4
2EJ

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE CONTADURIA Y ADMINISTRACION

CONSTRUCCION DE UTILERIAS DE COMUNICACION PARA CLIENTE-SERVIDOR BASADAS EN OBJETOS

SEMINARIO DE INVESTIGACION INFORMATICA
Que para obtener el Titulo de:

LICENCIADO EN INFORMATICA

PRESENTAN:

MOISES AVILA PEREZ
RAFAEL CARDENAS ORTEGA
MARIA TERESA GUTIERREZ RODEA

Asesor del seminario:

M. EN C. MARINA TORIZ GARCIA

MEXICO D.F. 1995 1994

Gracias, Padre del cielo, nos diste estudios para que los administraramos.
Este, es el fruto de la semilla que sembraste en nuestro corazón.
Gracias por nuestras familias, nuestros amigos y por
darnos lo mejor que tienes: tu amor...

Agradecemos...

...en especial a la **M. en C. Marina Toriz García**, asesora del seminario, por su experiencia, consejos y tiempo que compartió para que éste se realizara de la mejor manera.

...y a quienes ayudaron a que este seminario se hiciera realidad:

Ing. Grabriela Olivera Godínez

Dedicado a:

...mis padres **Martha Pérez** y **Luis Avila** con todo cariño por el amor y apoyo que me han brindado durante toda mi vida, a ellos, mi reconocimiento y respeto.

...a **Angélica**, amor de mi vida e inseparable compañera en todo momento, que siempre a creido en mi.

...a **Adriana**, la niña más hermosa de mi existencia.

...a mis familiares que siempre me han apoyado con sus consejos y guías.

...a mis amigos y compañeros de toda la carrera.

Agradezco especialmente a todas personas de la D.G.S.C.A que me apoyaron durante mi formación académica.

Dedicado a mis padres, **Leonor Ortega** y **Raúl Cárdenas**, por su amor, comprensión y apoyo, sin ellos este trabajo no tendría el significado que ahora tiene.

Gracias...

...a **Alejandro**, **Elizabeth**, **César** y **Gerardo**, hermanos de sangre y espíritu, de quienes he obtenido grandes enseñanzas.

...a **Evelyn Gómez H.**, por su amor y amistad, que han sido fuente de inspiración y fortaleza durante mis estudios, mi crecimiento espiritual y mi vida profesional, para lograr todos mis objetivos y sueños.

...a **Luis A. Viveros**, el mejor amigo que tengo, gracias por estar conmigo en todo momento.

...al **Ingeniero Norberto Arrieta Márquez,** por compartir sus conocimientos técnicos.

Dedicado a :

... mis queridos padres :

Teresa Rodea y **Delfino Gutiérrez** por su apoyo, enseñanza, cariño, comprensión y dedicación durante toda mi vida.

... mis hermanos:

Sergio, por su motivación en el transcurso de la carrera.

Verónica, por su ayuda en las etapas más difíciles de mis estudios.

Alberto, por su apoyo y ejemplo.

Jorge, por su cariño en toda ocasión.

Gracias a:

... mis tíos, **Ernesto**, **Humberto**, **Javier** e **Irma** y demás familiares por sus buenos consejos.

... mis compañeros de trabajo, **Lucy**, **Adriana** y **Adán** por su apoyo.

INDICE

INDICE DE ILUSTRACIONES

INDICE DE TABLAS

INDICE DE LISTADOS

INTRODUCCION

Hemos concluído nuestro Seminario de Investigación en Informática y hay un sentimiento mezclado de satisfacción por el deber cumplido, pero con la melancolía de ver terminado una etapa más de nuestras vidas.

Como resultado tenemos el proyecto denominado "**CONSTRUCCION DE UTILERIAS DE COMUNICACION PARA CLIENTE-SERVIDOR BASADAS EN OBJETOS**", y su implementación en la empresa "ELECTROPURA" para el enlace del corporativo y sus diferentes plantas purificadoras con el fin de obtener información en forma precisa, oportuna y veraz, así como en la cantidad y calidad esperada, que repercutirá en una mejor TOMA DE DECISIONES.

Otro de los logros importantes, es que con la construcción de dichas utilerías se puede incrementar el de por si poderío de la herramienta C++ versión 4.0 .

¡Sí!, BORLAND Co., propietaria de la herramienta y una de las empresas más importantes a NIVEL INTERNACIONAL en lo que a computación se refiere, a la fecha, en su biblioteca estándar NO tiene integradas estas utilerías, motivo por el cual, ya han sido enviadas para su consideración y posible implementación en su producto.

GRUPO SESER

México, D.F., febrero 20, 1995

A quien corresponda:

Por medio de la presente se da constancia de que las personas

Avila Pérez Moisés
Cardenas Ortega Rafael
Gutiérrez Rodea María Teresa

trabajaron durante los meses de septiembre a noviembre de 1994 en las oficinas corporativas de Electropura y en una de sus plantas, para la elaboración del Seminario de Investigación Informática llamado Construcción de Utilerías de Comunicación para Cliente-Servidor basadas en Objetos.

Las sesiones de trabajo se llevaron a cabo con el personal del área de sistemas, así como el personal del área contable. Los objetivos fundamentales de este trabajo fueron resolver en parte la problemática que vive Electropura y por otra, ayudar en el desarrollo del seminario de investigación de las personas antes mencionadas.

El programa final fue implementado y sirve de base para el desarrollo de nuevas alternativas de aplicación.

La presente se extiende para los fines que a los interesados convenga.

Esperando que sea de utilidad la información proporcionada, quedo de usted.

Atentamente

Act. Ma.Antonieta Elvira Sandoval
Gerente de Sistemas de Grupo SESER

Cliente-Servidor, redes de computadoras y objetos, son palabras que cada día toman más importancia en el medio informático de cualquier empresa o institución educativa. Esto, más que una moda, se perfila para dar solución a problemas propios del desarrollo de sistemas, tales como:

- El procesamiento distribuido.
- El mantenimiento de software.
- La facilidad de uso del software.
- La rapidez en el desarrollo de sistemas.

Por otra parte, la apertura de mercados provoca que las empresas se establezcan en diferentes lugares, geográficamente distantes y necesiten consultar y/o procesar información centralizada, para alcanzar sus objetivos. Es aquí donde las telecomunicaciones juegan un papel muy importante, para ayudar a resolver esta situación.

Ahora que está en auge la tendencia al **downsizing**, las empresas deberán lograr que las aplicaciones funcionen en un ambiente heterogéneo de cliente-servidor que, por supuesto, cuenta con distintos protocolos de red, sistemas operativos y bases de datos.

La realización del presente seminario denominado Construcción de Utilerías de Comunicación Basada en Objetos se llevó a cabo en la empresa Electropura para el enlace de las oficinas corporativas y sus plantas purificadoras.

El Grupo está constituido por 5 Plantas purificadoras de agua y las Oficinas Corporativas, ubicadas de la siguiente manera:

Corporativo	Distrito Federal.
Planta Andalucía	Col. Alamos, D.F.
Planta Vallejo	Zona Industrial Vallejo, D.F.
Planta Los Reyes	Edo. de México.
Planta Ecatepec	Edo. de México.
Planta Cancún	Quintana Roo.
Planta Tlalpan	D.F.,(inauguración en mayo de 1995).
Planta Tlalnepantla	Edo. de México, (inauguración en mayo de 1995).
Planta Cuernavaca	Morelos, (inauguración en septiembre de 1995).

El volumen de ventas de garrafón de 19 litros de agua purificada es el siguiente para cada planta:

Planta Andalucía	16,000 garrafones diarios.
Planta Vallejo	35,000 garrafones diarios.
Planta Los Reyes	50,000 garrafones diarios.
Planta Ecatepec	55,000 garrafones diarios.
Planta Cancún	20,000 garrafones diarios.
Planta Tlalpan	70,000 garrafones diarios (estimado).
Planta Tlalnepantla	70,000 garrafones diarios (estimado).
Planta Cuernavaca	35,000 garrafones diarios (estimado).

Lo que genera una venta diaria, actual, de 176,000 garrafones que se traducen en N$1,056,000.00 diarios (sin tomar en cuenta los estimados de las 3 plantas nuevas).

Como se puede apreciar el negocio de purificación de agua es muy rentable, y esto es reflejo de los métodos de producción y el monitoreo constante del área corporativa a cada planta, por lo que es necesario tener información fluida.

Actualmente, la comunicación entre Oficinas Corporativas y plantas purificadoras de agua se ha venido dando a través de mensajería, teléfono y fax.

Algunos problemas que se presentan en la comunicación de los reportes emitidos entre plantas y las Oficinas Corporativas son:

1. Las líneas del conmutador de las plantas están saturadas, por lo que los reportes se retrasan.

2. Al dictarse las cantidades de cada movimiento no se entienden.

3. El fax que se envía no es recibido con buena calidad, por lo que se pierde tiempo al rectificar la información.

4. En la captura del reporte final se incurre en error humano.

Por la importancia de consolidar información es necesario implementar un sistema de comunicación remota por medio de equipos de cómputo para tener enlazado a todo el grupo.

A través de una aplicación que se realice, por medio de un modelo cliente-servidor, se tendrá una comunicación directa entre la máquina donde se genera la información y la que la demanda, por medio de una línea telefónica, y de esta manera poder aprovechar la infraestructura instalada y agregar elementos que ayuden a completar este ciclo administrativo. Actualmente se utiliza el manejador de base de datos de Informix con el cual se han desarrollado aplicaciones que explotan una base de datos que contiene la información de las ventas realizadas en la planta. Este manejador lo definimos como el servidor de donde se explota la información para ser consultada en un programa, siendo éste el cliente.

La configuración de hardware y software de cada planta es la siguiente:

HARDWARE

Sede	Servidor de aplicaciones	Terminales	PC's
Oficinas Corporativas	IBM RS/6000	10	40
Planta Andalucía	IBM RS/6000	8	6
Planta Vallejo	HP Pentium	10	10
Planta Los Reyes	HP Pentium	15	10
Planta Ecatepec	HP Pentium	15	10
Planta Cancún	HP i486	8	8
Planta Tlalpan	*	*	*
Planta Tlalnepantla	*	*	*
Planta Cuernavaca	*	*	*

* Configuración aún no definida

SOFTWARE

Sede	Servidor de aplicaciones	Computadoras personales
Oficinas Corporativas	UNIX, Informix MCBA	Windows, DOS, Microsoft Office, Módulo de desarrollo (Borland C++)
Planta Andalucía	UNIX, Informix MCBA	Windows, DOS, Microsoft Office
Planta Vallejo	UNIX, Informix MCBA	Windows, DOS, Microsoft Office
Planta Los Reyes	UNIX, Informix MCBA	Windows, DOS, Microsoft Office
Planta Ecatepec	UNIX, Informix MCBA	Windows, DOS, Microsoft Office
Planta Cancún	C-Bos, UNIX, Informix MCBA	Windows, DOS, Microsoft Office
Planta Tlalpan	*	*
Planta Tlalnepantla	*	*
Planta Cuernavaca	*	*

* Configuración aún no definida

Es decir, la arquitectura que se tiene actualmente es una red de redes que enlaza a más de 100 computadoras diferentes, además la empresa cuenta con independencia de proveedores, lo que se traduce a contar con arquitecturas abiertas.

Todo esto se controla desde el corporativo, que a futuro contará con un servidor a gran escala.

El sistema que se desarrolla en este seminario de investigación lleva el nombre de SICE (Sistema Integral de Comunicación para Electropura). Los objetivos que se persiguen, de acuerdo a los elementos antes mencionados, son:

a) Objetivos Administrativos:

1. Facilitar la recopilación de la información de las plantas en un área central.

2. Aprovechamiento de recursos actuales de la empresa.

b) Objetivos Informáticos:

1. Aplicación de la arquitectura cliente-servidor en un procedimiento administrativo.
2. Desarrollo sencillo por medio de objetos.

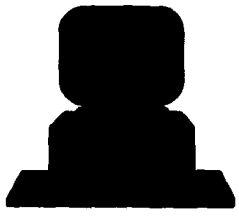
En la ilustración i.1 se observa que la tendencia de los grandes negocios es la utilización de la arquitectura cliente-servidor sobre otras opciones que se ofrecen actualmente.

En la ilustración i.2 se ve como ha evolucionado a través del tiempo la arquitectura cliente-servidor.

Estas ilustraciones refuerzan nuestra tesis de que por medio de esta arquitectura, la realización de sistemas a través de cliente-servidor y el paradigma de objetos, facilita el desarrollo de aplicaciones que dan solución al procesamiento de información.

Tendencias (Negocios grandes)

Cliente-Servidor	**69%**
Orientados a UNIX	**45%**
Alternativas UNIX	**70%**
Wan	**44%**
PCs/LAN 94-95	**54% / 69%**
Mercado P-3	**Profesionista nómada**
Minis actual/futuro	**73% / 64%**



La tendencia hacia la arquitectura cliente-servidor sigue creciendo en los corporativos mexicanos.

Ilustración i.1
Tendencia de los negocios grandes
Tomada de "BANCA Electrónica" No. 14, Enero de 1995.

