UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE INGENIERÍA INDUSTRIAL Y DE SISTEMAS

"TRANSMISIÓN DE DATOS VÍA TCP/IP EN TIEMPO REAL "

TESIS

Para optar por el título profesional de :

INGENIERO DE SISTEMAS

Christian Alfredo Perales Paz

Fritz Manuel Villajuán Guzmán

Lima -Perú
2003

INDICE

Pág.

INDICE DE GRAFICOS

Pág.

DESCRIPTORES TEMÁTICOS

- ***TIEMPO REAL***
- ***COMUNICACIONES***
- ***PROTOCOLO TCP/IP***
- ***INTERNET***
- ***JAVA***
- ***MODELO DE TRES CAPAS***
- ***CLIENTE-SERVIDOR***
- ***SOCKETS***

RESUMEN EJECUTIVO

La automatización de procesos no ha sido una tarea sencilla de solucionar. Mas aún si se trata de automatizar procesos de transferencia de información, es decir, tratar de lograr que el usuario obtenga de manera automática la información y que la pueda visualizar. Actualmente esta automatización se trata de solventar con aplicaciones Web; pero este tipo de aplicaciones tiene limitaciones al momento de actualizarla de manera automática en el Navegador Web.

Existen componentes que son utilizados dentro de las páginas Web para mostrar ese comportamiento dinámico que requiere la actualización automática de la información; esos componentes se denominan Applets, pero estos componentes por lo general consumen una regular cantidad de recursos de memoria, además son peligrosos en cuanto a la vulnerabilidad en la seguridad de la PC del Cliente.

Ante este inconveniente, se buscó una alternativa a este tipo de aplicaciones. Por ello se desarrolló una aplicación propietaria que cubra la actualización automática de la información. Se buscaron precedentes al tipo de aplicación que se estaba pensando desarrollar.

En la Bolsa de Valores de Lima, para la sección de consultas trabajan con páginas HTML estáticas, que son generadas mediante un proceso periódico.

Antes de realizar cualquier tipo de estudio previo para la aplicación propuesta, se definieron los objetivos que cubrirá la nueva solución, de modo que podamos hacer frente a la solución actual.

Para nuestro caso de estudio se busca que los usuarios tengan la información en tiempo real, mientras que su consulta se efectúe de manera automática; para el caso que se efectúen inserciones ó actualizaciones de información, no se ha incidido en detalle sobre ese tema.

El problema principal de la aplicación Web, se debía al tiempo que demoraba la actualización de la información, la cual significa pérdida de oportunidad de negocio, es decir, existía una posible oportunidad de perder información nueva, este tiempo estimado de pérdida era aproximadamente de 5 minutos, no era simplemente la actualización de la información.

Para la etapa de desarrollo de la nueva solución se trató con una metodología que permita el desarrollo de la aplicación propuesta. Se tomaron en cuenta las etapas de análisis y diseño, especificación de procedimientos por cada capa, especificación de los requerimientos de hardware y software, implementación del prototipo y las pruebas del prototipo.

Para el análisis y diseño, se utilizaron las herramientas proporcionadas por el UML. Estos diagramas nos permiten mostrar las distintas vistas que tendrá nuestra aplicación, de tal manera que puedan ser perceptibles a cualquier persona que desee comprender el funcionamiento de la aplicación. Además se diseñaron los formatos de los mensajes tanto para el envío como para la recepción en cada uno de los módulos que conforman la aplicación.

En la arquitectura de la aplicación se definieron tres capas que son la Base de Datos, la aplicación Servidor y la aplicación Cliente. Para las últimas capas mencionadas se especificaron sus procedimientos, cada uno de ellos corresponden a lo especificado en los diagramas UML.

De acuerdo a la herramienta de desarrollo en que se implementa la aplicación, se contará con determinado hardware. Se tuvieron en cuenta aspectos de performance, seguridad y estabilidad que requería la aplicación, para que ésta no entre en conflictos. Se recurrió también al Sistema

Operativo Linux para que aloje al módulo de la aplicación Servidor. Mientras que los módulos Clientes han sido desarrollados para Windows.

Las pruebas que se realizaron para determinar el grado de eficiencia de la aplicación y sobre todo del comportamiento del flujo de la información terminaron con éxito, lográndose cubrir los objetivos planteados. Se efectuaron las pruebas dentro de la red corporativa de la Bolsa de Valores de Lima así como también dentro de una red local (consistente en una Laptop con SO Windows XP y una PC con SO Linux). Surgieron inconvenientes cuando las pruebas se realizaron dentro de la Bolsa de Valores de Lima, que se debieron a mantenimiento del servicio de red.

Las pruebas no solamente fueron objeto de determinar si la aplicación permitía el flujo de información de modo automático sino que también se tomaron en cuenta algunas variables que permitirán realizar una medición del nivel de eficiencia de la aplicación propuesta, para luego llevarlo a una evaluación conjunta con la aplicación actual (aplicación Web). Entre las variables que se tomaron en cuenta son:

- El Tiempo que demora el cliente en recibir el mensaje.
- La Performance.
- La Seguridad.

A partir de la evaluación de estas variables, y de su posterior comparación con la solución Web, se definen las conclusiones y las recomendaciones. Estas recomendaciones tratan de cubrir deficiencias no estipuladas en el diseño de la aplicación.

INTRODUCCIÓN

PROLOGO

En estos momentos en que la Internet se ha convertido en uno de los medios que provee de información, existiendo una variedad infinita de aplicaciones orientadas a la Web, se tratan de implementar mecanismos de transferencia de información. Las distancias entre los entes participantes de esta situación parecen ser solucionados con la Internet.

La administración de este tipo de aplicaciones Web trae consigo limitados recursos de monitoreo entre otros, ante las limitaciones de este tipo de aplicaciones se va en búsqueda de nuevas alternativas.

El presente documento justamente trata de establecer tanto las fortalezas como las debilidades entre la solución Web y la propuesta por nosotros, como solución propuesta se desarrolla bajo el protocolo TCP/IP de modo exclusivo tratando de cubrir las deficiencias de las aplicaciones Web.

Se tiene por objetivo realizar la investigación, estudio, desarrollo e implementación de un sistema de transferencia de datos vía TCP/IP en tiempo real. Este sistema se encarga de emitir como de recibir paquetes de datos previamente, codificados de acuerdo a las circunstancias, a través de la red (corporativo ó local). En este escrito, se presenta todo el desarrollo de un prototipo de este sistema, así como también de antecedentes de estos tipos de sistema, un caso ejemplo, planes de contingencia, excepciones y las conclusiones.

La transferencia de la información, es una actividad de todos los días. Existen varios mecanismos de cómo llevarlo a cabo, desde el medio oral hasta el uso de la más sofisticada tecnología. Actualmente existen diversas aplicaciones Web y ante la necesidad de buscar nuevos mecanismos de satisfacer la transferencia de datos, se ha determinado en muchos casos acudir a la plataforma Web como la solución más acorde a la actualidad.

Se ha tomado como punto en cuestión la trasferencia de datos en la red, utilizando estándares universales, en este caso el protocolo TCP/IP. Se resalta que todas las aplicaciones Web se encuentran enmarcadas dentro del protocolo TCP/IP, pero los medios ya son establecidos y son difíciles de personalizar a las necesidades requeridas, en cambio en las aplicaciones tradicionales se empieza desde cero. Desde la definición de estructuras de la codificación de datos hasta la presentación misma de los datos, ofreciendo de esta manera flexibilidad y mantenimiento.

En líneas generales el prototipo constituye una solución práctica y efectiva no solamente desde el punto de vista económico sino también desde el punto de vista del desarrollo aplicativo. Específicamente los contratiempos que se tuvieron en su desarrollo fueron de aspectos generales, y no particulares, vale decir en las dificultades que podrían tener el desarrollar en un entorno en Windows (Java, Visual C++).

TEMÁTICA DEL PROBLEMA

A medida que la masificación de computadoras se ha ido desarrollando, las redes de computadoras han ido adquiriendo una importancia en el ámbito mundial. Estas redes son muy beneficiosas para la comunicación entre las personas (Internet), la automatización de las instituciones y muchas otras aplicaciones.

El tema central de este documento es buscar alternativas en cuanto al tratamiento de la transferencia de datos, existen muchos mecanismos de cómo llevarlo a cabo. Actualmente aquello se trabaja dentro de un entorno Web, Internet es un medio universal en el que todo tipo de aplicaciones Web convive, pero en el tratamiento de la transferencia de datos, la Internet no surge como el medio más adecuado para desarrollarlo.

Por mencionar entre algunas situaciones; mientras que en una aplicación Web el usuario simplemente interactúa con la aplicación a través de un Navegador Web, que es el medio estándar para acceder a la Web, se realiza

la conexión TCP, la petición y la respuesta determinada; en cambio en una aplicación Servidor se permite el envío de más de una respuesta a una petición, siendo esto una de las características que nos permite este tipo de aplicaciones.

Además este Navegador Web ya viene establecido, ya viene implementado sin tener alguna posibilidad de otorgarle mantenimiento, en cambio con la solución aplicación Servidor se busca la flexibilidad y el mantenimiento; y quizás absorber la solución Web, esto se puede conseguir implementando un propio servicio Web donde se establecen nuestras propias reglas.

TOPICOS DEL PROBLEMA

1. ¿Se tiene que requerir la actualización de su data? ¿Porqué se tiene que refrescar la data? ¿La data tiene que llegar sin que el usuario lo pida? ¿Porqué es necesario que esté en tiempo real?

La actualización de la información es necesaria por el mismo tipo de información que se involucra, la información bursátil durante un día de Rueda de Bolsa, sufre muchas variaciones, estas tienen suma importancia para las personas que negocian con Acciones ó Bonos, así como para aquellas que están involucradas con la Economía Nacional e Internacional.

Cada cambio de información que sufre durante la Rueda de Bolsa y su aprovechamiento produce grandes movimientos de capital.

El asunto no es que el usuario desarrolle algún mecanismo para poder ver reflejada la información actualizada al instante, sino que esta actualización se lleve a cabo de manera instantánea con respecto al momento en que ha sido generado este cambio. Esta actualización automática otorga un carácter dinámico a la aplicación.

La información que se maneja en tiempo real, puede ser determinante dependiendo de la ejecución de acciones ú operaciones de personas relacionadas al rubro de la información, el hecho que tengan éxito será debido a las oportunas decisiones tomadas a partir del conocimiento de la información en tiempo real.

2. Por otro lado, considero que nuestra aplicación en tiempo real, es un importante alcance para la estimación de beneficios, es decir, está presentando un producto con más facilidades y desde el punto de vista económico, es más rentable, por lo que recibes más y pagas menos.

Además, al tratarse de un producto con un mejor servicio, crearía una tendencia al crecimiento de nuestra cartera de clientes, por lo que estaríamos hablando de un beneficio cualitativo, al demostrarse que este

crecimiento de la cartera de clientes es producido por una aplicación que ha dado mejores resultados, en pocas palabras mejor calidad.

Otra razón por la que se implantaría la aplicación, es que en el plano comercial una aplicación tiene un promedio de vida aproximado de 5 años, y una entidad tan importante como la Bolsa de Valores de Lima tiene que mejorar sus aplicaciones que brindan servicios para sus clientes, por un principio muy elemental, si a los clientes de la Bolsa de Valores les va bien, entonces como consecuencia, a la Bolsa de Valores también le irá bien, debido a que los clientes seguirán depositando su confianza en la misma, por lo que la Bolsa de Valores crecerá mucho en referencia al reconocimiento, que está tan disminuido en estos días.

Estos últimos puntos formarían parte de un valor agregado que se le daría al producto y en el que tenemos que pensar de manera implícita, ya que con relación a estos intangibles nosotros podemos definir un estudio de marketing para obtener resultados significativos, desde el punto de vista económico, como también para hablar de la eficiencia de la aplicación.

3. ¿Porqué utilizar TCP/IP? ¿Porqué se ha extendido su uso?

Primero se tiene que considerar que TCP/IP es un conjunto de protocolos y servicios, cada uno con una meta determinada, que permite la conectividad de un conjunto de computadoras en un entorno determinado.

Su generalización ó estandarización, tiene dos potentes argumentos, uno está dado por la difusión de las PC, que provoca la necesidad de conectar computadoras de diferentes sistemas operativos (Windows, Linux, Unix, Solaris, AS/400) y ubicaciones físicas, los cuales para poder comunicarse necesitaban tener un protocolo común, y el otro argumento es el increíble desarrollo de Internet, que se basa en este protocolo para efectuar su conectividad.

Este protocolo TCP/IP tiene un mecanismo de seguridad que consiste en el concepto del "*checksum*", que viene a ser un identificador de la longitud del paquete de información enviado de una computadora a otra. Si el paquete recibido no llega de manera completa entonces se genera una excepción de tipo I/O. Un ejemplo claro de esto ocurre cuando se ejecuta el comando "*ping [dirección IP]* " en una PC Windows, se detalla la cantidad de bytes enviadas al computador remoto y luego el número de iteraciones donde se indica el porcentaje de bytes recibidos en el computador remoto.

Existen otros medios como alternativa al TCP/IP tal como el X.25 que es una red que aún se encuentra en funcionamiento sobre todo para las entidades bancarias. La característica principal de este tipo de redes es que mantiene un excesivo control de flujo de información y de errores. Esta característica retarda la transmisión de la información, a diferencia del TCP/IP. Tomando en cuenta este aspecto se recomienda TCP/IP.

4. Ventajas de la solución Web que no son tomadas en cuenta para la evaluación de las soluciones

Cuando se trabaja con una solución Web, las páginas que se ven por parte del cliente constituyen un reflejo, es decir en la transmisión de datos, el reflejo de la imagen llega al usuario, lo cual constituye una desventaja para nuestro cliente, debido a que se trata de una imagen que no la puedes reutilizar, en cambio al tratarse de nuestra aplicación la información transmitida es reutilizable, ya que guarda los datos (no imágenes) y no consume muchos recursos por que no hace conexión a la base de datos (como la mayoría de páginas Web) si no que es una conexión de un socket a través de un "*pipe*" (tubería).

Una ventaja de la solución Web es que sus aplicaciones son fáciles de implementar, como es bien sabido este tipo de aplicaciones son utilizadas en entorno Windows; pero la aplicación que se ha desarrollado también es para dicho entorno, por lo tanto esta supuesta ventaja ya no es significativa.

Si los sistemas operativos fuesen diferentes es muy posible que la aplicación Web sea más ventajosa debido a que en todos los sistemas operativos existen Navegadores Web para ver las páginas, aunque el servidor es el que determina la performance en la que se desenvuelve el cliente. En líneas generales como todas las aplicaciones para nuestro caso se desarrollan en

el sistema operativo Windows, esta ventaja aparente que tiene la aplicación Web no es considerada tal.

El asunto de que muchas personas puedan acceder a la aplicación Web, se le considera como una ventaja de esta solución, pero a su vez a mayor cantidad de personas que la accedan puede ocasionar una saturación del servicio.

En cambio para nuestra aplicación el número de usuarios es controlado, me refiero que podemos saber quienes están interactuando, lo cual permite una mejor administración de las conexiones.

OPTICA DE LA INVESTIGACIÓN

El graduado considera que la implementación del prototipo, como medida de solución del convencional método basado en la aplicación Web, resulto muy versátil, en el sentido de que no solamente se logra la necesidad de transmisión de datos en tiempo real, sino que también supera los cuantificadores establecidos.

Si bien es cierto, que la diferencia de muestreo de resultados, es solamente de segundos de lo que se está hablando puede decir que para nuestro caso, estos segundos si resultan significativos en vista de las múltiples tomas de decisiones que podrían ejecutar el usuario si tuviera la información a su

alcance, es más el hecho de perder unos segundos podría resultar perjudicial en algunos casos.

La mayor ventaja de Internet es que es una herramienta que provee el acceso a una vasta cantidad de información a lo ancho del mundo, con lo que cualquiera tiene acceso a ella, y ya no es tan costosa como algunos años atrás.

En una aplicación Web se necesita de interacción entre el Navegador y el usuario para que la información se actualice, en cambio, en la solución propuesta la información fluye de manera automática (en tiempo real), ya que no necesita ninguna acción por parte del cliente para que se refleje esta información.

Las aplicaciones Web que necesitan mayores niveles de seguridad, requieren de componentes y estos necesitan de licencias que ocasionan un gasto significativo. En una aplicación Web se necesita un Servidor Seguro, que utilice SSL, y anotamos como punto importante que en un servidor seguro (HTTPS), la carga de la aplicación se torna más lenta, y esto incomoda mucho a los clientes. También hay que prevenir con que facilidad se implementa dicha aplicación, esto tomándolo desde un punto de vista del desarrollador de esa aplicación.

Por otro lado una aparente solución para las aplicaciones Web de mostrar información de manera dinámica es mediante un Applet que viene a ser un componente desarrollado en Java. Desde que un navegador Web esta capacitado para descargar Applets para ejecutarlo de manera local en diferentes maquinas, la seguridad es un punto critico. Los riesgos en la seguridad de ejecutar Applets se refieren a 3 aspectos:

- Modificación del sistema, la mayoría de los programas tienen la habilidad de modificar datos en tiempo de ejecución. Java incluye clases predefinidas con métodos que pueden borrar o modificar archivos, memoria y algunas veces finalizar procesos. En los casos más severos esta la intrusión al sistema mismo.

- Invasión de la privacidad, este tipo de ataque se refiere a ser público algún tipo de información confidencial, como algunos archivos que deben mantenerse privados. Por ejemplo en un sistema Unix si el atacante obtiene acceso al archivo 'etc/password' (el cual contiene todos los usuarios y las contraseñas encriptadas) puede emplear un programa para descifrar las contraseñas.

- Negación de un servicio, los ataques de esta naturaleza hacen que los recursos de la máquina no estén disponibles. Esto ocurre cuando un proceso utiliza más recursos de los que tiene permitidos esencialmente

deteniendo la máquina. Existen muchas categorías para un ataque de negación de un servicio, algunos ejemplos son:

1. Saturación completa del sistema de archivos.

2. Utilizando todos los punteros de archivos disponibles.

3. Almacenando toda la memoria del sistema.

4. Creando miles de ventanas negando efectivamente el acceso a la salida de la pantalla o la ventana que esta en cola.

5. Usando todos los ciclos de la maquina creando muchos procesos de alta prioridad.

Las aplicaciones Java Standalone no tienen las mismas implicancias de seguridad que los Applets. Los primeros son capaces de abrir archivos para su lectura y escritura, comunicarse con diferentes servicios, realizar diferentes tipos de conexiones. No ocurre así con los Applets.

Si un Applet es descargado a través de Internet, no se tiene permiso para las siguientes acciones:

- Lectura de archivos en el sistema del cliente.

- Escritura de archivos en el mismo sistema.
- Eliminación de archivos en cualquier nivel.
- Creación de directorios.
- Listado del contenido del directorio.
- Verificación de la existencia de algún archivo.
- Creación de alguna conexión de tipo TCP/IP.

CAPITULO I

OBJETIVOS

El principal punto que se trata en este documento es que la actualización de la información tenga carácter automático, ésta justamente es una limitación que se busca cubrir con nuestra solución. Con esto se trata de brindar al usuario oportunidades de negocio que las pondrá en práctica basándose en sus decisiones.

Un concepto que se trata además en gran parte del documento es TCP/IP, que viene a ser un protocolo estándar de comunicación. Las aplicaciones Web utilizan también TCP/IP y nosotros interactuamos con ellos mediante un Navegador Web, pero estos Navegadores ya se encuentran desarrollados sin oportunidad de personalizarlos en cuanto a funcionalidad.

El concepto TCP/IP lo utilizamos como una herramienta para lograr construir nuestra solución, con fines de tener nuestro propio Navegador ó Interfaz Cliente, pero que se pueda personalizar.

OBJETIVO PRINCIPAL

"Presentar la información en TIEMPO REAL"

OBJETIVOS COMPLEMENTARIOS

- Experimentar conceptos del protocolo TCP/IP, durante la implementación de una aplicación prototipo y sus respectivas pruebas.

- Desarrollar una particular e innovadora interface de comunicación siempre basado en el protocolo TCP/IP, distinto al servicio convencional Http, porque éste tiene su propia forma de conectarse y sus propios formatos de salida.

- Analizar y determinar la factibilidad sobre la base del criterio económico tanto de las Aplicaciones Web como de la Aplicación de Transferencia de Datos. Teniendo como variables de análisis la seguridad, el desarrollo de programas, costo de los equipos de hardware, las licencias de software, etc.

- Analizar el comportamiento de los tiempos de recepción, por parte de los clientes, de los mensajes enviados de la aplicación Servidor. Este comportamiento se mostrará mediante una interface gráfica, la información que se reflejará en estas interfaces serán por hora y por día.

CAPITULO II:

SOLUCION ACTUAL

2.1. ESCENARIO

Existen aplicaciones Cliente / Servidor que trabajan bajo el protocolo TCP/IP, por ejemplo en la Bolsa de Valores de Lima ya ha implementado un Sistema de Negociación Electrónica, que labora bajo el protocolo TCP/IP, esta aplicación permite la comunicación entre las diferentes Sociedades Agentes de Bolsa que realizan las operaciones bursátiles durante la Rueda de Bolsa.

El Sistema de Negociación Electrónica (Elex), implementado en agosto de 1995, es un mecanismo que se ha desarrollado para facilitar la realización de operaciones en la Bolsa de Valores de Lima.

Este sistema ha sido desarrollado bajo la estructura Cliente / Servidor que optimiza el manejo de los datos permitiendo al usuario interactuar, en tiempo real, a través de una computadora personal instalada en su oficina con el Servidor central de la Bolsa y, de este modo, realizar sus transacciones bursátiles diarias.

La configuración del Sistema Elex ha permitido el diseño de un mecanismo de negociación que posibilita la coexistencia de la negociación en piso (en Sala de Rueda) con la negociación remota a través de terminales, habiéndose establecido este mecanismo para el mercado accionario. Así las propuestas y operaciones con acciones inscritas en Bolsa pueden ser ingresadas al sistema, en forma simultánea, a través de un terminal de Elex ó a través de la Sala de Rueda utilizando papeletas, las cuales son registradas mediante un sistema de lectura óptica por escáner.

Las Operaciones de Reporte y la Negociación con Instrumentos de Deuda se realizan exclusivamente a través de la plataforma tecnológica Elex. Esto significa que no se puede realizar operaciones de este tipo en piso (Sala de Rueda).

Mediante este sistema se permite a los intermediarios bursátiles ingresar sus propuestas y negociar valores desde los terminales ubicados en sus oficinas y en la Sala de Rueda a través de los siguientes módulos:

- Módulo de Renta Variable.
- Módulo de Bonos y Letras Hipotecarias.
- Módulo de Reporte (Acciones e Instrumentos de Deuda).
- Módulo de Mercado de Dinero.

([1]) Entre las ventajas que trae consigo el Sistema Elex, se cuenta con los siguientes:

[1] Referencia - Página Web de La Bolsa de Valores de Lima

- FÁCIL MANEJO: El usuario puede ingresar y aplicar propuestas a firme de compra y venta, y/o consultar diversas tablas de información del sistema, utilizando solamente el Mouse ó Trackball. Asimismo, si el usuario lo desea, puede trasladarse por las distintas opciones del sistema a través del uso del teclado.

- USO PERSONALIZADO: El usuario puede configurar el sistema de acuerdo a sus requerimientos o necesidades de negociación o consulta, dentro de un ambiente Windows, el mismo que utiliza la más moderna tecnología diseñada para sistemas de este tipo por la versatilidad de su manejo. De esta manera la Bolsa de Valores cumple con las recomendaciones de estandarización y permite, además, la futura interconexión con los Sistemas de Negociación de otras Bolsas.

- INFORMACIÓN ACTUALIZADA: El sistema realiza automáticamente, y en tiempo real, la actualización de la información del mercado, como precios, propuestas operaciones realizadas, índices, entre otros, en todas las tablas del sistema.

- ESTRUCTURA CLIENTE / SERVIDOR: La estructura diseñada para este sistema aprovecha el empleo de múltiples terminales para administrar, distribuir y utilizar grandes cantidades de información en forma simultánea, eficiente y veloz.

- SEGURIDAD Y CONFIABILIDAD: Elex asigna a los usuarios distintos niveles de acceso al sistema (negociación, lectura y consulta), así como también claves secretas, lo que otorga un adecuado grado de seguridad y confiabilidad.

La información pública generada en el Sistema Elex (cotizaciones, operaciones de mercado, índices, etc.) es transmitida desde el Servidor Central de la Bolsa de Valores hasta los terminales de los usuarios a través de la señal comercial de un canal de televisión y solamente es posible recibirla, mediante un decodificador programado para tal efecto, asegurando de esta manera la confidencialidad de la información. Este proceso es conocido como Broadcasting.

La utilización del Broadcasting elimina la probabilidad de una interrupción en la comunicación por desperfectos de las líneas o congestión de centrales, debido a la carencia absoluta de un medio físico para la recepción de información proveniente de la Bolsa de Valores.

Asimismo, el Broadcasting permite la simultaneidad en la recepción de la información produciéndose la propalación de la señal en forma instantánea a todos los usuarios.

La información de carácter privado generada en el Sistema Elex es transmitida de manera bidireccional a través de la Red Privada de

Comunicaciones, permitiendo a los intermediarios bursátiles autorizados interactuar con el Sistema, ingresando y aplicando propuestas de compra y/o venta y recibiendo toda la información de sus tablas privadas.

La información que es mandada por el Sistema Elex, es recibida por un proceso intermedio que se encarga de generar periódicamente las páginas Html conteniendo este tipo de información. En este caso, vemos que la aplicación Web consiste en páginas estáticas que publican información de interés.

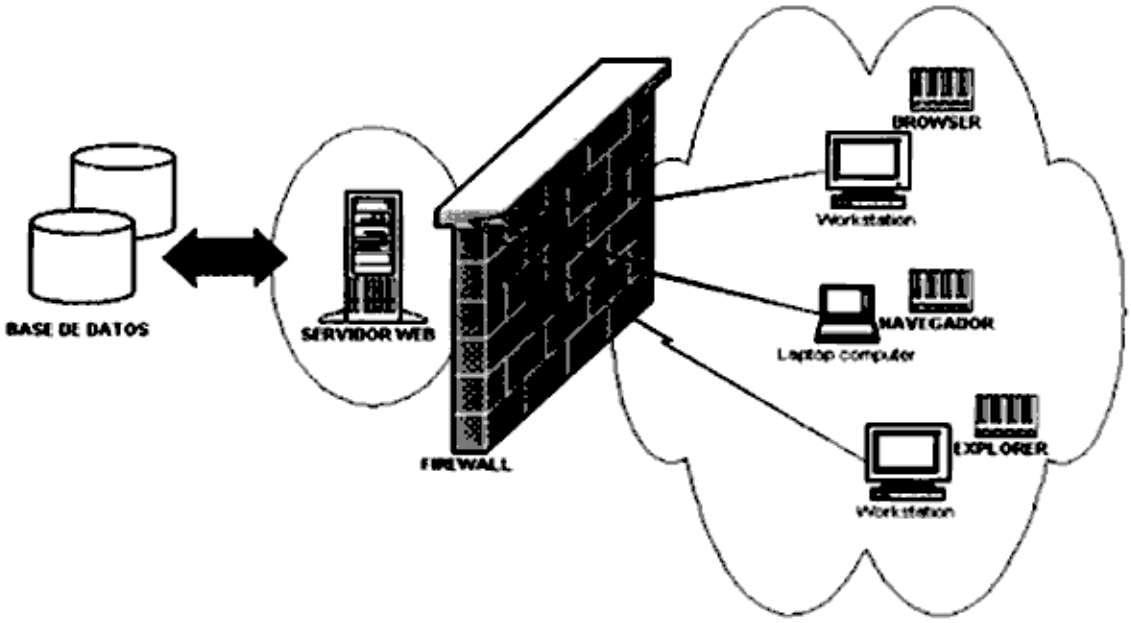

FIGURA 1: *Arquitectura de una aplicación Web, incluyendo un firewall.*

(Fuente Propia)

Actualmente las aplicaciones Web están saturando el mundo del software, a diario se desarrollan este tipo de aplicaciones, eligiéndose este tipo de aplicación para solucionar, por ejemplo, la transferencia de información.

Definitivamente un motivo esencial es la gran demanda que tiene Internet, existe a cada segundo un número infinito de personas que navegan por la Web, y por ende se chocará con "*n*" aplicaciones Web.

La mayor ventaja de Internet es que es una herramienta que provee el acceso a una vasta cantidad de información a lo ancho del mundo. Es un compuesto dinámico y está creciendo con una comunidad de usuarios interesados en diferentes temas y aspectos de vida, no solamente en asuntos de computadoras. Al utilizar efectivamente Internet se puede llegar a obtener abundante y variada información, generalmente con un alto nivel de actualización y vigencia, que posiblemente ayude a incrementar el conocimiento, la creatividad y la productividad.

También hay que contar la facilidad con que se implementa una aplicación, esto tomándolo desde un punto de vista del programador.

Las necesidades en que se incurre para desarrollar aplicaciones Web son hasta en cierta medida, fijas; la interface Web puede utilizarse para los siguientes casos:

- Tienda virtual

- Sitio Web de su compañía en Internet

- Sistemas de búsqueda como por ejemplo catálogos o productos

- Sistemas de procesamiento de órdenes

- Sistemas de contabilidad

- Sistemas de manejos de WORKFLOW

- Aplicaciones de diagramación gráfica.

CAPITULO III

SOLUCION PROPUESTA

3.1. CARACTERISTICAS

- Se cuenta con una instalación centralizada, donde se coloca el software en un único sitio y de inmediato todos los usuarios acceden a éste. Lo mismo ocurre cuando se requiere hacer alguna actualización.

- En esta aplicación dos clientes interactúan en el sistema y la información, es también conocida por el resto de clientes conectados al servicio, cabe mencionar que no necesariamente tienen que ser los mismos clientes.

- Se puede conocer el número de usuarios y estos pueden ser administrados de una manera más eficiente.

- El ciclo de vida del módulo Servidor se comporta como un "demonio". Mientras no haya flujo de información tal módulo se encontrará "dormido", de lo contrario ejecutará los procesos pertinentes.

- Se utiliza el protocolo TCP/IP como interface para las comunicaciones entre las aplicaciones desarrolladas.

- Se plantea para esta solución que el flujo de información se realice en tiempo real, siendo éste el objetivo principal.

3.2. FUNDAMENTO TEORICO

3.2.1. PROTOCOLO TCP/IP

Cuando pretendemos comunicar procesos que se encuentran en máquinas distintas, utilizamos una interfaz denominada "*sockets*". Dicha interfaz necesita un amplio conocimiento de "*las llamadas al sistema*" proporcionadas por la misma y es muy complicada de usar.

Para el usuario es mucho más atractivo utilizar una interfaz amigable que una interfaz directa de "*sockets*" ya que éste último requiere de mucha información técnica para conocer las operaciones que se deben realizar para ejecutar ciertas acciones. Esto nos conduce a construir una aplicación para realizar comunicaciones mediante la familia TCP/IP.

Dentro de la implementación de cada operación de aplicación para TCP/IP utilizaremos las funciones de la interfaz de "*sockets*" que permitirá la comunicación mediante TCP. Para la realización de los programas clientes

que utilicen la biblioteca de llamadas que hemos establecido en la aplicación, se crearán otras librerías que contendrán funciones que llamen a ésta biblioteca de llamadas, las cuales se han confeccionado en la interfaz de la aplicación para la comunicación por TCP/IP.

