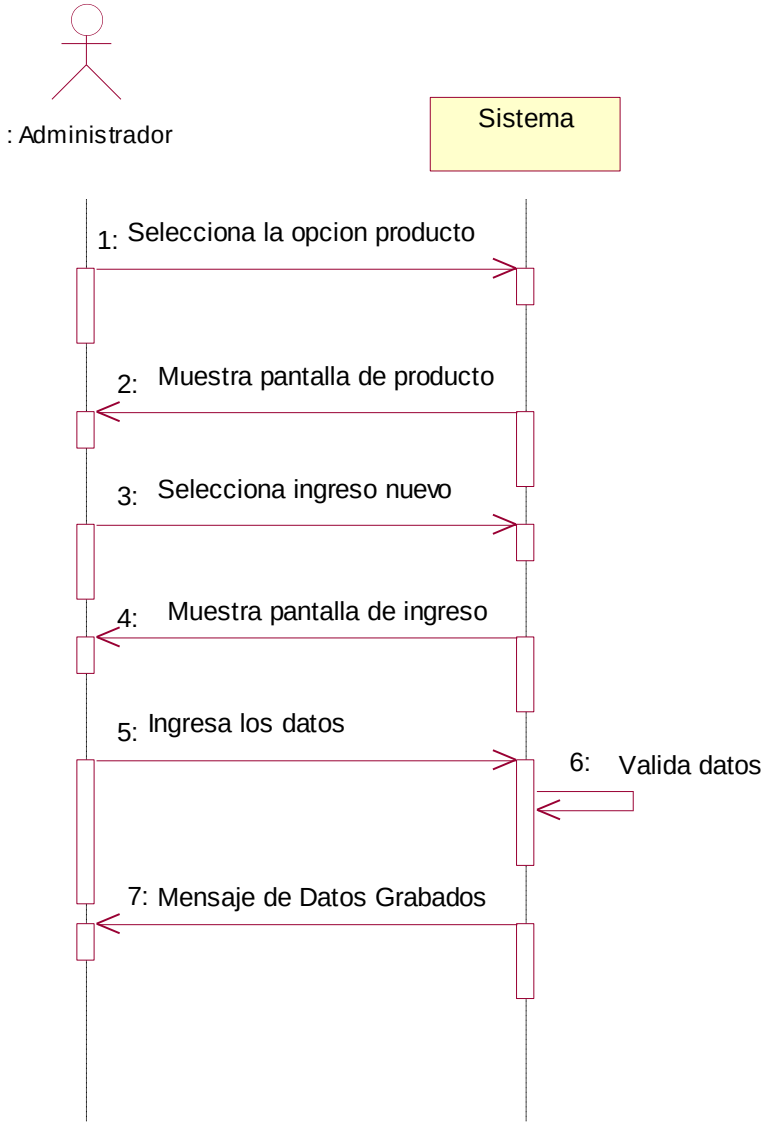

Figura 4.67 Diagrama de Secuencia Insertar Productos

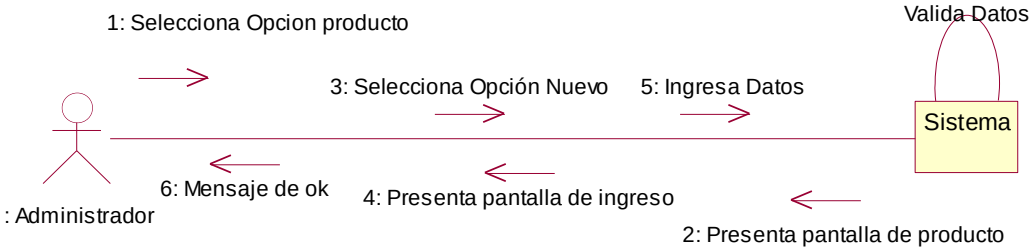

Figura 4.68 Diagrama de Colaboración Inserta Productos

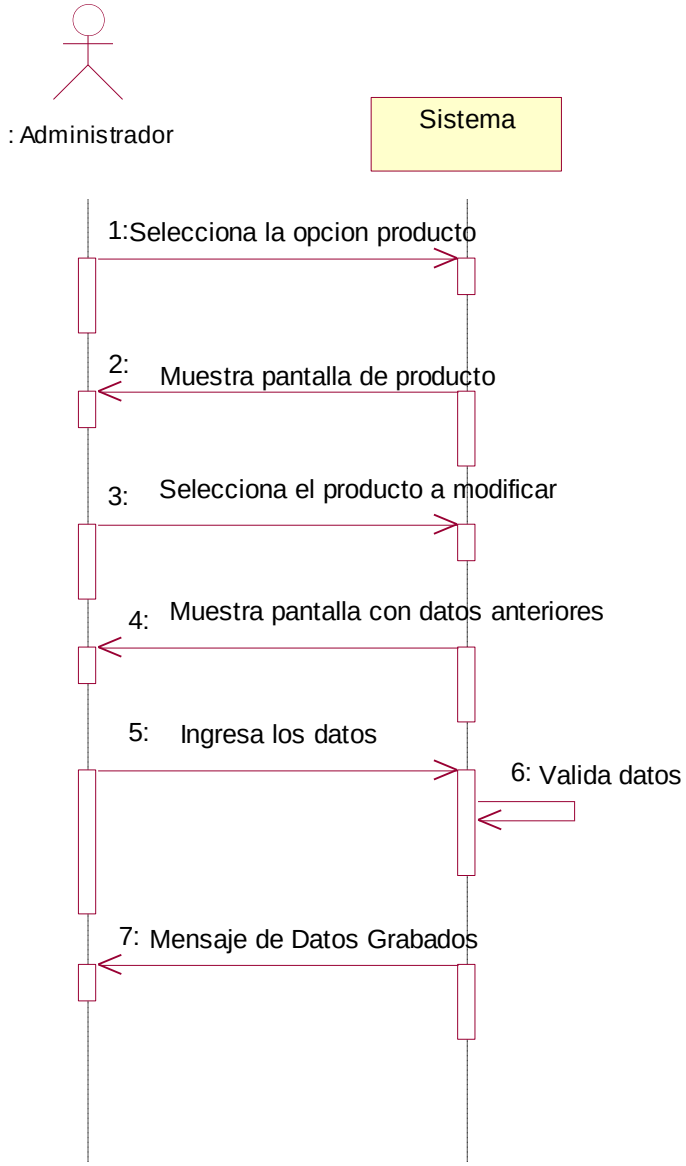

Figura 4.69 Diagrama de Secuencia Modifica Producto

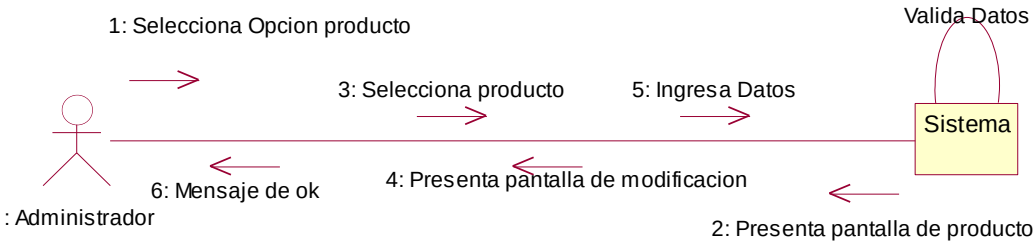

Figura 4.70 Diagrama de Colaboración Modifica Producto

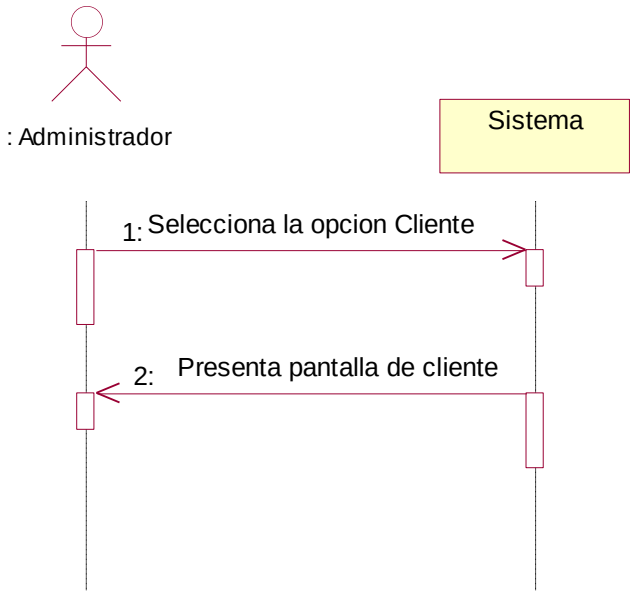

Figura 4.71 Diagrama de Secuencia Consulta de Clientes

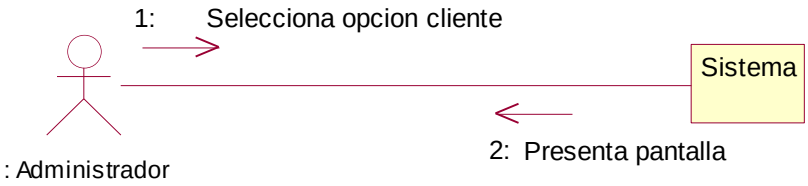

Figura 4.72 Diagrama de Colaboración Consulta de Clientes

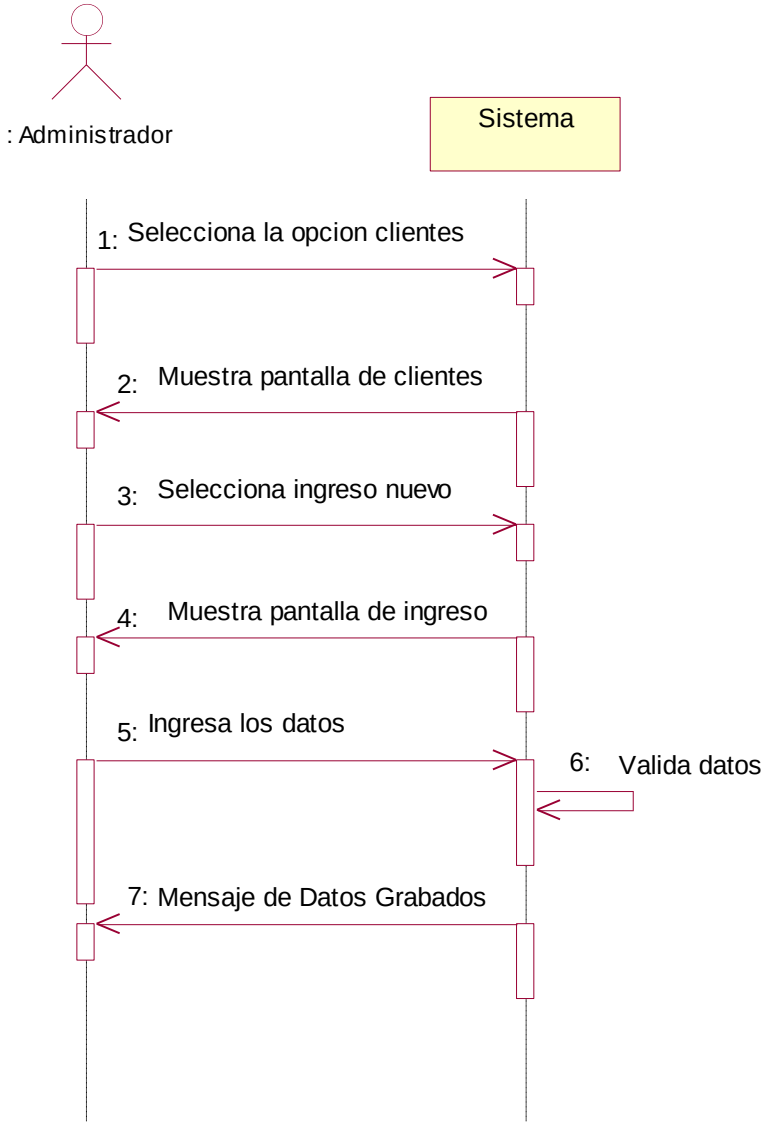

Figura 4.73 Diagrama de Secuencia Insertar Cliente

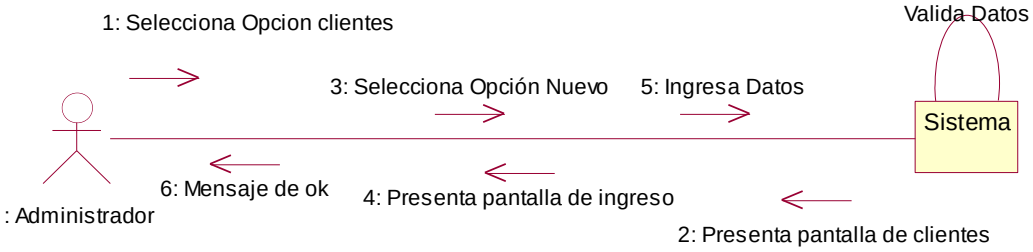