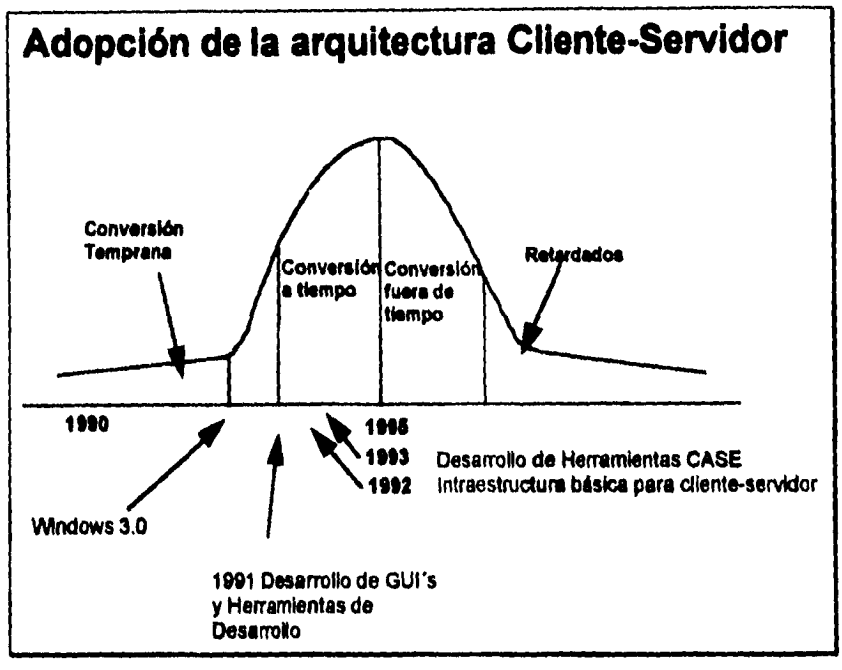


Ilustración I.2
Adopción de la arquitectura cliente-servidor
Tomado de "Soluciones Avanzadas " No. 1 Octubre 1992.

Para poder iniciar cualquier proyecto de sistemas, siempre es importante definir al principio del mismo:

- De qué manera se resolverán los problemas.
- Qué herramientas, métodos y recursos se pueden utilizar.

Cuando se inicia un trabajo con nueva tecnología se reflexiona sobre los beneficios e inconvenientes, además de los tabúes que esto genera, sabemos que no es tan sencillo implementar la tecnología cliente-servidor. Todo esto debido a las fantasías que existen alrededor de la arquitectura - que la tecnología mágicamente se hace sin la necesidad de planear y, por supuesto, con el costo de hacer nada-. ¿Suena esto absurdo?, aún cuando se sepa que estos mitos no tienen sentido, es sorprendente la frecuencia con que tales expectativas negativas hacen que se

realice un gran esfuerzo en el desarrollo, especialmente cuando se está trabajando con nuevas herramientas que son más poderosas y fáciles de usar que otras.

Por esto, en los primeros tres capítulos definimos los conceptos sobre la arquitectura cliente-servidor, tipos de redes y comunicaciones, y el paradigma de objetos, para tener las bases teóricas y la justificación de por qué utilizar, o no utilizar, las diferentes opciones del mundo de la informática.

En el primer capítulo, se describen las diferentes arquitecturas de cómputo que existen para procesar información:

- Arquitectura Mainframe.
- Arquitectura de Servidor de Archivos.
- Arquitectura Cliente-Servidor.

El tema de redes y comunicaciones se describe en el segundo capítulo; se hace una revisión a los diferentes tipos de comunicación, del modelo OSI y se señala la justificación del uso de la interfaz RS-232C.

El paradigma de objetos, su metodología y conceptos se describen en el tercer capítulo, así como una comparación con otras metodologías muy conocidas y utilizadas, como son el Diseño Estructurado y el modelo de Jackson, y además se señalan las herramientas a utilizar y se hace un análisis de las etapas del desarrollo de la metodología basada en objetos.

En el cuarto capítulo se presenta el análisis y desarrollo de la aplicación. Para la fase de análisis aplicamos una guía de 12 pasos para elaborar un modelo cliente-servidor. Los pasos son los siguientes:

1. Uso de lenguaje no técnico.
2. Especificar las metas y beneficios del proyecto función por función.
3. Partir el proyecto en fases y asignar las funciones que serán discutidas en cada fase.
4. Terminar el nuevo sistema del negocio, manejo, objetivos técnicos, identificar similaridades y diferencias entre estos objetivos y buscar concientizar cada uno.
5. Desarrollar la mejor estimación en cada fase.
6. Determinar el negocio y las barreras técnicas para encontrar los requerimientos.
7. Determinar cuáles procesos estarán en el servidor y cuáles en el cliente.
8. Decidir cuáles componentes del software deberán ser comprados y cuáles desarrollados, después encontrar las herramientas correctas.
9. Definir el impacto del proyecto en los recursos de la información.
10. Entender el impacto cultural y organizacional en la corporación y desarrollar un plan para manejar los cambios.
11. Desarrollar un plan a seguir para el grupo de técnicos.
12. Desarrollar sistemas utilizando metodología incremental

En la fase de desarrollo utilizamos el paradigma de objetos. Sus etapas son:

1. Identificación de objetos y clases.
2. Identificación de las operaciones de los objetos y clases.
3. Establecer la relación entre las clases y los objetos.
4. Implementar cada objeto.

Por último, mencionamos las conclusiones obtenidas por el desarrollo de este seminario de investigación, así como las referencias bibliográficas indicadas a lo largo del presente y la bibliografía de consulta utilizada para el mismo.

Como información complementaria se encuentran los apéndices y glosarios de términos utilizados en el desarrollo de esta investigación.

Agradecemos la ayuda prestada para el desarrollo de este seminario a Grupo SESER por permitirnos accesar a sus instalaciones y documentos de sistemas, así como al personal de las áreas involucradas para la mejor comprensión del objetivo del negocio.

Para el mejor entendimiento del presente documento se utiliza la siguiente notación:

Cada pie de página define un concepto al momento.

Las palabras remarcadas con **negrilla** se definen en el glosario con más detalle.

Los números encerrados en [ ] indican una referencia bibliográfica definida al final del documento. Estas nos permiten conocer el libro para obtener mayor información sobre el tema que se trata.

I ENTENDIENDO LA ARQUITECTURA CLIENTE-SERVIDOR

I.1 Procesos Distribuidos

La arquitectura cliente-servidor se originó por el cambio que hubo en el uso de la computación a mediados de los 80's, debido a que las redes de computadoras empezaron a tener un mayor auge sobre el uso de los **mainframes**, por lo que se popularizó su uso y el acceso a la información fue más fácil para los usuarios finales, los que ya no eran solamente profesionales en el ramo, sino además se sumaron los expertos en el análisis de ventas, mercadotecnia, servicios al cliente, etc., los cuales analizaban los datos y los usaban en el desarrollo de sus tareas.

Como la mayoría de los usuarios dejaron de ser programadores para convertirse en usuarios finales, la demanda por mejorar las interfaces gráficas se incrementó, y cabe comentar que el cómputo de los **mainframes** no se orientó a esas nuevas necesidades de trabajo.

El término "proceso distribuido" se refiere a la conexión de múltiples máquinas a través de una red de comunicaciones, con el fin de que entre todas realicen un sólo proceso.

Muchos niveles o variedades de proceso distribuido pueden ser posibles. Nosotros trataremos en este trabajo el caso de Electropura que se comenta ampliamente en el capítulo IV y que involucra un proceso de Base de Datos en una máquina (**backend**) y la aplicación en otra (**frontend**).

I.1.1 Backend vs. Frontend

El objetivo de muchos sistemas administrativos es soportar el desarrollo y ejecución de aplicaciones a una base de datos. Un sistema de base de datos puede ser dividido en dos estructuras simples, llamadas backend y frontend.

- El backend es el DBMS[1] en si. Este soporta todas las funciones básicas del DBMS, que son: definición, concurrencia, manipulación, seguridad e integridad de datos. En particular, ayuda a soportar los niveles externos, conceptuales e internos.
- El frontend es una aplicación que corre arriba del DBMS.

Un ejemplo de proceso distribuido en donde encontramos estas dos estructuras lo podemos ver en la ilustración I.1, a esta forma de procesamiento lo conocemos también como un sistema cliente-servidor (el frontend es el cliente, el backend es el servidor).

[1] Data Base Management System. Sistema Manejador de Base de Datos.

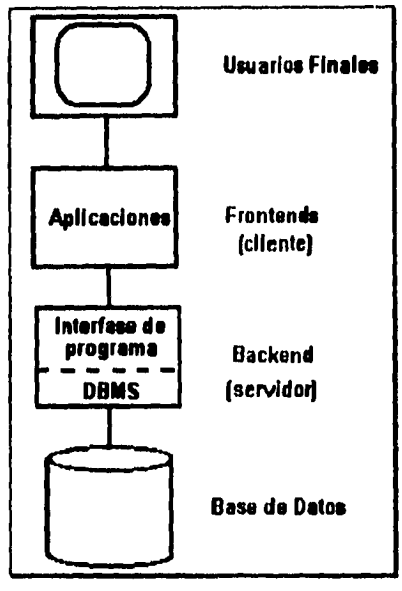


Ilustración I. 1
Backend y Frontend

Algunas definiciones del modelo cliente-servidor son las siguientes:

"El modelo de interacción cliente-servidor constituye la base de un gran número de aplicaciones que se ejecutan en red. El término servidor se aplica a cualquier proceso que ofrece un servicio, y el término cliente a cualquier proceso que utiliza los servicios ofrecidos por un servidor.", Dr. Marcelo Mejía. Ing. en sistemas, profesor del ITAM.

"El modelo cliente-servidor es un esquema de interconexión de diversas computadoras que presupone un medio para conectarlas y dos grupos de computadoras: unas llamadas servidores y otras llamadas clientes.", Ing. Enrique Montfort, especialista en sistemas, Digital de México.

Como podemos observar existen diferentes definiciones que tratan de describir qué es el cómputo cliente-servidor; sin embargo, este concepto no es fácil de explicar.

I.2 Evolución hacia la Arquitectura Cliente-Servidor

Por muchos años el procesamiento de datos se ha realizado en **mainframe** o **servidor de archivos** a través de transacciones orientadas para automatizar tareas repetitivas tales como:

Registro de empleados,
Pago a los clientes,
Registro de ventas,
Pago de nómina, etc.

Cabe mencionar que los programas por lotes o en línea pueden manejar esas tareas fácilmente. Algunas propiedades del estilo de cómputo en los **mainframes** se señalan en la Tabla I.1.

1. Es centralizado con procesamiento multiusuario. Toda la inteligencia[2] reside en el **host** y todos los usuarios la utilizan.

2. Interacción de terminal. Los usuarios interactúan con el mainframe a través de una terminal que sirve solamente para transmitir instrucciones al **host** y desplegar la respuesta del mismo.

3. Sistemas de administración ejecutiva. Un experto a través de "ritos secretos" hace un uso eficiente del **host**, verifica e instala nuevo software, hace copias de seguridad, y crea claves de acceso para tener un control sobre los usuarios que usan el **host**, los que en su mayoría son sólo personal que captura datos.

4. No son mecanismos apropiados para soportar **GUI's**[3].

5. No pueden incrementar el número de usuarios y aplicaciones fácilmente, lo cual se refiere como "escalabilidad".

Tabla I.1
Características de la arquitectura mainframe

Por otra parte el modelo de servidor de archivos se realizó a través de las redes de cómputo, su objetivo inicial fue simplemente compartir archivos y dispositivos entre usuarios, mas adelante esta arquitectura se utilizaba para manejar colecciones de información no estructurada, por lo que tenía las siguientes características (Tabla I.2)

[2] En este contexto se refiere al hecho de realizar tareas tales como los cálculos de operaciones, consultas a la base y llevar la lógica de la aplicación.
[3] GUI's Graphical User Interface. Interfaz Gráfica de Usuario.

1. Mínima Integridad en la información. Esto es porque se utilizan archivos aislados y no hay uniformidad y consistencia en los datos.

2. Seguridad ineficiente. Cualquier usuario puede ver los archivos sin restricción alguna.

3. Bajo Rendimiento. El tiempo de respuesta de cualquier petición es demasiado alto por el mismo modelo.

4. Una compleja administración de los archivos. Al no haber integridad, uniformidad y consistencia en los datos, se necesita invertir mucho tiempo en tener un control adecuado de la información.

5. Pérdida frecuente de datos. No hay control sobre los datos por lo que se pueden perder fácilmente.

Tabla I.2
Características del modelo servidor de archivos

La ilustración I.2 muestra el progreso del cómputo tradicional con mainframes al de cliente-servidor con sus diferentes estilos de trabajo.