PROTOCOLOS UTILIZADOS EN TCP/IP

Los protocolos, son aquellos procedimientos que utilizamos para comunicarnos y que poseen una serie de características que les distingue de otros que intentan solucionar su mismo problema en la comunicación.

En el mundo de las redes de ordenadores, existen distintas familias de protocolos que son la base de las redes de comunicaciones. Estas familias de protocolos están basadas en un modelo estándar que se creó para establecer una forma homogénea de protocolos de red. ([2])

NIVELES OSI

El estándar OSI contiene distintos niveles. Cada uno de estos niveles posee una característica que le define particularmente. Casi ninguna familia de protocolos mantiene todos sus niveles, aunque suelen utilizar parte de ellos. Sus niveles son los siguientes:

[2] Referencia : Comunicación y redes de computadores

- NIVEL FÍSICO

Este nivel se ocupa del envío y recepción de bits a través de un medio físico de transmisión.

- NIVEL DE ENLACE

Convierte lo que le ofrece el nivel anterior (físico) en tramas libres de errores de transmisión.

- NIVEL DE RED

Es el que se ocupa de lo que se denomina "*encaminamiento*", es decir, el que se encarga de decidir por qué camino se envía el paquete.

- NIVEL DE TRANSPORTE

Es el encargado de gestionar los puertos de la máquina, es decir, decide a que puerto se envía una determinada información y otorga fiabilidad en algunos casos.

- NIVEL DE SESIÓN

Es el que se ocupa de permitir establecer sesiones entre distintas máquinas, es decir, proporciona mecanismos para establecer un diálogo coherente.

- NIVEL DE PRESENTACIÓN

Se encarga de traducir la información, a un lenguaje común a todas las máquinas.

- NIVEL DE APLICACIÓN

Este es el último nivel del protocolo OSI, el más cercano, si se puede decir así, al usuario. Establece un conjunto de protocolos que son útiles para las aplicaciones que van a usar la red.

Esta secuencia de niveles no se usa íntegramente en ninguna familia de protocolos manteniendo cada familia sus propios niveles. ([3])

[3] Referencia : TCP/IP ilustrado Volumen 1.

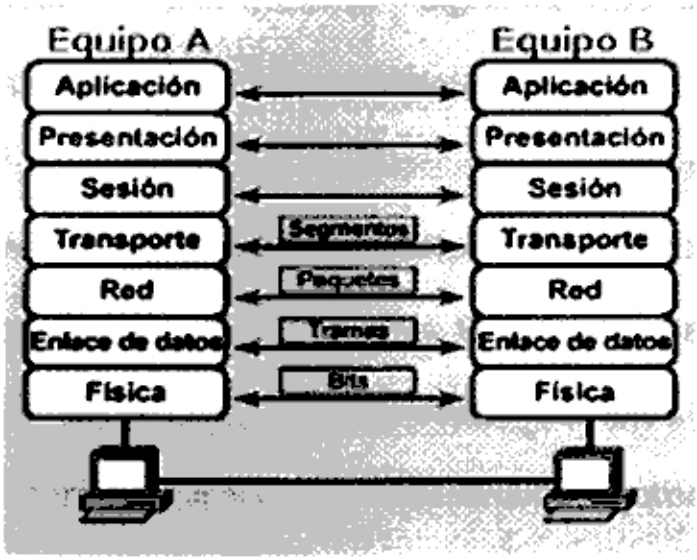

FIGURA 2: *Niveles del Modelo OSI.*

(Fuente - TCP/IP ilustrado Volumen 1. Stevens, W.Richard)

ENCAPSULAMIENTO DE DATOS

Si una computadora A desea enviar datos a otro B, en primer término los datos que se deben enviar se deben colocar en paquetes que se puedan administrar y rastrear a través de un proceso denominado encapsulamiento. Las tres capas superiores (aplicación, presentación y sesión) preparan los datos para su transmisión creando un formato común para la transmisión.

Una vez pasados a formato común, el encapsulamiento rodea los datos con la información de protocolo necesaria antes de que se una al transito de la red. Por lo tanto, a medida que los datos se desplazan a través de las capas del modelo OSI, reciben encabezados, información final y otros tipos de

información. La palabra "*encabezado*" significa que se ha agregado la información correspondiente a la dirección.

Una vez que se envían los datos desde el origen, viajan a través de la capa de aplicación directa hacia las otras capas. El empaquetamiento y el flujo de los datos que se intercambian experimentan cambios a medida que las redes ofrecen sus servicios a los usuarios finales. Como muestra la figura, las redes deben realizar los siguientes cinco pasos de conversión a fin de encapsular los datos:

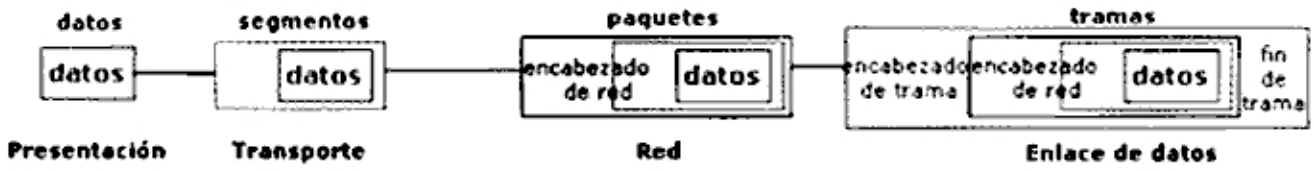

***FIGURA3:** Encapsulamiento de datos.*

(Fuente - TCP/IP ilustrado Volumen 1. Stevens, W.Richard)

1. Creación de los datos (Capa de Presentación).

Cuando un usuario envía un mensaje de correo electrónico, sus caracteres alfanuméricos se convierten en datos que pueden recorrer la Internet.

2. Empaquetar los datos para ser transportados de extremo a extremo (Capa Transporte).

Se dividen los datos en unidades de un tamaño que se pueda administrar, llamados segmentos, y se les asignan números de secuencia para asegurarse de que los "*host*" receptores vuelvan a unir los datos en el orden correcto. Luego los empaqueta para ser transportados por la Internet. Al utilizar segmentos, la función de transporte asegura que los "*host*" del mensaje en ambos extremos del sistema de correo electrónico se puedan comunicar de forma confiable.

3. Agregar la dirección de red al encabezado (Capa de Red).

El siguiente proceso se produce en la capa de red, que encapsula el segmento creando un paquete o datagrama, agregándole una dirección de red destino y origen, por lo general IP. Con esto, los datos se colocan en un paquete que contiene el encabezado de red con las direcciones lógicas de origen y destino. Estas direcciones ayudan a los dispositivos de red a enviar los paquetes a través de la red por una ruta seleccionada.

4. Agregar la dirección local al encabezado de enlace de datos (capa Enlace de datos).

En la capa de enlace de datos continúa el encapsulamiento del paquete, con la creación de una trama. Le agrega a la trama la dirección local (MAC de la tarjeta de red, única para cada tarjeta) origen y destino. Luego, la capa de

enlace de datos transmite los bits binarios de la trama a través de los medios de la capa física. La trama le permite conectarse al próximo dispositivo de red conectado directamente en el enlace. Cada dispositivo en la ruta de red seleccionada requiere el entramado para poder conectarse al siguiente dispositivo.

5. Transmisión del tren de bits creado (Capa Física).

Por último, el tren de bits originado se transmite a la red a través de los medios físicos (cableado, etc.). Una función de temporización permite que los dispositivos distingan estos bits a medida que se trasladan por el medio. El medio en la Internet física de redes puede variar a lo largo de la ruta utilizada. Por ejemplo, el mensaje de correo electrónico puede originarse en una Lan, cruzar el "*Backbone*" de un campus y salir por un enlace de Wan hasta llegar a su destino en otra Lan remota. Los encabezados y la información final se agregan a medida que los datos se desplazan a través de las capas del modelo OSI.

Cuando los datos se transmiten simplemente en una red de área local, se habla de las unidades de datos en términos de tramas, debido a que la dirección MAC es todo lo que se necesita para llegar desde el "*"host"*" origen hasta el "*"host"*" destino. Pero si se deben enviar los datos a otro "*"host"*" a través de una red interna o Internet, los paquetes se transforman en la unidad de datos a la que se hace referencia. Esto se debe a que la dirección

de red del paquete contiene la dirección destino final del *"host"* al que se envían los datos.

TCP/IP

Como decíamos con anterioridad, existen muchas familias de protocolos de comunicaciones en uso. Aunque tengamos tantas, hay algunas que son utilizadas más ampliamente que otras. Sin duda alguna, existe una familia de protocolos que monopoliza de alguna manera las comunicaciones en el mundo de la informática; esta es la familia TCP/IP.

Los protocolos más usados en las comunicaciones de ordenadores son los de la familia TCP/IP. Se subdivide en niveles en los cuales coexisten varios protocolos. Los recursos necesarios para su desarrollo los obtiene mediante la interfaz de *"sockets"*.

Esta familia contiene 4 niveles. Todos ellos pueden verse en el estándar OSI. Estos son los siguientes: nivel de enlace, nivel de red, nivel de transporte y nivel de aplicación.

- CAPA DE APLICACIÓN: Los diseñadores de TCP/IP sintieron que los protocolos de nivel superior deberían incluir los detalles de las capas de sesión y presentación. Simplemente crearon una capa de aplicación que maneja protocolos de alto nivel, aspectos de representación, codificación

y control de diálogo. El modelo TCP/IP combina todos los aspectos relacionados con las aplicaciones en una sola capa y da por sentado que estos datos están correctamente empaquetados para la siguiente capa.

- CAPA DE TRANSPORTE: Permite que capas pares en los *"host"* de fuente y destino puedan conversar. La capa de transporte se refiere a los aspectos de calidad del servicio con respecto a la confiabilidad, el control de flujo y la corrección de errores. Utiliza pues los servicios de la capa de red para proveer un servicio eficiente y confiable a los procesos de la capa de aplicación.

El hardware y el software dentro de la capa de transporte se denominan entidad de transporte, y pueden estar en el kernel, en un proceso de usuario, en una tarjeta, etc.

En esta capa se produce la segmentación de los datos producidos en la capa de Aplicación en unidades de menor tamaño, denominadas paquetes ó datagramas. Un datagrama es un conjunto de datos que se envía como un mensaje independiente.

La capa de transporte no se preocupa de la ruta que van a seguir los datos para llegar a su destino final. Simplemente considera que la comunicación entre ambos extremos está ya establecida y la utiliza.

- CAPA DE INTERNET O DE RED: El propósito de la capa de Internet es enviar paquetes origen desde cualquier red en Internet de redes y que estos paquetes lleguen a su destino independientemente de la ruta y de las redes que se utilizaron para llegar hasta allí.

  En esta capa se produce la determinación de la mejor ruta y la conmutación de paquetes. Durante su transmisión los paquetes pueden ser divididos en fragmentos, que se montan de nuevo en el destino.

  Para poder encaminar los datagramas de la capa de Transporte, éstos de encapsulan en unidades independientes, en las que se incorporan diferentes datos necesarios para el envío, como dirección de origen del datagrama, dirección de destino, longitud del mismo, etc.

  En una comunicación con arquitectura TCP/IP ambos *"host"* pueden introducir paquetes en la red, viajando estos independientemente de cual sea su destino. Por ello, no hay garantía ninguna de entrega de los paquetes ni de orden en los mismos.

- CAPA DE ACCESO A LA RED: El nombre de esta capa es muy amplio y se presta a confusión. También se denomina capa de *"host"* a red. Es la capa que se ocupa de todos los aspectos que requiere un paquete IP para realizar realmente un enlace físico y luego realizar otro enlace físico. Esta capa incluye los detalles de tecnología de Lan y Wan y todos los detalles de las capas física y de enlace de datos del modelo OSI.

Uno de los principales elementos que maneja esta capa es el de las direcciones físicas, números únicos de 6 bytes asignados a cada tarjeta de red, y que es el medio principal de localización de un *"host"* dentro de una red. Cada tarjeta tiene un número identificador, del que 3 primeros bytes son asignados por el fabricante de la misma, mientras que los otros 3 se asignan de forma especial. Cuando un *"host"* debe enviar un paquete a otro de su red busca a éste mediante su número de tarjeta de red (dirección física).

La interfaz de *"sockets"*, otorga sus recursos a varios de estos niveles. Uno de los niveles en los que se emplean *"sockets"* es en el de transporte. Este nivel, en el caso de la familia TCP/IP, puede emplear dos distintos protocolos. Estos son los denominados TCP.

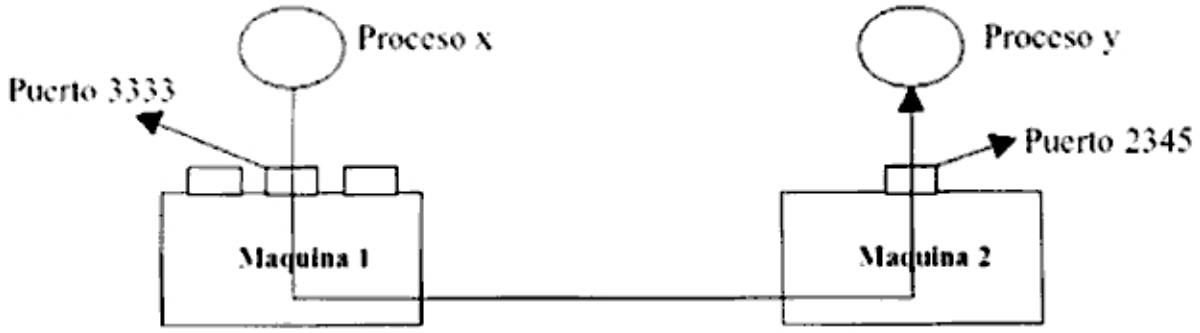

FIGURA 4: *Utilización de puertos.*

(Fuente - TCP/IP ilustrado Volumen 1. Stevens, W.Richard)

TCP, es un protocolo de transporte y se dice que es orientado a conexión. Esto se refiere a la facultad que tiene para establecer aquello que llama conexiones propiamente dichas.

Cuando hablamos de conexiones en cuestiones relacionadas con protocolos de comunicaciones, nos referimos a la creación de un flujo único de datos que establecen el emisor y el receptor, que va a ser el camino por el cual van a transferirse todos los datos de un lado al otro de la propia conexión.

Además de esta característica, TCP es fiable, es decir, nos garantiza la llegada de los datos al otro extremo de la conexión. Sin embargo, el otro protocolo de transporte de esta familia, UDP, es no orientado a conexión (los datos enviados en distintos paquetes pueden utilizar distintos caminos para llegar a su destino) y no fiable (no se garantiza la llegada al otro extremo de

los datos enviados). La interfaz de *"sockets"*, permite utilizar conexiones TCP.

INTERFACES TCP

Existen dos tipos de interfaces entre la conexión TCP y los otros programas.

El primero es utilizar la pila de los programas de la capa de red. Como en esta capa solamente está el protocolo IP, la interface lo determina este protocolo. El segundo tipo es la interfaz del programa de usuario. Esta interface puede variar según el sistema operativo, pero en general tiene las siguientes características.

La interface envuelve el programa de usuario llamando a una rutina que introduce entradas en una estructura de datos llamada el bloque de control de transmisión (TCB). Las entradas se realizan inicialmente en la pila de hardware y transferidas al TCB por medio de una rutina de sistema. Estas entradas permiten al TCP asociar un usuario con una conexión particular, de modo que pueda aceptar comandos de un usuario y mandarlos a otro usuario en la otra parte de la conexión. TCP utiliza unos identificadores únicos para cada parte de la conexión. Esto se utiliza para recordar la asociación entre dos usuarios. Al usuario se le asigna un nombre de conexión para utilizarlo en futuras entradas del TCB. Los identificadores para cada extremo de la conexión se llaman *"sockets"*. El "*socket*" local se

construye concatenando la dirección IP de origen y el número de puerto de origen. El "*socket*" remoto se obtiene concatenando la dirección IP de destino y el número de puerto de destino.

El par de *"sockets"* de una conexión forman un único numero en Internet. El UDP tiene los mismos *"sockets"*, pero no los recuerda. Esta las diferencia entre un protocolo orientado a conexión y otro sin conexión. A continuación se explican los comandos más usuales:

- OPEN: Inicia una conexión ó comienza a escuchar un "*socket*". El usuario tiene un nombre de conexión local que actúa como un puntero dentro del TCB.

- SEND: El comando SEND envía datos del buffer especificado.

- RECEIVE: El comando RECEIVE es un mensaje de error si el nombre local proporcionado no es utilizado antes con el comando OPEN.

- CLOSE: El comando CLOSE hace que se cierre una conexión. Se produce un error si la conexión especificada no ha sido abierta, o si no se tiene autorización para cerrar la conexión.

- STATUS: El comando STATUS solamente tiene una variable asociada, que es el nombre de la conexión.

- ABORT: El comando ABORT hace que todos los comandos SEND y RECEIVE asociados al nombre de la conexión local se interrumpan. La entrada del usuario del TCB se elimina y se envía un mensaje especial de reinicio a la entidad del otro lado de la conexión.

El TCP recuerda el estado de cada conexión por medio del TCB. Cuando se abre una conexión, se efectúa una entrada única en el TCB. Un nombre de conexión se le asigna al usuario para activar los comandos de la conexión. Cuando se cierra una conexión se elimina su entrada del TCB.

COMUNICACIÓN CONFIABLE

La transmisión se hace de manera confiable por medio del uso de secuencias de números y confirmaciones. A cada segmento de datos enviado se le asigna un número de secuencia. Los segmentos también llevan un número de confirmación, el cual es el número de secuencia del siguiente segmento de datos que se espera recibir en la dirección opuesta. Cuando TCP envía un segmento de información, coloca una copia de tal segmento en una cola e inicializa un reloj; cuando la confirmación para ese segmento de datos es recibida, el segmento se borra de la cola. Si la confirmación no es recibida antes de que el reloj expire, el segmento es retransmitido. El proceso del TCP que recibe los segmentos, utiliza los

números de secuencia para reconstruir la información de manera correcta, ya que los segmentos pueden ser recibidos fuera de orden y para eliminar los duplicados. Los errores son manejados al añadir un valor de verificación (*checksum*) a cada segmento. El valor de verificación es chequeado por el receptor, y si éste no es correcto, el segmento se desecha.

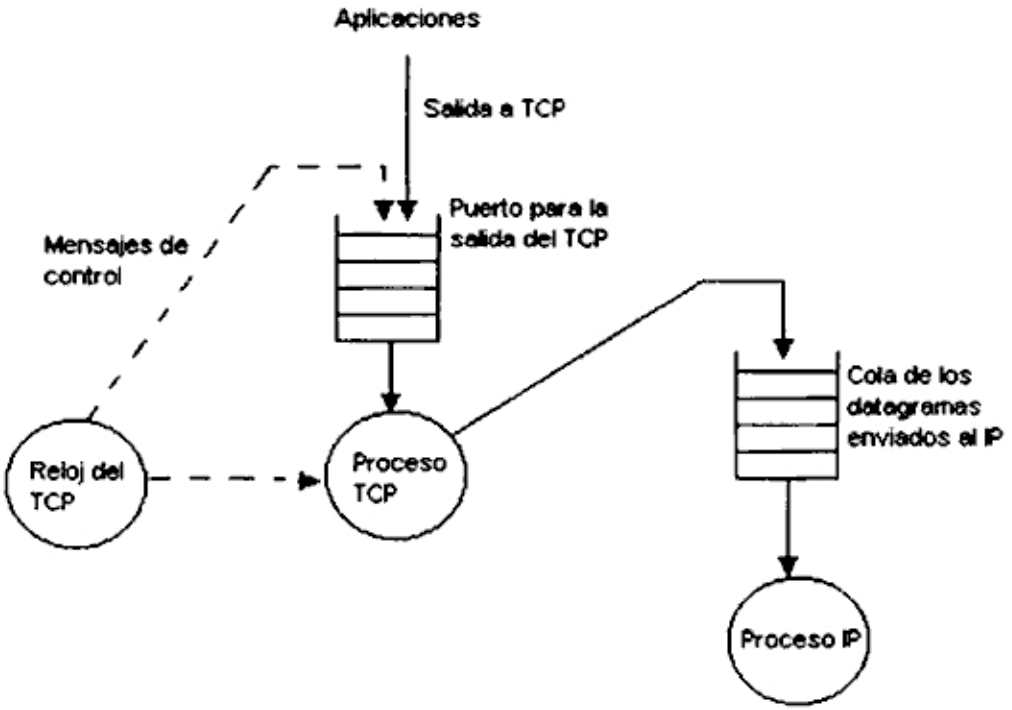

FIGURA 5: *Comunicación confiable.*

(Fuente - TCP/IP ilustrado Volumen 1. Stevens, W.Richard)

La conclusión a la que se ha llegado tras realizar este trabajo ha sido la siguiente: El conjunto de protocolos TCP/IP ha sido de vital importancia para el desarrollo de las redes de comunicación, sobre todo para Internet. El ritmo de expansión de Internet también es una consecuencia de estos protocolos, sin los cuales, conectar redes de distintas naturalezas (diferente Hardware, sistema operativo, etc.), hubiera sido mucho más difícil, por no decir

imposible. Así pues, podemos decir que los protocolos TCP/IP fueron y son el motor necesario para que las redes en general, e Internet en particular, se mejoren y se pueda lograr una buena "autopista de la información".([4])

DIFICULTADES CON LAS INTERFACES DE LAS COMUNICACIONES

Las interfaces que emplean las aplicaciones del sistema Linux en el campo de las comunicaciones, están basadas en una abstracción llamada *"socket"*. Para establecer comunicaciones en el sistema operativo Linux, se emplea una interfaz poco intuitiva y bastante pesada de utilizar. Esta interfaz está basada en lo que se denominan *"sockets"* y si nos ceñimos a los casos típicos de comunicación como TCP, se puede afirmar que es algo redundante.

Los *"sockets"*, son unos *"enchufes"* que utilizamos para comunicar procesos entre sí. Estos procesos pueden ser locales (pueden estar en la misma máquina) ó remotos (se encuentran en máquinas distintas).

La interfaz de comunicaciones para Linux, permite emplear muchas opciones distintas, lo que hace que existan multitud de funciones en dicha interfaz. Por la tanto para realizar una operación típica, tendríamos que llamar de manera secuencial a una extensa serie de funciones que podrían intercambiarse por una única que nos facilitará el proceso.

[4] Referencia : Comunicación de datos, redes de computadores y sistemas abiertos.

Esto nos plantea el problema de la utilización de la interfaz de *"sockets"*. Por un lado nos permite efectuar multitud de variaciones en nuestras comunicaciones, ya sean orientadas a conexión o no. Pero cuando queremos desarrollar algo más genérico nos percatamos de que debemos realizar muchas llamadas al sistema para efectuar algo que de alguna forma es bastante frecuente utilizar y que podría integrarse en un menor número de operaciones. En la interfaz de *"sockets"*, las estructuras que nos permiten identificar a estos son descriptores de ficheros. Cuando hacemos una llamada a la función socket() de esta interfaz, ésta nos devuelve un descriptor que va a identificar nuestro extremo en la comunicación. Éste se podrá conectar, por así decirlo, con otro descriptor de fichero, estableciéndose entre ellos un flujo de datos.

Además de la interfaz de *"sockets"*, el kernel del sistema nos facilita unos recursos denominados puertos, los cuales son los identificadores (por así decirlo) de los procesos que se van a comunicar.

Lo normal es que un proceso, que se quiere comunicar con otro local ó remoto, esté escuchando a través de un puerto concreto. Para simplificar las cosas, en Linux los puertos preestablecidos son certificados para el usuario *"root"*. Estos puertos se suelen utilizar para lo que se denominan *"demonios"* (procesos que siempre están en ejecución mientras el sistema está en funcionamiento, y que suelen esperar peticiones de algún tipo). Hay otros

puertos que pueden ser utilizados por cualquier usuario y que son libres por tanto para la utilización de las demás aplicaciones.

En general cuando queremos enviar bytes o recibir a través de un proceso que se encuentra en otra máquina, lo que hacemos es conectarnos a un determinado puerto y enviar o recibir lo que tengamos pendiente. También son una herramienta de multiplexación de recepción de información. Cuando nos llegan datos de las capas más bajas de nuestro protocolo de comunicación, estos datos son distribuidos por el nivel de transporte al puerto que le corresponda dicha información:

La interfaz de *"sockets"*, nos permite asignar a un "*socket*", un puerto que va a ser el recurso por el cual se va a comunicar el propio proceso. El sistema nos facilita unas estructuras de datos que nos ofrecen la posibilidad de establecer el puerto al que se refiere cada "*socket*" abierto.

Los *"sockets"* y los puertos son abstracciones distintas, aunque ambos tienen la particularidad de que son recursos del nivel de transporte. Estos dos componentes, "*sockets*" y puertos, son la base de la interfaz del nivel de transporte en varios sistemas. Por ejemplo, en el sistema operativo Windows se utiliza una interfaz muy parecida a la de Linux para el empleo de comunicaciones, en la cual se emplean los denominados "*winsockets*", que son una versión de los mismos utilizados por Linux.

3.2.2. PROTOCOLO SSL

SSL fue diseñado con el propósito de crear un canal de comunicaciones seguro entre un cliente y un servidor que fuese independiente del sistema operativo usado por ambos y que se beneficiara de forma dinámica y flexible de los nuevos adelantos en materia de cifrado a medida de que estos estuvieran disponibles.

SSL trabaja sobre el protocolo TCP y por debajo de protocolos como HTTP, IMAP, LDAP, etc., y puede ser usado por todos ellos de forma transparente para el usuario. Opera entre la capa de transporte y la de sesión del modelo OSI (o entre la capa de transporte y la de aplicación del modelo TCP) y está formado, a su vez, por dos capas y cuatro componentes bien diferenciados.

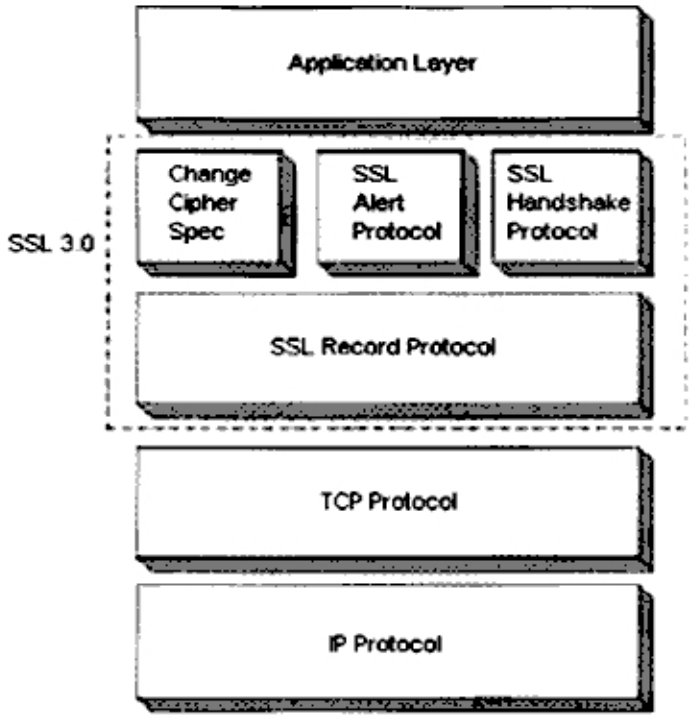

FIGURA 6: *Protocolo SSL.*

(Fuente - TCP/IP ilustrado Volumen 1. Stevens, W.Richard)

El protocolo de registro (*SSL Record Protocol*) se encarga de encapsular el trabajo de los elementos de la capa superior, construyendo un canal de comunicaciones entre los dos extremos objeto de la comunicación. El verdadero corazón de SSL está en el protocolo de "*Handshake*" que es el encargado de intercambiar la clave que se utilizará para crear un canal seguro mediante un algoritmo eficiente de cifrado simétrico. También es responsabilidad de este protocolo coordinar los estados de ambos extremos de la transmisión.

El protocolo de Alerta es el encargado de señalizar problemas y errores concernientes a la sesión SSL establecida. Por último, el "*Change Cipher Spec Protocol*" está formado por un único mensaje consistente en un único byte de valor 1 y se utiliza para notificar un cambio en la estrategia de cifrado.

A grandes rasgos, podríamos decir que SSL trabaja de la siguiente forma; en primer lugar intercambiamos una clave de longitud suficiente mediante un algoritmo de cifrado asimétrico.

Mediante esa clave establecemos un canal seguro utilizando para ello un algoritmo simétrico previamente negociado. A continuación, toma los mensajes a ser transmitidos, los fragmenta en bloques, los comprime, aplica un algoritmo "*hash*" para obtener un resumen (MAC) que es concatenado a cada uno de los bloques comprimidos para asegurar la integridad de los

mismos, realiza el cifrado y envía los resultados. El estado de todas estas operaciones son controladas mediante una máquina de control de estados. Una sesión SSL puede comprender múltiples conexiones. Adicionalmente, se pueden establecer múltiples sesiones SSL simultaneas.

A continuación veremos todos estos procesos con un poco más de detalle.

El protocolo de "*Handshake*" es el encargado de negociar los atributos de la sesión SSL que permitirán construir un canal seguro de comunicaciones. En primer lugar el cliente envía un mensaje "*Client Hello*" al servidor el cual debe de responder con un mensaje similar de "*Server Hello*". Estos mensajes son utilizados para dar a conocer ciertas características de ambos: versión del protocolo usada, algoritmos de cifrado conocidos y preferidos, longitudes máximas de clave que admite para cada uno de ellos, funciones "*hash*" y métodos de compresión a utilizar. Adicionalmente cliente y servidor pueden intercambiar dos números generados aleatoriamente para ser usados como sal. En este momento, además, el servidor asigna un identificador a la sesión y se hace constar la fecha y hora de la misma. El identificador de sesión es enviado al cliente en el mensaje de "*Server Hello*". Si el servidor no respondiera con un mensaje de "*Server Hello*" o éste no fuese válido o reconocible la sesión abortaría inmediatamente. Generalmente el servidor, el segundo en contestar, elige los algoritmos más fuertes de entre los soportados por el cliente. Si no hay acuerdo en este punto se envia un mensaje de error y se aborta la sesión.

A continuación del mensaje de *"Server Hello"*, el servidor puede enviar su Certificado (típicamente un X.509) de forma que sea autenticado por el cliente y que, además, éste reciba su clave pública. Si no es así, le envía al cliente su clave pública mediante un mensaje de *"Server Key Exchange"*. Está claro es que al menos uno de estos dos mensajes es necesario para establecer el canal seguro. Un último mensaje que puede enviar el servidor en esta fase de negociación es una solicitud de certificado al cliente. Por último, la fase concluye con el envío, por parte del servidor, de un mensaje de *"Server Hello Done"*.