Figura 4.74 Diagrama de Colaboración Inserta Cliente

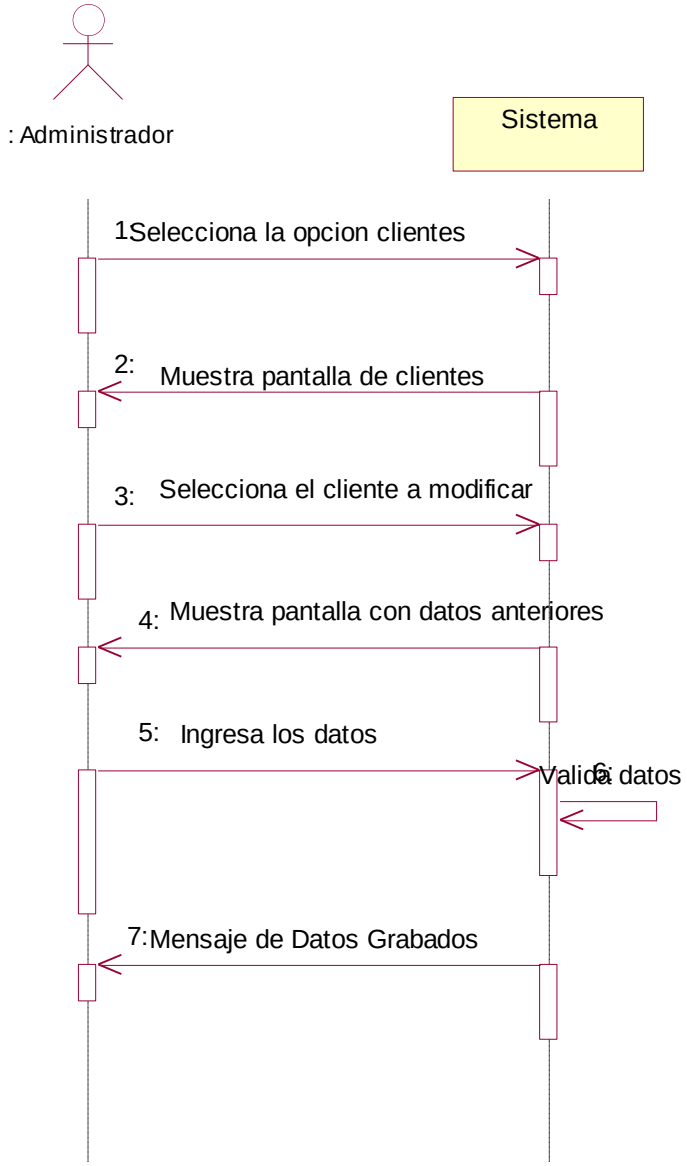

Figura 4.75 Diagrama de Secuencia Modifica Cliente

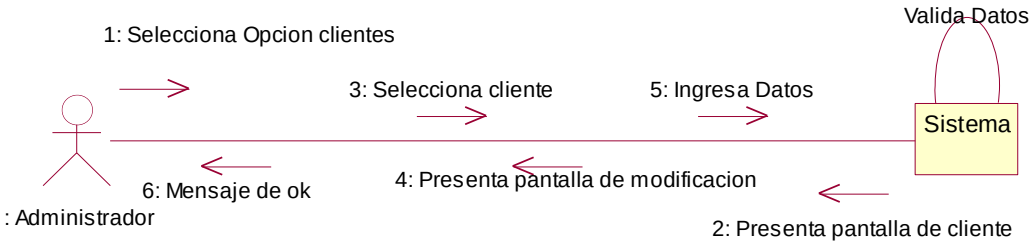

Figura 4.76 Diagrama de Colaboración Modifica Cliente

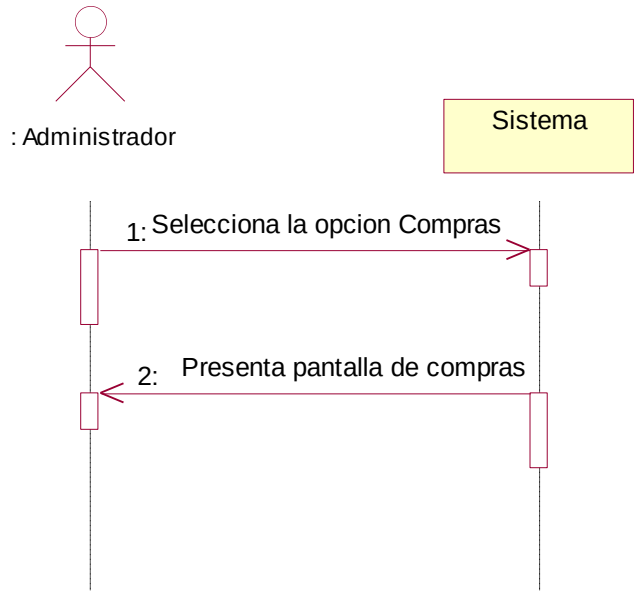

Figura 4.77 Diagrama de Secuencia Consulta de Compras

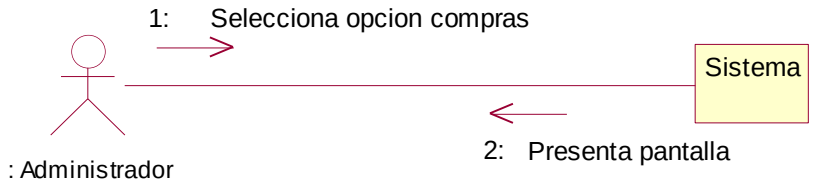

Figura 4.78 Diagrama de Colaboración Consulta de Compras

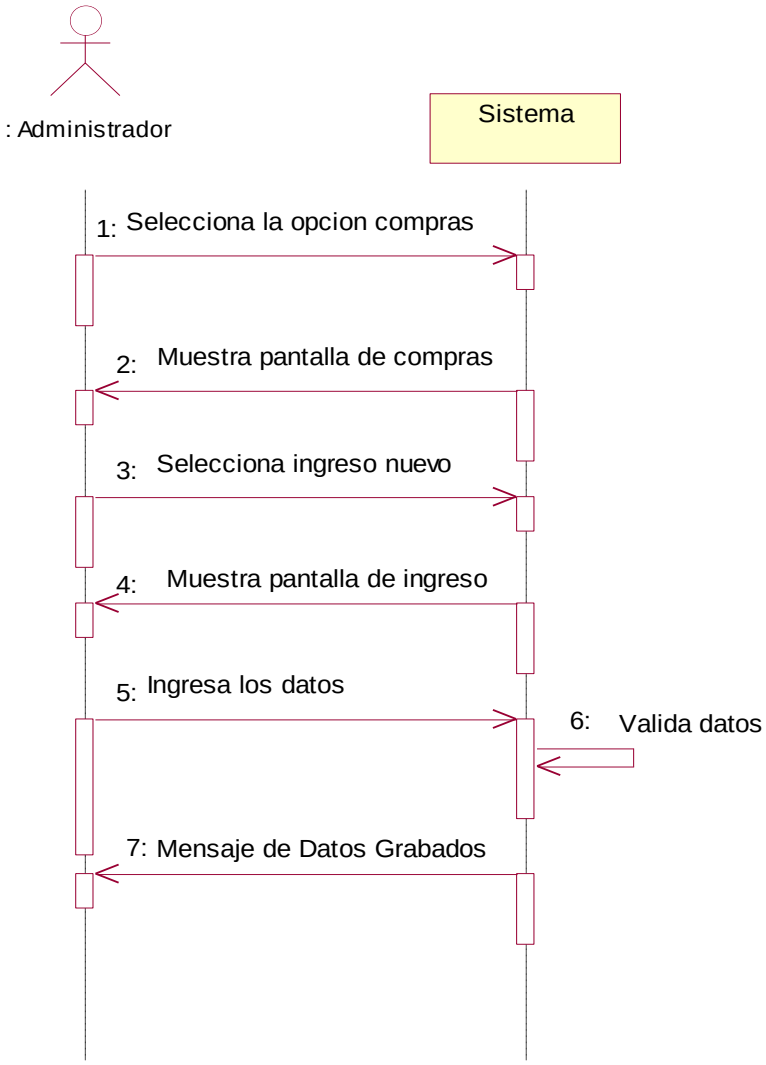

Figura 4.79 Diagrama de Secuencia Insertar Compra

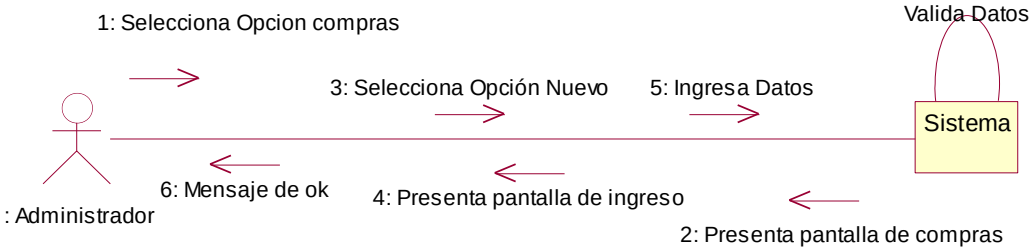

Figura 4.80 Diagrama de Colaboración Inserta Compra

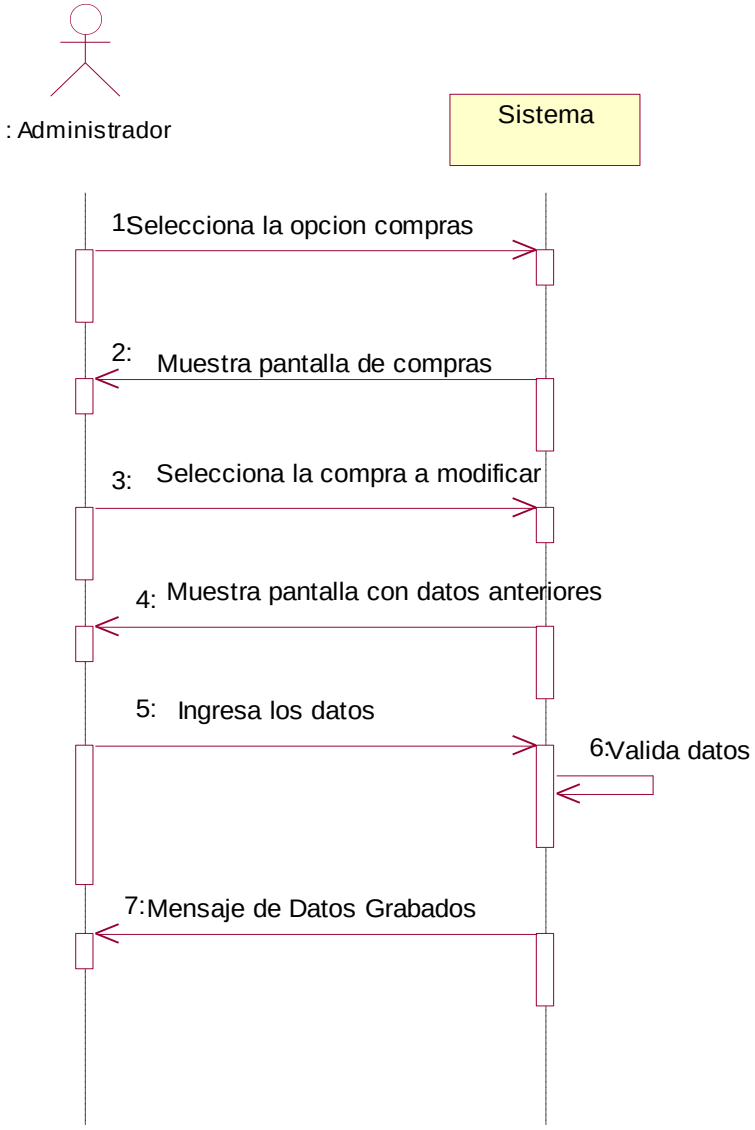