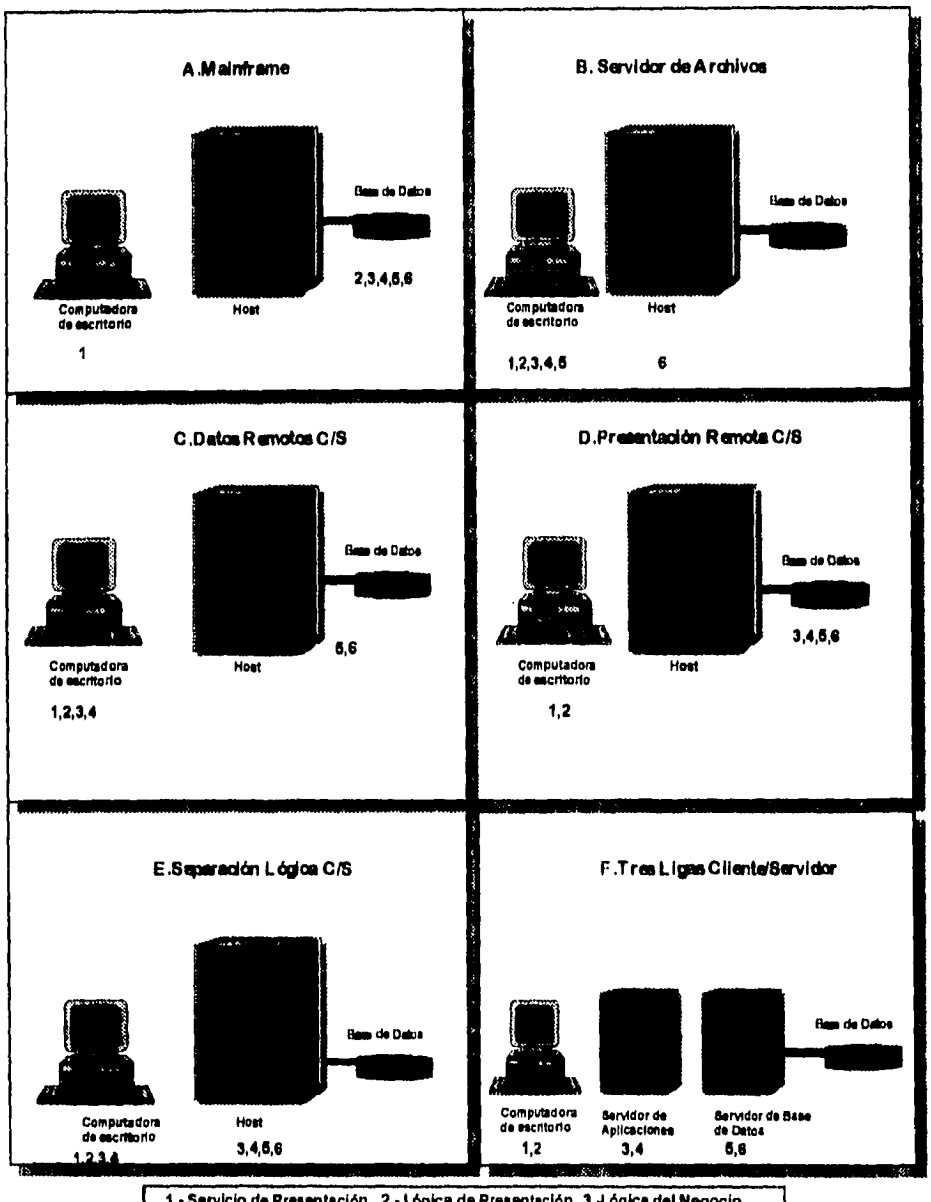


Ilustración I. 2
Evolución del cómputo

I.3 División de una Aplicación

Para entender mejor los límites del cómputo en los **mainframes** y el servicio de archivos, así como las variedades de la arquitectura cliente-servidor dividimos una aplicación en seis partes.

1 Servicio de presentación. Cuenta con un dispositivo que acepta entradas del usuario y las despliega en la lógica de presentación. Este dispositivo puede simplemente desplegar caracteres recibidos de un teclado o un dispositivo que reciba señales de un mouse.

2 Lógica de presentación. Controla la interacción entre el usuario y la computadora. Esta es la especificación que el usuario realiza cuando selecciona un menú de opciones, un botón de selección o escoge un elemento de una lista.

3 Lógica del negocio (o aplicación). Es una colección de decisiones, cálculos y operaciones que la aplicación puede llevar a cabo. Esta puede incluir los cálculos del pago a empleados, la decisión de aceptar una orden de compra, la evaluación de cargar una aplicación o el procedimiento de realizar una transacción la cual ha sido transferida.

4 Lógica de datos. Es la expresión de las operaciones desarrolladas en la base de datos que son necesarias para realizar la lógica del negocio. Como la mayoría de los manejadores de bases de datos se basan sobre el modelo relacional, las expresiones son instrucciones del **SQL** como el SELECT, INSERT y UPDATE.

5 Servicio de datos. Son las acciones que el DBMS toma para realizar la lógica de datos, incluyendo la manipulación, la definición y la transacción de los mismos. Para manipular datos el DBMS típicamente compila las instrucciones del SQL.

6 Servicio de archivos. Accesa el disco y trae los bits que se convierten en datos cuando son vistos a través del DBMS. Los servicios de archivos son usualmente realizados por funciones del sistema operativo.

A continuación se describe como se maneja esta división en las arquitecturas de la ilustración I.2.

I.3.1 Mainframe (Ilustración I.2)

En el ambiente del mainframe la terminal o la computadora de escritorio ayudan al servicio de presentación (1), mientras que el host participa con las funciones restantes (2,3,4,5,6). El host determina las respuestas de las consultas del usuario, realiza la lógica del negocio y recupera la información de la base de datos. De esta forma vemos dos serios problemas en el cómputo del mainframe Tabla I.3.

1. La interfaz del usuario es limitada al estilo de interacción de la terminal. Es difícil, si no es que imposible, proveer una GUI eficiente donde todos los procesos residan en una máquina host. Desafortunadamente, una terminal da una interfaz mediocre para muchas aplicaciones.
2. Cada usuario adicional con su aplicación da una carga sustancial al mainframe. Esto da como resultado la pérdida de escalabilidad, lo cual significa que el cómputo del host está limitado al número de usuarios y aplicaciones que se pueden manejar. Ya que la carga de la red es más alta que la que soporta el host, porque el host no solamente manda datos a la terminal, sino además manda instrucciones de cómo poner los datos en la pantalla, incluyendo etiquetas y prompts.

Tabla I.3
Problemas en el cómputo del mainframe

I.3.2 Servidor de Archivos (ilustración I.2)

La arquitectura servidor de archivos maneja los problemas del mainframe, pero introduce sus propias fallas. Pone todas las partes de la aplicación, excepto los servicios de archivos, en el escritorio. El cliente (computadora de escritorio) maneja la lógica de presentación y de negocios, así como las funciones de la base de datos. El servidor de archivos solo trae los archivos necesarios y los pasa al cliente (ver Tabla I.4).

Mainframe	**Servidor de Archivos**
Host	***Host***
1 Lógica de presentación	1 Servicio de Archivos
2 Lógica del negocio	
3 Lógica de datos	***Computadora de escritorio***
4 Servicio de datos	1 Servicio de presentación
5 Servicio de archivos	2 Lógica de presentación
	3 Lógica del negocio
Computadora de escritorio	4 Lógica de datos
1 Servicio de presentación	5 Servicio de datos

Tabla I.4
Cuadro comparativo de mainframe y servidor de archivos

Sin el retraso de una red, tanto la lógica de presentación y los servicios de despliegue se encuentran juntos, lo que hace más fácil crear una interfaz gráfica para el usuario. El servicio de archivos maneja sólo los archivos que el usuario y la aplicación adicional utilizan incrementando la carga del CPU.

La arquitectura servidor de archivos tiene dos principales desventajas Tabla I.5.

1. La demanda de la computadora de escritorio es muy alta. Una interfaz basada en Windows de Microsoft con la complejidad de código, y un DBMS en la misma máquina pueden saturar a la PC más poderosa.
2. La más seria, es el gran incremento de la carga en la red.

Tabla I.5
Desventajas de la arquitectura servidor de archivos

Podemos ilustrar lo anterior con la aplicación de este trabajo. Se tiene una base de datos con las siguientes tablas de pago de vendedores (liquidaciones) que está compuesta por:

Empleados

Campo	Tipo
rmdb_rn	integer
no_empleado	char(10)
frecuencia	char(2)
periodo	char(2)
codigo_ded_per	char(4)
no_empleado	char(10)
cod_per_ded	char(4)
cod_frecuencia	char(2)
num_periodo	char(2)
monto_dias	decimal(11,2)
pctje_unid	decimal(15,6)
flag	char(2)

Documentos

Campo	Tipo
td_tipo	char(1)
td_descrip	char(20)
td_factor	decimal(2,0)
td_cliente	decimal(1,0)
td_especie	decimal(1,0)
td_folio	decimal(1,0)
td_afecta	decimal(1,0)
td_afectaIq	decimal(1,0)
td_destino	decimal(1,0)
td_comodato	decimal(1,0)
td_pago	decimal(1,0)
td_credito	decimal(1,0)
td_efectivo	decimal(1,0)
td_bonifica	decimal(1,0)

Venta diaria

Campo	Tipo
vd_folio	integer
vd_num_empl	char(10)
vd_num_ruta	smallint
vd_fecha_venta	date
vd_categoria	char(1)
vd_status	char(1)
vd_ayudante_1	char(10)
vd_ayudante_2	char(10)
vd_ayudante_3	char(10)
vd_ayudante_4	char(10)
vd_ayudante_5	char(10)
vd_bandera	char(1)

Detalle_venta

Campo	Tipo
dv_folio	integer
dv_producto	char(15)
dv_num_viaje	decimal(2,0)
dv_salida	smallint
dv_entrada	smallint
dv_repos	decimal(4,0)
dv_cantidad	integer
dv_prestamo	smallint
dv_devolucion	smallint
dv_precio	decimal(16,4)
dv_no_unidad	char(10)
dv_hr_sal	datetime hour to minute
dv_hr_ent	datetime hour to minute

Liquidación_diaria

Campo	Tipo
ld_tipo_doc	char(1)
ld_folio	integer
ld_num_liquida	serial
ld_impte_a_liq	decimal(11,4)
ld_impte_liq	decimal(11,4)
ld_adeudo	decimal(14,2)
ld_status	char(1)
ld_impreso	char(1)

Detalle_liquidación

Campo	Tipo
dl_tipo_doc	char(1)
dl_num_liquida	integer
dl_tipo	char(1)
dl_importe	decimal(11,4)
dl_folio	integer
dl_num_clte	char(6)
dl_especie	decimal(3,0)

Detalle_producto

Campo	Tipo
dp_tipo_doc	char(1)
dp_num_liquida	integer
dp_num_clte	char(6)
dp_tipo	char(1)
dp_folio	integer
dp_producto	char(15)
dp_cantidad	smallint
dp_precio	decimal(15,4)

Una consulta común es la lista de movimientos de cada vendedor por artículo y su detalle de documentos con los que liquidó en una fecha determinada. En un sistema de servidor de archivos, se requiere mover el contenido de las 7 tablas a través de la red para el cliente, considerando la cantidad de datos que son involucrados en una respuesta de este tipo se pueden hacer los siguientes cálculos: Tabla I.6.

Tabla	Bytes por cada tabla
Empleado	80
Documentos	34
Venta_diaria	90
Detalle_venta	103
Liquidación_diaria	57
Detalle_liquidación	34
Detalle_producto	52
Por registro	**450**

Tabla I.6
Cálculo de bytes

Teniendo 66 rutas y cada ruta lleva 3 trabajadores de reparto se tienen 198 trabajadores. Se hacen en promedio 3 viajes al día con 8 productos, y se utilizan en

promedio 3 documentos para pagar (efectivo, crédito y bonificación) con lo que se tiene el siguiente resultado:

198 * 3 * 8 * 3 * 450 = 6,415,200 bytes

Que pasan a través de la red para cada usuario, tomando en cuenta que se tienen 8 o más usuarios en la red al mismo tiempo se puede congestionar, y el cliente, en Electropura, tendrá probablemente dificultad para ejecutar la respuesta con eficiencia. Esta cantidad de información es un factor importante a considerar dentro del Grupo.

El cómputo de la arquitectura cliente-servidor está diseñado para manejar estos problemas separando los componentes de las aplicaciones y poniéndolos donde son más efectivos. Se tienen diferentes argumentos en favor de esta arquitectura: Tabla I.7.

1. Procesamiento paralelo. Múltiples procesadores son aplicados en la tarea a realizar y su trabajo es independiente de los otros pero se complementan.

2. La máquina servidora (backend) se puede considerar una máquina que realiza las funciones del DBMS (Una máquina exclusiva para la base de datos) para de esta forma darle el mejor desarrollo a la base.

3. La máquina cliente por otro lado (frontend) puede ser una estación de trabajo personal, que necesita una buena interfaz para el usuario final, teniendo alto grado de respuesta y fácil utilización por parte del mismo.

4. Diferentes máquinas cliente pueden accesar a la misma máquina servidora. De esta manera una base de datos es compartida por diferentes sistemas frontend (ver ilustración I.3).

5. Escalabilidad. Puede incrementar el número de usuarios y aplicaciones fácilmente sin que el rendimiento se vea afectado.

6. Tolerancia a fallas. Por la forma de la arquitectura se puede recuperar la información aun cuando suceda alguna falla.

7. Compatibilidad. Los datos elementales preparados para una aplicación están disponibles para todas las aplicaciones o consultas. Ningún dato elemental es "propiedad" de una aplicación.

Tabla I.7
Ventajas de la arquitectura cliente-servidor

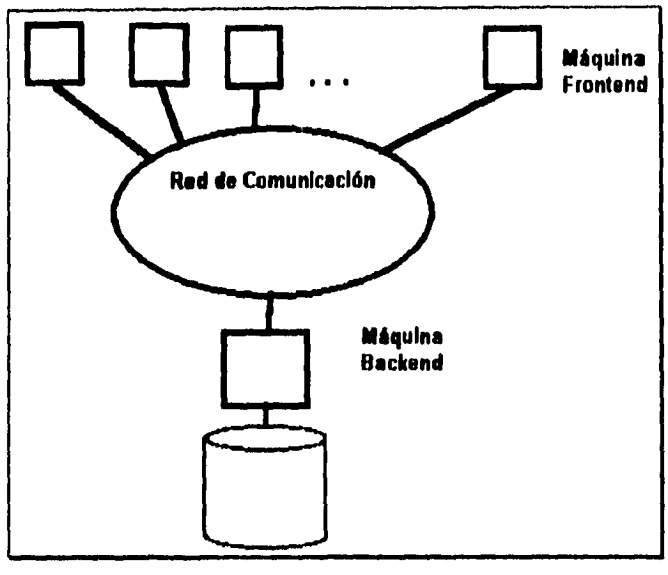


Ilustración I.3
Un backend con múltiples frontend

Como hay diversas posibles divisiones, es necesario entender las diferentes arquitecturas y para cuales clases de aplicaciones son apropiadas cada una.