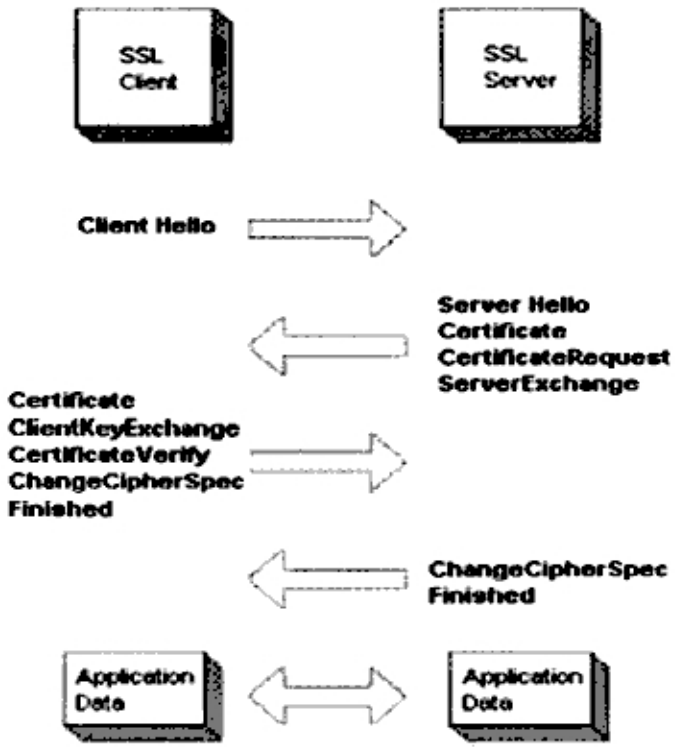

FIGURA 7: *Protocolo SSL.*

(Fuente - TCP/IP ilustrado Volumen 1. Stevens, W.Richard)

Si el Servidor ha solicitado su certificado al cliente, éste debe de responder con él ó con un mensaje de alerta indicando que no lo posee. A continuación se envía un mensaje de *"Client Key Exchange"* donde el cliente envía al servidor, cifrada mediante la clave pública de éste, la clave maestra, un número aleatorio generado por él y que actuará como clave del algoritmo simétrico acordado para el intercambio de datos. Por último, si el cliente ha enviado un certificado y éste tiene capacidades de firma, enviará adicionalmente un mensaje de *"Certificate Verify"* firmado digitalmente con objeto de que el servidor pueda verificar que la firma es válida. En este punto el cliente da por concluida la fase mediante un mensaje de *"Change Cipher Spec"* seguido, inmediatamente, de un mensaje de *"Finished"* que ya va cifrado mediante los algoritmos y claves recién negociados.

En respuesta, el servidor envía su propio mensaje de *"Change Cipher Spec"* y, a continuación, su mensaje de *"Finished"* cifrado con los parámetros negociados. En este momento finaliza la fase de *"Handshake"* y cliente y servidor pueden intercambiar datos libremente.

Durante la transmisión de datos los mensajes son fragmentados y comprimidos por el protocolo de registro antes de su envío y descomprimidos y reconstruidos por el mismo protocolo al otro extremo de la comunicación. El algoritmo de compresión utilizado es característico de la sesión y se negocia, como hemos visto, en la fase de *"Handshake"*.

SSL puede establecer múltiples conexiones dentro de una misma sesión o reanudar una sesión previamente interrumpida. En ambos casos el intercambio de mensajes de la fase *"Handshake"* es mucho más reducido como veremos a continuación. El cliente envía un mensaje de *"Client Hello"* usando el identificador de la sesión previamente negociada. El servidor verifica si ese identificador es válido y en caso afirmativo devuelve un mensaje de *"Server Hello"* usando el mismo identificador de sesión. Acto seguido, envía al cliente un mensaje de *"Change Cipher Spec"* y a continuación un mensaje de *"Finished"* cifrado ya con los parámetros de la sesión reanudada. El cliente responde con sus propios mensajes de *"Change Cipher Spec"* y *"Finished"* y seguidamente comienzan a intercambiar datos.

3.2.3. MODELO CLIENTE / SERVIDOR

- Es el modelo de base utilizado en la mayoría de las redes de computadoras.

- El término servidor se aplica a cualquier programa que ofrece un servicio y que puede ser accesado a través de la red.

- Los servidores aceptan peticiones recibidas a través de la red, realizan el servicio y regresa el resultado al que generó la petición.

- Un programa en ejecución, se convierte en cliente cuando envíe a una petición a un servidor y espera una respuesta.

- Los servidores pueden realizar tareas simples o bastante complejas.

- Si el principal objetivo de una máquina es el de soportar un programa servidor en particular, entonces el término "servidor" puede ser aplicado también a dicha máquina.

Un servidor comienza su ejecución antes del inicio de cualquier interacción y (usualmente) continúa aceptando peticiones y enviando respuestas indefinidamente. Un cliente es cualquier programa que efectúa una petición y espera una respuesta; (generalmente) termina después de usar un servidor un número finito de veces.

Un servidor espera una petición sobre un puerto bien conocido, el cual ha sido reservado para un cierto servicio. Un cliente reserva un puerto arbitrario y no usado para poder comunicarse.

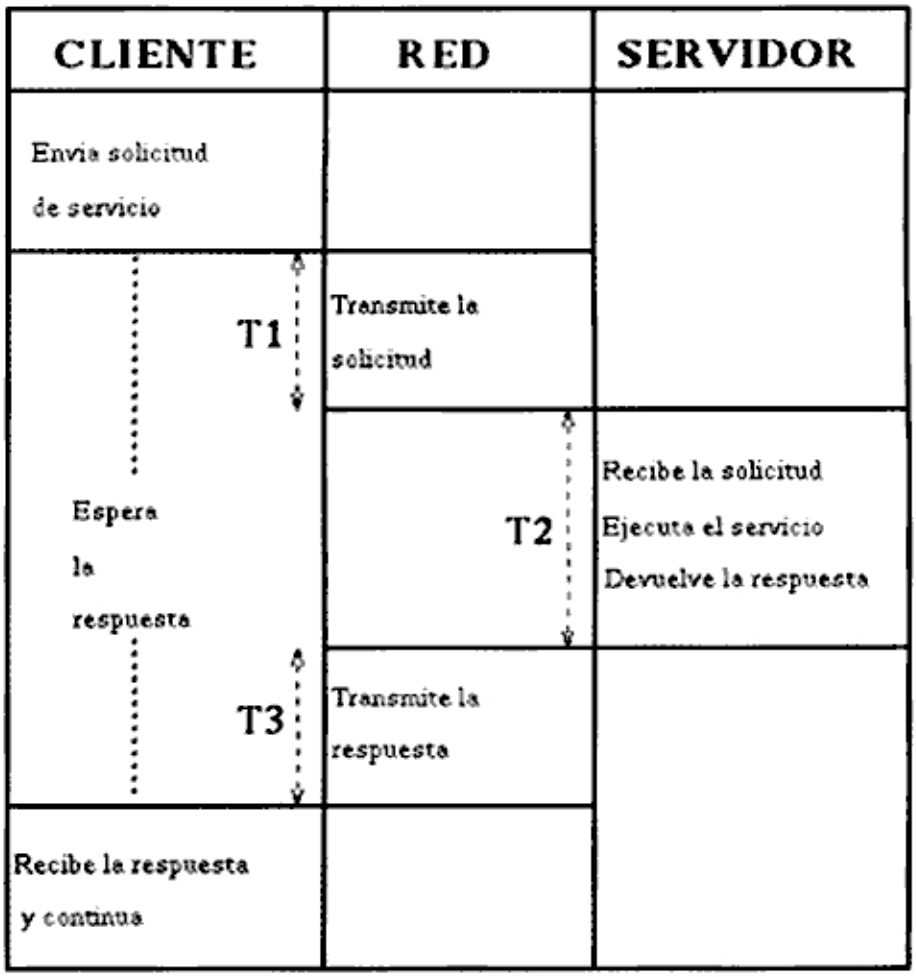

***FIGURA 8:** Modelo Cliente /Servidor.*

(Fuente Propia)

El primer problema con el que se debe lidiar al querer comunicar dos procesos es la sincronización. Suponer por ejemplo que existen dos programas que desean comunicarse en forma sincronizada a través de un "*pipe*". La mejor forma de lograr la sincronización es que uno de los dos programas espere hasta que llegue algún indicio de que existe otro programa que se quiere comunicar con él. En el modelo Cliente / Servidor, el Servidor siempre está esperando una conexión (inicia la conexión en forma pasiva) y el Cliente inicia la conexión en forma activa. En general, un

Servidor debe ser capaz de atender simultáneamente a varios clientes, por esto son en general más complejos.

Desde el punto de vista de una aplicación, el TCP/IP, al igual que muchos otros protocolos de comunicación, implementa un mecanismo fiable para la transmisión de datos entre ordenadores.

En concreto, el TCP/IP permite que dos procesos puedan establecer una comunicación, tanto si ambos procesos se están ejecutando en el mismo ordenador o en ordenadores remotos.

Hay que tener presente que el protocolo TCP/IP especifica los detalles y mecanismos para la transmisión de datos entre dos aplicaciones comunicantes, pero no dictamina cuando ni por que deben interactuar ambas aplicaciones, ni siquiera especifica como debería estar organizada una aplicación que se va a ejecutar en un entorno distribuido. Es tarea del diseñador de la aplicación distribuida el establecer un protocolo de comunicación y sincronización adecuado.

En la práctica el esquema de programación más utilizado para la implementación de aplicaciones distribuidas es el paradigma Cliente / Servidor. Imaginemos un técnico de computadoras que inicia la ejecución de dos programas en máquinas distintas y que tiene la intención de que dichos programas puedan comunicar entre sí. Una vez iniciado el primer programa

éste envía un mensaje a su contertulio. La conexión con la maquina a la cual va dirigido el mensaje se puede establecer en un intervalo de unos pocos milisegundos (recordar que las computadoras funcionan a velocidades muchos ordenes de magnitud por encima de los humanos), por lo que el proceso recién lanzado determina que su contertulio todavía no existe, con lo cual emite un mensaje de error y finaliza su ejecución. Mientras tanto, el técnico inicia la ejecución del segundo proceso.

Desdichadamente, el segundo proceso no puede comunicar con el primero ya que éste ha concluido su ejecución. Incluso si los dos procesos intentan establecer la comunicación continuamente, estos pueden ejecutarse tan rápidamente, que la probabilidad de colisiones es muy alta.

Según este modelo, en cualquier par de aplicaciones comunicantes, una de las partes implicadas en la comunicación debe iniciar su ejecución y esperar (indefinidamente) hasta que la otra parte contacte con ella. Esta solución es importante, ya que el TCP/IP no proporciona ningún mecanismo que automáticamente lance procesos para atender mensajes entrantes, debe de existir un proceso en espera que sea capaz de atender dichos mensajes (peticiones de servicio).

Muchos administradores hacen que ciertos programas de comunicaciones se inicien automáticamente cuando el sistema arranca, de este modo se aseguran que la computadora estará preparada para aceptar ciertas

solicitudes de servicio. Después de iniciar su ejecución, cada uno de estos programas se queda en espera de la siguiente petición para el servicio que oferta.

3.2.4. ARQUITECTURA DE TRES CAPAS

Cuando se trata el tema de la arquitectura de 3 capas, también aparecen involucrados elementos ajenos al desarrollo en sí del sistema, como los detalles relacionados con la distribución de diferentes componentes de la aplicación en diferentes servidores, cada uno de ellos especializado en una función particular: entregar los datos, validar las normas del negocio y asegurarse de que las transacciones se procesen de la manera debida, generar los reportes, o los formularios de entrada, etcétera.

Pero la esencia de los modelos de desarrollo por capas es el concepto de "*separación*", manteniendo cada componente tan separado del contexto global como sea posible. Y cada capa es, simplemente, la agrupación de todos los componentes que tienen una funcionalidad común.

La CAPA DE SERVICIOS DE INTERFAZ, o de servicios de usuario se ocupa fundamentalmente de la comunicación con el operador. Incluye todos los componentes cuya responsabilidad primaria es recibir las entradas del usuario, o presentarle la información al mismo. Las ventanas de diálogo, o los presentadores de reportes, son los ejemplos clásicos de esta capa.

Las ventanas de entrada, donde se definen las operaciones o los atributos de las entidades, no hacen nada por sí mismas. Se limitan a enviar los datos recibidos a los componentes en la capa de servicios de negocio. En esta capa residen todas las funciones que validan las entradas, obtienen los datos (accediendo a los servicios de la capa inferior), presentan los datos (enviándolos a los componentes de interfaz, que los formatean y los muestran) y ejecutan las operaciones. Esto es más sencillo de usar que de explicar. La aplicación Cliente puede llamarse de múltiples formas: Servicio de usuario, Cliente, aplicación, FRONT-END, capa de presentación, GUI, etc. La función de la aplicación Cliente es la de permitir al usuario una interfaz para los servicios de negocios. Una aplicación Cliente bien diseñada permite que el usuario entienda los servicios de negocios como un todo y navegar eficientemente por estos servicios. Esta es la capa que se crea con lenguajes de "*Cuarta Generación*" como VISUAL C++.

La Capa de Servicio de Negocios crean la unión entre las aplicaciones cliente y los servicios de datos. La función de esta capa lógica es primordialmente la de hacer valer las políticas del negocio y encapsular un modelo de los negocios así como exponer tal modelo a las aplicaciones cliente.

Las políticas de negocios son reglas que restringen y controlan el flujo de las tareas. Encapsulan políticas como en los siguientes ejemplos:

- Pedidos que han sido embarcados y no pueden ser cancelados.

- Envío de camiones en alguna ruta de servicio con un código más económico.

- Un mensaje por correo electrónico deberá ser enviado a una cuenta ejecutiva cuyo destinatario no ha hecho ningún pedido desde hace varias semanas.

Las reglas de negocios cambian más rápidamente de lo que una aplicación es útil. Supóngase que una regla adicional de crédito se ponga a efecto. El trabajo en el servidor de negocios, por lo tanto, deberá cambiar no así el flujo de datos. Dado que estos cambios son frecuentes en un entorno empresarial, las reglas del negocio son excelentes cambios para su encapsulamiento.

Finalmente la Capa de Servicio de Datos es aquello cuyo manejo se lleva a cabo mediante Sistemas Manejadores de Base de Datos (DBMS) basados en SQL, como SQL Server y Oracle. Estos son los que manejan los datos, información y transacciones para los servidores de negocios. Así, estos servidores mantendrán inalterable solamente la integración de datos que manipulan.

A. DIAGRAMA DE CASOS DE USO

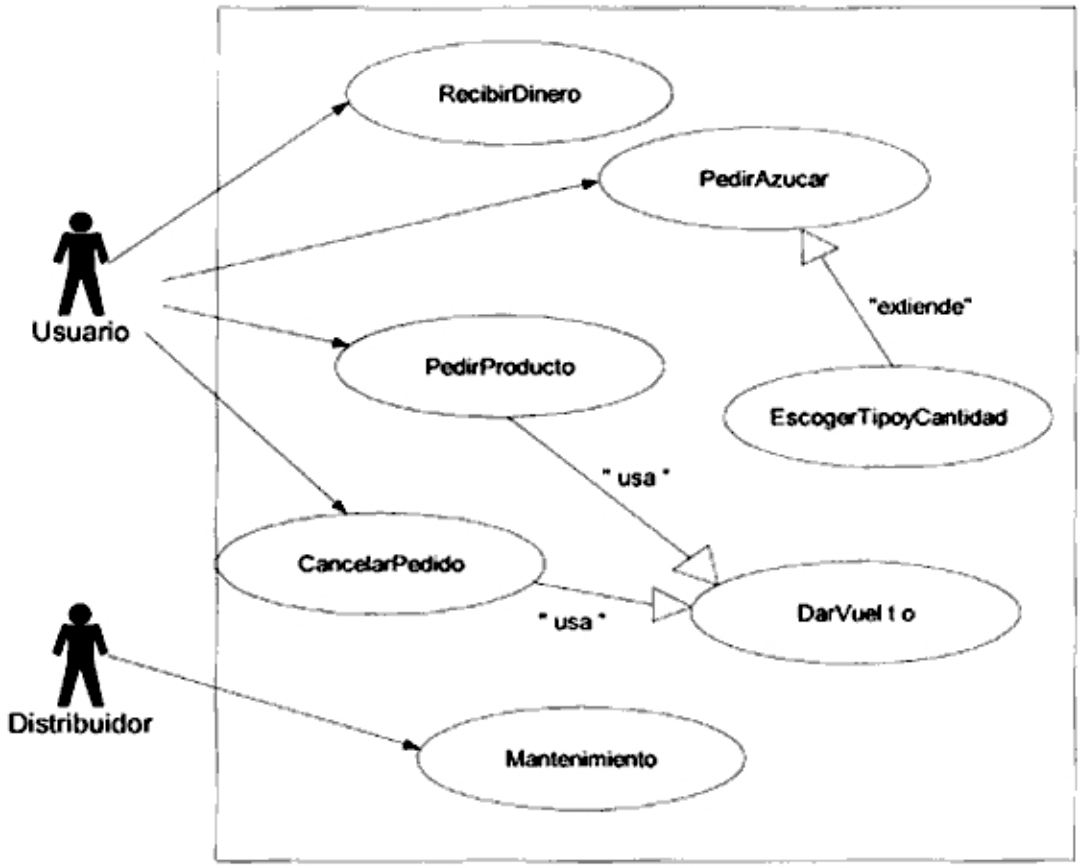

FIGURA 9: *Ejemplo de un Diagrama de Caso de Uso.*

(Página Web – Elementos Notacionales del UML 1.0)

Los diagramas de caso de uso son creados para visualizar las relaciones entre actores y casos de uso.

Un diagrama de casos de uso muestra las distintas operaciones que se esperan de una aplicación o sistema y como se relaciona en su entorno (usuarios y otras aplicaciones).

- **Caso de uso:** Un caso de uso es un patrón de comportamiento que el sistema exhibe, cada caso de uso es una secuencia de transacciones relacionadas y ejecutadas por un actor y el sistema mediante un dialogo. Se representa en el diagrama por una elipse, denota un requerimiento solucionado por el sistema. Cada caso de uso es una operación completa desarrollada por los actores y por el sistema de un dialogo. El conjunto de casos de uso representa la totalidad de operaciones desarrolladas por el sistema. Va acompañado de un nombre significativo.

- **Actor:** es el usuario del sistema que necesita o usa algunos de los casos de uso. Se representa mediante un icono, acompañado de un nombre significativo si es necesario.

RELACIONES DE UN DIAGRAMA DE CASOS DE USO

Entre los elementos de un diagrama de caso de uso se pueden presentar tres tipos de relaciones, representadas por líneas dirigidas entre ellos (del elemento dependiente al independiente).

- **Comunicación:** relación entre un actor y un caso de uso, denota la participación del actor en el caso de uso determinado.

- **Usa (uses):** relación entre dos casos de uso, denota la inclusión del comportamiento de un escenario en otro.

- **Extiende (extends):** relación entre dos casos de uso, denota cuando un caso de uso es una especialización de otro.

B. DIAGRAMA DE CLASES

Un diagrama de clases muestra la existencia de clases y sus relaciones en la vista lógica de un sistema.

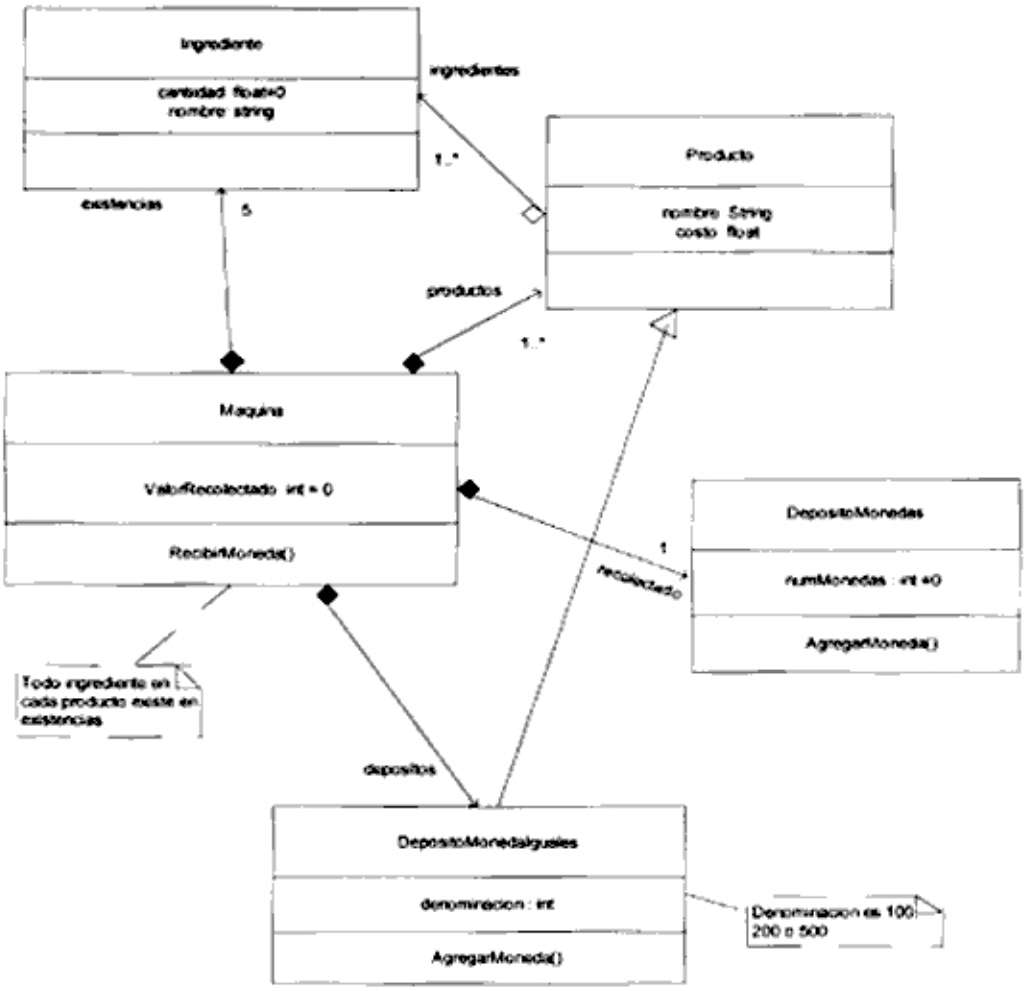

FIGURA 10: *Ejemplo de un Diagrama de Clases.*

(Página Web – Elementos Notacionales del UML 1.0)

CLASE

Representada por un rectángulo con tres divisiones internas, son los elementos fundamentales del diagrama. Una clase describe un conjunto de objetos con características y comportamiento idéntico.

ATRIBUTO

Identifican las características propias de cada clase. Generalmente son de tipos simples, ya que los atributos de tipos compuestos se representan mediante asociaciones de composición con otras clases.

OPERACION

El conjunto de operaciones describe el comportamiento de los objetos de una clase.

ASOCIACIÓN

Una de las asociaciones en general es una línea que une dos o más símbolos. Pueden tener dos tipos de adornos, que definen su semántica y características. Los tipos de asociaciones entre clases presentes en un diagrama estático son:

1. Asociación binaria
2. Asociación n-aria
3. Composición
4. Generalización
5. Refinamiento

Cada asociación puede presentar algunos elementos adicionales que dan detalle a la relación, como son:

- Rol, identificado como un nombre en los finales de la línea, describe la semántica de la relación en el sentido indicado. Por ejemplo, la asociación de composición entre maquina e ingrediente recibe el nombre de existencias, como rol en ese sentido.

- Multiplicidad, describe la cardinalidad de la relación. En el ejemplo anterior se utilizan 1, 1...*,5,*, como indicadores de multiplicidad.

ASOCIACION BINARIA

Se identifica como una línea sólida que une dos clases. Representa una relación de algún tipo entre las dos clases, no muy fuerte (es decir, no se exige dependencia existencial ni encapsulamiento).

COMPOSICION

En una asociación fuerte, que implica tres casos de independencia existencial. El elemento dependiente desaparece al destruirse el que lo contiene y si es de cardinalidad 1, es creado al mismo tiempo. Hay una pertenencia fuerte. Se puede decir que el objeto contenido es parte constitutiva y vital del que lo contiene.

Los objetos contenidos no son compartidos, esto es, no hacen parte del estado de otro objeto. Se denota dibujando un rombo relleno del lado de la clase que contiene a la otra en la relación. En el ejemplo inicial de esta hoja se presentan varios ejemplos de relaciones de composición entre 'Maquina' y 'Producto', 'Maquina' y 'DepositoMonedas' y 'Maquina' y 'DepositoMonedasIguales'.

Existe también una relación de composición menos fuerte (no se exige dependencia existencial, por ejemplo) que es denotada por un rombo sin rellenar en uno de los extremos. Un ejemplo puede encontrarse entre productos e ingrediente.

GENERALIZACIÓN

La relación de generalización denota una relación de herencia entre clases. Se representa dibujando un triangulo sin rellenar en el lado de la superclase. La subclase hereda todos los atributos y mensajes descritos en la superclase. En el ejemplo se encuentra una generalización entre 'DepositoMonedas' (superclase) y 'DepositoMonedasIguales' (subclase).

PAQUETE

Un paquete es una forma de agrupar clases(u otros elementos en otros tipos de diagramas) en modelos grandes. Pueden tener asociaciones de dependencia o de generalización entre ellos.

DEPENDENCIA

Denota una relación semántica entre dos elementos (clases, o paquetes por el momento) del modelo. Indica que cambiar el elemento independiente puede requerir cambios en los dependientes. Se muestra como una línea punteada direccional, indicando el sentido de la dependencia. Puede tener por medio de estereotipos una explicación del tipo de dependencia presentada.

C. CONCEPTOS BÁSICOS DE UN DIAGRAMA DE ESTADO

Muestra el estado espacial de un estado a otro y las acciones que resultan. Muestra el conjunto de estados por los cuales pasa un objeto durante su vida en una aplicación, junto con los cambios que permitan pasar de un estado a otro.

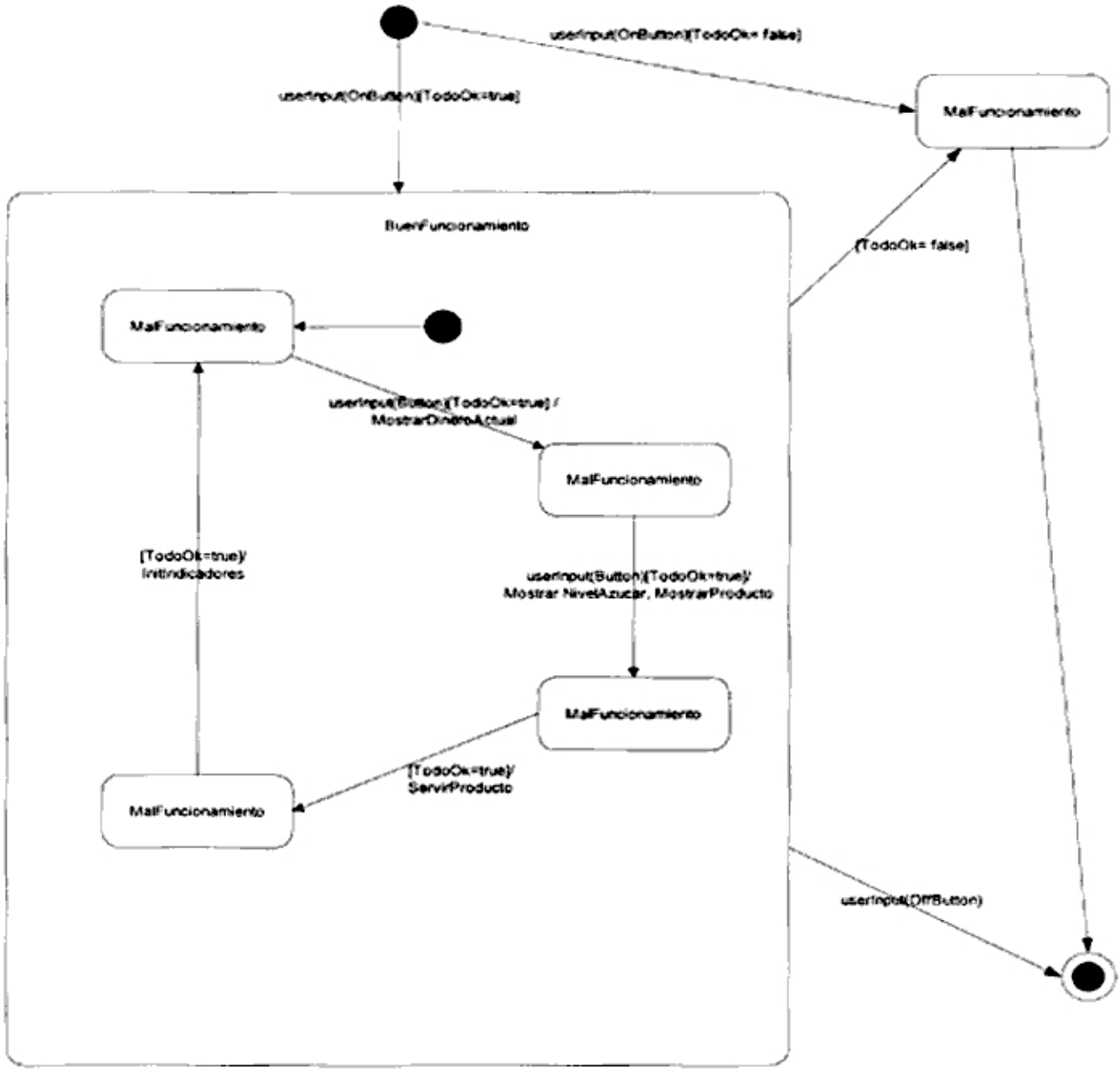

FIGURA11: *Ejemplo de un Diagrama de Estados para una máquina de café.*

(Página Web – Elementos Notacionales del UML 1.0)

ESTADO

Identifica un periodo de tiempo del objeto (no instantáneo) en el cual el objeto esta esperando alguna operación, tiene cierto estado característico o puede recibir cierto tipo de estímulos. Se representa mediante un rectángulo con los bordes redondeados, que puede tener tres compartimientos: uno

para el nombre, otro para el valor característico de los atributos del objeto en ese estado y otro para las acciones que se realizan al entrar, salir o estar en un estado.

EVENTOS

Es una ocurrencia que puede causar la transición de un estado a otro de un objeto. Esta ocurrencia puede ser una de varias cosas:

- Condición que toma el valor de verdadero o falso.
- Recepción de una señal de otro objeto en el modelo.
- Recepción de un mensaje.
- Paso de cierto periodo de tiempo, después de entrar al estado o de cierta hora y fecha particular.

El nombre de un evento tiene alcance dentro del paquete en el cual esta definido, no es local a la clase que lo nombre.