Figura 4.81 Diagrama de Secuencia Modifica Compra

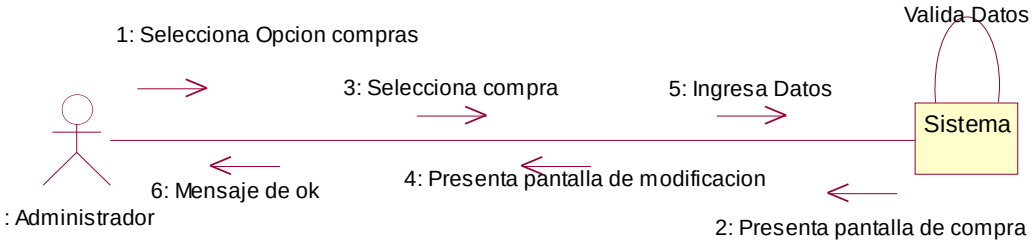

Figura 4.82 Diagrama de Colaboración Modifica Compra

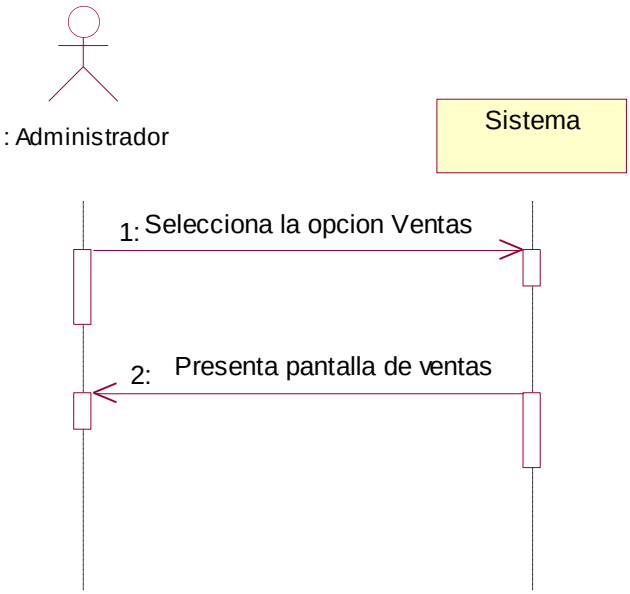

Figura 4.83 Diagrama de Secuencia Consulta de Ventas

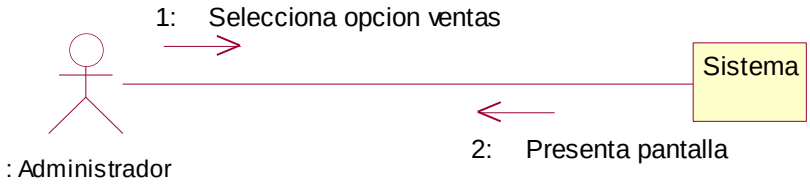

Figura 4.84 Diagrama de Colaboración Consulta de Ventas

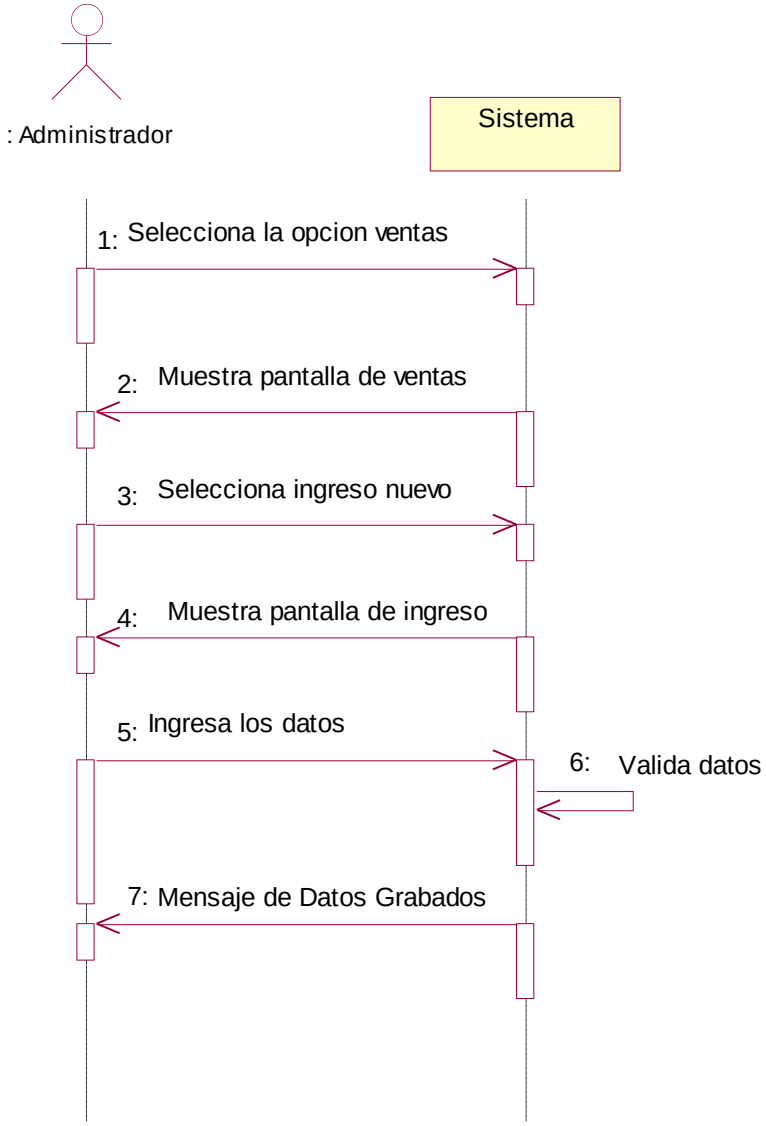

Figura 4.85 Diagrama de Secuencia Insertar Venta

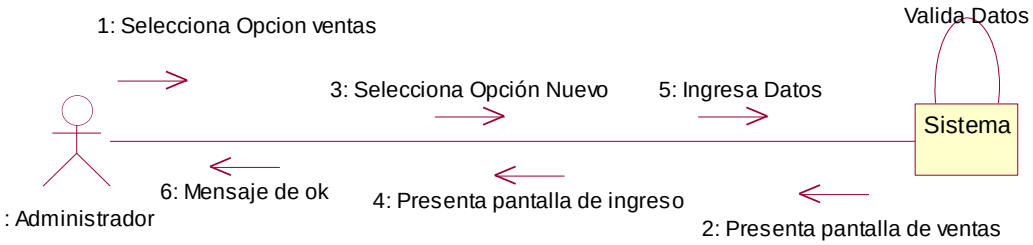

Figura 4.86 Diagrama de Colaboración Inserta Venta

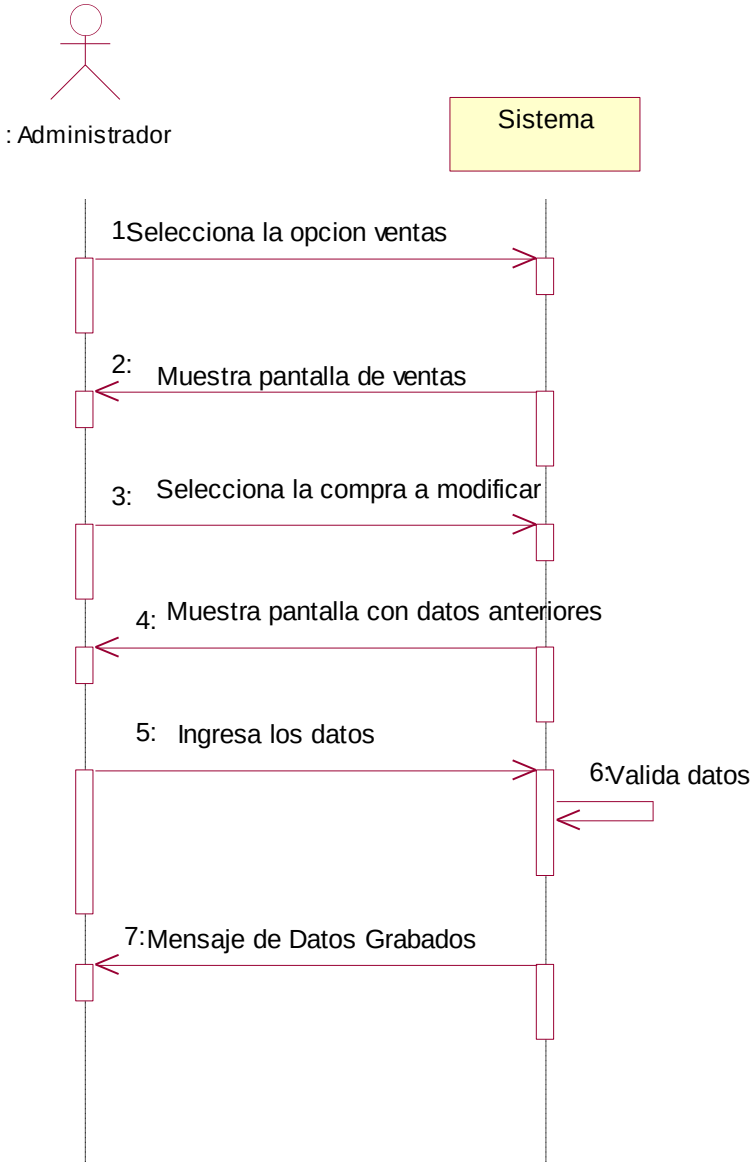

Figura 4.87 Diagrama de Secuencia Modifica Venta

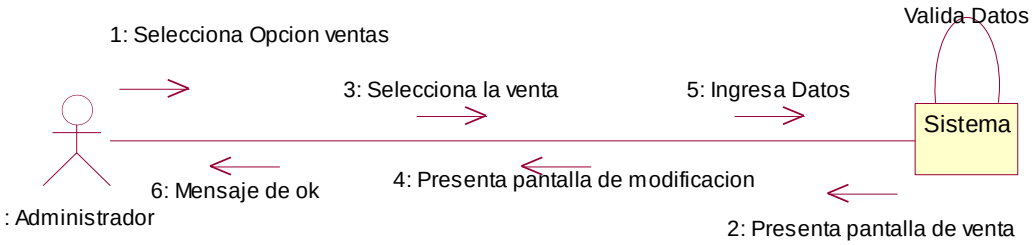

Figura 4.88 Diagrama de Colaboración Modifica Venta

4.5. FASE DE ELABORACION