I.3.3 Cliente-Servidor con Dos Ligas. (ilustración I.2)

La mayoría de las configuraciones cliente-servidor de hoy en día usan el modelo dos ligas, que consiste de un cliente que invoca los servicios de un servidor. Es posible que un servidor tenga la función de cliente para un servidor diferente, en un orden de una arquitectura jerárquica cliente-servidor, pero esto sigue teniendo la aproximación de dos ligas.

Como mínimo, una arquitectura cliente-servidor asume que la presencia de servicios de presentación (1), y la lógica de presentación (2) residen en el cliente. Esto claramente maneja las contrariedades que se presentaban con la interfaz de

terminal permitiendo un manejo de la interfaz local para poder tener así un GUI. La pregunta a contestar es ¿Dónde se colocan las demás cosas?.

La primera posibilidad es poner solamente los servicios de datos (5) y los servicios de archivos (6) en el servidor. Los servicios de presentación (1), la lógica de presentación (2), la lógica del negocio (3), y la lógica de los datos (4) en el cliente. Esta forma de arquitectura es algunas veces llamado "remoto dbms". La mayoría de las definiciones del cómputo cliente-servidor se basan en este modelo, "la aplicación está en el cliente y el DBMS está en el servidor".

Cuando se tienen pocas consultas en el servidor, el modelo remoto del DBMS da la mejor escalabilidad. Sin embargo, para aplicaciones complejas en las cuales se realiza una gran cantidad de interacciones con la base de datos, el trabajo del cliente y el de la red se hacen más pesados. Los resultados de las instrucciones del SQL deben regresar al cliente para que los procese y tome una decisión lógica. El modelo remoto DBMS también crea una burda administración de la aplicación ya que todo el código está en cada cliente. Así cuando se cambia el código, se debe cambiar en todos los clientes.

Se puede reducir la carga en el cliente y la red moviendo parte de la lógica de negocio (3) al servidor (la separación del modelo lógico del cómputo cliente-servidor). El siguiente paso, sería mover toda la lógica de datos (4) al servidor, dejando solamente los servicios de presentación (1) y la lógica de presentación (2) en el cliente (el modelo de presentación remota) para minimizar el impacto en el cliente.

Para decidir entre estos 3 modelos, el diseñador del sistema debe determinar la localización más apropiada para la lógica del negocio. En general no es buena idea dejar que las restricciones del hardware dominen en la planeación.

El ahorro en el desarrollo, operación y administración de un sistema diseñado apropiadamente es equivalente al costo de contar con servidores y clientes poderosos.

El mecanismo más usado es poner la lógica de negocios en los servidores con procedimientos almacenados, lo cual son colecciones de código de procedimientos que incluyen el SQL para accesar a la base de datos. El programador explícitamente invoca un procedimiento almacenado por nombre, pasando valores de los parámetros que éste requiere.

Al ser compilados, el uso atinado de los procedimientos almacenados puede mejorar el desarrollo y reducir la carga del servidor (comparado con el SQL que no está compilado y que es enviado por el cliente). Al igual que en un lenguaje de programación, la compilación acelera la ejecución de un procedimiento almacenado. La compilación consiste de dos partes.

1. El SQL es procesado. Esto asegura el análisis de SQL, revisando la validación de la tabla y los nombres de las columnas, verificando que el usuario de la consulta del SQL tenga los privilegios requeridos para ejecutar el SQL, y así optimizarlo. La optimización es el proceso donde el DBMS usa las definiciones de la estructura en el almacenamiento físico (tal como los índices) para determinar la mejor ruta de acceso a los datos.

2. Cualquier código procedural también se debe compilar. El resultado es almacenado como parte de la base de datos.

Si encimamos procedimientos almacenados con una compleja lógica del negocio, se perderá alguna de las ventajas del desarrollo. Si no se pone la lógica del negocio en procedimientos almacenados (como en el modelo remoto del DBMS), se pueden usar simplemente para mejorar el desarrollo de la base de datos. En este

caso, se almacenaría solamente lo que está compilado en el SQL en el servidor de una manera similar al proceso de **DB2**.

Otro aspecto en el desarrollo es la carga en la red. En vez de enviar instrucciones en el SQL desde el cliente al servidor que tiene la base de datos y regresar los resultados intermedios al cliente, todo los procesos y decisiones pueden realizarse en el servidor con una simple llamada al procedimiento almacenado. Por ejemplo, se podría usar procedimientos del servidor de SQL de Sybase para actualizar una parte de la tabla de tal forma que el precio promedio de todas las partes se incrementaría una cantidad especificada en el parámetro promedio. Para hacer esto, el procedimiento incrementa el precio de todas las partes mayores de cien pesos en un diez por ciento y las que son menores o iguales a cien, un 20 por ciento.

```
CREATE PROCEDURE incremento_precio promedio AS
    while
        (select promedio(precio) from part) <= promedio
    update parte
        set precio = 1.1 * precio
        where precio > 100
    update parte
        set precio = 1.2 * precio
        where precio <= 100
```

Los procedimientos almacenados también reducen la entrada y salida porque invocan a un procedimiento que usualmente está en la memoria cache por lo cual no requiere acceso al disco. Típicamente el procedimiento almacenado en la memoria cache está separado de los datos que están en la misma memoria por lo que de esta forma los datos de entrada y salida no forzarán al DBMS a sacar al procedimiento de dicha memoria.

Los procedimientos almacenados mejoran la base de datos y la integridad de la aplicación porque son compartidos, con lo cual se asegura la consistencia de las operaciones y cálculos. Esto también trae como resultado un incremento en la productividad ya que se escribe y se despliegan los procedimientos comunes una sola vez. Los procedimientos almacenados también pueden mejorar la seguridad permitiendo al DBA[4] poner solamente un usuario que accesa los datos a través de un procedimiento almacenado (no directamente).

Las arquitecturas de dos ligas se ejecutan con algunos problemas en aplicaciones complejas que tienen muchos clientes y bases de datos heterogéneas, tales como: Tabla I.8.

[4] Database Access. Acceso a base de datos.

1. Hacer frente a la administración de un gran número de clientes. Poner la lógica del negocio en el cliente, significa que se debe actualizar una aplicación o arreglarla en todos los clientes. Se debe liberar, instalar y probar cada aplicación que se altere en el cliente, con un esfuerzo substancial. En particular se debe distribuir con la falta de uniformidad en las configuraciones del cliente y con la falta de control en los cambios subsecuentes de la configuración. Mover la lógica del negocio fuera del cliente minimiza este problema. La pregunta restante es ¿Cómo mover la lógica del negocio?.
2. Tratar de usar procedimientos almacenados para implementar lógicas del negocio complejas. Desafortunadamente, los lenguajes procedurales usados por los procedimientos almacenados, generalmente no son capaces de hacerlo como los lenguajes de programación estándar, ninguno de ellos dan un ambiente de programación que tenga capacidades en áreas como verificación y seguimiento, control de versión, y manejo de librerías. Lo más significante es que cada implementación de los procedimientos almacenados usa un lenguaje propio asociado con un DBMS específico. Hoy en día, los procedimientos almacenados no son portables entre los DBMS. Esto restringe la habilidad de cambiar de DBMS sin reescribir todas las aplicaciones, tanto como la habilidad de conectar clientes a bases de datos heterogéneas sin escribir procedimientos para cada DBMS.

Tabla I.8
Fallas en el modelo 2 ligas

La arquitectura de dos ligas también tiene otros defectos. Por ejemplo, ¿Cómo se pueden correr programas por lotes en un sistema de dos ligas ?. El cliente tiene que esperar hasta que el programa por lotes termine, aun cuando el trabajo se ejecute en el servidor. También, mientras se ejecuta en el servidor, los trabajos por lotes y los usuarios interactivos tendrán que interferir unos con los otros.

Finalmente, existe el problema de las transacciones que abarcan múltiples bases de datos bajo diferentes DBMS's. ¿Cómo puede el usuario garantizar la integridad de una transacción distribuida en una base de datos distribuida heterogénea?. Un DBMS tiene una habilidad limitada para desarrollar un COMMIT[5] de dos fases, el protocolo típicamente usado para asegurar la integridad de la base de datos con múltiples DBMS's.

I.3.4 La Arquitectura de Tres Ligas (Ilustración I.2)

La arquitectura de tres ligas resuelve los defectos de la arquitectura cliente-servidor de dos ligas. En el modelo de tres ligas, el cliente se dedica a la presentación lógica de los servicios (2), y tiene un **API** para invocar la aplicación en una capa media. La base de datos del servidor se dedica a los servicios de datos (5) y los servicios de archivos (6) con el cual se podría optimizar sin el riesgo de usar procedimientos almacenados. La capa media es un servidor de aplicaciones en el cual se ejecuta la lógica del negocio, y desde la lógica de los datos invoca los servicios de los datos.

El servidor de aplicaciones puede manejar las transacciones y asegurar la integridad de la base de datos distribuida manejando un proceso de COMMIT de dos fases heterogéneas. El servidor de aplicaciones puede manejar consultas **asíncronas** para asegurar que las transacciones se realicen. También, centraliza la lógica de la aplicación para hacer más fácil la administración y el manejo de los cambios, y dar acceso a los recursos basados en nombres en vez de localizaciones.

[5] El commit de 2 fases es importante cuando se tiene una transacción que puede interactuar con múltiples e independientes manejadores, cada uno de estos manejadores son dueños del conjunto de transacciones recuperables (en particular, esto es importante en el contexto de un sistema distribuido). En estos casos el commit o el rollback es manejado por un componente del sistema llamado Coordinador, quien tiene la tarea de garantizar que ambos manejadores hagan commit o rollback a las actualizaciones al mismo tiempo.

Como no es posible distinguir líneas de separación entre la lógica de presentación, la lógica del negocio, y la lógica de los datos, la lógica del negocio puede aparecer en las tres ligas. Así que se debe determinar el lugar de una función en particular en una liga usando criterios tales como facilidad de desarrollo y verificación, escalabilidad de los servidores, y desarrollo (incluyendo procesamiento y carga de la red). Estos criterios frecuentemente estarán en conflicto uno con otro, así que llevar a cabo un balance satisfactorio puede ser un ejercicio interesante.

La arquitectura de tres ligas tiene otros beneficios para el proceso de desarrollar aplicaciones. Tabla I.9.

1. Desarrollar buenas interfaces requiere de habilidades y disciplinas especializadas. No cualquiera que escribe una buena lógica del negocio puede escribir el código de presentación.
2. Separando las plataformas en las cuales estos componentes corren, se hace más fácil organizar la implementación reflejando los diferentes conjuntos de habilidades en la empresa.

Tabla I.9
Beneficios de la arquitectura de tres ligas

Sin embargo, un buen lenguaje para construir interfaces puede fallar en la implementación de la lógica del negocio.

Finalmente, los cambios a la lógica de la aplicación no requieren remendar con código de la interfaz, porque éstos son dos cosas totalmente diferentes, el cómo se realiza contra el cómo se presenta.

Con todos estos elementos referentes a la evolución de la arquitectura cliente-servidor tenemos un panorama claro sobre las ventajas y desventajas de utilizar dicha arquitectura en el desarrollo de la aplicación de Electropura.

Para finalizar presentamos en la ilustración I.4 un cuadro resumen de la infraestructura de la arquitectura cliente-servidor, haciendo una división por los elementos de Cliente, Red y Servidor de algunos productos representativos, en el Apéndice A se amplia a cada una de estas divisiones.

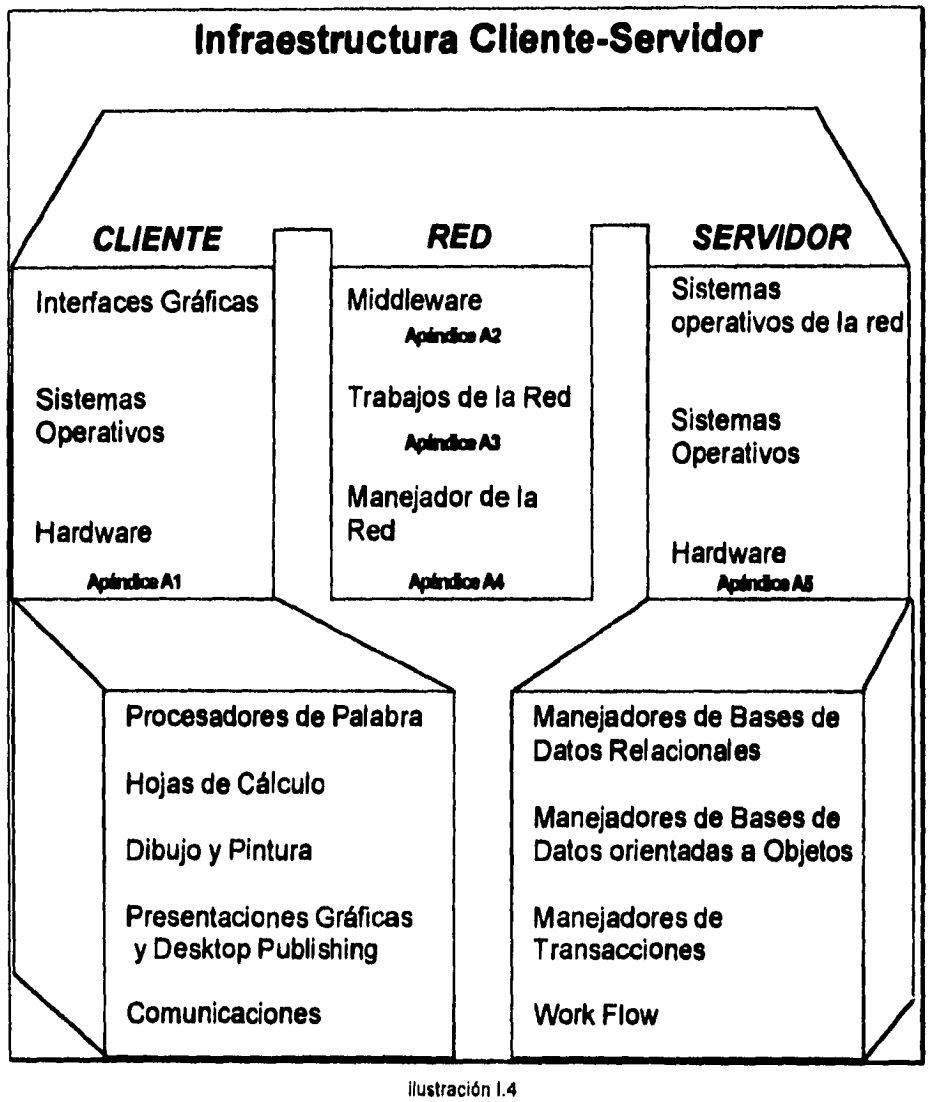


Ilustración I.4
Cuadro Resumen de la Infraestructura cliente-servidor

II COMUNICACIONES Y REDES

II.1 Transmisión de datos

En el capítulo anterior tratamos las diferentes arquitecturas de cómputo y discutimos las ventajas y desventajas de cada una de ellas, ahora revisaremos cómo las redes han influido en el desarrollo de la arquitectura cliente-servidor.