ENVIO DE MENSAJES

Además de mostrar y transición de estados por medio de eventos, puede representarse el momento en el cual se envían mensajes a otros objetos. Esto se realiza mediante una línea punteada dirigida al diagrama de estados del objeto receptor del mensaje. Si tomamos como ejemplo un control

remoto que puede enviar ordenes de encender o apagar al televisor o a la videograbadora se puede obtener un diagrama de estados como el siguiente:

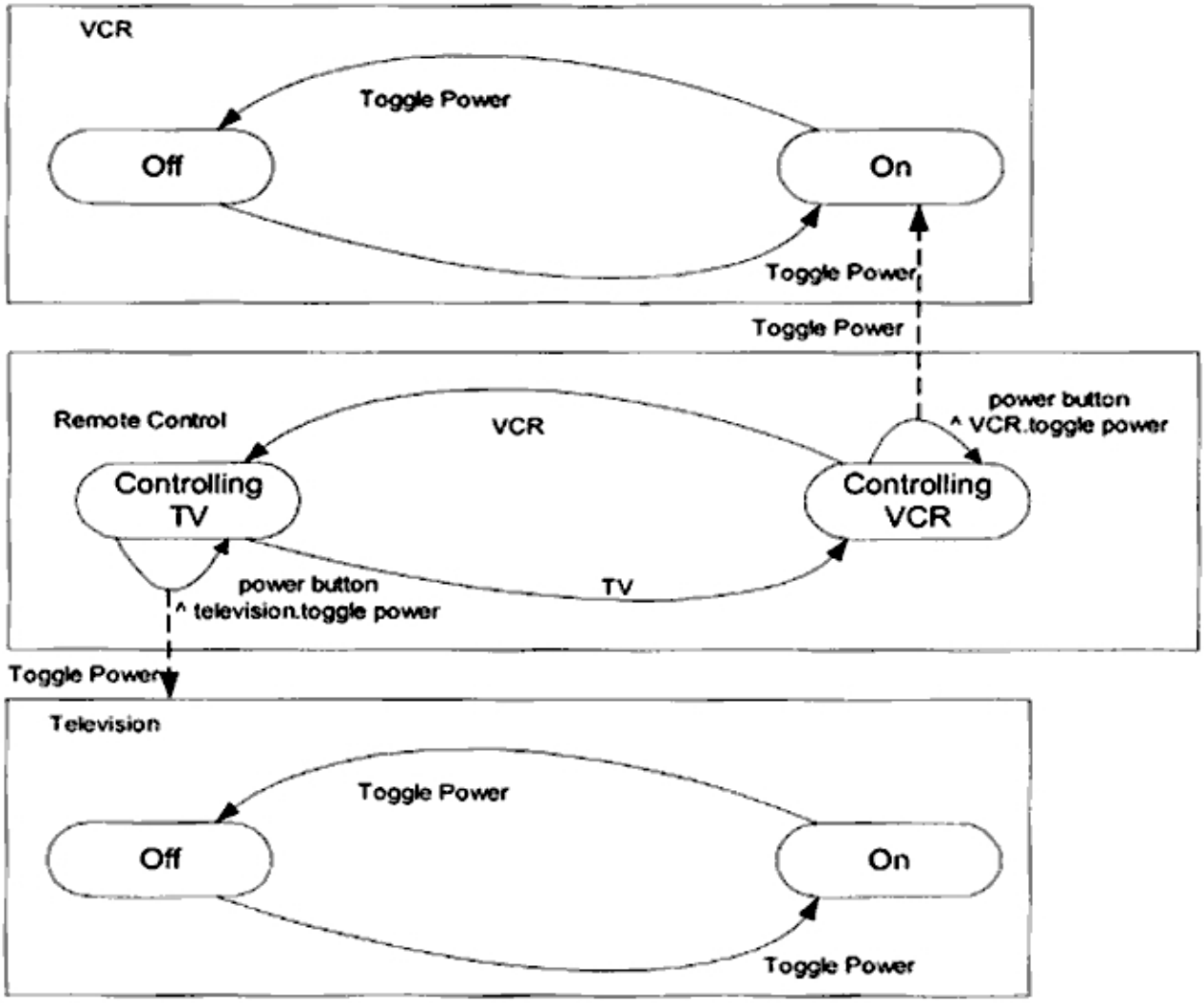

FIGURA12: *Ejemplo de un envio de mensajes.*

(Página Web – Elementos Notacionales del UML 1.0)

TRANSICIÓN SIMPLE

Una transición simple son una relación ente dos estados que indican que un objeto en el primer estado puede entrar al segundo estado y ejecutar ciertas operaciones, cuando un evento ocurre y si ciertas condiciones son satisfechas. Se representa como una línea sólida entre dos estado, que puede venir acompañada de un texto con el siguiente formato:

event-signature '[' guard-condition] '/' action-expression '^' send-clause

'event-signature' es la descripción del evento que da a lugar la transición, 'guard-condition' son las condiciones adicionales al evento necesarias para que l transición ocurra, 'actino-expression' es un mensaje al objeto o a otro objeto que se ejecuta como resultado de l transición y el cambio de estado, por ejemplo, el envío de eventos a otros paquetes o clases.

En el caso del ejemplo inicial de esta hoja se tiene una transición entre los estados 'IntroduciendoMoneda' y 'SeleccionadoAzucaryProducto' que tiene una transición con el siguiente detalle:

Userinput(Button)| [TodoOk=True] MostrarNivel Azucar, MostrarProducto

El evento que dispara el cambio de estado es 'userinput'. Se requiere como condición adicional que no se haya detectado ninguna falla (TodoOk = True) y se ejecuta 'MostrarNivelAzucar' y 'MostrarProducto', que deberían ser ejecutados por el objeto al cual pertenece el diagrama.

D. CONCEPTOS DE UN DIAGRAMA DE IMPLEMENTACION

Un diagrama de implementación muestra la estructura del código (Diagrama de Componentes) y la estructura del sistema de ejecución (Diagrama de ejecución)

DIAGRAMA DE COMPONENTES

Un diagrama de componentes muestra las dependencias lógicas entre componentes software, tanto componentes fuentes, binarios o ejecutables. Los componentes de software tienen tipo, que indica si son útiles en tiempo de compilación, enlace o ejecución. Se consideran en este tipo de diagramas solamente tipos de componentes. Instancias especificas se encuentran en el diagrama de ejecución.

Se representa como un grafo de componentes software unidos por medio de relaciones de dependencia (generalmente de compilación). Puede mostrar

también contenencia de entre los componentes software e interfaces soportadas. Un ejemplo es el siguiente:

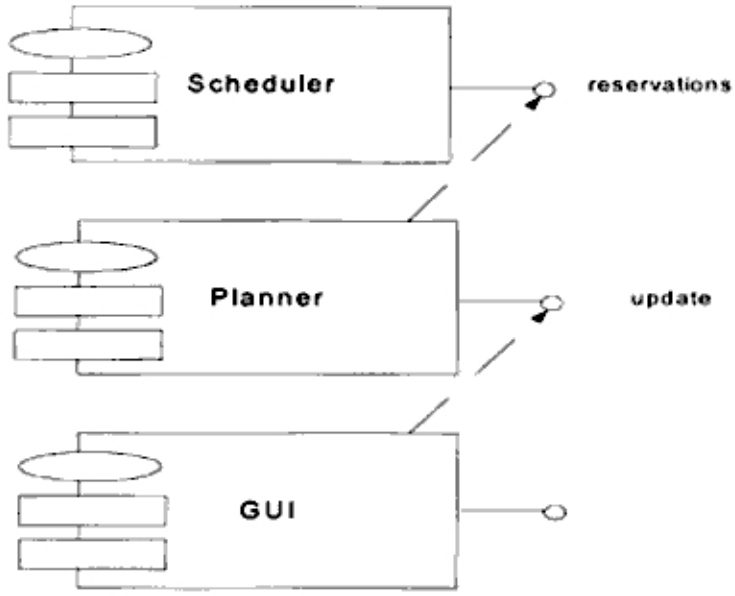

FIGURA13: *Ejemplo de un Diagrama de Componentes.*

(Página Web – Elementos Notacionales del UML 1.0)

En este caso tenemos tres componentes, GUI dependiendo de la interfaz 'update' provista por 'Planner', 'Planner' dependiendo de la interfaz 'reservations' por 'Scheduler'.

DIAGRAMA DE EJECUCIÓN

Un diagrama de ejecución muestra la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes software, procesos y objetos que se ejecutan en ellos. Instancias de los componentes software representan manifestaciones en tiempo de ejecución del código. Componentes que solamente sean utilizados en tiempo de compilación deben mostrarse en el diagrama de componentes.

Un diagrama de ejecución es un grafo de nodos conectados por asociaciones de comunicación. Un nodo puede contener instancias de componentes software, objetos, procesos(un caso particular de un objeto). Las instancias de componentes software pueden estar unidos por relaciones de dependencia, posiblemente a interfaces.

Un diagrama de ejecución es el siguiente:

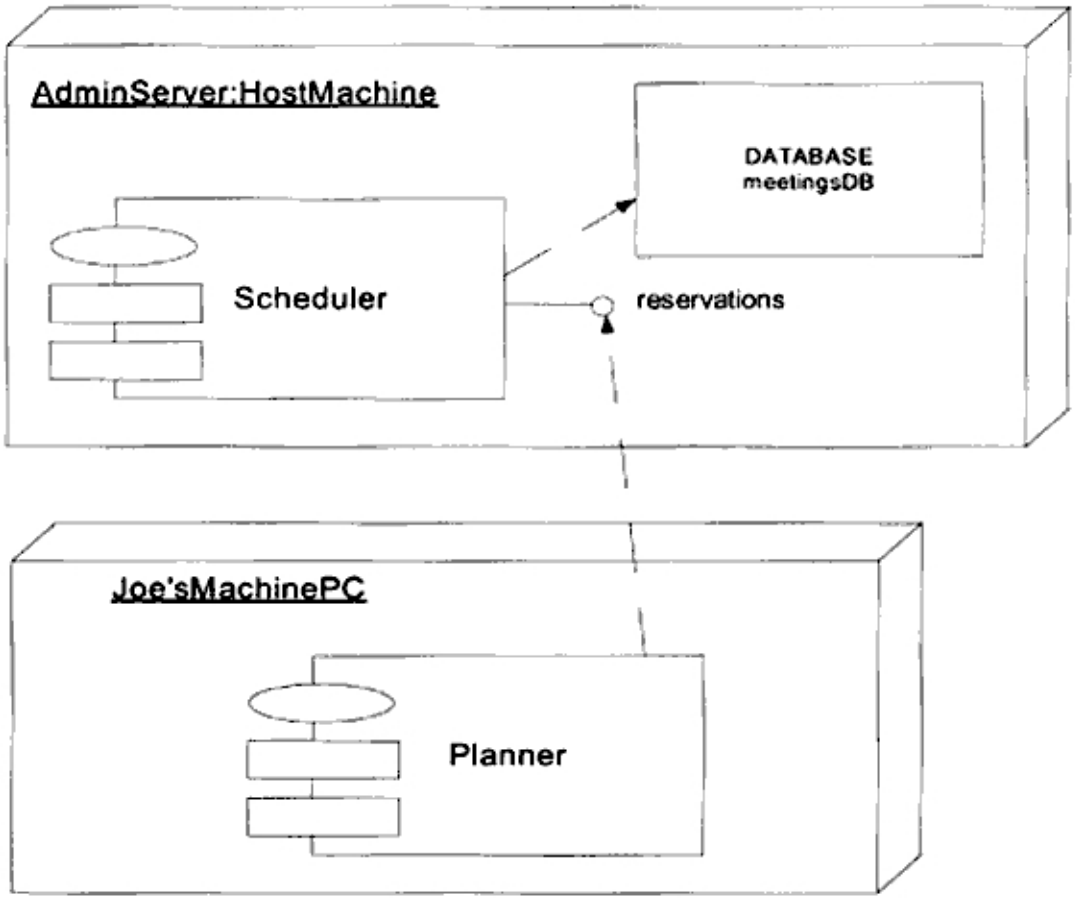

FIGURA14: *Ejemplo de un Diagrama de Ejecución.*

(Página Web – Elementos Notacionales del UML 1.0)

En este caso se tienen dos nodos, 'AdminServer' y 'Joe'sMachine'. 'AdminServer' contiene la instancia del componente 'Scheduler' y un objeto activo (proceso) denominado 'meetingsDB'. En 'Joe'sMachine' se encuentra la instancia del componente software 'Planner', que depende de la interfaz 'reservations', definida por 'Scheduler'.

NODOS

Un nodo es un objeto físico en tiempo de ejecución que representa un recurso computacional, generalmente con memoria y capacidad de procesamiento. Pueden representarse instancias o tipos de nodos. Se representa como un cubo 3D en los diagramas de implementación.

COMPONENTES

Un componente representa una unidad de código (fuente, binario o ejecutable) que permite mostrar las dependencias en tiempo de compilación y ejecución. Las instancias de componentes de software muestran unidades de software en tiempo de ejecución y generalmente ayudan a identificar sus dependencias y su localización en nodos.

E. CONCEPTOS BÁSICOS DE UN DIAGRAMA DE SECUENCIA

Un diagrama de secuencia muestra interacciones ordenadas en una secuencia de tiempo. Esta descripción es importante porque puede dar detalle a los casos de uso, aclarándolos al nivel de mensajes de los objetos existentes, como también muestra el lado de los mensajes de las clases diseñadas en el contexto de una operación.

A continuación se muestra un ejemplo de diagrama de secuencia, que da detalle al caso de uso 'PedirProducto' del ejemplo de la cafetera.

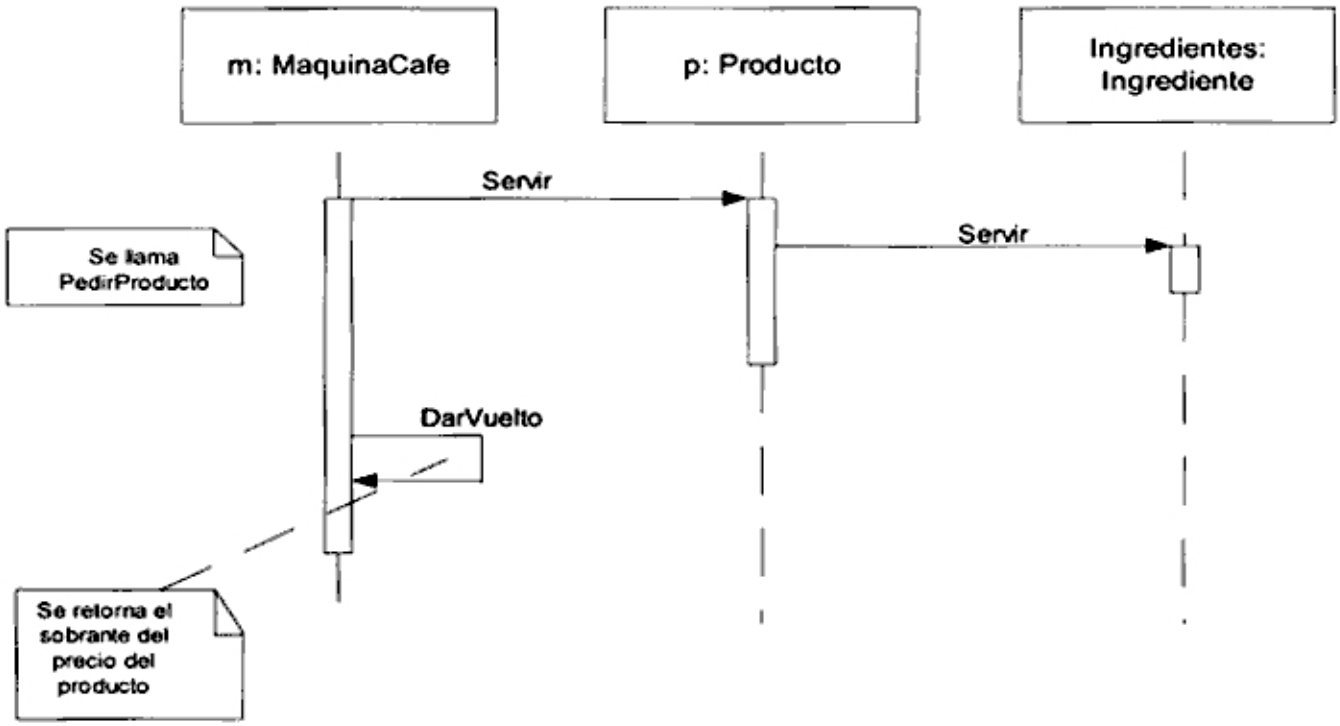

FIGURA 15: *Ejemplo de un Diagrama de Secuencia.*

(Página Web – Elementos Notacionales del UML 1.0)

LINEA DE VIDA DE UN OBJETO

Un objeto se representa como una línea vertical punteada con un rectángulo de encabezado y con rectángulos a través de la línea principal que denotan la ejecución de métodos (véase activación). El rectángulo de encabezado contiene el nombre del objeto y el de su clase, en un formato 'nombreObjeto': 'nombreClase'. Por ejemplo, el objeto m, instancia de la

clase 'MaquinaCafe' envia dos mensajes seguidos para dar respuesta a la operación 'PedirProducto': servir al objeto p de la clase 'Producto' y 'DarVuelto' a sí mismo.

ACTIVACION

Muestra el periodo de tiempo en el cual el objeto se encuentra desarrollando alguna operación, bien sea por sí mismo por medio de delegación a alguno de sus atributos. Se denota como un rectángulo delgado sobre la línea de vida del objeto. En el ejemplo anterior el objeto Ingredientes se encuentra activado mientras ejecuta el método correspondiente al mensaje Servir (que ejecuta 'IngredientesServir') y el objeto m se encuentra activo mientras se ejecuta 'p.Servir' y 'DarVuelto'.

F. CONCEPTOS DE UN DIAGRAMA DE ACTIVIDADES

Un diagrama de actividades es un caso especial de un diagrama de estados en el que todos los estados son de acción (identifican que acción se ejecuta al estar en él) y todas las transiciones son enviadas al terminar la acción ejecutada en el estado anterior. Puede dar detalle a un caso de uso, un objeto o un mensaje en un objeto. Sirven para representar transacciones internas, sin hacer mucho énfasis en transacciones o eventos externos. Se presenta a continuación un ejemplo de un diagrama de actividades para un mensaje de un objeto. Generalmente modelan los pasos de un algoritmo.

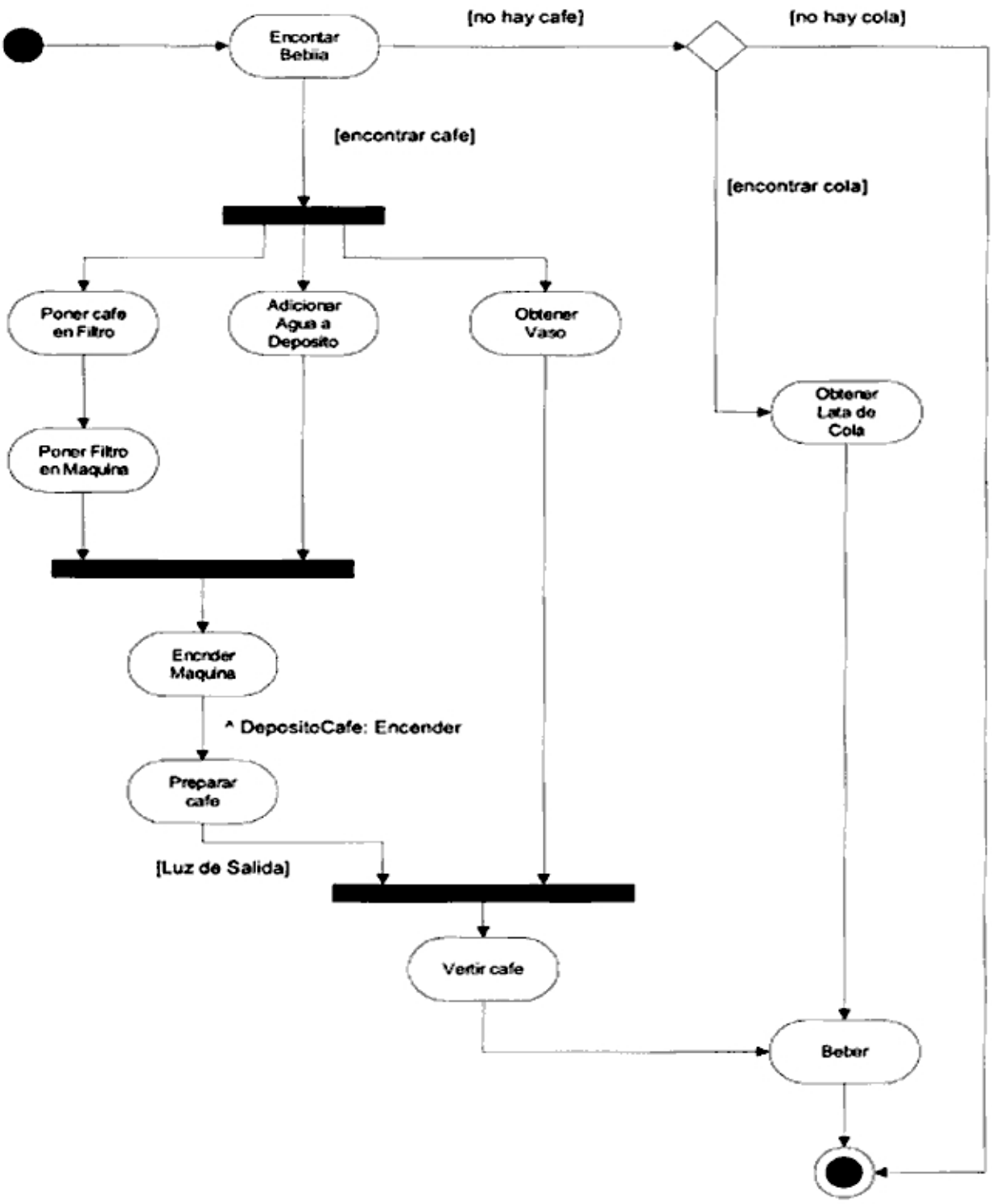

FIGURA16: *Ejemplo de un Diagrama de Actividades.*

(Página Web – Elementos Notacionales del UML 1.0)

ESTADO DE ACCION

Representa un estado de acción interna, con por lo menos una transición que identifica la culminación de la acción (por medio de un evento implícito). No deben tener transiciones internas ni transiciones basadas en eventos (sí éste es el caso, representarlo en un diagrama de estados). Permite modelar un paso dentro del algoritmo.

TRANSICIONES

Las flechas entre estados representan transiciones con evento implícito. Pueden tener una condición en el caso de decisiones.

DECISIONES

Se representa mediante una transición múltiple que sale de un estado, donde cada camino tiene una etiqueta distinta. Se representa mediante un diamante al cual llega la transición del estado inicial y del cual salen las múltiples transiciones de los estados finales. Un ejemplo se ve en la figura cuando no hay café y se toma una decisión entre hay cola y no hay cola.

3.3. ESCENARIO

Esta aplicación ha sido desarrollada para el entorno de la Bolsa de Valores de Lima, los procedimientos que se llevan acabo en la aplicación tratan información relativa a cotizaciones de valores y bonos, operaciones bursátiles, índices bursátiles y "*Brokers*".

Las estructuras de los mensajes que fluyen a través de toda la aplicación van de acuerdo justamente a las cotizaciones, operaciones e índices.

Dentro de la Bolsa de Valores, existe una aplicación llamada Elex, esta aplicación registra todas las operaciones bursátiles que se ejecutan en el mercado de valores, de acuerdo a como se dan estas operaciones y a ciertos criterios realizan modificaciones en las cotizaciones de los valores o bonos, así como variaciones en los índices bursátiles.

Además de la aplicación Elex existe una aplicación llamada Venser está se alimenta de la información que le envía el Elex en tiempo real durante la rueda, la información que recoge la aplicación Venser la recibe de acuerdo a ciertas estructuras, de acuerdo a esto se da modificaciones en el buffer de datos de la aplicación.

La aplicación Venser tiene como clientes a medios de comunicación tanto a nacionales como internacionales entre ellos tenemos a Bloomberg y Reuters.

Estos clientes reciben la información que les envía la aplicación Venser todo esto a través de mecanismos que utilizan el protocolo TCP/IP, esto quiere decir que estos clientes reciben información bursátil en tiempo real.

3.4. SUPUESTOS

- La aplicación Cliente fue desarrollado para la plataforma Windows, con esto nos referimos a la aplicación con la que el cliente puede interactuar, ya que existen otras aplicaciones Cliente pero que su ejecución es exclusiva para administración dentro del Servidor.

- El número de usuarios de la aplicación es limitado, ya que una de los fines de la aplicación es poder conocer a los Clientes que se conectan al servicio y desde luego llevar a cabo la administración de ellos.

- La aplicación tiene alcance a Internet.

- Se deja el Navegador Web como aplicación Cliente, se plantea una alternativa en cuanto a la interfaz del Cliente con las aplicaciones orientadas a Internet.

- No existe firewall dentro de la red, es decir, no hay ningún tipo de restricciones que se haya utilizado para la aplicación.

- La información que se transmite dentro del entorno de la aplicación, por el momento no dispone de un protocolo de seguridad, tal como SSL (Socket Secure Layer).

- Asumimos que la empresa en la que desarrollamos la aplicación cuenta con las licencias requeridas, es por eso que no las consideramos dentro de los costos estimados, en este caso nos referimos a las licencias de Visual Studio y de Oracle.

3.5. IMPORTANCIA

En muchas ocasiones el tratamiento que se le da a la información otorga armas que uno las utiliza, de acuerdo a criterios propios, para tomar decisiones. Estas decisiones en cuestión de segundos determinan la ganancia o pérdida de millones de dólares.

La exclusividad con que uno obtiene la información determina mayores probabilidades de llevar a cabo decisiones efectivas. Esta exclusividad se define en tener la información en vivo, en tiempo real.

En el escenario de la Bolsa de Valores de Lima es vital contar con la información en vivo y en directo, para que sus decisiones en cuanto a las operaciones de compra y/o venta sean las adecuadas ya que lo que el agente bursátil busca es realizar compra de paquete de acciones a una cotización baja, ó una venta de paquete de acciones a una alta cotización.

Debido al alto impacto que tienen actualmente las aplicaciones Web, y por ende los navegadores Web. Las aplicaciones tradicionales están en vías de migración a la interface Web. Para el tema de la transferencia de datos, debido a que la interface con la aplicación, como lo es el navegador, ya esta desarrollado; no otorga muchas ventajas en monitoreo y administración de la red de comunicación que se desarrolla bajo esta aplicación.

Debido a este inconveniente suscitado, se trata de instanciar nuevas interfaces de comunicación vía TCP. Esta nueva interface tendrá sus propios formatos de salida, mecanismos de conexión, recepción y emisión de información.

3.6. METODOLOGÍA DE DESARROLLO

Para llevar a cabo el desarrollo de la aplicación (que conforma la solución propuesta para el caso de estudio de la Bolsa de Valores de Lima), se definieron una serie de métodos que intervinieron en la implementación de la

solución propuesta con el propósito de cubrir las limitaciones encontradas en la solución Web.

ANALISIS Y DISEÑO

Para esta etapa se optó por utilizar figuras y diagramas de modelamiento, tales como el diagrama de bloques, los diagramas del UML, un gráfico que representa la arquitectura física de la aplicación. Esto con el objeto de obtener una idea clara de lo que se espera de la aplicación.

- Diagrama de Bloques. Los procedimientos que ejecutan cada uno de los módulos que conforman la aplicación son esquematizados en este diagrama, por ejemplo se muestran los procedimientos de la aplicación Servidor.

- Diagrama de Clases. Aquí se muestra el diseño lógico de la aplicación, desde el punto de vista de los objetos, se muestran las distintas clases utilizadas y desarrolladas para la aplicación. Como punto aparte se menciona que el lenguaje de programación utilizado para lograr este diseño, desde el punto de vista del UML, es Java.

- Diagrama de Actividades. Se muestran las distintas tareas que involucra a la aplicación dentro del contexto de una Rueda de Bolsa. Desde que se inicia tal (8:51am) hasta el fin del mismo (16:10pm).

- Diagrama de Casos de Uso. En este diagrama se muestran las distintas relaciones que existen entre los actores, que toman un carácter dinámico dentro de la aplicación. En este caso de estudio, Rueda de Bolsa, se tomaron como actores a la aplicación Servidor, a la aplicación Cliente y a la Tabla de Memoria.

- Diagrama de Componentes. Denota la interacción entre los programas (ejecutables), las librerías (necesarias para la ejecución de programas), el hardware (tanto para la aplicación Servidor como para la aplicación Cliente) y el medio de comunicación (la red y el protocolo TCP/IP).

A diferencia del Diagrama Físico, los distintos diagramas constituyen diferentes vistas hacia la aplicación a desarrollar, estas vistas reflejan el funcionamiento de la aplicación, el posicionamiento de los distintos componentes dentro de la aplicación así como la interacción de los distintos actores dentro de la aplicación.

Luego de haber desarrollado estos diagramas se procedieron a implementar el formato de los mensajes, los identificadores para cada tipo de mensaje, el control del "checksum" ó de la longitud de los mensajes en el equipo remoto, para evitar perdida de información.

ESPECIFICACIÓN DE PROCEDIMIENTOS POR CADA CAPA

Para la aplicación se especificaron tres capas, la Base de Datos, la aplicación Servidor y las distintas aplicaciones Cliente. Para la Base de Datos, sus procedimientos ya se encuentran establecidos. En cambio para los módulos a desarrollar se tienen que especificar sus procedimientos ó tareas a ejecutar por cada uno de ellos.

Para la aplicación Servidor, entre sus tareas elementales vienen a corresponder la ejecución del mismo, de manera simultánea, con el inicio de la Rueda de Bolsa. Además debe enviar los mensajes que recepciona, a todos los Clientes conectados en ese momento, así como también, tiene que almacenar estos mensajes en la Tabla de Memoria correspondiente.
Para cada aplicación Cliente, se definieron las siguientes funciones:

- Conexión a la aplicación Servidor luego de que éste se haya ejecutado satisfactoriamente.
- Descarga inicial del contenido de la Tabla de Memoria desde el Servidor.
- Solamente para el caso del Cliente Oracle, se determinó, que se llevaría a cabo la actualización de la información en la Base de Datos, así como la confirmación de ello al Servidor.

REQUERIMIENTOS DE HARDWARE Y SOFTWARE

En esta sección se especifica los elementos de Hardware y Software, que son necesarios para poder poner en acción la aplicación.

El desarrollo de la aplicación requirió de las siguientes herramientas de programación:

- Java, para el desarrollo de la aplicación Servidor.
- Visual C++, para el desarrollo de las aplicación Cliente.

Para el módulo de la aplicación Servidor, por tener éste un carácter crítico, se convino colocarlo dentro de un equipo que reúna las mínimas características de un Servidor, tanto en Hardware como en Software, contando además con una Base de Datos, que para el caso de estudio es Oracle.

Debido a la estabilidad de su Sistema Operativo para funciones y servicios TCP/IP, se decidió utilizar el Sistema Operativo Linux (Distribución Suse 7.1), que es compatible con el equipo disponible para la realización de la aplicación.

Para el caso de la aplicación Cliente, el nivel de Hardware no tendría que disponer de muchos recursos, debido a que solamente se limita a conectarse a través de TCP/IP hacia la aplicación Servidor, recibir y enviar los mensajes correspondientes, sin tener contacto con la Base de Datos. Para esta aplicación Cliente se utilizó el Sistema Operativo Windows 98 junto con procesadores a partir de Pentium II.

IMPLEMENTACION DEL PROTOTIPO

En la implementación del prototipo se consideró desarrollar los módulos de tal manera que realicen los procedimientos elementales que se definieron en el ítem anterior. Tomando como referencia la información que fluye debido a la Rueda de la Bolsa, se desarrollaron mecanismos, de tal modo que se utilicen para construir la fuente de datos inicial que alimenta a las Tablas de Memoria del Servidor, así como la elaboración de mensajes para realizar una simulación de nuestra aplicación.

Los módulos del Servidor y las librerías, que se desarrollaron y utilizaron para su ejecución, se diseñaron bajo entorno Java (JDK1.3.1 Distribución de Sun Microsystems). En cambio los módulos correspondientes al Cliente se desarrollaron bajo Visual C++.

PRUEBAS DEL PROTOTIPO

Las pruebas se realizaron en dos ambientes distintos, en primer lugar se desarrollaron en una red local que consistía en una PC donde se encontraba el módulo de la aplicación Servidor, el Cliente Oracle y la Base de Datos; y una Laptop que albergaba a la aplicación Cliente; estos equipos estaban conectados mediante un Cable Crossover.

Para este caso las pruebas se realizaron se realizaron de manera satisfactoria. Por otro lado, en el entorno Bolsa de Valores de Lima, se encontraron dificultades que conciernen a tareas de mantenimiento del Servicio de Red, que provocaba aislamiento en la comunicación entre los equipos de prueba.

Mediante una herramienta de "*scan*" de información se observaba como ésta (información) viajaba en la aplicación de forma vulnerable a cualquier ente. Es decir la información viajaba de manera insegura, esto demuestra la necesidad de mecanismos de encriptación de la información.