4.5.1. MODELO DE ANALISIS Y DISEÑO

En este punto se explicara los nombres y tipos de datos de cada una de las tablas que se utilizaran en el sistema.

SCV_AGENCIA

Descripción: Tabla que almacena las agencias.

Campos:

Campo	Tipo de dato	Descripción
SCVAGEN_CODIGO	varchar (10)	Código de la Agencia
SCVAGEN_NOMBRE	Varchar (30)	Nombre de la Agencia
SCVAGEN_DIRECCION	Varchar (50)	Dirección de la Agencia
SCVAGEN_TELEFONO	Varchar (50)	Teléfono de la Agencia

Claves:

Nombre	Tipo	Campo	Tabla Referencia
PK_SCV_AGENCIA	PRIMARY KEY	SCVAGEN_CODIGO	

SCV_CLIENTES

Descripción: Tabla que almacena los clientes de la empresa.

Campos:

Campo	Tipo de dato	Descripción
SCVCLIE_CODIGO	Varchar (10)	Código del Cliente
SCVCLIE_NOMBRE	Varchar (40)	Nombre del Cliente
SCVCLIE_APELLIDO	Varchar (40)	Apellido del Cliente
SCVCLIE_DIRECCION	Varchar (40)	Dirección del Cliente
SCVCLIE_TELEFONO	Varchar (10)	Teléfono del Cliente
SCVCLIE_TELEFONOOFICINA	Varchar (10)	Teléfono Oficina del Cliente
SCVCLIE_EMAIL	Varchar (20)	Email del Cliente
SCVCLIE_CATEGORIA	Varchar (1)	Categoría del Cliente

Claves:

Nombre	Tipo	Campo	Tabla Referencia
PK_SCV_CLIENTES	PRIMARY KEY	SCVCLIE_CODIGO	

SCV_TIPOPRODUCTO

Descripción: Tabla que almacena los tipos de productos.

Campos:

Campo	Tipo de dato	Descripción

SCVTIPR_CODIGO	Varchar (10)	Código del Tipo de Producto
SCVTIPR_NOMBRE	Varchar (30)	Nombre del Tipo de Producto
SCVTIPR_DESCRIPCION	Varchar (30)	Descripción del Tipo de Producto

Claves:

Nombre	Tipo	Campo	Tabla Referencia
PK_SCV_TIPOPRODUCTO	PRIMARY KEY	SCVTIPR_CODIGO	

SCV_PRODUCTOS

Descripción: Tabla que almacena los productos que tiene la empresa.

Campos:

Campo	Tipo de dato	Descripción
SCVPROD_CODIGO	Varchar (10)	Código del Producto
SCVTIPR_CODIGO	Varchar (10)	Tipo del Producto
SCVPROD_NOMBRE	Varchar (20)	Nombre del Producto
SCVPROD_MARCA	Varchar (20)	Marca del Producto
SCVPROD_PRECIO	float	Precio del Producto

Claves:

Nombre	Tipo	Campo	Tabla Referencia
PK_SCV_PRODUCTOS	PRIMARY KEY	SCVPROD_CODIGO	
TIPO_PRODUCTO_FK		SCVTIPR_CODIGO	SCV_TIPOPRODUCTO

SCV_PROVEEDORES

Descripción: Tabla que almacena los proveedores.