Históricamente nos encontramos con que dos tecnologías como la informática y las telecomunicaciones, que tenían muy poco en común, empiezan a tener puntos de contacto cuando, a partir de la década de los sesenta, se consigue lograr que la información de los mainframes, (que procesaban centralmente el conjunto de las aplicaciones administrativas y científicas de una organización), pudiera llegar a puntos distantes haciendo uso de las líneas de comunicación, con lo que las compañías de telecomunicaciones se enfrentaron a un problema similar al existente en el comienzo de las comunicaciones telefónicas: la conexión entre dos puntos terminales.

A partir de la década de los ochentas, la unión de la informática y las telecomunicaciones han logrado que el hombre pueda hacer mejor uso de la información.

Entendemos como transmisión de datos el movimiento de información que ha sido o va a ser procesada, codificada generalmente en forma binaria, sobre algún sistema de transmisión eléctrica. La ilustración II.1 esquematiza los elementos que constituyen un sistema de transmisión de datos entre los puntos A y B.

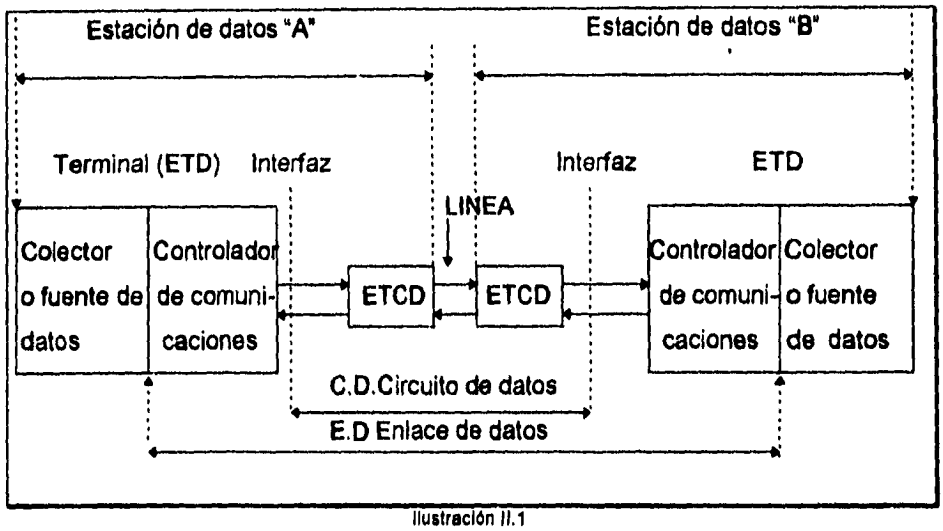


Ilustración II.1
Elementos de un sistema de transmisión de datos

Será preciso pues, la existencia de una fuente de datos, un destinatario de los mismos y un camino de unión entre ellos.

Las técnicas y medios empleados para llevar a cabo esta transmisión varían en función a la distancia.

ETD (Equipo Terminal de Datos) cubre dos funciones básicas. Ser fuente o destino final de los datos y controlar la comunicación. Este concepto engloba tanto las normalmente llamadas terminales como la computadora más compleja.

ETCD (Equipo de Terminación de Circuitos de Datos). Elemento de suma importancia en el tema que nos ocupa, cuya misión consiste en transformar las señales portadoras de la información a transmitir, utilizadas por los ETD, en otras, que, conteniendo aquella misma información más alguna adicional de uso exclusivo entre ambos ETCD, sean susceptibles de ser enviadas hasta el ETD distante, mediante los medios de comunicación clásicos.

LINEA: Conjunto de medios de transmisión que une los dos ETCD, cuya constitución dependerá de la distancia, velocidad, etc. y que debe cumplir determinadas especificaciones, apoyándose siempre en la infraestructura de comunicación.

ED: Enlace de Datos: unión entre fuente y colector de datos, formado por los controladores de comunicaciones, ETCD y LINEA.

CD: Circuito de datos: Conjunto formado por los ETCD (modems) y la LINEA, cuya misión será entregar en la interfaz con el ETD colector, las señales bajo la misma forma y con idéntica información que recibió en la interfaz con el ETD fuente.

RUIDO: Distorsión en la emisión de la información.

II.2 Estructura Física de los Circuitos para la Transmisión de Datos

La interfaz entre el Equipo Terminal de Datos (ETD) y Equipo de Terminación de Circuitos de Datos (ETCD) es la interfaz **RS-232C**, la que utiliza el intercambio de datos binarios en serie. Esta interfaz cubre tres características:

- Mecánicas.
- Eléctricas.
- Funcionales.

II.2.1 Características mecánicas (ISO-2110)

Al tratar de establecer la conexión de un equipo con otro, lo primero que nos vamos a plantear es la compatibilidad física de los conectores que usaremos. La

interfaz RS-232C define un conector de 25 patillas, (para mayor información sobre las funciones de las patillas de un RS-232C ver apéndice B) el macho está asociado al ETD (terminal) y la hembra está asociada al ETCD (equipo de comunicaciones). Típicamente se usa el conector DB-25, aunque como éste no está definido en el estándar, otros fabricantes utilizan uno diferente (el DB-9).

Lo que si define perfectamente la norma, ISO-2110, es la asignación de señales a los contactos del conector, así como la longitud máxima recomendada, determinada por la capacidad del mismo.

II.2.2 Características eléctricas

Parte fundamental, dentro de la recomendación RS-232C es la definición de las características de las señales que por éste transitan; el nivel eléctrico cubre el margen de tensiones y corriente en cada patilla, estando limitado el voltaje entre ± 3 y ±25 voltios, y la intensidad a 3 miliamperios, de tal manera que aún en el caso de cortocircuito entre patillas no se cause daño alguno a los diversos componentes.

II.2.3 Características funcionales

Son éstas las más interesantes desde el punto de vista del usuario, pues son las que debe conocer para poder realizar así su correcta aplicación.

Dentro de éstas, destacan las siguientes:

- Transferencia de datos a través de la interfaz.
- Control de las diversas señales en la interfaz.

- Proporcionar las señales de sincronización que regulan la transferencia de bits.
- Referencia de señal eléctrica.
- Descripción funcional de los circuitos de intercambio, teniendo en consideración que pueden existir diversas opciones, por lo que debemos examinarlos cuidadosamente.

  Circuitos de intercambio de datos

  TD (Transmitted Data)

  RD (Received Data)

Estos son los más importantes, pues en definitiva representan la información que deseamos transmitir.

II.3 Comunicación y Modems

"Si se desea comunicar dos computadoras en lugares geográficos distintos, se puede por medio de la línea telefónica y de los satélites, entre otros. En el caso de línea telefónica, cada computadora deberá conectarse a un modem, que traducirá señales digitales (manejadas por la computadora) a señales analógicas (transportadas por el teléfono) y viceversa (ilustración II.2). La conexión es computadora-modem-teléfono-modem-computadora. Intervienen dos interfaces RS-232C, una entre cada PC y modem." [1].

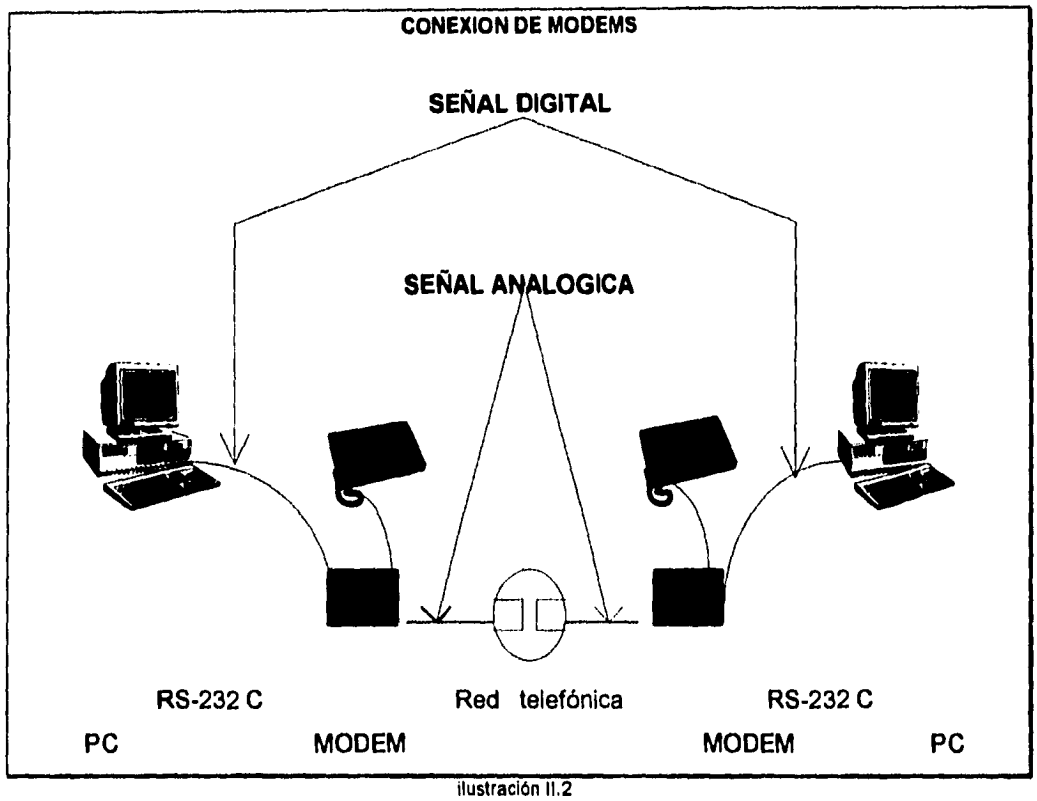


Ilustración II.2
Conexión de Modems

El cable de interfaz que une una computadora con un modem utiliza la configuración de señales denominadas uno a uno como se observa en la tabla II.1

Terminal (DB25)		Señal	Terminal (DB25)
2	———	TX	2
3	———	RX	3
7	———	GND	7
4	———	RTS	4
5	———	CTS	5
6	———	DSR	6
8	———	CD	8
20	———	DTR	20

tabla II.1
Configuración de cable para Modem

Esta configuración es diferente a la que se utiliza cuando se conectan dos computadoras sin tener un modem de por medio, ésta configuración es la llamada null modem como se observa en la tabla II.2.

Es importante tomar en cuenta lo anterior para la programación de SICE.

Terminal (DB25)		Señal		Terminal (DB25)
2	→	TX		3
3		RX	←	2
7	—	GND	—	7
4		RTS		5
5		CTS		4
8	←	CD	→	8
6		DTR	←	20
20	→	DSR		6

tabla II.2
Configuración de cable Null-Modem

El RS-232C cubre los protocolos a usar para:

- contestar la llamada recibida,
- control de datos,
- control general del modem.

Los niveles de voltaje utilizados por RS-232C no son muy convenientes, ya que no son los utilizados por los estandarizados en las tecnologías **TTL** y **MOS** y necesitan una fuente de poder adicional.

La palabra **MODEM** proviene de **Mo**dulación y **Dem**odulación, entendiendo por modulación el proceso por el cual un tren de datos entrante genera una señal analógica compatible con la línea de transmisión y por demodulación el proceso inverso, que consiste en reconstruir, a partir de la señal recibida en la línea, el tren de datos que las originó. Los dos procesos anteriormente descritos se realizan en un único equipo eléctrico que recibe este nombre.

Pueden ser externos, independientes, o residir dentro del gabinete del procesador central. Según el caso, se les llama modulares o integrados respectivamente.

Desde un punto de vista puramente técnico existen multitud de soluciones a la hora de diseñar un modem. Sin embargo, a fin de facilitar la instalación de enlaces internacionales y evitar la proliferación innecesaria y antieconómica de soluciones particulares, existen diversos organismos internacionales (EIA en USA y CCITT en Europa) que han normalizado una serie de modems que cubren prácticamente el espectro de necesidades presentes (ilustración II.3).

CARACTERISTICA	V.19	V.20	V.21	V.22	V.23	V.26	V.27	V.29	V.36
Velocidad de transmisión bits/s	<=10	<240	<=300	<=600/ 1,200	<=600/ 1,200	2,400	4,800	9,600	48,000
Tipo de transmisión	Paralelo	Asinc./ paralelo	Asin- crona.	Sincrona/ asincrona	Sincrona/ asincrona	Sin- crona	Sin- crona	Sin- crona	Sin- crona
Velocidad de modulación : bd	<=10	<=40	<=300	600	<=600/ 1,200	1,200	1,600	2,400	9,600

Ilustración II.3
Modems normalizados. Resumen de características

La velocidad de transmisión se define como el número máximo de elementos binarios (bits) que pueden transmitirse por un determinado circuito de datos durante un segundo, la unidad es el bit/s.