En conclusión, las pruebas se desarrollaron exitosamente, lográndose el objetivo principal que era lograr el flujo sincronizado de la información y sobre todo realizarlo en tiempo real.

3.7. CRONOGRAMA DE ACTIVIDADES

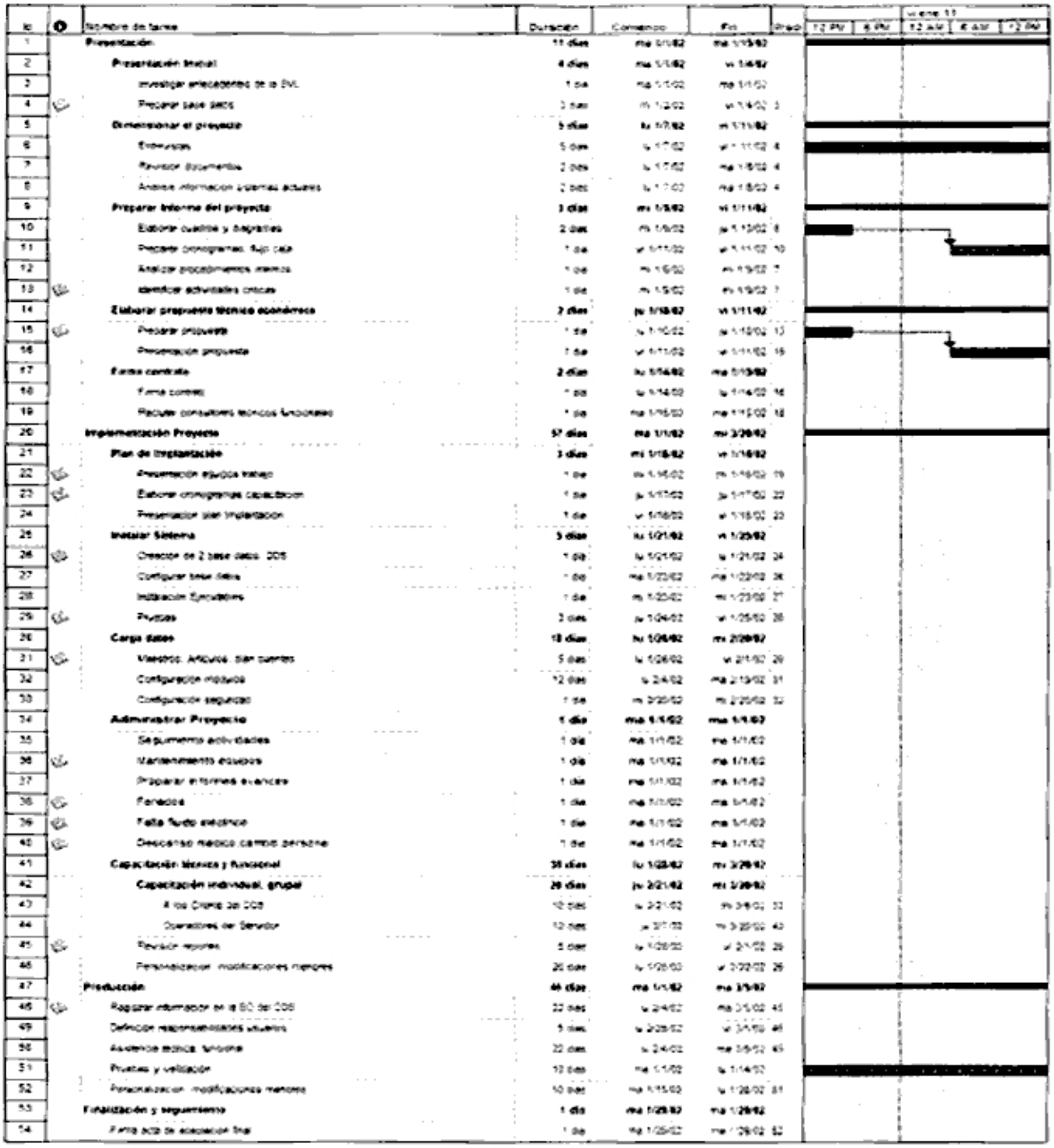

FIGURA17: *Cronograma de Actividades.*

(Fuente Propia)

CAPITULO IV

DESARROLLO DE LA SOLUCION PROPUESTA

4.1. ANÁLISIS Y DISEÑO

4.1.1. DIAGRAMAS DEL MODELO

1. Diagrama de Bloques

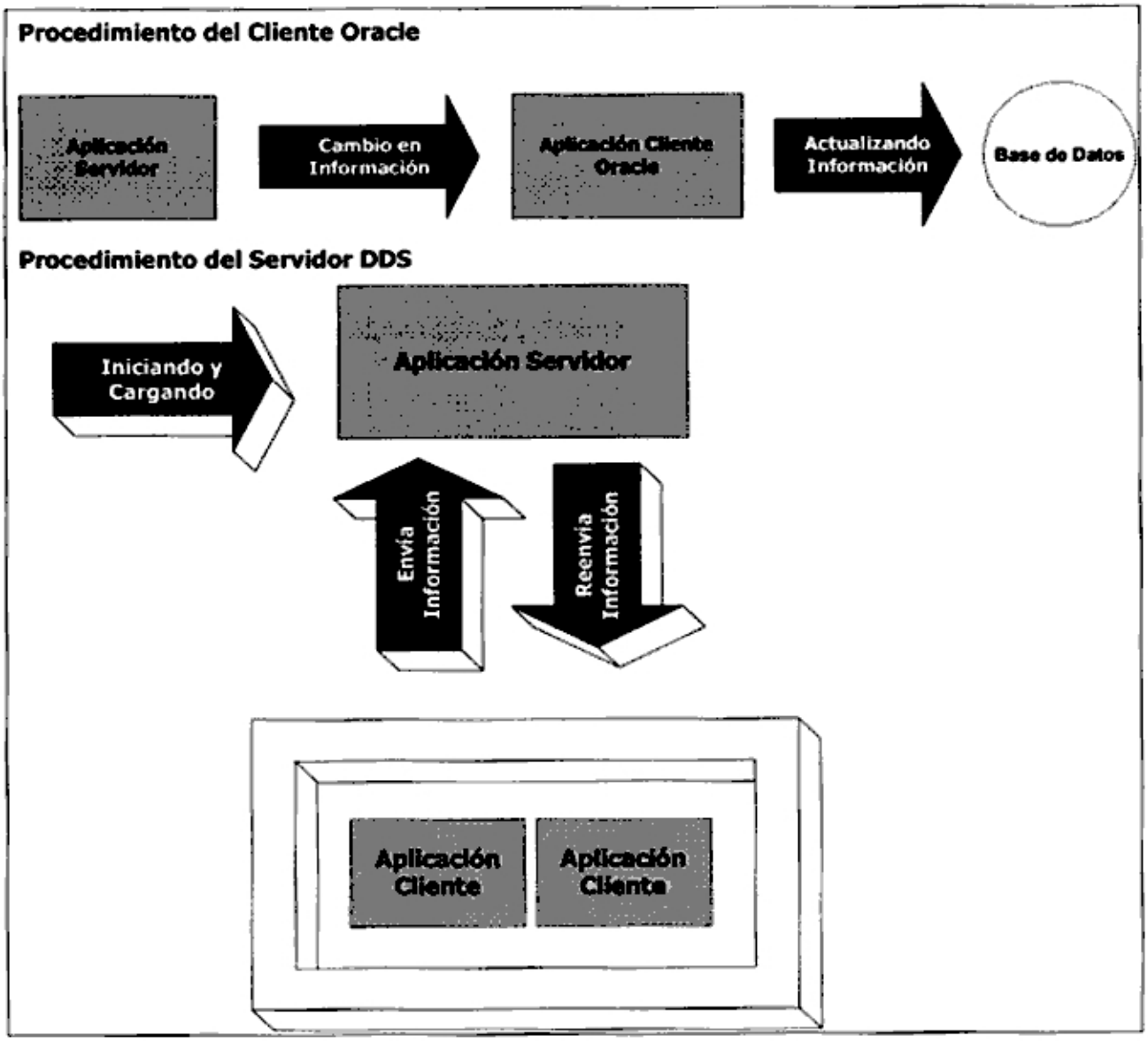

FIGURA18: *Diagrama de Bloques.*

(Fuente Propia)

2. Diagrama de Clases

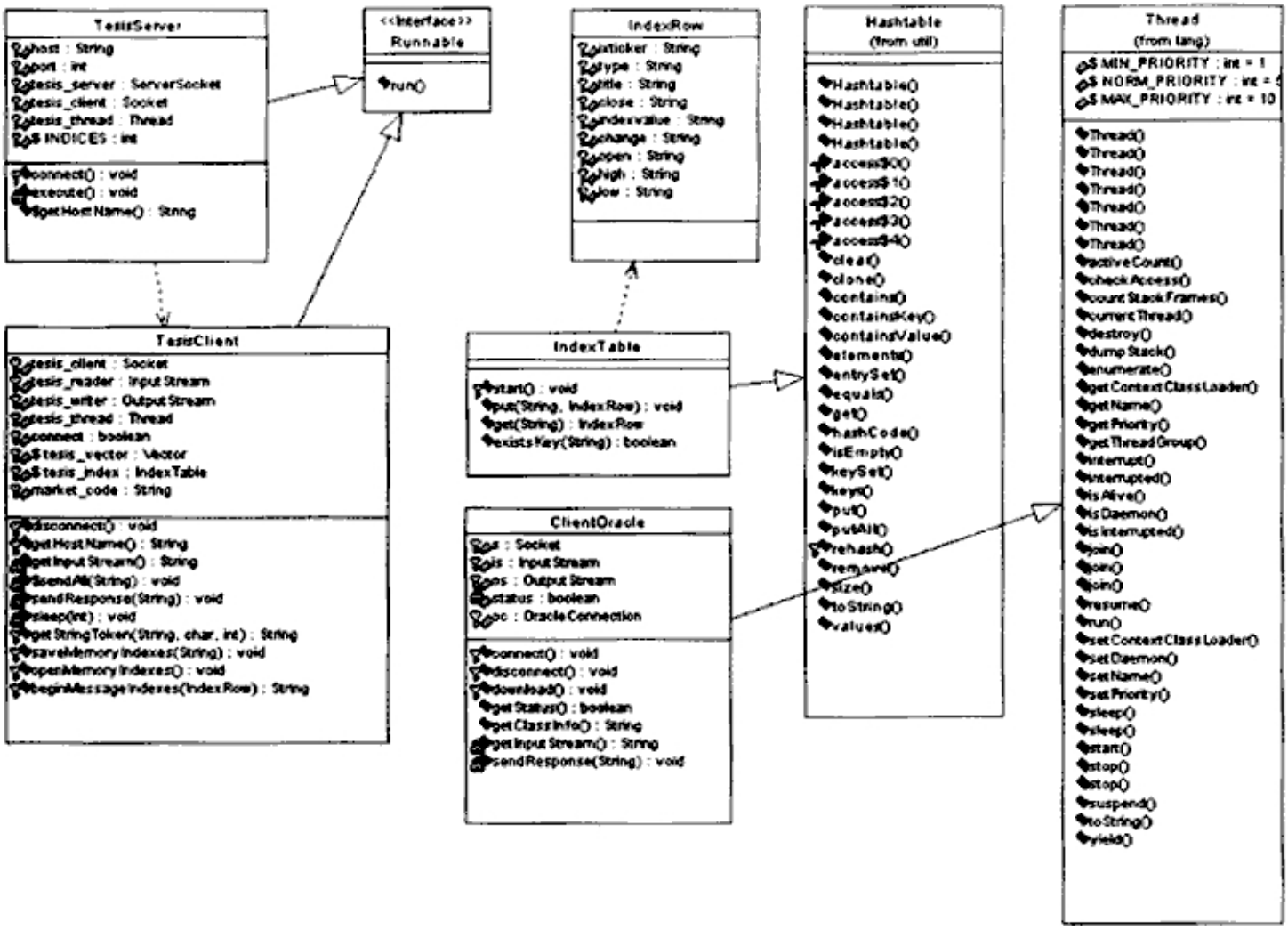

FIGURA19: *Diagrama de Clases.*

(Fuente Propia)

3. Diagrama de Actividades

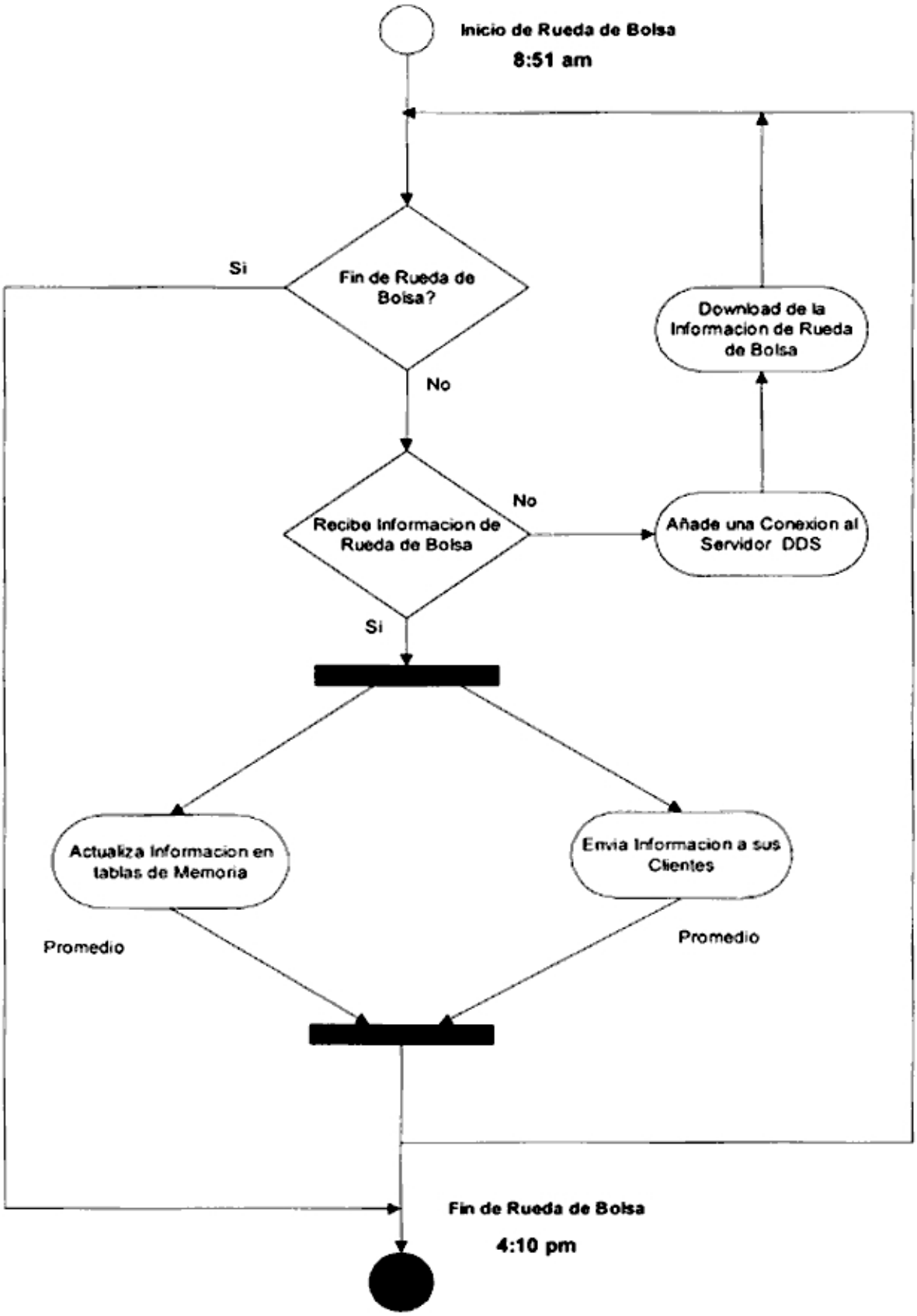

FIGURA 20: *Diagrama de Actividades.*

(Fuente Propia)

4. Diagrama de Casos de Uso

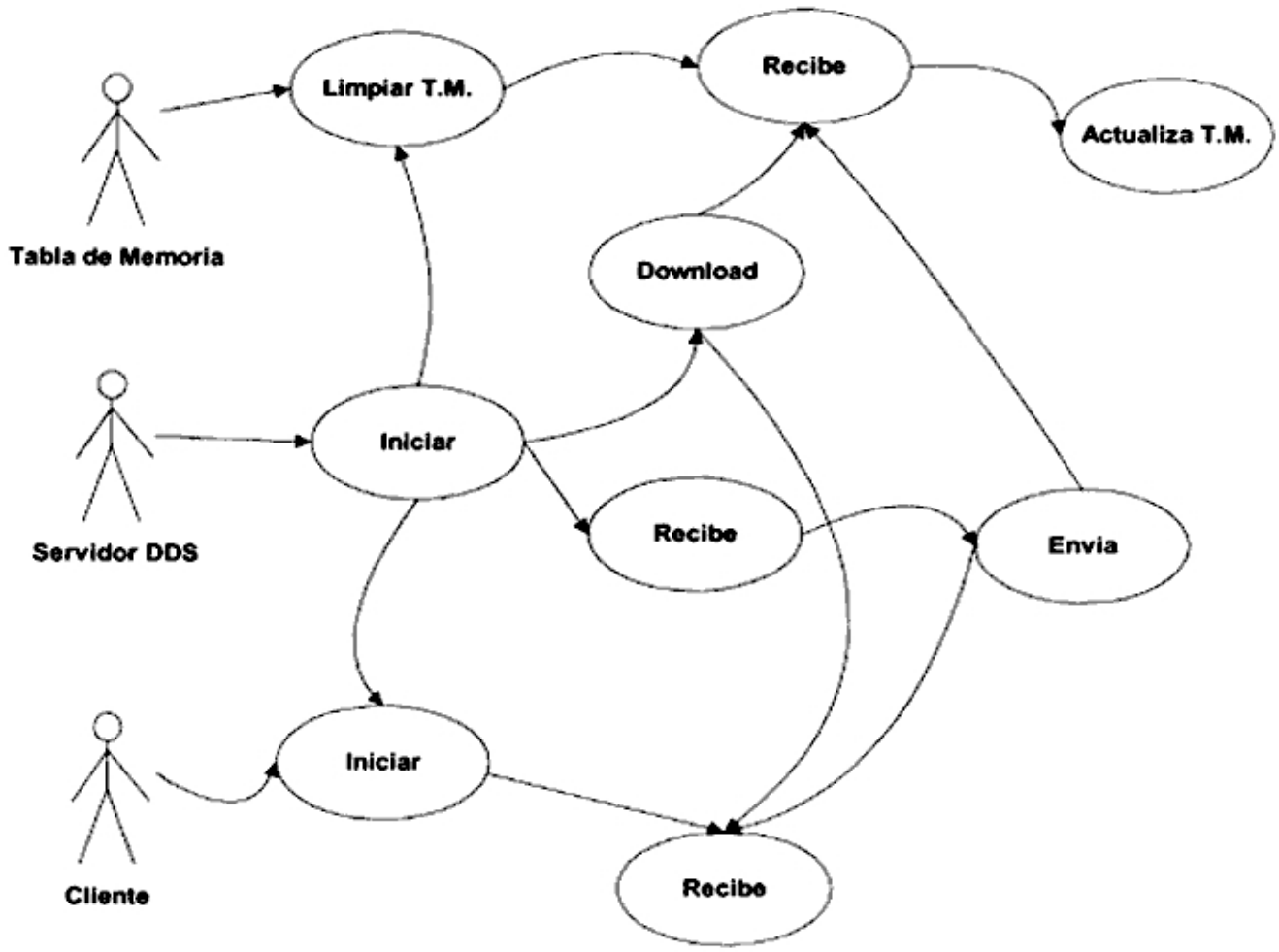

FIGURA 21: *Diagrama de Casos de Uso.*

(Fuente Propia)

5. Diagrama de Componentes

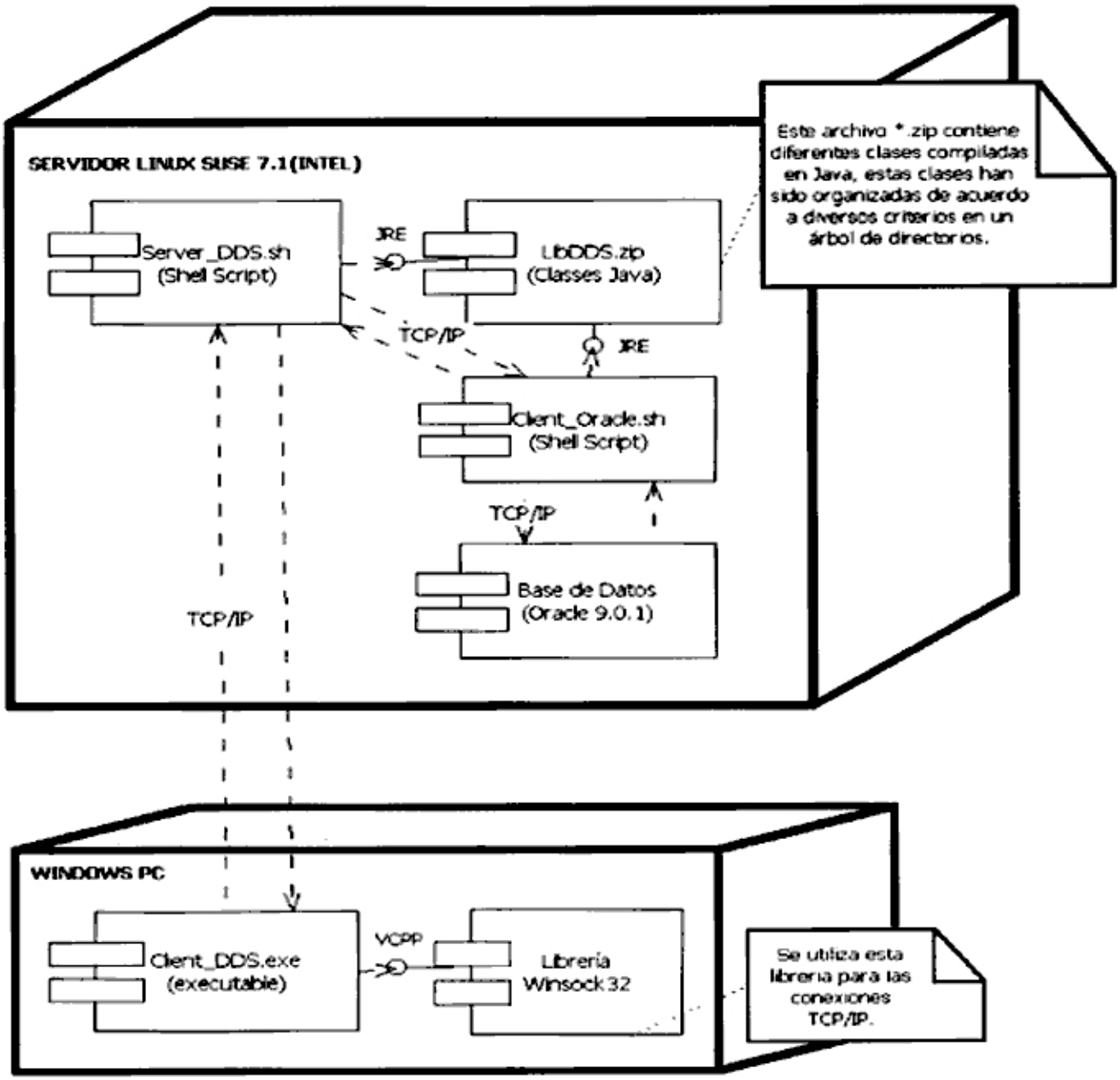

***FIGURA 22:** Diagrama de Componentes.*

(Fuente Propia)

4.1.2. ARQUITECTURA

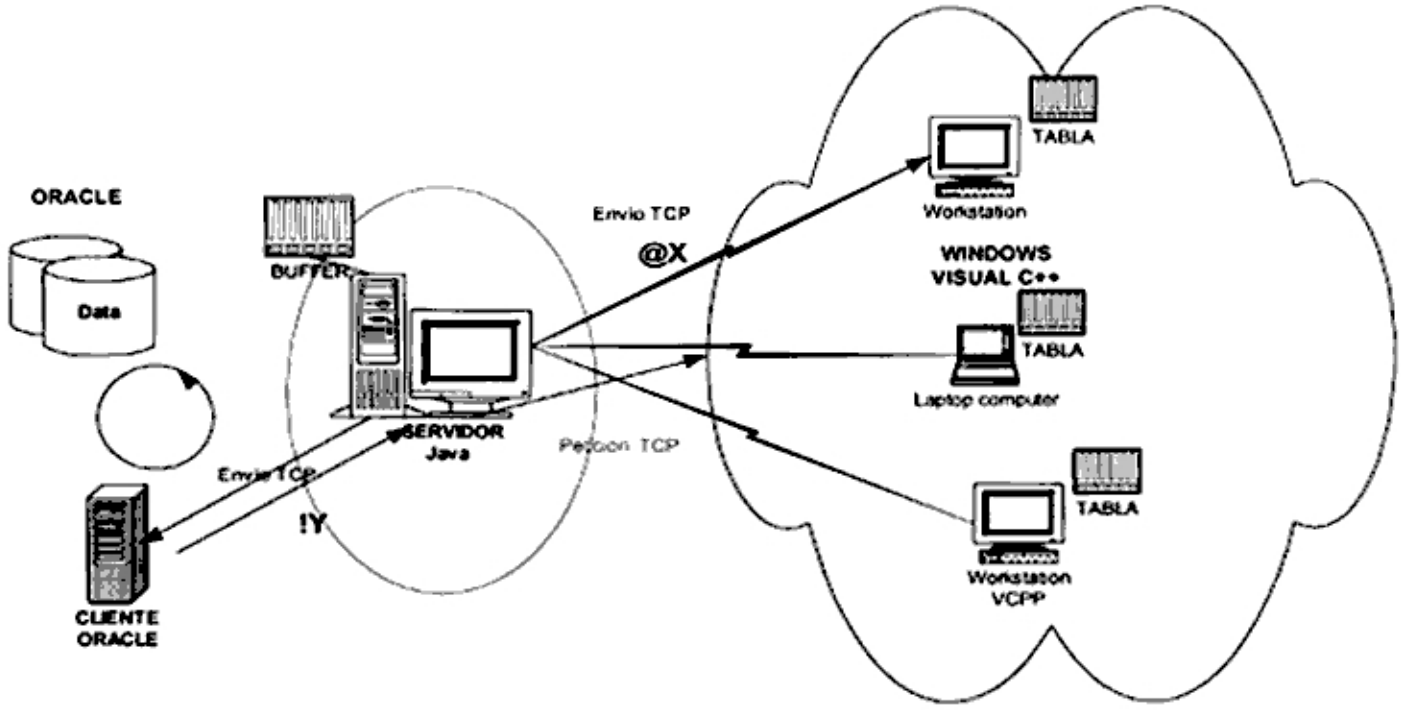

FIGURA 23: *Arquitectura de la aplicación Cliente / Servidor.*

(Fuente Propia)

4.1.3. ESTRUCTURA DE LOS MENSAJES DE DATOS

El mensaje de datos es la forma como las personas pueden ponerse en contacto para desarrollar un lenguaje, un intercambio de signos que puedan entender y, de los cuales puede originar una transacción.

Los mensajes de datos en su estructura genérica se define de la siguiente manera:

[HEADER]|[BODY]

En cuanto a los caracteres delimitadores entre campo de datos se convino utilizar el carácter '|'.

Los mensajes de datos que se establecieron para la aplicación Servidor se componen de dos partes:

- HEADER (cabecera), en esta parte del mensaje se guarda el código descriptor de la tabla de datos ó el tipo de información del que se envía; por ejemplo tenemos hasta el momento solamente un buffer de datos que corresponde a la tabla de índices, diseñados tanto los mensajes como los componentes para su almacenamiento, no confundir los buffer de datos con las tablas de datos. El buffer de datos es aquel que se manipula a nivel aplicación Servidor, en cambio la tabla de datos es el que almacena en la Base de Datos.

- BODY (cuerpo de datos), en esta parte del mensaje se almacena la información que es de importancia para la aplicación Cliente. Por ejemplo para el buffer de datos relativo a Índices Bursátiles se tiene la siguiente estructura.

[TICKER] | [TIPO] | [TITULO] | [CIERRE] | [ACTUAL] | [VARIACION] | [APERTURA] | [MAXIMA] | [MINIMA]

Como se observa son campos que identifican datos propios de los Índices Bursátiles.

Para observar como son los mensajes que se manejan en esta aplicación se muestra la siguiente figura.

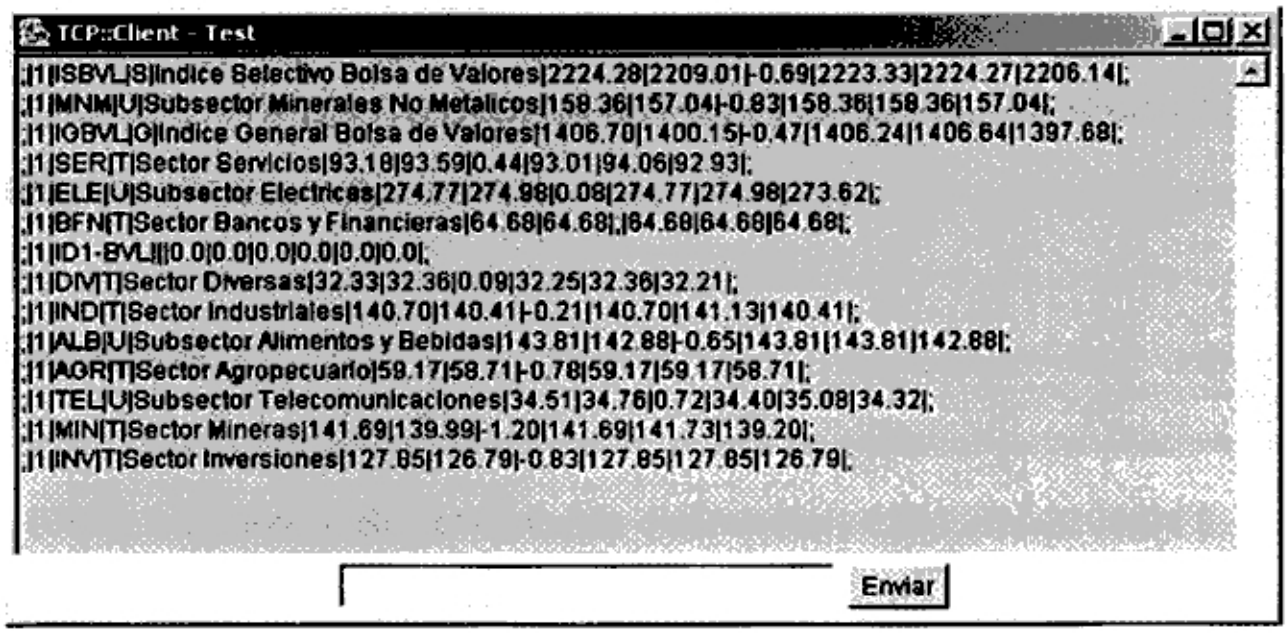

FIGURA 24: *Aplicación extra, esta se desarrolló con el objetivo de tener una forma de visualizar las estructuras de los mensajes de datos.*

(Fuente Propia)

Como se observa en la figura se tienen distintos mensajes, en cada unos de los mensajes se distinguen el primer campo que identifica un inicio de

mensaje que es el carácter ';' y seguidamente se observa el código descriptor del buffer de datos que es '1', esto indica que se trata de información relativa a la tabla de Índices Bursátiles.([5])

Como se definió anteriormente el campo '1' es la cabecera, a continuación sigue el cuerpo de datos, tomando la estructura del cuerpo de datos y comparando con el primer mensaje mostrado en la figura, tenemos:

ESTRUCTURA DEL CUERPO DE DATOS(BODY)

[TICKER] | [TIPO] | [TITULO] | [CIERRE] | [ACTUAL] | [VARIACION] | [APERTURA] | [MAXIMA] | [MINIMA]

MENSAJE

ISBVL | S | Indice Selectivo Bolsa de Valores | 2224.28 | 2209.01 | -0.69 | 2223.33 | 2224.27 | 2206.14

4.2. ESPECIFICACION DE PROCEDIMIENTOS

PROCEDIMIENTOS EJECUTADOS DEL LADO DEL CLIENTE ORACLE

- Cuando la información que fluye por la red sufre modificaciones tanto por la aplicación Cliente como por la aplicación Servidor, estas

[5] Referencia : Documento Interno – Distribución de Datos a los Vendors

modificaciones tienen como destino a la aplicación Cliente Oracle, que a su vez los actualiza en la Base de Datos.