Campos:

Campo	Tipo de dato	Descripción
SCVPROV_CODIGO	Varchar (10)	Código del Proveedor
SCVPROV_NOMRE	Varchar (40)	Nombre del Proveedor
SCVPROV_DIRECCION	Varchar (40)	Dirección del Proveedor
SCVPROV_TELEFONO	Varchar (10)	Teléfono del Proveedor
SCVPROV_EMAIL	Varchar (20)	E-mail del Proveedor
SCVPROV_REPRESENTANTE	Varchar (40)	Representante del Proveedor
SCVPROV_RUC	int	RUC del Proveedor

Claves:

Nombre	Tipo	Campo	Tabla Referencia
PK_SCV_PROVEEDORES	PRIMARY KEY	SCVPROV_CODIGO	

SCV_USUARIOS

Descripción: Tabla que almacena los usuarios del sistema.

Campos:

Campo	Tipo de dato	Descripción
SCVUSUA_ID	Varchar (10)	identificación del Usuario
SCVAGEN_CODIGO	Varchar (10)	Código del Usuario
SCVUSUA_NOMBRE	Varchar (40)	Nombre del Usuario
SCVUSUA_APELLIDO	Varchar (40)	Apellido del Usuario

SCVUSUA_LOGIN	Varchar (20)	Login del Usuario
SCVUSUA_PASSWORD	Varchar (20)	Password del Usuario
SCVUSUA_EMAIL	Varchar (20)	e-mail del Usuario

Claves:

Nombre	Tipo	Campo	Tabla Referencia
PERFIL_FK	PRIMARY KEY	SCVAGEN_CODIGO	SCV_AGENCIA
PK_SCV_USUARIOS		SCVUSUA_ID	

SCV_FACTURACOMPRA

Descripción: Tabla que almacena las facturas de compra.

Campos:

Campo	Tipo de dato	Descripción
SCVFACO_NROFACTURA	numeric	Número de la Factura de Compra
SCVPROV_CODIGO	Varchar (10)	Código del Proveedor
SCVFACO_CODIGOSUCURSAL	Varchar (10)	Código de la Agencia
SCVFACO_DESCUENTO	float	Descuento

SCVFACO_IVA	decimal	IVA
SCVFACO_SUBTOTAL	decimal	Subtotal
SCVFACO_FORMADEPAGO	int	Forma de Pago
SCVFACO_FECHA	datetime	Fecha de la Factura de Compra

Claves:

Nombre	Tipo	Campo	Tabla Referencia
COMPRA_FK	PRIMARY	SCVPROV_CODIGO	SCV_PROVEEDORES
PK_SCV_FACTURACOMPRA	KEY	SCVFACO_NROFACTURA	

SCV_DETALLECOMPRA

Descripción: Tabla que almacena el detalle de la Factura de Compra.

Campos:

Campo	Tipo de dato	Descripción
SCVPROD_CODIGO	Varchar (10)	Código del Producto
SCVFACO_NROFACTURA	numeric	Número de la Factura
SCVDFAC_CANTIDAD	int	Cantidad
SCVDFAC_VALORUNITARIO	decimal	Valor Unitario

Claves:

Nombre	Tipo	Campo	Tabla Referencia
PRODUCTO_COMPRA_FK SCV_DETALLECOMPRA_PK	PRIMARY KEY	SCVPROD_CODIGO SCVFACO_NROFACTURA	SCV_PRODUCTOS SCV_FACTURACOMPRA

SCV_ FACTURAVENTA

Descripción: Tabla que almacena las facturas de ventas.

Campos:

Campo	Tipo de dato	Descripción
SCVFAVE_NROFACTURA	numeric	Número de la Factura de Venta
SCVCLIE_CODIGO	Varchar (10)	Código del Cliente
SCVFAVE_CODIGOSUCURSAL	Varchar (10)	Código de la Agencia
SCVFAVE_DESCUENTO	float	Descuento
SCVFAVE_IVA	decimal	IVA
SCVFAVE_SUBTOTAL	decimal	Subtotal

SCVFAVE_FORMADEPAGO	int	Forma de Pago
SCVFAVE_FECHA	datetime	Fecha de la Factura de Compra

Claves:

Nombre	Tipo	Campo	Tabla Referencia
PK_SCV_FACTURAVENTA VENTA_FK	PRIMARY KEY	SCVFAVE_NROFACTURA SCVCLIE_CODIGO	SCV_CLIENTES

SCV _DETALLEFACTURAVENTA

Descripción: Tabla que almacena el detalle de la Factura de Venta.

Campos:

Campo	Tipo de dato	Descripción
SCVPROD_CODIGO	Varchar (10)	Código del Producto
SCVFAVE_NROFACTURA	numeric	Número de la Factura
SCVDFAV_CANTIDAD	int	Cantidad
SCVDFAV_VALORUNITARIO	decimal	Valor Unitario

Claves:

Nombre	Tipo	Campo	Tabla Referencia

Nombre	Tipo	Campo	Tabla Referencia
PRODUCTO_VENTA_FK SCV_DETALLEFACTURAVENTA_PK	PRIMARY KEY	SCVPROD_CODIGO SCVFACO_NROFACTURA	SCVPROD_CODIGO SCVFAVE_NROFACTURA

SCV_KARDEX

Descripción: Tabla que almacena el stock de los productos.

Campos:

Campo	Tipo de dato	Descripción
SCVPROD_CODIGO	Varchar (10)	Código del Producto
SCVAGEN_CODIGO	Varchar (10)	Código de la Agencia
SCVKARD_STOCK	int	Stock del Producto

Claves:

Nombre	Tipo	Campo	Tabla Referencia
AGENCIAKARDEX_FK PRODUCTOSKARDEX_FK	PRIMARY KEY	SCVAGEN_CODIGO SCVPROD_CODIGO	SCV_AGENCIA SCV_PRODUCTOS

4.6. PRUEBAS

4.6.1. INTRODUCCION

Propósito

El propósito del Plan de Pruebas es recabar toda la información necesaria para plantear y controlar las pruebas.

Alcance

El Plan de Pruebas realiza la comprobación como la funcionalidad, utilidad, fiabilidad las mismas que serán dirigidas por este plan de prueba.

4.6.1.1. PERSONAS A LA QUE SE DIRIJE EL PLAN

Este Plan de Pruebas esta dirigido exclusivamente para las personas encargadas de la verificación funcional del sistema.

4.6.1.2. PREPARACION Y PRUEBAS PLANEADAS

Lo que se presenta a continuación, permitirá determinar para cada requisito la característica a ser probada y los tipos de prueba que se emplearán.