La velocidad de modulación fija las características de la línea de transmisión. Es el número máximo de veces por segundo que puede cambiar el estado de señalización en la línea, se utiliza como unidad el baudio, equivalente a un intervalo significativo por segundo.

II.4 Tipos y Modos de Transmisión

Puede hablarse básicamente de dos tipos de transmisión:

Sincrónicos o asincrónicos, dependiendo del tipo de mensaje a transmitir. Pueden tener diagnósticos residentes y disponer de mecanismos de detección y corrección de errores, la rapidez de reacción de los circuitos del modem, es una variable que juega en los tiempos de respuesta de las terminales remotas.

II.4.1 Tipos

II.4.1.1 Asíncrona

Los "n" bits que forman la palabra del código correspondiente, van siempre precedidos de un bit "0" llamado arranque y seguidos de al menos un bit "1" llamado de parada, que en algunos casos es de 2 bits. El conjunto citado constituye un caracter, pudiendo mediar, entre dos consecutivos cualquier separación. La transmisión asíncrona se emplea bastante, ya que las interfaces de este tipo entre los ETD y ETCD son sencillos y económicos. Como la sincronización entre el emisor y el receptor tiene lugar caracter a caracter, es admisible una cierta desviación entre las características de ambos extremos, puesto que tales diferencias pueden ser corregidas antes de que llegue el siguiente byte.

II.4.1.2 Sincrona

Los datos proporcionados por la terminal, lo son a una cadena fija y constante, marcada por una base de tiempo común para todos los elementos que

intervienen en la transmisión, conocida como reloj. En los formatos síncronos se suprimen las señales intermitentes de arranque/parada que acompañan a cada caracter. Las señales preliminares suelen llamarse bytes de sincronización, o banderas. Su misión principal consiste en alertar al receptor de la llegada de datos.

Este tipo de transmisión requiere de modems y terminales complejos, pero supone una mejor utilización de la línea y permite mayores velocidades por ser menos sensibles al ruido y además a otros efectos de los medios de transmisión.

II.4.2 Modos

En la práctica existen tres modos básicos de explotar un circuito de datos:

- *Simplex*

La transmisión se realiza solamente en un sentido, sin posibilidad de hacerlo en el opuesto.

- *Semiduplex o "Half-Duplex"*

La transmisión se realiza alternativamente en uno u otro sentido. (También llamada bidireccional alternada).

- *Duplex o "Full-Duplex"*

Consiste en transmitir simultáneamente e independientemente en ambos sentidos. Ya sea enviando datos en los dos, o bien datos en uno y control de los mismos en el otro. Este método, si bien consigue la mayor eficiencia en la utilización de las líneas, exige terminales y modems muy sofisticados, por lo que raramente se usa, salvo en la unión entre computadoras. (También llamada bidireccional simultánea).

La transmisión en modo simplex es habitual en televisión y en radiodifusión comercial. La transmisión semiduplex aparece en muchos sistemas. Un ejemplo de ellos son las aplicaciones del tipo pregunta/respuesta, en las cuales un ETD envía una pregunta a otro ETD y queda a la espera de que el proceso de aplicación obtenga la respuesta o la calcule y devuelva el resultado. El duplex permite transmitir en ambas direcciones a la vez, sin estar sometido a la estructura de parada y espera del semiduplex.

II.4.3 Líneas de Transmisión

El modo de transmisión de datos y la velocidad a que se pretenda transmitir, determina el tipo de línea a utilizar en cada caso.

- *Dos hilos*

  Se entiende por línea de dos hilos aquella en la que la unión física entre modems se realiza por un par físico. Debido al pequeño ancho de banda que ésta posee, normalmente va asociada a transmisiones semiduplex, salvo el caso de modems de baja velocidad. Se puede utilizar en circuitos urbanos.

- *Cuatro hilos*

  Esta es la constituida por dos pares de enlace, que a su vez pueden ser dedicados y conmutados. En aquellas aplicaciones que son susceptibles al ruido, los circuitos dedicados ofrecen una mejor inmunidad. Estos circuitos permiten la transmisión duplex, aunque la mayor parte de las veces se encuentran infrautilizados, bien por el desconocimiento que puede tener el usuario, o bien por las propias limitaciones de equipos.

Los circuitos de cuatro hilos, o circuitos de cuadretes, suelen conocerse como circuitos duplex. Incluyen dos pares de hilos cada uno; dos de los hilos transmiten los datos, y los otros dos cierran los correspondientes circuitos. Para las compañías telefónicas, un enlace de dos hilos suele corresponder a un circuito telefónico conmutado normal, mientras que un circuito de cuatro hilos suele ser una línea alquilada, no conmutada (ilustración II.4).

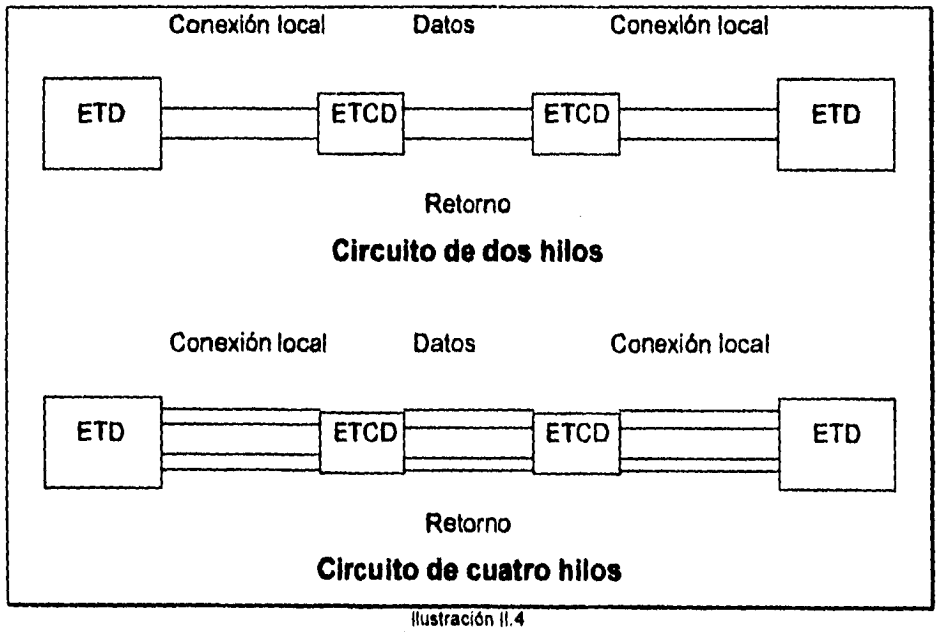


Ilustración II.4
Tipos de circuitos

II.4.4 Velocidad de Transmisión

Se entiende por tal, la capacidad del sistema para transferir información.

- Velocidad de modulación.

Es la que interesa al técnico de comunicaciones para fijar y definir las características de la línea de enlace. Se define como el número máximo de veces por segundo que cambia el estado de la línea. Su unidad de medida es el BAUDIO.

- Velocidad de transmisión serie.

Se define como el número máximo de elementos binarios (bits) que se pueden transmitir por un determinado circuito de datos durante un segundo.

- Velocidad de transferencia de datos.

Este es un concepto que tiene interés para el Analista de Sistemas, y representa la cantidad de información que puede transferirse por unidad de tiempo, bien sean bits, caracteres o bloques, pero siempre netos y de información.

II.5 Redes

En el ambiente informático, es un conjunto de computadoras interconectadas a través de uno o varios caminos o medios de transmisión, a cada computadora se le conoce como nodo. Su finalidad es transferir e intercambiar datos entre estas.

II.5.1 Funciones

Entre las funciones de red más importantes se destacan las siguientes:

- Comunicaciones entre tareas. Intercambio de datos entre dos programas, en el mismo o en diferentes nodos, a través de un camino lógico establecido entre ambos en base a consentimiento mutuo (establecimiento de sesiones).
- Acceso a recursos remotos. Tanto programas como terminales de usuario pueden accesar a archivos y/o dispositivos en nodos remotos. Entre las operaciones que se pueden realizar están las siguientes:

    - Transferencia de archivos entre dos nodos.
    - Manipulación de archivos residentes en nodos remotos.
    - Utilización compartida de periféricos conectados a cualquier otro nodo remoto de la red.
    - Envío de archivos conteniendo comandos a sistemas operativos remotos con ejecución, a su recepción, en el nodo remoto.

- Comunicación entre terminales de la red. Intercambio de mensajes entre terminales de usuario mediante una rutina de utilerías.
- Terminales remotas. Un terminal local de un nodo puede ser usado como local de un nodo remoto, estableciendo una conexión lógica entre esta terminal y el nodo remoto, ejecutando este nodo remoto los comandos que se tecleen en dicha terminal.
- Carga a través de la línea. En nodos satélites, con sistema operativo "sólo memoria", es posible cargar éstos, tanto el sistema operativo, como las aplicaciones de forma remota, a través de la línea desde otro nodo con recursos de almacenamiento propios, produciendo el arranque automático de las aplicaciones en el nodo satélite.

II.5.2 Topologías De Red

La configuración de una red suele conocerse como topología de la misma. El término "topología" es un concepto geométrico con el que se alude al aspecto de una cosa. Cuando se establece la red, el diseñador debe plantearse los siguientes tres objetivos:

- Proporcionar la máxima fiabilidad posible, para garantizar la recepción correcta de todo el tráfico.
- Encaminar el tráfico entre el ETD transmisor y el receptor a través del camino más económico dentro de la red.
- Proporcionar al usuario final un tiempo de respuesta óptimo y un caudal eficaz máximo.

Al tratar el aspecto de fiabilidad, se habla de la capacidad que tiene la red para transportar datos correctamente (sin errores) de un ETD a otro.

II.5.2.1 Topología Jerárquica

La estructura jerárquica es una de las más extendidas en la actualidad. El software que controla la red es relativamente simple, y la topología proporciona un punto de concentración de las tareas de control y de resolución de errores. En la mayoría de los casos, el ETD situado en el nivel más elevado de la jerarquía es el que controla la red. En determinadas situaciones, el ETD más elevado, normalmente una computadora central, ha de controlar todo el tráfico entre los distintos ETD. Si ese ETD principal falla, toda la red deja de funcionar, a no ser que exista otra computadora de reserva capaz de hacerse cargo de todas las funciones del ETD averiado.

Las redes con topología jerárquica también son conocidas como redes verticales o en árbol.

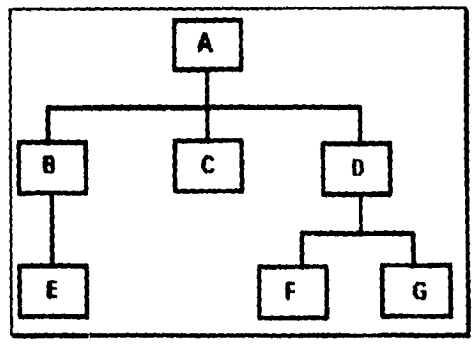


Ilustración II.5
Topología jerárquica o en árbol

II.5.2.2 Topología Horizontal (bus)

Esta estructura es frecuente en las redes de área local. Es relativamente fácil controlar el flujo de tráfico entre los distintos ETD, ya que el bus permite que todas las estaciones reciban todas las transmisiones, es decir, una estación puede difundir la información a todas las demás. La principal limitación de esta topología está en el hecho de que suele existir un solo canal de comunicaciones para todos los dispositivos de la red, por lo tanto, si el canal de comunicaciones falla, toda la red deja de funcionar. Algunos fabricantes proporcionan canales completamente redundantes por si falla el canal principal, y otros ofrecen conmutadores que permiten rodear un nodo en caso de que falle.

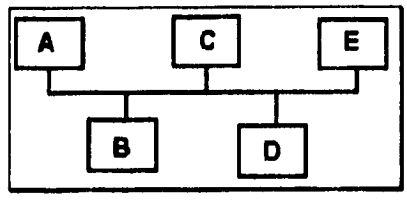


Ilustración II.6
Topología horizontal o en bus

II.5.2.3 Topología en Estrella

Es una de las más utilizadas en los sistemas de comunicaciones de datos, su software no es complicado y su flujo de tráfico es sencillo. Todo el tráfico emana del núcleo de la estrella, por lo general la computadora central, posee el control total de los ETD conectados a éste. La configuración de esta topología es una estructura muy similar a la de la topología jerárquica, aunque su capacidad de procesamiento distribuido es limitada. El nodo central es responsable de encaminar el tráfico hacia el resto de los componentes; se encarga, además, de localizar las averías. Esta tarea es relativamente sencilla en el caso de una topología en estrella, ya que es posible aislar las líneas para identificar el problema, pero al igual que en la estructura jerárquica también esta topología puede sufrir saturaciones y problemas en caso de avería del nodo central.

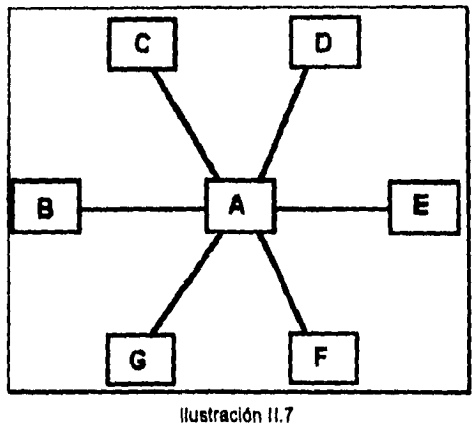


Ilustración II.7
Topología en estrella

II.5.2.4 Topología en Anillo

Se llama así por el aspecto circular del flujo de datos. En esta topología, los datos fluyen en una sola dirección, y cada estación recibe la señal y la retransmite a la siguiente del anillo. Cada componente sólo debe de llevar a cabo una serie de tareas muy sencillas: aceptar los datos, enviarlos al ETD conectado al anillo o retransmitirlos al próximo componente del mismo.