- Al siguiente día cuando comienza una nueva rutina, la aplicación Servidor le solicita la última actualización, y la aplicación Cliente se la envía.

- La función de la aplicación Servidor en el Sistema, consiste en cargar la información al final del día (al final de la rutina) y en enviarla al inicio de éste (inicio de la rutina), son las únicas intervenciones que cumple la aplicación Servidor, a menos que necesite que actúe un DBA para actualizar la base de datos en una exigencia o requerimiento determinado.

- Aquí se encuentran todas las tablas almacenadas diarias y todos los cambios realizados en el día, su función solamente es de almacenamiento, no existe una interacción directa con la Base de Datos.

PROCEDIMIENTOS EJECUTADOS DEL LADO DE LA APLICACIÓN SERVIDOR

- Se inicia el día, ejecutando la aplicación Servidor, con la última actualización de las tablas que envía la Base de Datos a la aplicación

Servidor, así es como comienza la rutina diaria, estas tablas vendrían a ser las mismas que la aplicación manda al cerrar la última actualización de las tablas, al finalizar la rutina del día anterior.

- Una vez llegada la información a la aplicación Servidor, realiza los movimientos correspondientes a ella (aplicación), gestiona las operaciones respectivas y envía la información procesada a las aplicaciones Cliente, de acuerdo a los requerimientos de cada una de estas.

- La aplicación Servidor enviará los mensajes recibidos a cada uno de las aplicaciones Cliente conectadas al servicio.

- A partir de los mensajes recibidos por la aplicación Servidor, éste actualiza las tablas de memoria; luego son transmitidos a las aplicaciones Cliente conectadas.

PROCEDIMIENTOS EJECUTADOS DEL LADO DE LA APLICACIÓN CLIENTE

- Al iniciar la aplicación Cliente, esta se ejecuta correctamente si la aplicación Servidor está ejecutado, si es así se envía una petición y se ejecuta un "*download*" previo con la información de cierre del día anterior.

- Cada aplicación Cliente necesitará tablas determinadas a sus necesidades.

4.3. REQUERIMIENTOS HARDWARE Y SOFTWARE

- ***MODULO SERVIDOR***

HARDWARE

Pentium IV de 1.8Ghz.

512 MBytes de memoria RAM.

Disco duro libre de 1GB.

Tarjeta de red Ethernet.

SOFTWARE

Plataforma: Linux (Distribución Suse 7.1)

DBMS Oracle (Versión 8.1.7)

Herramienta de Desarrollo: JSDK 2.0 (Java Development Kit)

- ***MODULO CLIENTE***

HARDWARE

Pentium de 350 Mhz ó superior.

64Mbytes de memoria RAM.

Disco duro libre de 10 MB.

Tarjeta de red Ethernet.

Nota: Si existe firewall en la red de la aplicación Cliente, se debe habilitar la salida al puerto 7000.

SOFTWARE

Plataforma: Windows (Todas las versiones).

Herramienta de Desarrollo: Visual C++.

Servicio de Internet.

4.4. SÍNTESIS

Se tomaron en cuenta 3 capas: la primera, la Base de Datos; la segunda, la aplicación Servidor; y la tercera, la aplicación Cliente. En cuanto a la aplicación Servidor se menciona que las tareas ejecutadas por éste se limitan a recibir información, guardar en un buffer y enviar a todos las aplicaciones Cliente conectados a él.

Las aplicaciones Cliente tienen dos ambientes de ejecución, en el ambiente Windows, tenemos a los clientes Visual C++, que simplemente observan la información guardada en tiempo real. En el módulo Servidor, se tiene al

cliente Oracle, este cliente se encarga de actualizar la Base de Datos con la información que le envía la aplicación Servidor.

La información que estamos utilizando para el desarrollo de esta aplicación es proveniente de la Base de Datos que se utiliza de manera exclusiva en la Rueda de Bolsa.

La aplicación Servidor se ejecuta sobre plataforma Linux, mientras que las aplicaciones Cliente bajo Windows.

4.5. PRUEBAS DE PROTOTIPO

Las pruebas que se han dado para esta aplicación se realizaron en dos entornos distintos.
Las pruebas se desarrollaron en el mes Diciembre de manera constante y diaria, ya que se estaba trabajando con información que eran enviados por el Sistema Elex, en la Bolsa de Valores de Lima.

PRUEBAS EN RED LOCAL

Uno de esos entornos fue dentro de una Red Local. Los equipos que se utilizaron fueron los siguientes:

- 1 PC PENTIUM IV con SERVIDOR SUSE LINUX V 7.1

- 1 LAPTOP PENTIUM IV WINDOWS XP
- CABLE CROSSOVER

Las pruebas se iniciaron, con el levantamiento de la aplicación del servidor desde LINUX, seguidamente se procedió a levantar el cliente Oracle, en vista de que se estaba dentro de una Red Local y no dentro de la Red de Bolsa de Valores, se procedió a desarrollar dentro de la aplicación Cliente Oracle una tarea acerca de la lectura de un archivo "*log*" que se genera dentro del Sistema Elex de la Bolsa de Valores, este archivo guarda todos los mensajes que envía el Sistema Elex a la aplicación Servidor de esta manera nos sirve como simulación de un download previo de la información al inicio de la Rueda de Bolsa.

Una vez que se ha ejecutado la lectura de este archivo, la aplicación Servidor carga su buffer de datos seguidamente los clientes Visual C++ que se encuentran en la portátil, se ejecuta al inicio, se ejecuta internamente el envío de un comando a la aplicación Servidor, pidiendo el download de una tabla determinada, la aplicación Servidor automáticamente le envía toda la información requerida, la información viaja a través del Cable Crossover hacia la Laptop y automáticamente se muestra en la tabla de la aplicación Cliente bajo Windows.

Para esta Red Local se desarrolló una aplicación adicional, esta aplicación se encargaba de enviar los mensajes modificados de tal manera que simule modificación en cotizaciones, inserción de operaciones y cálculo de índices.

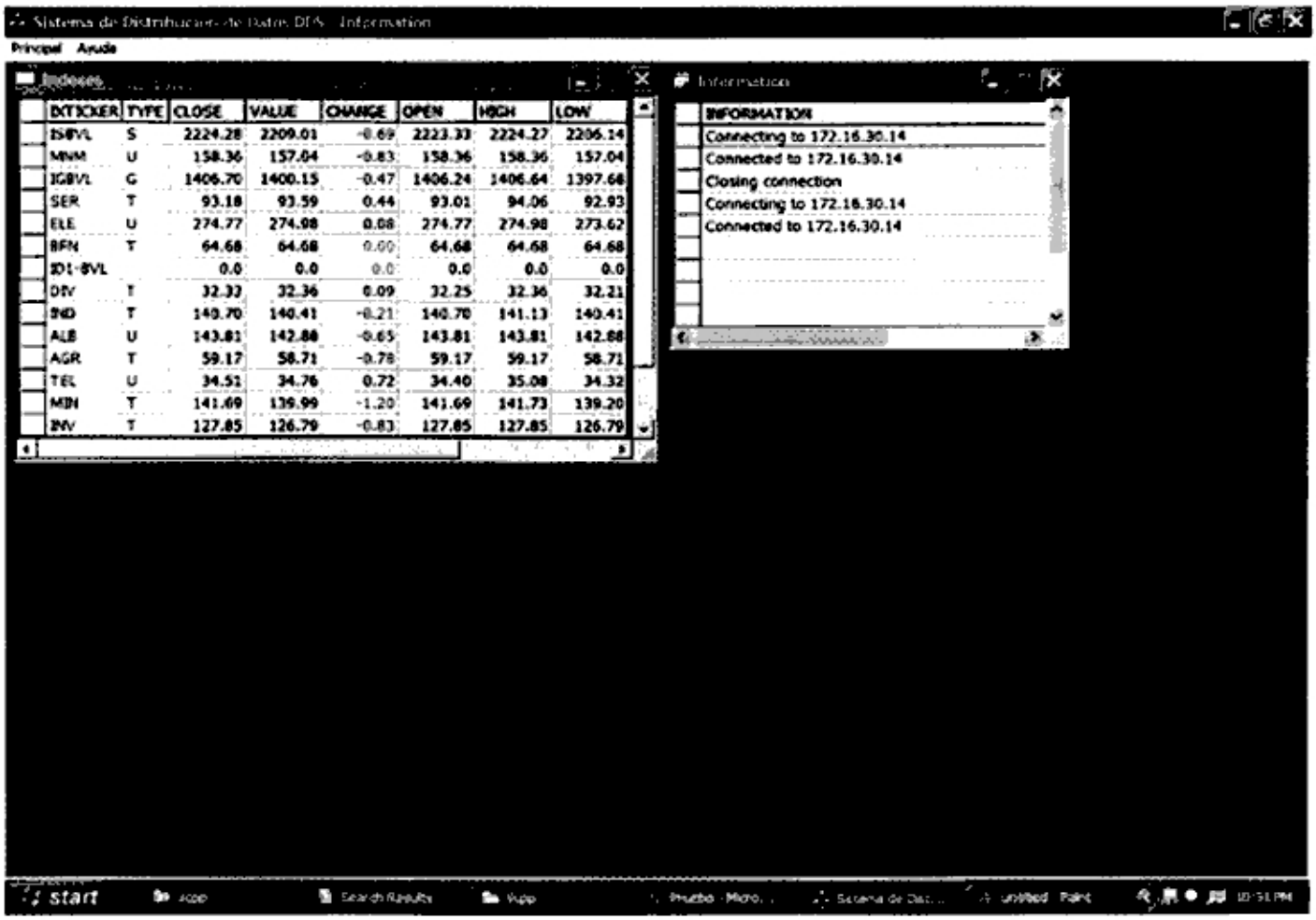

FIGURA 25: *Aplicación Cliente, implementado en Visual C++, como se observa existen dos ventanas en una de ellas se muestra información relativa a los Índices Bursátiles.*

(Fuente Propia)

Estos mensajes son recibidos por la aplicación Servidor los guarda en el buffer y se los envía a la aplicación Cliente en la Laptop.

FIGURA 26: *Resultado diario debido a la interacción entre las aplicaciones Cliente y la aplicación Servidor.*

(Fuente Propia)

PRUEBAS EN RED BVL

Dentro de la Red Corporativa de Bolsa de Valores, las pruebas se realizaron directamente con el Sistema Elex, se utiliza una aplicación intermedia denominada IDServer, esta aplicación recibe todos los mensajes del Sistema Elex.

Para que los mensajes que viene recibiendo la aplicación IDServer, sean también recibidos por la aplicación Servidor, se implementó una aplicación que acceda como Cliente a la aplicación IDServer, esta aplicación una vez que recibe los mensajes que recibe el IDServer los envía automáticamente a la aplicación Servidor. Los mensajes recibidos por la aplicación Servidor, los guarda en un buffer de datos.

Las aplicaciones Cliente, que se han conectado con la aplicación Servidor, recuperan en el inicio de la conexión el contenido del buffer de datos y luego los mensajes que la aplicación Servidor, envía a todas sus aplicaciones Cliente.

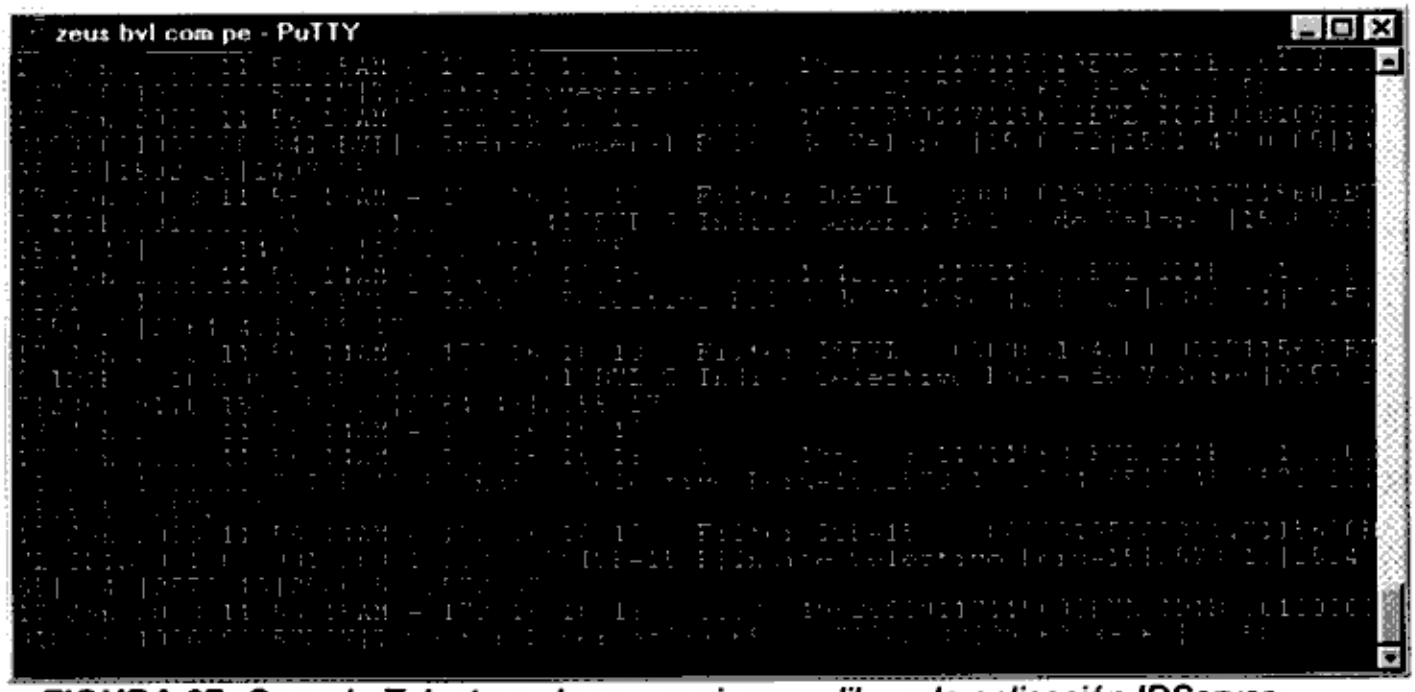

***FIGURA 27:** Consola Telnet con los mensajes que libera la aplicación IDServer, esta aplicación está desarrollada en Java y es ejecutada sobre un Servidor Unix.*

(Fuente Propia)

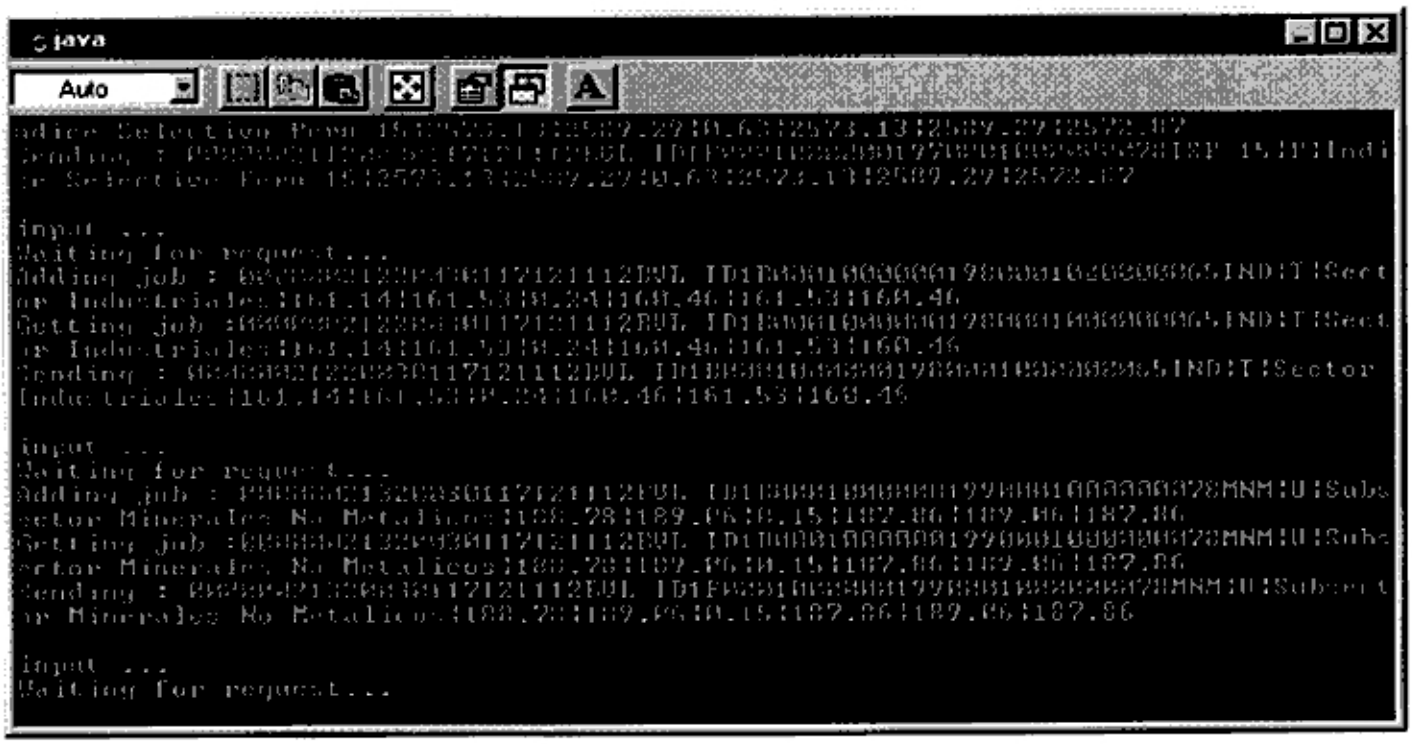

FIGURA 28:** Aplicación nexo entre la aplicación IDServer y la aplicación Servidor, esta aplicación está desarrollada en Java y es ejecutada (para esta prueba) en Windows. **(Fuente Propia)

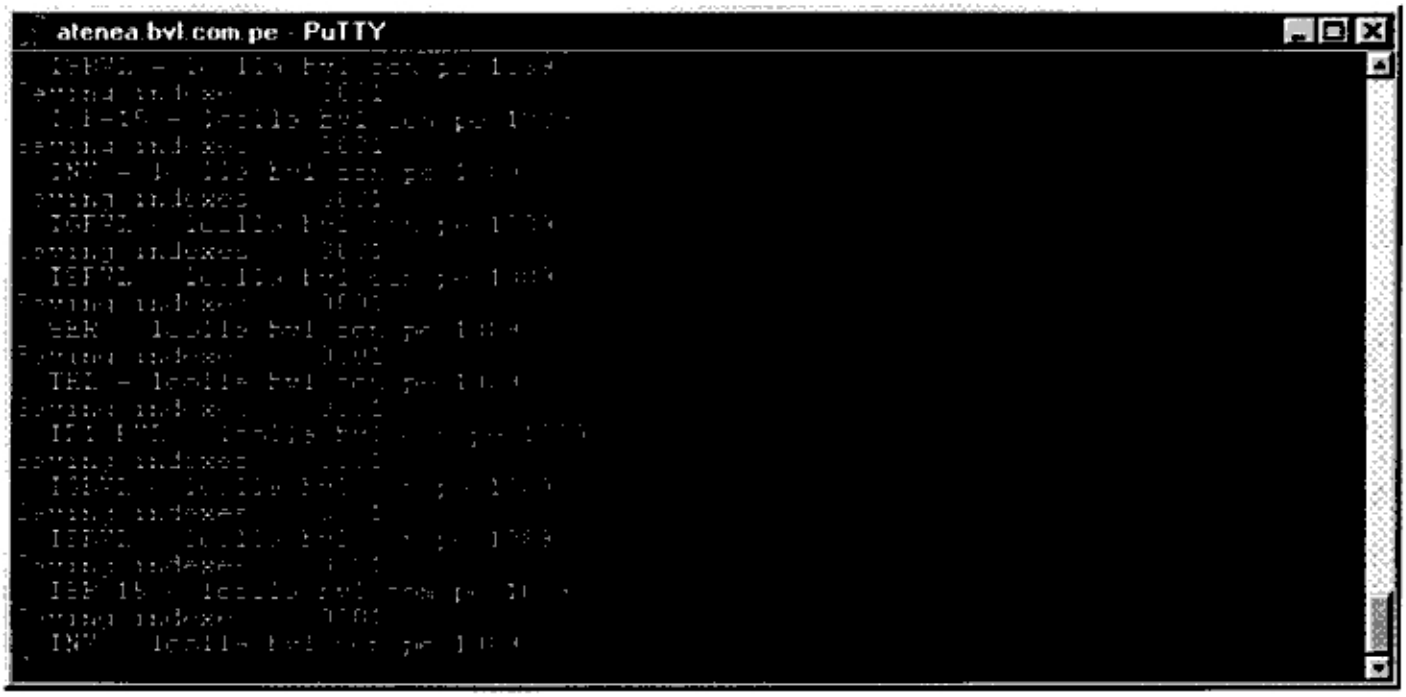

***FIGURA 29:** Consola Telnet con los mensajes que libera la aplicación Servidor.*

(Fuente Propia)

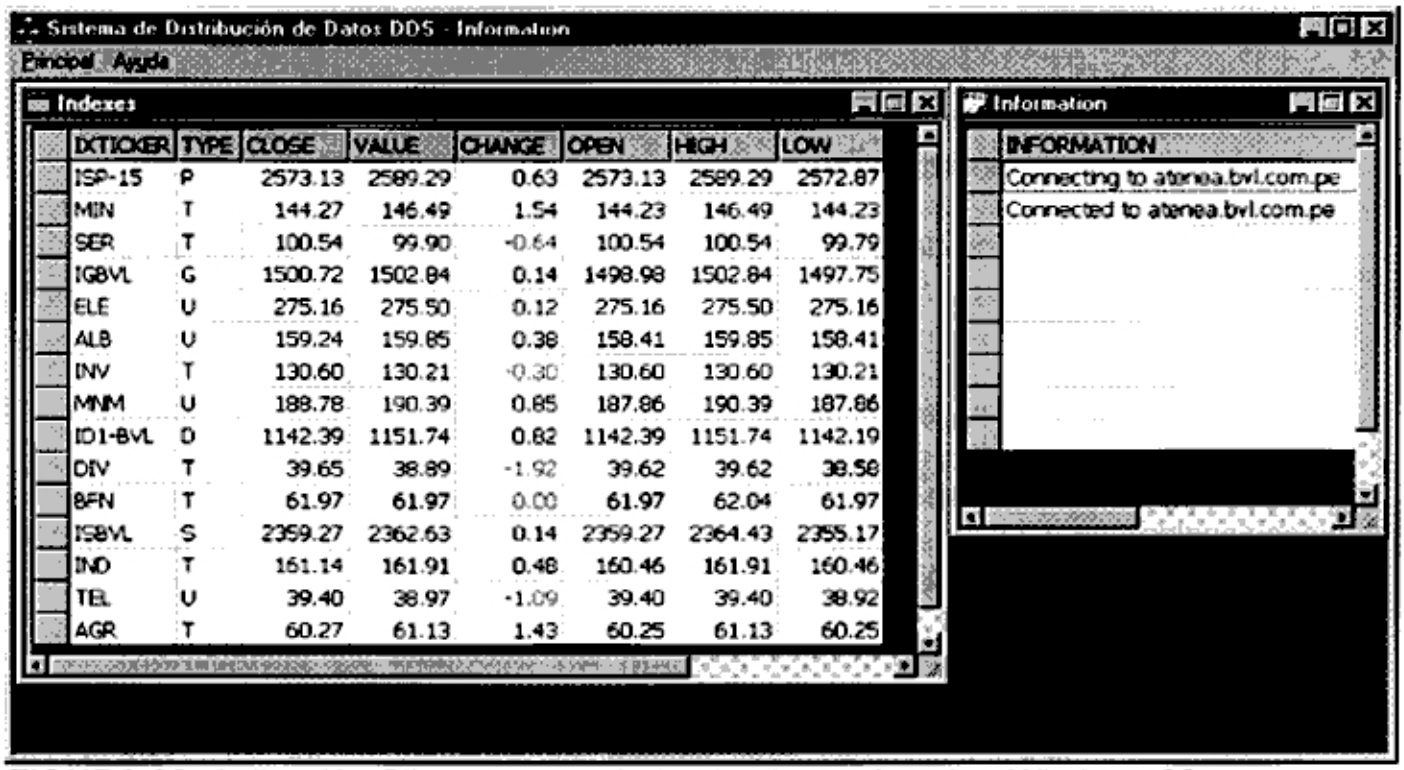

***FIGURA 30:** Aplicación Cliente que está siendo ejecutado sobre Windows 98.*

(Fuente Propia)

CAPITULO V

EVALUACIÓN DE RESULTADOS

5.1. SELECCIÓN DE VARIABLES

Para determinar cual de las dos propuestas mencionadas dentro del presente documento, es la que tiene el mayor índice de factibilidad, se definieron unas variables que a criterio del autor medirán de manera cualitativa el grado de factibilidad de cada una de las propuestas (Aplicación Web, aplicación Servidor).

Dentro de las variables cuantitativas y cualitativas se tienen los siguientes:

- Seguridad, se utilizó el protocolo SSL como mecanismo de seguridad para la información que fluye en la red de aplicación.

- Performance, se realizaran pruebas tanto para una aplicación Web sencilla y la aplicación prototipo desarrollada y se medirá el grado de consumo de recursos del sistema (%CPU, %Memoria). Estas mediciones se realizará de manera cuantitativa solamente para el caso de Linux ya que se posee una función propia del Sistema Operativo de obtener el consumo %CPU y %Memoria.

- Costo en Hardware y Software, se evaluaran los distintos costos que conllevan a desarrollar tal propuesta bajo tal herramienta de desarrollo dentro de una plataforma determinada, así como el hardware ideal para cada propuesta.

- Tiempo de Procesamiento, se realizaran mediciones de tiempo desde la emisión de los mensajes de información hasta la recepción de los mismos dentro de la aplicación Servidor versus el tiempo de procesamiento que implica una aplicación Web por sesión de usuario. Se establecerá un tiempo promedio en cada uno de los casos.

Se definirán pesos por cada uno de estas variables tanto cuantitativas como cualitativas, para luego determinar grado de factibilidad de cada uno de las propuestas teniendo como criterios de medición estas variables.

5.2. EVALUACION DE VARIABLES

De las variables que se han definido para la evaluación de las soluciones, solamente se ha realizado un estudio detallado acerca del Tiempo de Procesamiento. En cambio para las demás variables solamente se han determinado aproximaciones en rendimiento entre las soluciones puestas en cuestión.

- En el tema de la seguridad, si nuestra aplicación no contaba con elementos de seguridad se encontraría en desventaja respecto a la aplicación Web. Ya que para el procedimiento actual se cuenta con procedimientos de generación de páginas Web que por ser estáticas tienen únicamente carácter de lectura.

  En cambio para nuestra aplicación por existir flujo de información en tiempo real entre diversos usuarios, se requiere que la información fluya de manera segura, y entonces se dispone del SSL como alternativa para lograr este cometido.

  En la siguiente figura se explica como se utiliza el protocolo SSL como mecanismo de seguridad, cifrando la información entre emisor y receptor de información.

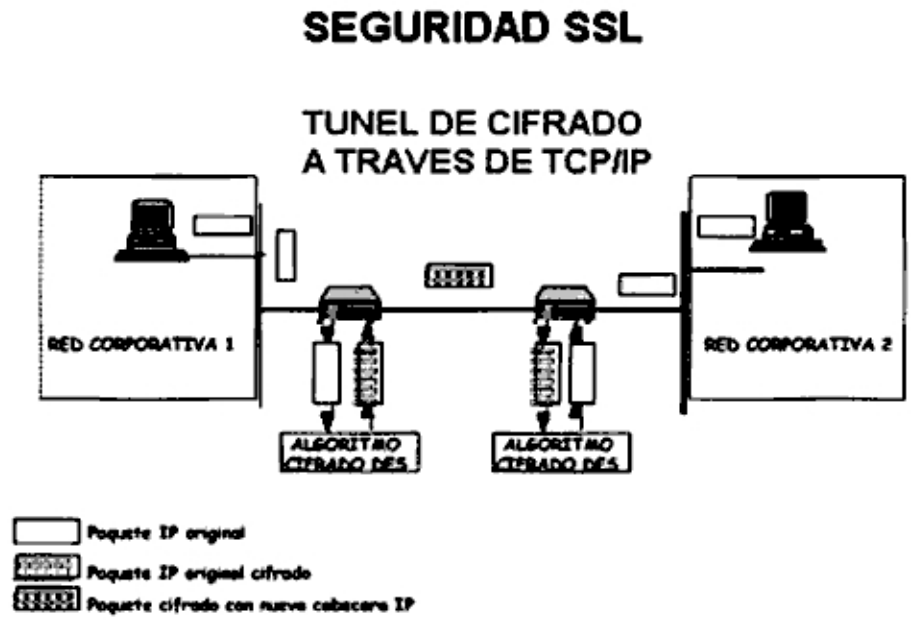

FIGURA 31: *Seguridad SSL.*

(Fuente Propia)

- En cuanto a la variable de Performance se ha observado mediante un monitoreo del rendimiento de CPU y de la Memoria, que la solución Web consume más recursos sobre todo de CPU, ya que cada persona que acceda a la aplicación implícitamente realiza una consulta a la Base de Datos generando un consumo regular de recursos. Sin embargo en la solución propuesta este consumo de recursos es mucho menor, debido a que por cada movimiento de información se accede a la Base de Datos, pero para permitir el flujo de información a los usuarios no se tienen que realizar consultas a la Base de Datos, en todo caso es un asunto de protocolo de comunicación.

- El costo de Hardware y Software no determina una marcada diferencia entre ambas soluciones. Ya que por lo general para ambas soluciones se necesita tanto un Servidor con elevados recursos de CPU y Memoria, para albergar tanto al Servidor de Aplicaciones como a la Base de Datos. Tomando además en cuenta que aplicación Servidor para la solución propuesta esta desarrollada en Java y por ende no cuenta con costo en licencia, en cambio en la solución Web se utilizan muchas veces productos Freeware. Hasta el momento solamente se mencionó la capa de Servicio de Aplicaciones; pero para la capa del Cliente, en la solución Web no se necesita demasiados recursos para un Navegador Web, y para la solución propuesta se requieren de mínimos requerimientos de Hardware, pero existe el tema de la licencia acerca de la herramienta que se desarrollo esta.