Se ha diseñados un conjunto de pruebas para comprobar el cumplimiento de las especificaciones de requisitos.

4.6.1.3. INGRESO AL APLICATIVO

Ingresar a la aplicación con usuario previamente creado, para comprobar la conexión de RMI y verificar sus funciones.

4.6.1.4. GESTIÓN DE AGENCIAS

Crea Agencias

Modifica Agencias

Consulta Agencias

4.6.1.5. GESTIÓN DE PROVEEDORES

Crea Proveedor

Modifica Proveedor

Consulta Proveedores

4.6.1.6. GESTIÓN DE CLIENTES

Crea Cliente

Modifica Cliente

Consulta Clientes

4.6.1.7. GESTIÓN DE USUARIOS

Crea Usuario

Modifica Usuario

Consulta Usuarios

4.6.1.8. GESTIÓN DE PRODUCTOS

Crea Producto

Modifica Producto

Consulta Productos

4.6.1.9. GESTIÓN DE ADQUISICIONES

Crea Pedido

Modifica Pedido

Consulta Pedidos

4.6.1.10. GESTIÓN DE VENTAS

Crea Venta
Modifica Venta
Consulta Ventas

4.6.1.11. GESTIÓN DE CONSULTAS

Se verificara la ejecución de RMI para las consultas.

4.6.1.12. RECURSOS REQUERIDOS

Hardware y Software
A continuación se mostrara los recursos del sistema para realizar el Plan de Pruebas.

Servidor Base Datos
Servidor para instalar la aplicación (debe estar instalado el servidor Tomcat)
Red
Cliente

CAPÍTULO V

CONCLUSIONES Y RECOMENDACIONES

5.1. CONCLUSIONES

Al finalizar el presente trabajo de investigación podemos concluir lo siguiente:

La utilización de objetos en el desarrollo de sistemas distribuidos, presenta varias ventajas que permiten ocultar las dificultades inherentes a la distribución en niveles de abstracción inferiores, dejando así a los programadores la tarea de resolver únicamente su problema en particular.

La invocación remota de métodos en Java parte del hecho de correr sobre una plataforma homogénea. Está diseñada para tomar ventaja de esta característica, lo que le permite presentar propiedades que otros modelos de objetos como CORBA no poseen. Ejemplo de esto lo tenemos en la capacidad que tiene RMI de migrar dinámicamente a las implantaciones de los objetos, lo que le puede permitir a un cliente enviar un objeto para que se ejecute en una máquina con mayor poder de cómputo. Por otro lado, RMI posee todas las características de seguridad que hereda de la plataforma Java misma. Pero a diferencia de CORBA, que posee una arquitectura que proporciona independencia del lenguaje de programación, RMI está diseñada exclusivamente para Java.

Por último, podemos mencionar que Java RMI se utiliza como infraestructura fundamental de otras tecnologías Java sumamente importantes, como los Enterprise JavaBeans y JINI.

5.2. RECOMENDACIONES

Al finalizar el presente trabajo de investigación recomendamos lo siguiente:

Utilizar páginas jsp porque son mas entendibles y permiten la modificación rápida.

RMI actualmente posee un muy primitivo servicio de nombres (que no es persistente).

Que cada ves que se modifique las funciones del servidor hay que generar unas clases "especiales" que se incorporan una al cliente (Stub), y otra al servidor (Skel), para lo cual hay que ejecutar el comando rmic Exportado que las produce automáticamente después de haber compilado el objeto exportado.

Sistema de Compra y Venta por estar basado en la plataforma Java, la cual es una plataforma en constante desarrollo, puede estar a la par introduciendo características heredadas de la plataforma o usando nuevos paquetes y tecnologías como Jini, EJB, XML y más.

Tener cuidado de NO dejar puertos ocupados con el rmiregistry

CONTENIDO

REFERENCIAS BIBLIOGRAFICAS

- **RMI Specification.**

http://web2.java.sun.com/products/jdk/1.2/docs/guide/rmi/spec/rmiTOC.doc.html

- **JavaTM 2 Platform, Standard Edition, v1.2.2 API Specification.**

http://java.sun.com/products//jdk/1.2/docs/api/

- **Getting Started Using RMI.**

http://web2.java.sun.com/products/jdk/1.2/docs/guide/rmi/getstart.doc.html

- **Oracle TopLink Developer's Guide 10g Release 3 (10.1.3)**

http://www.oracle.com/technology/products/ias/toplink/doc/1013/main/_html/sesdef004.htm

- **Tutorial de java**

http://www.cica.es/formacion/JavaTut/

- **Introducción a UML**

http://www.programacion.com/Tutoriales/UML

- **UML Resource Center. Rational Software.**

http://www.rational.com/uml/

- **Manual de Jsp**

http://www.desarrolloweb.com/articulos/831.php?manual=15

- **Products & Technologies. Java Technology. 2005**

http://www.java.sun.com

Anexo 1

Cd que contiene lo siguiente:

Archivo de la tesis: "DESARROLLO DE APLICACIONES DISTRIBUIDAS BAJO INVOCACION DE METODOS REMOTOS EN ENTORNO TCP-IP".

- Manual del Sistema de Compra y Venta.

Anexo 2.1

GLOSARIO

RMI (Remote Method Invocation).- Método de Invocación Remota.

JSP.- Página HTML con código JSP entremezclado en el documento.

DFD's.- Diagrama de Flujo de datos.

POO.- Paradigma Orientado a Objetos.

AOO.- Análisis Orientado a Objetos.

UML.- Lenguaje Modeling Unified, Lenguaje de Modelado Unificado.

JVM.- entorno de ejecución de instrucciones que genera un compilador java.
Objetos locales: son objetos que se ejecutan en un host determinado.
Objetos remotos.- objetos que se ejecutan en todos los demás host.
ORB Object Request Broker.- Intermediario de Solicitud de Objetos. Conexión de
objetos distribuidos.

IIOP.- Internet InterORB Protocol.

RPC.- Remote Procedure Call. Llamada a Procedimientos Remotos.

JRMP JAVA Remote Method Protocol.- RMI usa el Protocolo de Método Remoto de Java (JRMP) para la comunicación de los objetos remotos de Java.

Latacunga, Julio 2006

_____

Sr. Wilson Francisco Zambrano Serrano
C.I. 0502655129

_____

Ing. José Luis Carrillo
COORDINADOR DE LA CARRERA DE SISTEMAS E INFORMATICA

_____

Dr. Rodrigo Vaca
SECRETARIO ACADEMICO