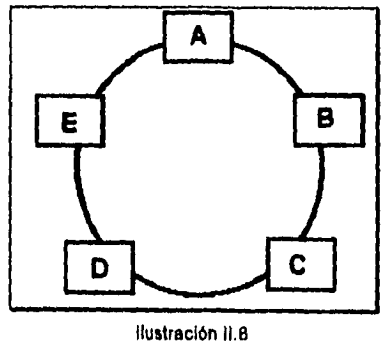


Ilustración II.8
Topología en anillo

II.5.2.5 Topología en Malla

Gracias a la multiplicidad de caminos que ofrece a través de los distintos ETD y ECD, es posible orientar el tráfico por trayectorias alternativas en caso de que algún nodo esté averiado u ocupado.

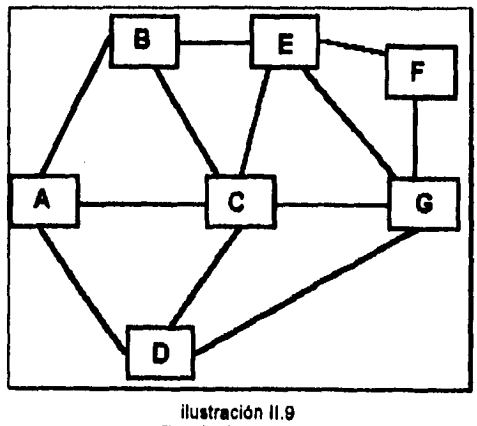


Ilustración II.9
Topología en malla

II.6 Los Niveles de OSI

El modelo de referencia de OSI[1], que se ha tomado como base de estandarización de la mayor parte de las arquitecturas comerciales de comunicaciones está basado en una estratificación en siete niveles (Tabla II.3), los cuales son:

Nivel 1, o nivel de control físico
Nivel 2, o nivel de control de enlace
Nivel 3, o nivel de control de red (encaminamiento)
Nivel 4, o nivel de control de transporte
Nivel 5, o nivel de control de sesiones
Nivel 6, o nivel de control de presentación
Nivel 7, o nivel de aplicación

Tabla II.3
Niveles de OSI

Los objetivos que persigue el modelo OSI son:

- Proporcionar una serie de normas para la comunicación entre sistemas.
- Eliminar todos los impedimentos técnicos que pudieran existir para la comunicación entre sistemas.
- Abstraer el funcionamiento interno de los sistemas individuales.
- Definir los puntos de interconexión para el intercambio de información entre los sistemas.
- Limitar el número de opciones, para incrementar las posibilidades de comunicación sin necesidad de onerosas conversiones y traducciones entre diferentes productos.

[1] Siglas de "Open Systems Interconnect". Estructura lógica y estándar de 7 niveles de protocolos definida por ISO.

El nivel más bajo del modelo es el *nivel físico*, las funciones incluidas dentro de este estrato se encargan de activar, mantener y desactivar un circuito físico entre un ETCD y un ETD siendo el estándar que se utiliza para éste, el RS-232C, el cual se ocupa para la comunicación entre circuitos en la realización del presente seminario de investigación.

El *nivel de enlace* es el responsable de la transferencia de datos por el canal. Proporciona a los datos la sincronización necesaria para delimitar el flujo de bits del nivel físico. Garantiza la identidad de los bits, encargándose de que los datos lleguen sin errores al ETD receptor. Se ocupa de controlar el flujo de datos para impedir que el ETD se desborde en ningún momento. Una de sus funciones más importantes consiste en detectar errores en la transmisión y en recuperar, por distintos mecanismos, los datos perdidos, duplicados o erróneos. En nuestro caso, si la petición del cliente hacia el servidor sufre una pérdida de información, el nivel de enlace se encarga de recuperar la misma para que llegue como lo había deseado el cliente.

El *nivel de red* define la interfaz entre el ETD de usuario y la red de conmutación de paquetes, además de la interfaz de un ETD con otro a través de esta red. Especifica las operaciones de encaminamiento por la red, y la comunicación entre distintas redes. Este nivel es el que se encarga de mantener y terminar las conexiones de la red la cual ha llevado a cabo la comunicación desde el servidor hasta el cliente. En algunos casos los sistemas son terminales de datos destinatario, en el caso de este desarrollo es el cliente quien realiza la petición del reporte, y es en este nivel donde se lleva a cabo la interfaz entre las terminales. Aquí mismo se proporciona la trayectoria de la conexión entre una pareja de entidades del nivel de transporte.

El *nivel de transporte* proporciona la interfaz entre la red de comunicación de datos y los tres niveles superiores. Es el nivel que permite al usuario elegir entre diversas opciones de calidad dentro de una misma red (es decir, dentro del nivel de

red). Está diseñado para mantener al usuario al margen de algunos de los aspectos físicos y funcionales de la red de paquetes. Este nivel sólo se encarga de que la información llegue desde el sistema fuente hasta el sistema destino, es decir desde el cliente hasta el servidor. El propósito de este nivel es el proporcionar un servicio de transferencia transparente de datos (de fin a fin) entre las entidades del nivel de sesión. Es decir que las entidades de sesión no necesitan conocer los detalles por los cuales se consigue una transferencia de datos fiable.

El *nivel de sesión* funciona como interfaz del usuario con el nivel de transporte. Ofrece un mecanismo organizado de intercambio de datos entre usuarios. Cada usuario puede seleccionar el tipo de control y de sincronización que desea de la red. El objetivo de este nivel, es organizar y sincronizar el diálogo, y gestionar la transferencia de datos entre entidades del nivel de presentación.

El *nivel de presentación* asigna una sintaxis a los datos, es decir, determina la forma de presentación de los datos según este modelo, sin preocuparse de su significado o semántica. Su principal misión es, por ejemplo, aceptar tipos de datos (caracteres, enteros, etc.) procedentes del nivel de aplicación y negociar con el nivel homólogo del otro extremo la sintaxis escogida.

El *nivel de aplicación* se encarga de atender al proceso de aplicación del usuario final. A diferencia del nivel de presentación, este nivel tiene en cuenta la semántica de los datos. Contiene varios elementos de servicio capaces de gestionar procesos de aplicación tales como la gestión de trabajos, el intercambio de datos financieros, sentencias **send/receive** de distintos lenguajes de programación, etc. Todos los niveles anteriores existen para dar soporte a este nivel y funciona de "ventana" entre los usuarios comunicantes, por ejemplo, en el desarrollo del presente seminario es la presenteción que se da entre el cliente y el servidor, es decir se plasma la información en pantalla conforme se haya hecho la petición de cliente. Claro está, la totalidad de una aplicación final no se encuentra

completamente en este nivel. Sólo una parte de ésta forma parte de este nivel y utiliza protocolos del mismo.

En la tabla II.4 se hace mención de las relaciones que se encontraron entre los niveles de OSI y SICE (Sistema Integral de Comunicación para Electropura).

NIVELES	SICE
Nivel físico	Modem
Nivel de enlace	Modem, línea telefónica
Nivel de red	Modem
Nivel de transporte	Modem, línea telefónica
Nivel de sesión	Programa
Nivel de presentación	Programa
Nivel de aplicación	Programa

Tabla II.4
Modelo OSI y SICE

Todos estos niveles se muestran en la ilustración II.10.

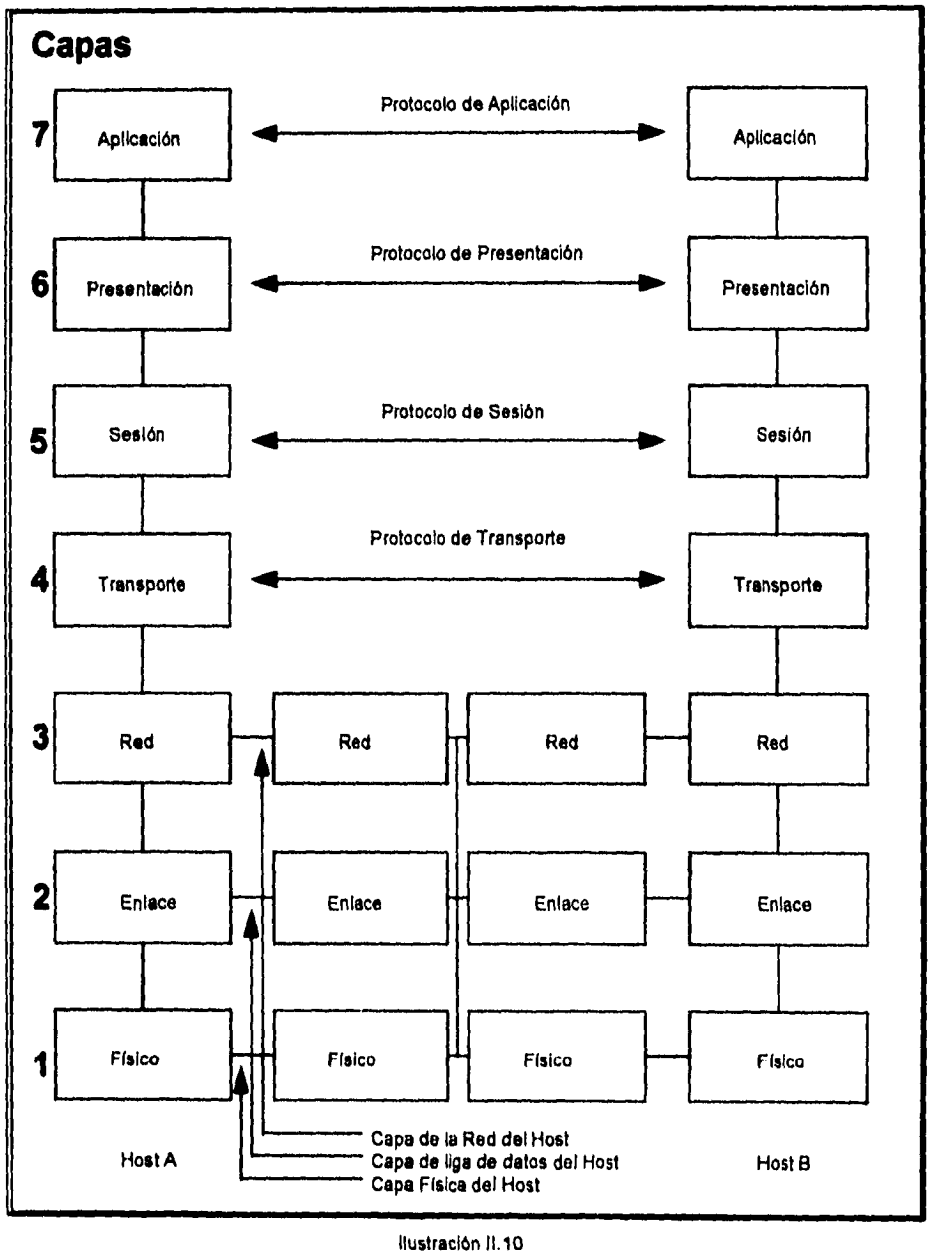


Ilustración II.10
Arquitectura de une red según el modelo OSI

Cliente-servidor en el modelo OSI representa una comunicación que se ha utilizado en diversas organizaciones y siendo Electropura una de ellas no podría dejar de usarla. En el desarrollo del presente seminario se hace la comunicación de un cliente con un servidor vía modem, además de contar con la ayuda de un programa en C++ para obtener la comunicación deseada con el fin de facilitar la petición de nuestro cliente.

II.7 Middleware: El Buen Intermediario

El *middleware* es un fenómeno computacional ubicado entre las aplicaciones, los sistemas operativos, las bases de datos y los protocolos de red.

Gracias al middleware, ni los desarrolladores de aplicaciones, ni las aplicaciones mismas reportan disparidad alguna entre los sistemas operativos de red y los protocolos. El middleware es transparente para el usuario final y está diseñado fundamentalmente para los desarrolladores de aplicaciones y los programadores de las empresas.

Son fundamentalmente dos las áreas en que se siente el impacto del middleware.

1. La primera de éstas son las redes; éste hace posible que sistemas y protocolos diferentes funcionen cordialmente.

2. La segunda de estas son las bases de datos, ya que el middleware permite que cualquier interfaz - o frontend - de éstas pueda funcionar con el backend de cualquier otra.

El middleware para redes está diseñado para hacer la vida mucho más sencilla a los desarrolladores de aplicaciones.

Supongamos que una empresa tiene tres tipos diferentes de redes que ejecutan, por ejemplo, **SNA**, **TCP/IP** e **IPX**. Si un programador fuera contratado para escribir un programa para el procesamiento de transacciones en línea, se vería obligado a escribir tres programas distintos. Pero si contara con el middleware, el desarrollador sólo tendrá que escribir un sólo programa al nivel del middleware - que da soporte a los tres protocolos - para que la aplicación se ejecute en todas las redes.

> " El middleware representa un cambio total en el paradigma de cómo se escriben las aplicaciones distribuidas " [2] - resalta Frank Dzubeck, presidente de una firma consultora llamada Communication Networks Architects, que tiene sus oficinas en Washington, D.C.

Al frente de las bases de datos, el middleware capta la atención de muchos administradores de sistemas de información, quienes podrán ahora adquirir un frontend para una base de datos como Paradox o FoxPro, sin tener que preocuparse de su conectividad con backends como Oracle, Sybase o Informix.

El middleware para bases de datos está diseñado para que las interfaces cliente de todas las bases de datos operen junto con cualquier base backend, siempre y cuando ambos se conformen y den soporte a los mismos niveles al middleware. Sin éste, las interfaces de cliente sólo pueden conectarse a bases de datos backend en caso de que se haya escrito un conjunto de manejadores para cada programa.