- Para el Tiempo de Procesamiento se tiene que para la solución propuesta llega a niveles de milisegundos mientras que para la solución Web depende del Tiempo de Respuesta de la Base de Datos para una consulta realizada desde un Navegador Web. Seguidamente se verá la justificación del nivel optimo en este aspecto de la solución propuesta.

Una razón por la que se implantaría la solución propuesta es que en el plano comercial una aplicación tiene un promedio de vida aproximado de 5 años, y una entidad tan importante para la Bolsa de Valores tiene que mejorar para sus clientes, por un principio muy elemental, si a los clientes de la Bolsa de

Valores les va bien, entonces como consecuencia, a la Bolsa de Valores también le irá bien, debido a que los clientes seguirán depositando su confianza en la misma, por lo que la Bolsa de Valores crecerá mucho en lo referente al reconocimiento, que esta tan llevado a menos en estos días por muchas empresas.

Para el caso particular de la Bolsa de Valores se debe conocer que por cada operación bursátil existe una comisión que se atribuye la Bolsa de Valores, esta comisión depende del Mercado en donde se desarrolle la operación; por ejemplo, para el Mercado de Renta Variable por cada operación realizada la Bolsa de Valores obtiene una comisión equivalente al 0.08625% del Monto Negociado de la operación. Esto quiere decir que si la solución propuesta brinda oportunidad a la Bolsa de Valores de aumentar el nivel de operaciones bursátiles determinando de esta manera un incremento en la comisión que percibe la Bolsa de Valores.

ESTIMACIONES DEL COSTO DEL DESARROLLO DE LA APLICACION

Para obtener un estimado del costo del producto en lo referente solamente a desarrollo del prototipo, se determinó el cálculo de éste, teniendo conocimiento del número de horas que se demoró en implementar las aplicaciones, así como también el costo de una hora hombre.

Se ha estimado, que el tiempo de desarrollo que se utilizara para la aplicación equivale a 264 horas, equivalentes a 3 meses, durante la etapa de implementación del proyecto, trabajando 4 horas diarias exclusivamente en el proyecto durante 22 días útiles.

TP = (3 meses) (22 días / meses) (4 horas / días) = 264 horas

Donde: TP , es igual al tiempo de desarrollo del proyecto.

Para obtener el costo por hora de cada uno de las personas que intervinieron en la implementación del proyecto, se tuvieron en cuenta las remuneraciones mensuales que perciben éstas, en la implementación del proyecto.

La remuneración mensual promedio calculada fue de 2200 nuevos soles, teniendo en cuenta que, los días útiles por cada mes son 22, y que por cada día útil se laboran 8 horas.

Tomando estas acotaciones, el cálculo del costo por hora promedio del desarrollador se calculó de la siguiente manera:

CH = (2200 nuevos soles) / (22 días) / (8 horas / días) = 12.5 nuevos soles /hora

Donde: CH , es igual al costo por hora de cada desarrollador.

El costo total del desarrollo sería el siguiente:

CT = (264 horas) (12.5 nuevos soles / hora) = 3300 nuevos soles.

Donde: CT , es igual al costo total del desarrollo del proyecto.

En conclusión, tomando en cuenta estos resultados, tenemos que el costo referente exclusivo del desarrollo de la aplicación resulta significativamente rentable, si lo comparamos con cotizaciones de otras empresas, al solicitarle servicios de outsourcing, tales cotizaciones tienen bordean los 10 000 dólares, lo que significa que el costo se estaría incrementado en 10 veces al de nuestra propuesta.

RESULTADOS ACERCA DE LOS TIEMPOS DE RECEPCION DE LOS CLIENTES

Antes que nada quiero precisar que la que la información que esta sirviendo de prueba para la aplicación es sobre Índices Bursátiles, únicamente sobre eso, nos estamos basando.

Para la aplicación que se ha propuesto como alternativa a la aplicación Web, se creyó conveniente realizar un análisis de carácter cuantitativo. Este

análisis tuvo en cuenta el tiempo de recepción por parte de los clientes de los mensajes enviados por la aplicación Servidor. El periodo de pruebas para este análisis cuantitativo fue desde el 05 de Febrero del 2003 hasta el 11 de Marzo del 2003. No obstante para el siguiente análisis individual solamente se tomaron los días 27 de Febrero, 28 de Febrero y el 3 de Marzo del 2003. Las imágenes, que se van mostrando a continuación, reflejan el comportamiento en cada uno de los días que se tomaron como pruebas de la cantidad de segundos que se consumen en el envío de mensajes desde la aplicación Servidor a las aplicaciones Cliente respectivas.

Se notarán claras diferencias en las figuras, no solamente en los días que se tomaron como fuente de análisis sino en la data que se empleó, para algunos casos se incluye el Download. Este Download es una operación previa que ejecuta la aplicación Servidor junto con el Proveedor de la Información, esta operación se ejecuta al inicio de la Rueda de Bolsa (aprox. 08:51am). Como se mencionó anteriormente existe una clara diferencia en los gráficos al incluir el Download, cuando es así surge un "*pico*" que sobresale al inicio de la Rueda de Bolsa, conforme va avanzando la Rueda de Bolsa el consumo de tiempo va disminuyendo llegando a tener un nivel constante.

En otros casos no se toma en cuenta el Download sino todo el periodo de Envío de Mensajes Directo a las aplicaciones Cliente.

La diferencia que existe entre estos dos tipos de figura, es que el Download demora mucho más que un simple Envío de Mensajes a las aplicaciones Cliente. El Promedio de Tiempo en segundos que demora el Download es de 170 segundos es decir 3 minutos. En cuanto al Promedio de Tiempo que demora en enviar los mensajes a las aplicaciones Cliente está en 2 segundos. Por último se muestra una figura con el comparativo de Tiempos de Consumo por Día desde el 05 de Febrero hasta el 11 de Marzo del 2003.

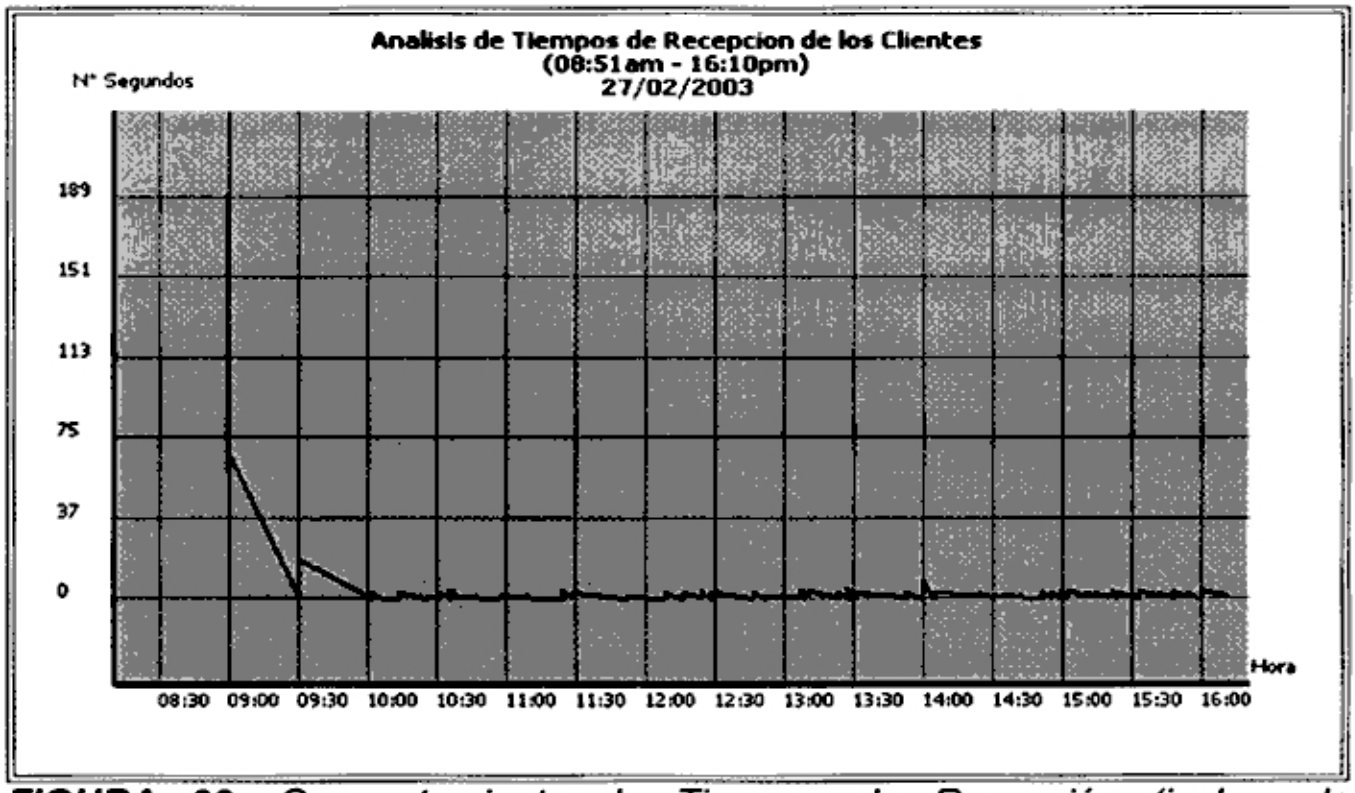

FIGURA 32: *Comportamiento de Tiempos de Recepción (incluyendo Download a partir de las 08:51am) correspondiente a la Rueda de Bolsa del día 27 de Febrero del 2003.*

(Fuente Propia)

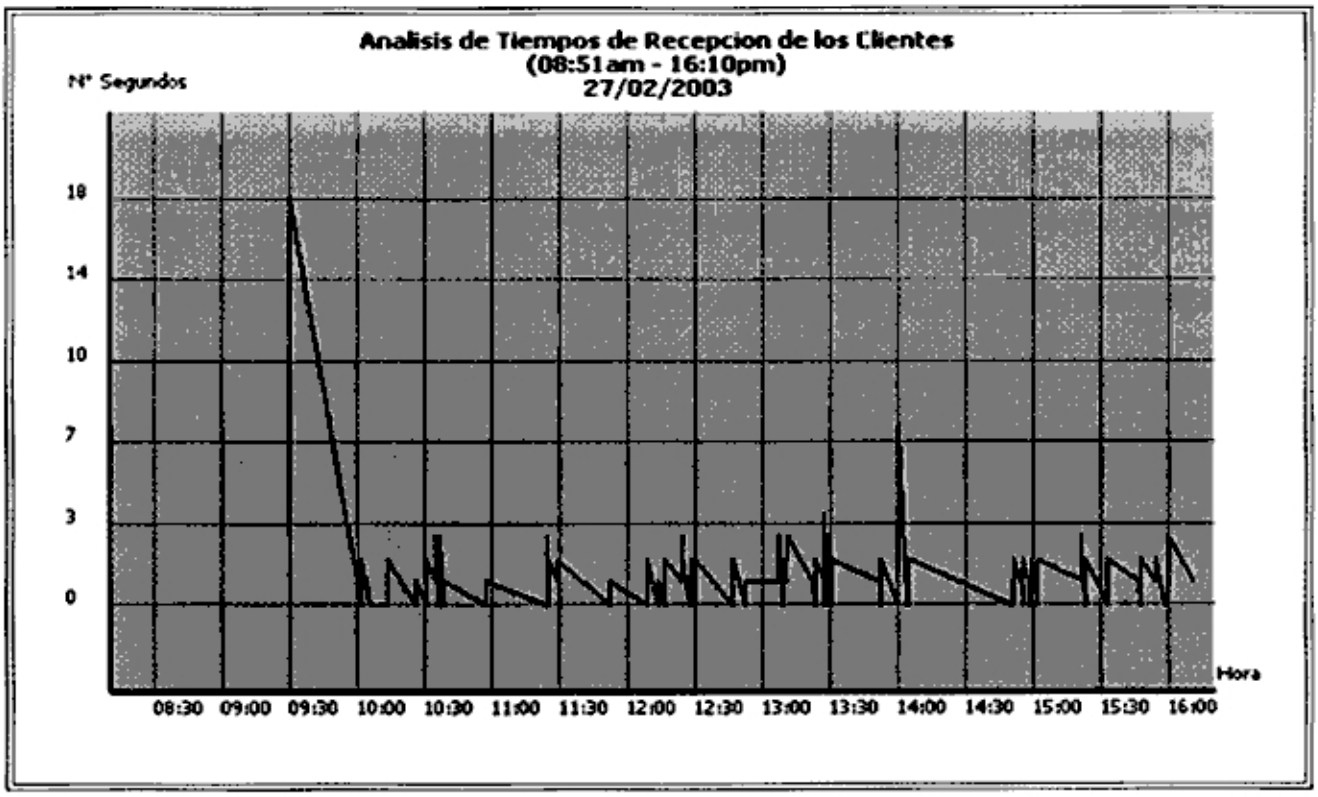

FIGURA33: *Tiempos de Recepción del día 27 de Febrero del 2003.*

(Fuente Propia)

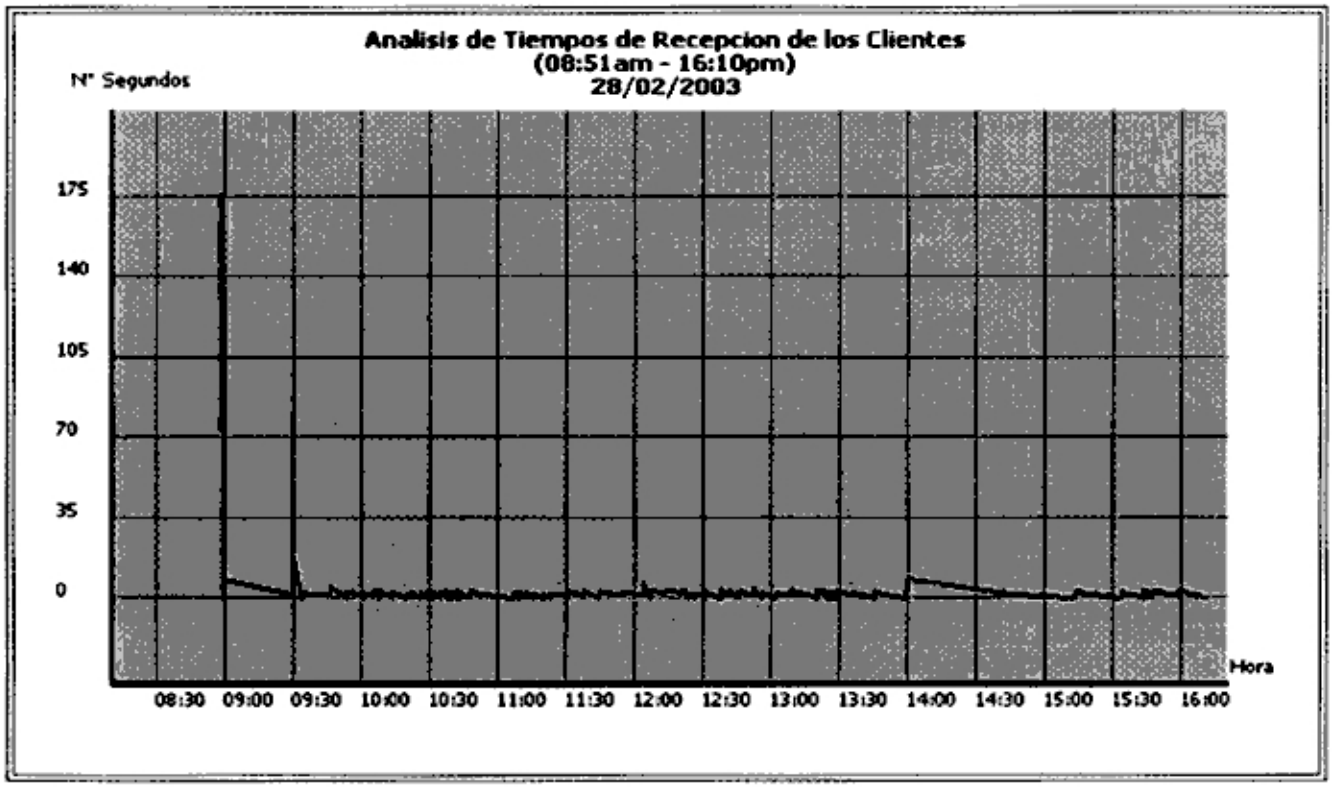

FIGURA 34: *Tiempos de Recepción (incluyendo Download a partir de las 08:51am) correspondiente a la Rueda de Bolsa del día 28 de Febrero del 2003.*

(Fuente Propia)

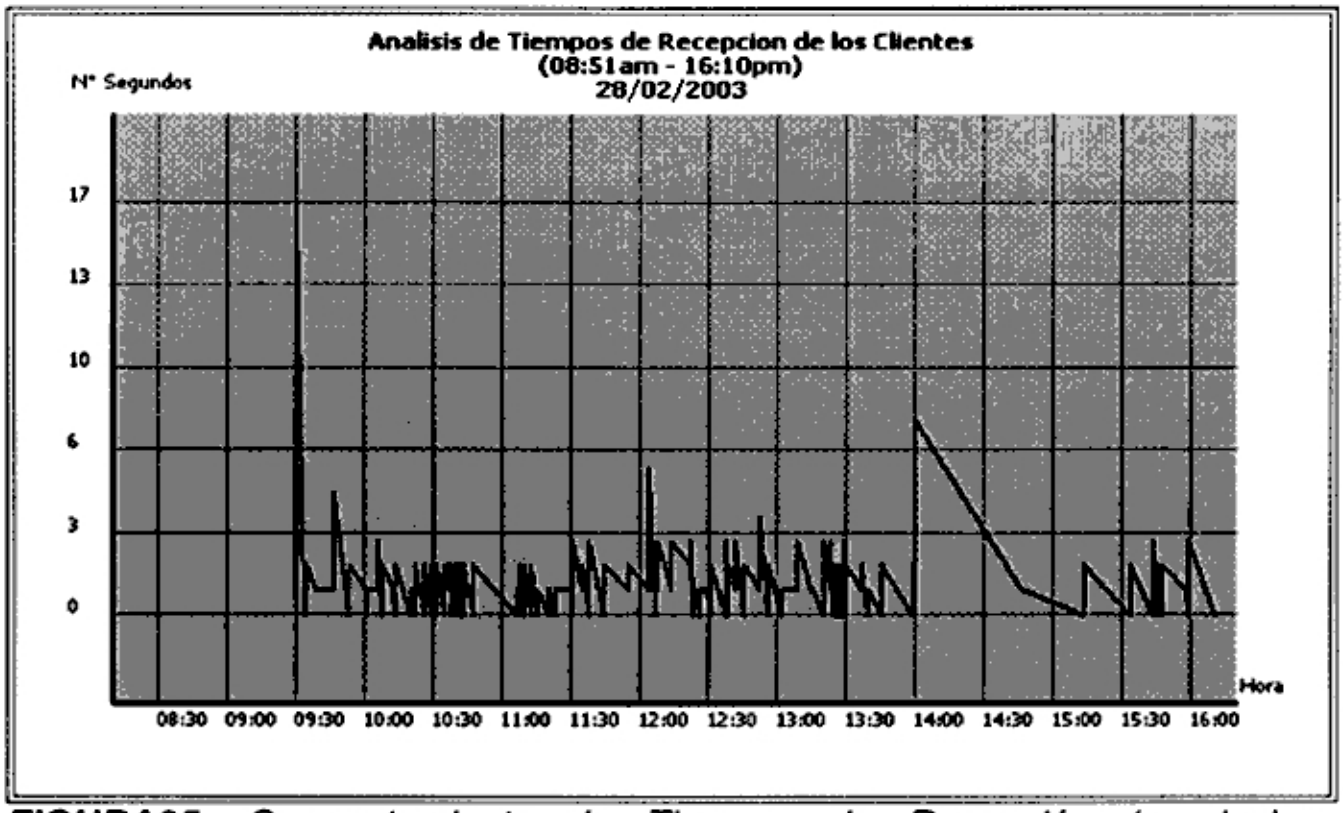

FIGURA35: *Comportamiento de Tiempos de Recepción (no incluye Download) correspondiente a la Rueda de Bolsa del día 28 de Febrero del 2003.*

(Fuente Propia)

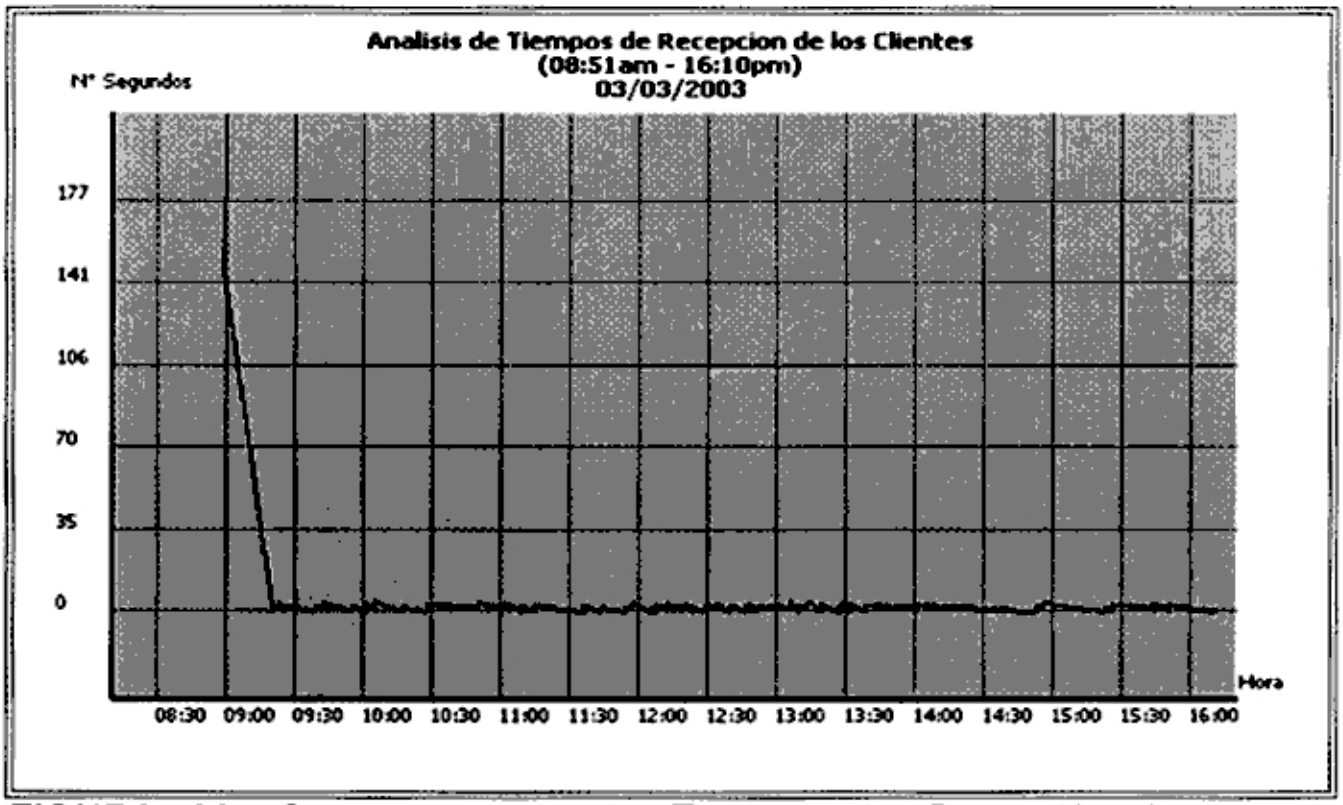

FIGURA 36: *Comportamiento de Tiempos de Recepción (incluyendo Download a partir de las 08:51am) correspondiente a la Rueda de Bolsa del día 03 de Marzo del 2003.*

(Fuente Propia)

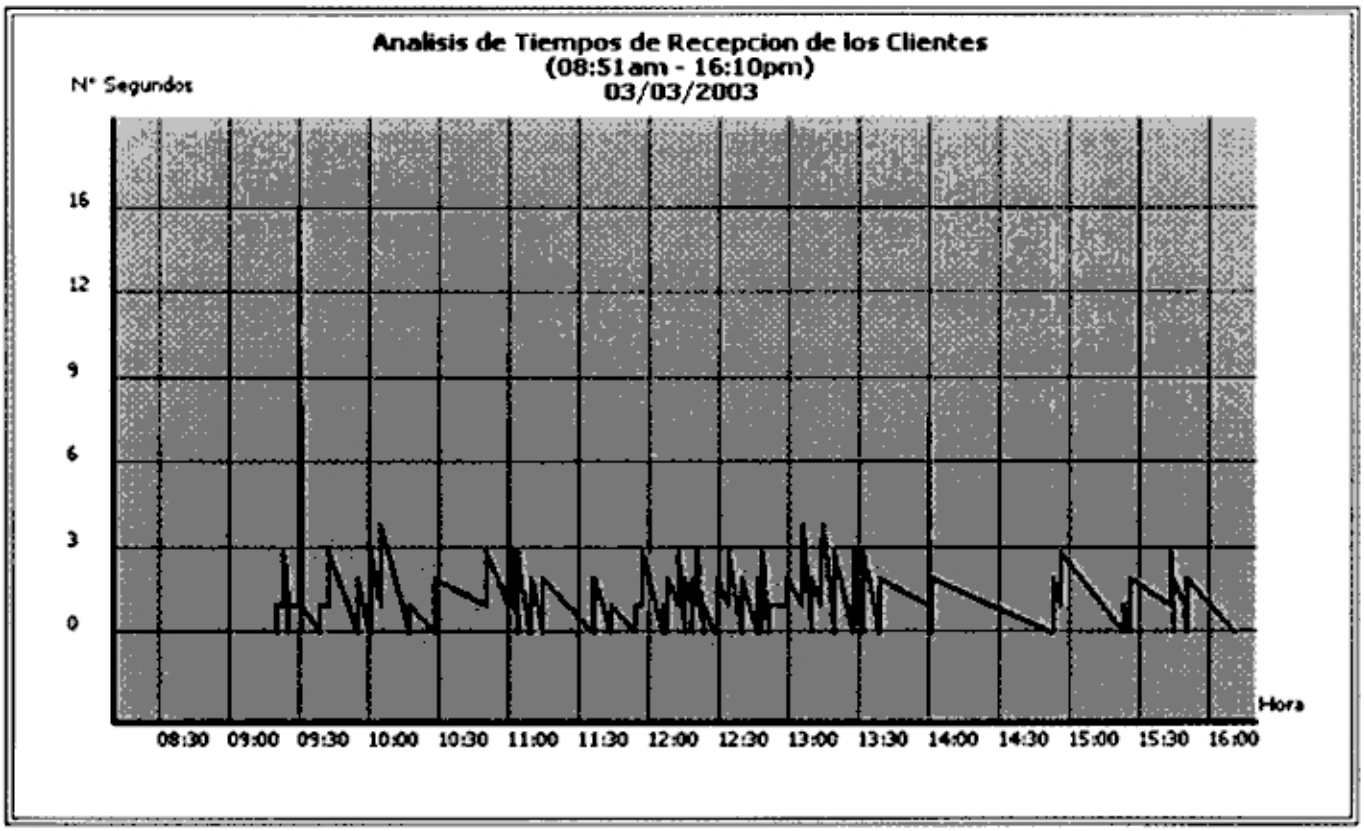

FIGURA37: *Comportamiento de Tiempos de Recepción (no incluye Download) correspondiente a la Rueda de Bolsa del día 03 de Marzo del 2003.*

(Fuente Propia)

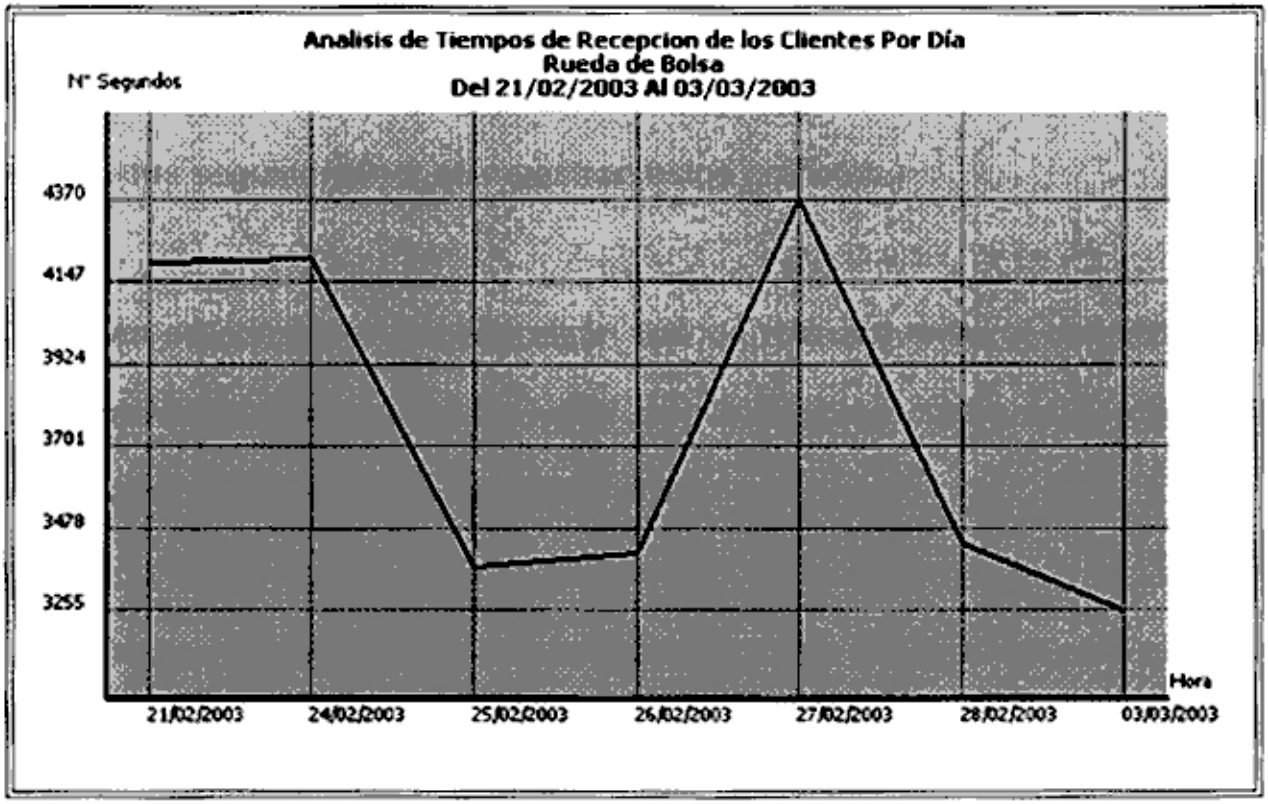

FIGURA38: *Comportamiento de Tiempos de Recepción por Día (Incluyendo Download desde las 08:51am), desde el 21 de Febrero hasta el 3 de Marzo del 2003.*

(Fuente Propia)

Anteriormente solamente se había incluido dentro de las variables de análisis el tiempo de recepción de la información por parte de los clientes. Además del tiempo de recepción se determinó una variable adicional que viene a ser el tamaño de la información que recibe cada una de las aplicaciones Cliente dentro de la Rueda de Bolsa.

Se desarrollaron gráficos, que al contrario de los mostrados anteriormente, muestran el comportamiento del tiempo de recepción dentro de condiciones

no tan normales, esto debido a excepciones que ocurrieron durante algunos días que se suscitó dentro de la Rueda de Bolsa. Estas excepciones serán explicadas seguidamente se muestren estos gráficos.

También se consideró mostrar algún tipo de relación entre el tamaño de la información recibida por cada unos de los clientes contra el tiempo que tomaba recepcionarlos, esto durante el día.

Se tiene que mencionar que la fuente de datos que se utilizaron para obtener los gráficos, se refiere a una serie de archivos "*log*" o bitácoras de mensajes que emite la aplicación Servidor. Una muestra de los archivos "*logs*" originales se muestra a continuación:

- 05 Feb 2003 09:30:05AM - 172.16.10.18 : 00000002320030205093001BVL ID1B000100000002200010000000056DIV|T|Sector Diversas|36.06|36.55|1.36|36.55|36.55|36.55
- 05 Feb 2003 09:30:06AM - 172.16.10.18 : 00000002420030205093001BVL ID1B000100000002300010000000062IND|T|Sector Industriales|176.01|176.01|;|176.01|176.01|176.01

A partir de estos archivos "*logs*" se generaron archivos con información consolidada por día de Rueda de Bolsa, esto con la intención de obtener una

serie de datos que muestre el comportamiento entre las variables mencionadas, como el tamaño de información recepcionada y el tiempo de recepción.

El contenido de archivo que se muestra a continuación se definió para graficar información en cualquier instante de la Rueda de Bolsa.

08:58|169

08:58|170

08:58|170

08:58|171

08:58|172

Por ejemplo el formato que se muestra es:

[Hora]|[N° Segundos]

Esto se traduce que a una determinada hora hubo una recepción de información que tomó N° segundos recepcionarla.

El siguiente contenido del archivo, que se muestra a continuación, se definió para graficar información consolidada diaria, esta información surge a partir de los archivos logs originales.

20030205|3442|23026

20030206|3412|35977

20030207|3657|24380

20030210|3666|27704

20030211|3460|36652

20030212|3238|23620

20030213|8446|44086

20030214|19288|39550

20030217|3269|23753

20030218|16668|37549

20030219|11216|48391

20030220|10183|38896

20030221|4198|29402

20030224|4209|21493

20030225|3371|28988

20030226|3407|23258

20030227|4370|21662

20030228|3436|30704

20030303|3255|23657

20030304|3500|34087

20030305|4206|29144

20030306|3536|34272

20030307|3251|22443

20030310|2945|7134

20030311|3416|32522

La información mostrada anteriormente solamente es utilizada para graficar el comportamiento de tal ó cual variable por día de Rueda de Bolsa, al contrario de los primeros gráficos que reflejaban el comportamiento del Tiempo de Recepción por horas ó teniendo en cuenta la hora, en rangos de 0.5 hora, como variable dinámica.

La información mostrada corresponde al siguiente formato:

[Fecha de Rueda de Bolsa]|[Tiempo de Actividad]|[Tamaño de Información]

Cada uno de estos conceptos se explica seguidamente:

- Fecha de Rueda de Bolsa, este ítem aparece en primer lugar, es la fecha en que se efectúo la Rueda de Bolsa, el formato de fecha es el "YYYYMMDD".

- Tiempo de Actividad, este ítem se refiere al tiempo acumulado que se empleó para recibir la información desde la aplicación Servidor hasta cada una de las aplicaciones Cliente. La unidad de medida de este ítem es el segundo.

- Tamaño de Información, este ítem se refiere al tamaño acumulado de los mensajes que fueron enviados desde la aplicación Servidor hasta cada una de las aplicaciones Cliente. La unidad de medida de este ítem es el byte.

A continuación se presentara el gráfico de Tamaño de Información contra el Tiempo de Recepción por cada uno de los clientes.

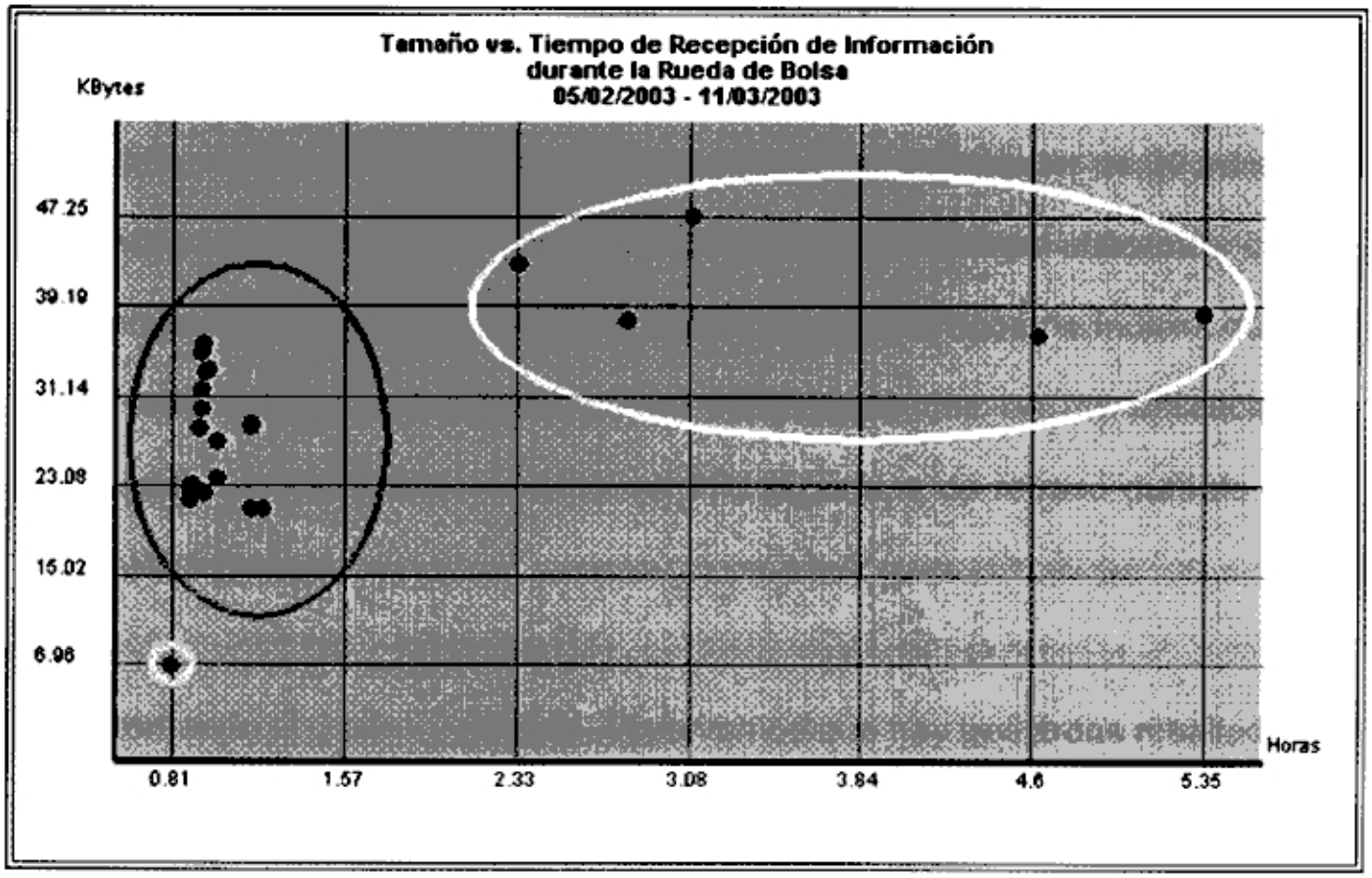

FIGURA39: Se muestra el comportamiento entre el Tamaño vs. Tiempo de Recepción de la información durante la Rueda de Bolsa (incluyendo Download).

(Fuente Propia)

En algunos gráficos se ha mencionado el término "*Download*", este download ocurre, para el caso particular de la Bolsa de Valores, al inicio de la Rueda de Bolsa y al Término de la Rueda de Bolsa de Valores Nacionales. El inicio de la Rueda de Bolsa se realiza a las 8:51am y el Término de Rueda de Bolsa para Valores Nacionales a las 2:00pm. A estas horas se realiza un cálculo adicional que tiene que ver con inicializar las cotizaciones de los distintos valores así como los distintos Índices Bursátiles con los resultados con que finalizaron el día anterior de Rueda de Bolsa.

Es por ello que el download toma un tiempo regular alrededor de 30 minutos. Cuidado que este tiempo es acumulado que solamente utilizamos como información de prueba los Índices Bursátiles, son 12 tipos de índices que se envían desde la aplicación Servidor hasta cada una de las aplicaciones Cliente. Cada índice toma cerca de 170 segundos de retraso durante este download.

Retornando al gráfico mostrado, observamos que hay tres áreas resaltadas.

- El área que se resalta de color verde es en el que la Rueda de Bolsa se desarrolla en Condiciones Normales, como se observara el Tiempo de Recepción promedio en esta área es aproximadamente 1 hora. En tanto la cantidad de información fluctúa entre 21KBytes a 39Kbytes. Hay que entender que no siempre el Tamaño de Información (solamente Indices Bursátiles) es el mismo para cada Rueda de Bolsa. La variación de este

tamaño de información depende de la cantidad de movimientos de las cotizaciones de determinadas cartera de valores.

- El área que se resalta de color amarillo, presenta una excepción en la Rueda de Bolsa, ocurrieron inconvenientes con la Red. El medio TCP/IP había producido errores que dentro de 30 minutos se solucionaron. Se había suspendido la Rueda de Bolsa por espacio de 2 minutos y luego se procedió a la medida de contingencia que era la conexión mediante X.25. Este cambio hizo imposible que la aplicación de prueba recibiera la información a impartir a las aplicaciones Cliente.

  Por eso el tiempo de recepción acumulado es tan bajo, la razón es que hubo suspensión de Rueda de Bolsa.

- El área que se resalta de color blanco presenta un amplio tiempo de recepción, de 2.3 a 5.3 horas, se puede explicar este fenómeno con que en esos días de Rueda de Bolsa se desarrollaron tareas de mantenimiento de Red de Bolsa de Valores. Explicando de antemano que estas tareas de mantenimiento no perjudicaron a las aplicaciones de producción de Rueda de Bolsa. Estas tareas de mantenimiento tuvieron que ver con actualizaciones en el firewall, que como se sabe es el que regula las entradas y salidas de accesos entre ordenadores dentro de una Red. Por alrededor de 30 minutos se estuvieron llevando a cabo

estas tareas, llegando a bloquear el puerto que servia de escucha para las aplicaciones Cliente, por espacio de 30 minutos.

Es solamente coincidencia que esta área pertenezca a los altos tamaños de información, para esos días de Rueda de Bolsa, se había agregado una fuente adicional de información para la aplicación de prueba (que es la que se discute en este proyecto), que era la de Operaciones de Stock (Acciones).

Se mostrarán seguidamente gráficos donde se observa claramente el tiempo de desfase que ocasionaron estas tareas imprevistas de mantenimiento de Red.

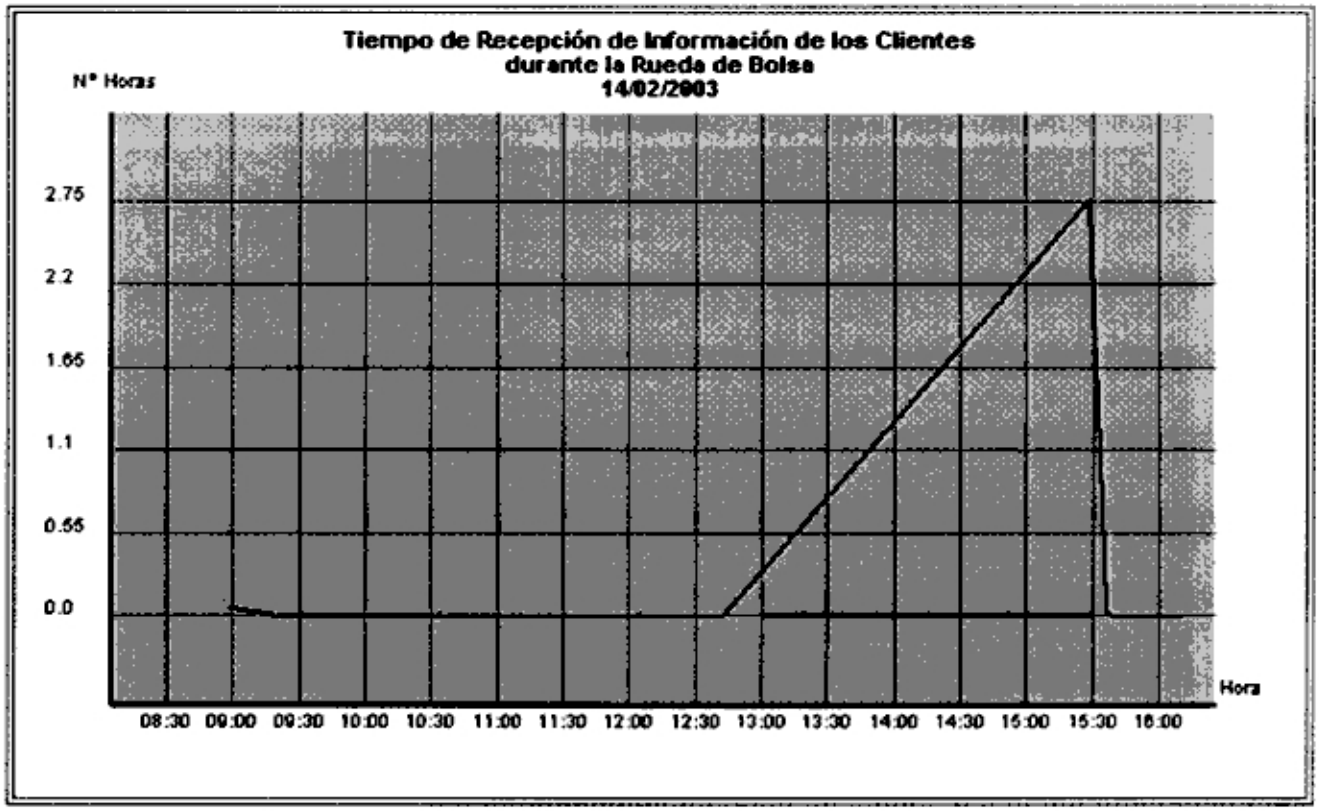

FIGURA 40: Rueda de Bolsa del 14 de Febrero del 2003. Aquí ocurre un desfase de 2.5 horas aproximadamente (Observar línea verde).

(Fuente Propia).

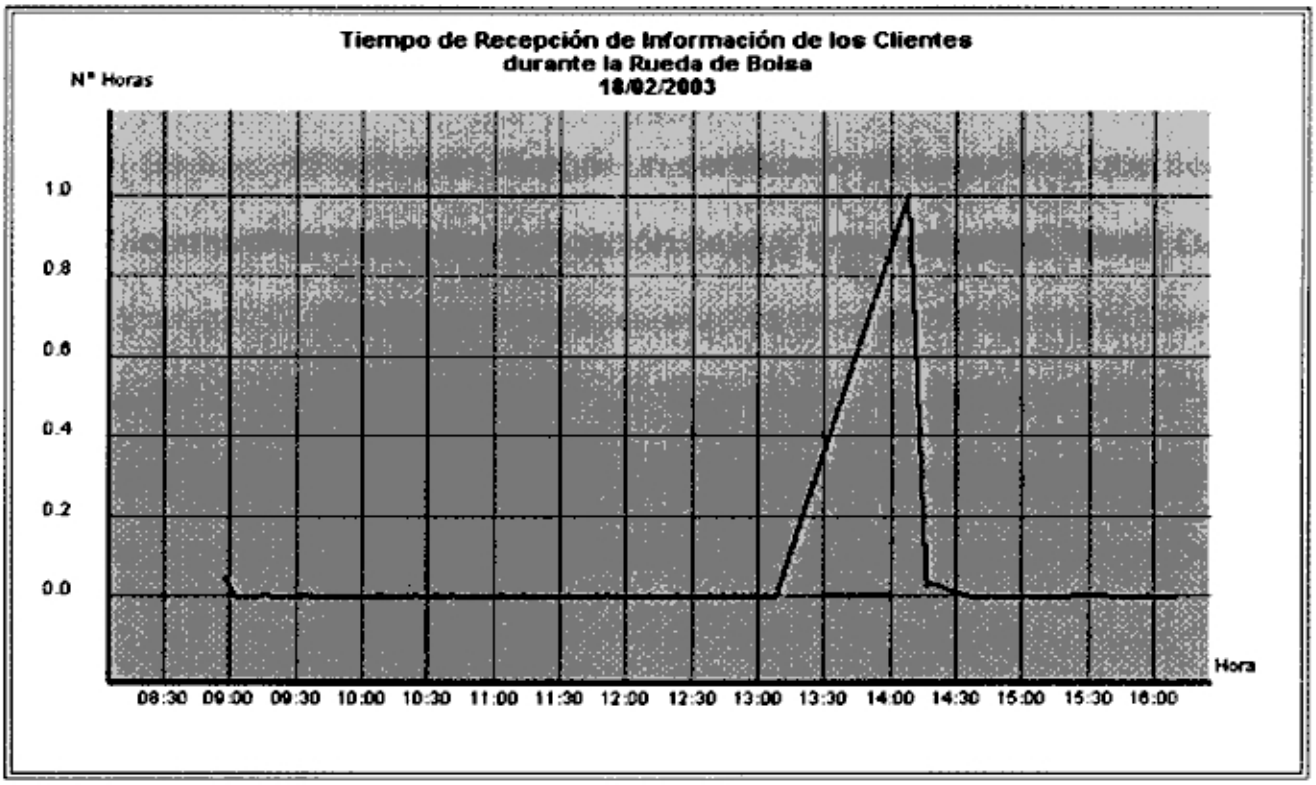

FIGURA 41: Rueda de Bolsa del 18 de Febrero del 2003. Aquí ocurre un desfase de 0.5 horas aproximadamente (Observar línea verde).

(Fuente Propia)

5.3. EXCEPCIONES Y DIFICULTADES

- Bloqueo ó interrupción en la comunicación de la red, esto permitirá que se corte con lecciones y no se pueda iniciar la aplicación Servidor.

- Demasiado tráfico de red, saturando los paquetes de información por la red.

- Filtración por firewall.

- Se tuvo dificultad en otorgar un certificado de seguridad que sea compatible entre una aplicación Java y una aplicación Visual C++.

CAPITULO VI

LECCIONES APRENDIDAS

- Se recomienda utilizar un archivo de configuración para una aplicación con el propósito de personalizarla, así se tendrá la oportunidad de cambiar las variables dinámicamente sin necesidad de compilar una nueva versión de la aplicación.

- Mecanismos de seguimiento de la aplicación, además de la detección de errores (I /O).

- Otorgar la responsabilidad a un operador de Sistemas para el monitoreo de la aplicación.

- Debemos tener un ambiente paralelo, para que no interfiera con los avances en el prototipo y se desenvuelvan normalmente.

- El uso de certificados para tener un nivel más alto de autenticación, controlará los intentos de violación de seguridad.

- Tenemos que considerar un procedimiento de Backup ó copia diario de la base de datos, por si se presenta algún tipo de pérdida de información de la misma.

CAPITULO VII

MEJORES PRACTICAS

- Hemos conectado correctamente las múltiples interfaces de tal manera que nos permita la comunicación con diferentes plataformas.

- Automatización del proceso, presentación de la información en tiempo real.

- Aplicación del modelo de tres capas, desarrollando nuestro propio servidor de aplicaciones y acoplándolo a la Base de Datos y a cada una de las aplicaciones Cliente.

- Mejoramiento de la performance debido a una menor interacción con la base de datos; la cual resulta ser mucho más significativa.

- El uso de gráficos para mostrar el comportamiento de la aplicación, por ejemplo en los tiempos de respuesta de la información hacia los clientes.

CAPITULO VIII

CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

- Se logró la comunicación mediante TCP entre diferentes aplicaciones desarrolladas en distintas herramientas de software (Microsoft, Sun) y sobre diferentes plataformas (Windows, Unix, Linux), estas fueron una aplicación Servidor desarrollado en ANSI C, aplicación Servidor Java y la aplicación Cliente bajo Windows.

- La información que proviene de la aplicación Servidor a la aplicación Cliente es actualizada de manera automática, sin haber de por medio algún mecanismo ejecutado por parte del usuario; en cambio en la aplicación Web se requiere que el usuario ejecute un evento para refrescar la información en una página Web (presionar el botón Actualizar del Navegador).

- Actualmente en la Bolsa de Valores de Lima, se muestra información de Negociación de Rueda en una página Html. El periodo de tiempo

que demora en actualizar la información de Negociación de Rueda es de 5 minutos. Este periodo de tiempo se tomó en cuenta debido a:

- Cálculo de Índices Consolidados y demás ratios, entre ellos se tienen el Índice General y el Índice Selectivo.

- Actualización en la Base de Datos.

- Luego de estas tareas se procede a enviar la información a todos los usuarios conectados al servicio.

- En cambio en la aplicación Servidor, la información fluye de manera automática, siendo la tardanza en nivel de milisegundos que es lo que tarda en cargar las cotizaciones eliminando de este modo la tarea de actualización en la Base de Datos, de la cual se encarga otro proceso sincronizado con la aplicación.

- En performance la aplicación Servidor resulta positivo respecto a la aplicación Web, esto se remitió a una medición del nivel de recursos del sistema. Durante la ejecución de una aplicación Web, que consulta indirectamente la Base de Datos, su nivel de consumo de recursos de memoria en promedio llega a ser 90%. Mientras que en nuestra aplicación, que envía información a través de TCP, su nivel de consumo de memoria y CPU (en actividad) puede llegar hasta 5%.

- El entorno corporativo (Bolsa de Valores de Lima) seleccionado para nuestra aplicación fue el idóneo en vista que la afluencia de información se comporto de manera constante y las consultas de los usuarios no tuvo ningún inconveniente.

RECOMENDACIONES

- Utilizar un archivo de configuración para los casos en el cual se definen las variables que puedan sufrir cambios de tal manera que la aplicación sea flexible, permitiéndonos cambiar algunas variables de manera dinámica, por ejemplo, la ruta completa donde se encuentre la aplicación dependiendo del sistema operativo Windows, Linux ó Unix ya que cada uno de ellos maneja diferentes sistemas de archivos, además otras variables que pueden cambiar son el *"host"* donde se ubica la aplicación Servidor, el puerto que servirá de escucha para la aplicación Servidor.

- Mecanismos de detección de errores. Se pueden tomar en cuenta alertas vía mail ó archivos *"log"*, para evitar tiempos de desinformación durante una caída de la aplicación Servidor ó durante su mal funcionamiento.

- La aplicación Servidor se debería manejar en los entornos Unix y Linux, por ser más estables que Windows y por tener aquellos características de Servidor.

- Debe existir una persona encargada de administrar esta aplicación (operador), adjuntándose a las herramientas de monitoreo de Red; esta persona tiene que estar calificada en la administración de redes.

- Utilizar un ambiente de prueba, paralelo al que esta activo, para que no interfiera con los avances en el prototipo y de desenvuelva normalmente.

- Tener presente el desarrollo de módulos de encriptación, durante la autenticación y durante el envío de información ya que esto; es parte del nivel de seguridad que debería tener todo sistema de transferencia de datos.

- Tomar en cuenta un número de 5 usuarios para las pruebas de la aplicación. Esto contribuye a una aceptable concurrencia en el tráfico de información.

- Realizar diariamente backup del directorio de la aplicación, para tener respaldo ante una posible pérdida de datos.

CAPITULO IX

GLOSARIO DE TERMINOS

APPLET: Se llama Applet a cualquier programa pequeño hecho en Java que puede referenciarse en una página Html. Los Applets difieren de los programas hechos específicamente para Java en que no serán autorizados a acceder ciertos recursos de la PC local, como archivos y dispositivos de Hardware, y no puede comunicarse con otras computadoras conectadas a una Red Local.

ARCHIVO: Cualquier archivo creado dentro de una aplicación: por ejemplo, un documento creado por un procesador de textos, una hoja de cálculo, una base de datos o un gráfico. También denominado Documento.

BACKBONE: Nivel más alto en una red jerárquica. Las redes aisladas ("STUB") y de tránsito ("TRANSIT") conectadas al mismo eje central están interconectadas.

CABLE CROSSOVER: Cable coaxial de transmisión de datos de punto a punto por medio de tarjetas de red

CCITT: International Consultative Committee on Telegraphy and Telephony. Comité Consultivo de Telegrafía y Telefonía. Organización que establece estándares internacionales sobre telecomunicaciones.

CHECKSUM: (Suma de comprobación). Es el más elemental de los controles de errores. Consiste normalmente en un único octeto obtenido mediante una o varias operaciones matemáticas realizadas sobre todos los octetos de un bloque de datos con el fin de controlar los posibles errores que se produzcan durante una transmisión. Si el "*checksum*" no es el esperado se reenvía el bloque erróneo.

CLIENTE/SERVIDOR: Se denomina así al binomio consistente en un programa Cliente que consigue datos de otro llamado servidor sin tener que estar obligatoriamente ubicados en el mismo ordenador. Esta técnica de consulta remota se utiliza frecuentemente en redes como Internet.

DATO: Término general para la información procesada por un ordenador.

DBA: Data Base Administrator, como su nombre lo dice administra y organiza las diferentes bases de datos.

DBMS: Data Base Manager System que significa Sistema Manejador de Base de Datos.

EMAIL: Correo enviado a través de medios electrónicos. Aunque originalmente se trataba de mensajes de texto, actualmente puede ser otro todo tipo de información.

FIREWALL: Sistema que se coloca entre una Red Local e Intenet. La regla básica es asegurar que todas las comunicaciones entre dicha red e Internet se realicen conforme a las políticas de seguridad de la organización que lo instala. Además, estos sistemas suelen incorporar elementos de privacidad, autentificación, etc.

FRONT-END: Es lo que se ve, en el entorno usuario pudiendo ser en una aplicación Windows o en una aplicación Web.

HARDWARE: Son todos los componentes físicos que componen una PC.

HOST: Computadora que permite a los usuarios comunicarse con otros sistemas centrales de una red. Los usuarios se comunican utilizando programas de aplicación, tales como el correo electrónico, Telnet, WWW y FTP.

HTML: Lenguaje utilizado para crear los documentos de hipertexto que se emplean en la WWW. Los documentos Html son simples archivos de texto que contienen instrucciones (llamadas tags) entendibles por el Navegador Web.

HTTP: Protocolo que se utiliza para transferir archivos de hipertexto a través de Internet. Requiere de un programa Cliente de HTTP en un extremo y una aplicación Servidor de HTTP en el otro extremo. Es el protocolo más importante de la WWW.

HTTPS: El protocolo de comunicación seguro empleado por los servidores de WWW con en clave. Esto es usado para transportar por Internet información confidencial como el número de tarjeta de crédito.

ICMP: Es una extensión del Protocolo de Internet (IP), y permite generar mensajes de error, paquetes de prueba y mensajes informativos relacionados con IP. Básicamente, se usa para comprobar la existencia de la máquina consultada.

INTERNET: Conjunto de redes conectadas entre sí, que utilizan El protocolo TCP/IP para comunicarse.

JAVA: Lenguaje de programación orientado a redes. Fué diseñado por Sun Microsystems específicamente para escribir programas que pudieran bajarse

y ejecutarse en la computadora local, sin temor que éstos pudieran contener virus o provocar algún daño en los archivos. Las páginas Web pueden contener pequeños programas hechos en Java llamados Applets, por ejemplo: para producir animaciones u otros efectos vistosos (también se puede programar aplicaciones completas, como calculadoras, chat, etc.).

KERNEL: Núcleo o parte esencial de un Sistema Operativo. Provee los servicios básicos del resto del sistema.

LAN: (Red de Area Local). Red de computadoras ubicadas en el mismo ambiente, piso o edificio.

LINUX: Versión de libre distribución del sistema operativo Unix.

ORACLE: Manejador de Base de Datos, robusto que permite un manejo muy seguro y confiable con su información, es capaz de soportar más información que el propio SQL

OSI: OPEN SYSTEMS INTERCONNECTION (Interconexión de Sistemas Abiertos) Conjunto de protocolos diseñados por comités ISO con el objetivo de convertirlos en estándares internacionales de arquitectura de redes de computadoras.

PC: Acrónimo de Personal Computer (Computadora Personal).

PROTOCOLO: Conjunto de reglas que definen la forma en que las computadoras se comunican entre sí.

PROXY: Es un ordenador que pertenece a una Red y que hace de intermediario entre los diferentes puestos de la Red y la conexión a Internet para estos puestos. Esto facilita que todos los ordenadores de esta red tengan que pasar por el Proxy para poder entrar en Internet, de forma que podamos aumentar la seguridad, administrar el número de conexiones y realizar función de caché de páginas vistas en las Web de Internet.

RFC1700: Acrónimo de Requests For Comments (Peticiones de comentarios) Propuestas de estándares plasmadas en documentos numerados, siendo '1700' el número identificador de cada RFC. Su uso es muy frecuente en Internet.

RUNTIME: En tiempo de ejecución.

SCRIPT: Programa informático que se interpreta.

SOCKET: Tipo de conexión (zócalo) entre el microprocesador y la placa madre.

TCP/IP: Conjunto de protocolos que definen a la Internet. Fueron originalmente diseñados para el sistema operativo Unix, pero actualmente puede encontrarse en cualquier sistema operativo.

TELNET: Acrónimo de TELECOMUNICATIONS NETWORK (Red de telecomunicaciones) Protocolo de alto nivel que permite a un ordenador conectarse remotamente a otro de tal forma que el ordenador emisor parece ser el ordenador llamado. Es uno de los recursos más utilizados en 'Internet'. Para que el ordenador remoto no rechace la llamada es necesario disponer de una cuenta autorizada que se denomina "*userid*" (identificador de usuario). Esta cuenta se protege con un "*password*" (palabra clave) con el fin de evitar usurpaciones de identidad. Existen bastantes servicios en Internet que permiten utilizar una cuenta especial de dominio público con el fin de que no haga falta una cuenta específica. Evidentemente esta cuenta tiene movimientos limitados y, normalmente, se llama "*guest*" (invitado).

TIEMPO REAL: Cuando una acción realizada en el ordenador progresa paralelamente al tiempo del "*mundo real*", se dice que la acción ocurre en tiempo real. Un ejemplo sería un programa que mostrará el desarrollo de una colonia de bacterias que se reprodujeran con el mismo ritmo de crecimiento con que lo haría una colonia real. Últimamente han aparecido muchos juegos que requieren reacciones en tiempo real. La mayoría de las máquinas de los salones recreativos transcurren en tiempo real.

UNIX: Sistema operativo que fue diseñado para que lo usaran muchas personas a la vez (es multiusuario) a la vez tenia incorporado al protocolo TCP/IP. Es el sistema operativo más utilizado por los servidores de Internet.

VISUAL C++: Lenguaje de programación basado en ANSI C, implementado para aplicaciones visuales.

WAN: Una red de computadoras de gran tamaño, dispersa por un país o incluso por todo el planeta.

WEB: Conjunto de recursos que pueden accederse utilizando un Navegador, mediante el protocolo HTTP.

WINDOWS: Sistema operativo desarrollado por Microsoft, bastante comercial en nuestro medio. Utiliza interfaces visuales.

CAPITULO X

BIBLIOGRAFÍA

- Comunicación de datos, redes de computadores y sistemas abiertos (Cuarta Edición). 1996
  Editorial Addison Wesley Longman
- Stallings,W. Comunicaciones y redes de computadores (Quinta Edición). 1997
  Editorial Prentice Hall
- TCP/IP ilustrado Volumen 1. 1997
  Stevens, W.Richard)
- Documento Interno – Distribución de Datos a los Vendors
  Bolsa de Valores de Lima
- Página Web de la Bolsa de Valores de Lima
  http://www.bvl.com.pe
- Página Web de Elementos Notacionales del UML 1.0
  http://www.cs.ualberta.ca/~pfiguero/soo/uml/