II.7.1 Productos de Middleware

Los dos productos middleware más importantes que se están desarrollando actualmente son ODBC (Open Database Connectivity) de Microsoft e IDAPI, que desarrolla un consorcio integrado por Novell, IBM, WordPerfect y Borland. Ambos productos se ajustan a los estándares desarrollados por ANSI y X/Open.

Estos son los productos que tienen más auge en el mercado para usuarios finales en plataforma Windows.

Tanto ODBC como IDAPI suministran una interfaz para tener acceso a datos en un ambiente heterogéneo de sistemas para el manejo de bases de datos relacionales.

La manera más adecuada de describir ODBC e IDAPI consiste en compararlos con los manejadores para impresoras que se desarrollaron hace unos años. Actualmente, cuando un usuario carga un sistema operativo, simplemente escoge el tipo de impresora a la que el sistema operativo da soporte, de entre las que vienen en una lista. Una vez hecho esto, cuando un usuario necesita imprimir un documento - sin importar la aplicación que esté utilizando -, el manejador de la impresora conecta automáticamente la aplicación y la impresora, y el documento se puede imprimir.

IDAPI y ODBC funcionan más o menos de la misma manera. Cuando un usuario carga una base de datos frontal, se le proporciona una lista de las bases de datos de fondo a las que se les da soporte. Sin embargo, son los desarrolladores de aplicaciones quienes reconocen el verdadero valor de las API de middleware por que en vez de escribir diferentes conjuntos de código para cada base de datos a la que se va a dar soporte, basta con que el desarrollador escriba un conjunto de

código - para ODBC o IDAPI - y la base de datos de fondo se conectará de manera automática con la frontal, y viceversa.

Una diferencia importante entre ODBC e IDAPI es que ésta última se conecta tanto con bases de datos navegacionales como de conjunto, mientras que ODBC trabaja únicamente con las de conjunto. Las bases de datos de conjunto se diferencian de las navegacionales en la forma como el usuario busca la información. En una base de datos de conjunto como es ACCESS, los usuarios elaboran una lista de detalles y hechos que hay que localizar al momento de consultar una base en busca de un elemento. En una base de datos navegacional como PARADOX, el usuario nada más tiene acceso a una base y "le hecha un vistazo" a los datos hasta que encuentra el elemento.

Otra diferencia entre los dos "estándares" de middleware es que IDAPI permite que los usuarios consulten dos bases de datos al mismo tiempo, mientras que los usuarios de ODBC nada más pueden consultar una.

> "Básicamente, ODBC es un conjunto de manejadores para bases de datos, pero IDAPI realmente es un programa middleware porque toma en consideración todos los tipos de bases de datos. Tanto IDAPI como ODBC representan un cambio fundamental en la conexión de bases de datos, pero IDAPI ha ido más allá porque da soporte a bases de datos navegacionales y de conjunto, cosa que no sucede con ODBC". [2]

Otro producto, OMNIS 7(de Blyth Software), puede comunicarse con tantas bases de datos como la memoria de la computadora lo permita. Por medio de módulos conocidos como DAM (Data Access Module) puede crear la interfaz con cada base de datos, siendo transparente para el desarrollador o para el usuario utilizar una base de datos u otra. El DAM tiene la función de ser un middleware.

Como puede apreciarse las redes son parte fundamental en el desarrollo de aplicaciones cliente-servidor, y como va avanzando el desarrollo de las mismas la interconectividad entre backend y frontend diferentes será mucho más sencilla y efectiva, tanto para usuarios finales como para desarrolladores.

III PARADIGMAS DE SOFTWARE

III.1 Evolución de la Ingeniería de Software

En este capítulo se trata cómo ha evolucionado la ingeniería de software a lo largo del tiempo. Consideramos que es importante conocer cómo se ha venido analizando, diseñando y desarrollando con las diferentes metodologías que han existido para poder revisar la metodología basada en objetos y así dar fundamentos de por qué se utiliza en el desarrollo de esta aplicación.

La Ingeniería de Software surge como respuesta a los problemas de la construcción y el mantenimiento de sistemas de información.

Durante la primera época de la era de la informática[1], se desarrollaron fundamentalmente los aspectos físicos, el equipo o hardware. Pero en la actualidad, el reto mayor en la informática es la reducción de costos y mejoramiento de los procedimientos de construcción de software.

La velocidad a la que avanza la tecnología de hardware de computadoras es sorprendente; año con año se logran avances que conducen a la construcción de computadoras más veloces, más compactas y más baratas. Sin embargo, en el aspecto de software no parece haber un desarrollo similar. Mientras los costos del hardware han disminuido continuamente, los de software han hecho lo contrario. La construcción de software no es una tarea fácil y en muchas ocasiones los proyectos de programación sobregiran los presupuestos de tiempo y dinero.

[1] La aplicación de las computadoras al procesamiento de la información comenzó en 1954 cuando uno de los primeros computadores se programó para procesar nóminas.[4]

Cabe hacer notar que la planificación de la realización de un software no se puede llevar a cabo de la misma forma como se planifica la construcción de un bien inmueble.

¿Por qué?. La complejidad del software no se ha comprendido al mismo grado que un ingeniero civil experto en puentes y caminos entiende la complejidad de la construcción de una carretera.

Es necesario adaptar las técnicas existentes de otras disciplinas al contexto específico de la realización de software.

Los tiempos de entrega del software son siempre muy inciertos y largos. La razón fundamental de estos problemas es la complejidad de éstos que se quieren resolver, y las técnicas de fabricación del software mal adaptadas a ésta.

Es necesario adaptar las fases de construcción de un software:

- Fase de planificación,
- Definición del sistema,
- Análisis de requerimientos,
- Fase de desarrollo y
- Fase de implantación

a la complejidad de los sistemas que se quieren construir.

La construcción de software ha recibido la atención de los expertos desde hace mucho tiempo; en la década de los 70's se consiguieron avances significativos hacia el desarrollo de metodologías o paradigmas[2] para construir programas en forma sistemática y a bajo costo. Como resultado de esos esfuerzos surgieron técnicas, como el diseño estructurado y el desarrollo descendente (top-down), que

[2] Paradigma. Según el diccionario es una palabra tipo que se da como modelo para una declinación, una conjugación. Es un ejemplo o modelo a seguir.

durante mucho tiempo han sido las herramientas utilizadas por los programadores para construir software. Aunque dichas técnicas han sido empleadas durante mucho tiempo en proyectos complejos, la mayoría de los ingenieros de software coinciden en afirmar que sufren de tres grandes deficiencias:

- Los productos que resultan son pocos flexibles.
- Los programadores que las usan tienden a concentrarse en el diseño y la implementación inicial del sistema, sin tomar en cuenta su vida posterior.
- No alientan al programador a aprovechar el trabajo de proyectos anteriores.

La desventaja de utilizar una metodología que se concentra en el diseño inicial del sistema se hace evidente si se toma en cuenta que la vida útil de un producto de software puede ser cuatro o cinco veces más grande que el lapso en que se desarrolla; por ejemplo, un sistema que se desarrolla en uno o dos años puede mantenerse trabajando durante un período que va de cuatro a diez años. Los gastos que se hacen durante este último período (gastos de mantenimiento) representan alrededor del 70% del costo total del sistema.

III.2 Fases de Construcción de Software

Los paradigmas actuales de construcción de software tienen tres grandes fases:

1a fase: definición. Esta fase se enfoca en el "qué".

- ¿Qué tipo de información se procesará?
- ¿Qué tipo de información y desempeño se desean obtener?
- ¿Qué tipo de interfaz se quiere establecer con el sistema?

- ¿Qué tipo de restricciones de diseño existen?

2a fase: desarrollo. Esta fase se centra en el "cómo".

- ¿Cómo se diseñarán la estructura de datos y la arquitectura del software?
- ¿Cómo se implantarán los procedimientos o algoritmos?
- ¿Cómo se traducirá el diseño a lenguaje de máquina?
- ¿Cómo se realizarán las pruebas y el mantenimiento del producto?

3a fase: mantenimiento. Esta fase se centra en la administración de los cambios.

- ¿Será posible realizar, a un costo razonable, las modificaciones correctivas?. Estas son las modificaciones debidas a errores de funcionamiento.

- ¿Será posible realizar, a un costo razonable, las modificaciones evolutivas?. Estas son las modificaciones debidas a mejoras dentro de algún componente del sistema.

Estas tres fases las encontramos en varios paradigmas. Podemos profundizar un poco cada una de éstas dando algunos elementos del ciclo de vida de un software.

III.3 Ciclo de Vida de un Software (modelo clásico)

1a etapa: Análisis de requerimientos

Comprende el dominio de información del problema. Es necesario establecer una representación del flujo de información y de su estructura, la cual debe ser lo más cercana al dominio del problema. Requiere de una comunicación intensa entre

los usuarios y desarrolladores. Esta es una etapa de descubrimiento y refinamiento. Se tienen diversas actividades: reconocimiento del problema, evaluación y síntesis, especificación; revisión.

El análisis de un sistema es una de las partes fundamentales para la construcción del mismo, porque es la etapa donde se realiza un estudio de la complejidad de éste y se detectan las grandes partes del mundo real.

2a etapa: Diseño

En esta etapa los requerimientos del software se traducen a un conjunto de representaciones que describen la estructura de datos, arquitectura y procedimientos del sistema. El objetivo es construir una representación o modelo de una entidad, que será desarrollada posteriormente, y que debe reflejar completamente los requerimientos del sistema.

La definición o diseño es la primera fase en el desarrollo de cualquier producto o sistema. En ésta es donde se establece la calidad del desarrollo del sistema. Determina en gran parte la **robustez** y características del sistema.

3a etapa: Codificación

El diseño debe codificarse en un lenguaje de programación y traducirse a lenguaje de máquina.

La selección del lenguaje de programación para llevar a cabo la codificación es una tarea delicada. ¿Cuál es el lenguaje que mejor se adapta al diseño? y en general ¿Cuál es el lenguaje que mejor soporta el paradigma que se está utilizando en el desarrollo del sistema?

Es conocido que un buen diseño es independiente del lenguaje de implantación y que todo diseño correctamente realizado podría ser codificado en lenguaje ensamblador. Pero la evolución de los lenguajes siempre ha ido adelante de las metodologías de desarrollo. Por lo que siempre podremos seleccionar un lenguaje de programación que soporte ampliamente el paradigma.

Por ejemplo, notemos que C++ no surgió como un proyecto para dar soporte al paradigma basado en objetos, sino como un proyecto de incluir en el lenguaje C ciertas características que le permitieran al autor, Bjarne Stroustrup, modelar ciertos fenómenos con eventos.

4a etapa: Prueba

La verificación o prueba del software principia una vez que se terminó la codificación. Se plantea la necesidad de poder verificar un software por componentes, a medida que éstos se van terminando, sin necesidad de tener que esperar la codificación de todo el producto.

La prueba del software es un elemento crítico para la garantía de calidad. Sin embargo, es conocido que una prueba sólo demuestra la existencia de alguna falla, pero no indica que el sistema no les tenga. Es decir, una prueba nunca demuestra que el sistema sea totalmente correcto.

5a etapa: Mantenimiento

Se plantea la necesidad de realizar mantenimiento al software. Los diferentes tipos de mantenimiento: correctivo y evolutivo.

El mantenimiento correctivo es el proceso que incluye el diagnóstico y la corrección de uno o más errores.

El mantenimiento evolutivo es una actividad que modifica al software para que interactúe adecuadamente con su entorno cambiante. Esto se realiza comúnmente cuando cambia el hardware en el cual fue diseñado el sistema. También se realiza este tipo de mantenimiento cuando por sugerencias de los usuarios el sistema debe adaptarse a sus necesidades mejor comprendidas por el uso del sistema.

III.4 Paradigmas de la Ingeniería de Software

Existen varios paradigmas para la realización de software. En todos la idea es la descomposición de un problema para una mejor comprensión de la complejidad.

Un paradigma para la realización de software se inserta en la gran problemática de cómo el hombre emplea constantemente tres métodos de organización para comprender el mundo real:

- Las diferencias de los diversos objetos y sus atributos: una planta, el tamaño, el color.
- Las diferencias entre los objetos y sus componentes: una planta, las hojas, el tallo.
- La clasificación en diferentes clases de objetos: las diferentes clases de plantas.

En la informática, un paradigma trata de ofrecer instrumentos para comprender la complejidad de un sistema de información.

III.4.1 Programación en Escala Reducida

Los sistemas de cómputo inicialmente fueron desarrollados por una sola persona o un grupo reducido, 2 o 3 personas. Este tipo de sistemas generalmente

tiene un objetivo limitado y un periodo de vida corto. Este tipo de programación se denomina programación en escala reducida.

Con el avance del hardware se pudieron generar sistemas para grandes proyectos de software. Este tipo de software se caracteriza por su gran complejidad que hace imposible se realice por una sola persona o grupo pequeño, como al principio de la informática.

III.4.2 Programación en Escala Mayor

La programación en escala mayor tiene una complejidad inherente y esencial, ésta puede dominarse pero no evitarse. Es necesario siempre tratar de descomponer un problema en varias partes.

III.4.2.1 La Descomposición de un Problema

El enfoque de la descomposición es tradicional en todas las disciplinas. En la informática lo proponen varios autores entre otros Dijkstra, Parnas, etc.

La idea de la descomposición es que al diseñar un sistema complejo de software es esencial descomponerlo en partes más pequeñas cada vez, cada una de las cuales podrá descomponerse a su vez hasta poder tener componentes que sean accesibles a la técnica de desarrollo que se tenga.

III.4.2.1.1 Descomposición Algorítmica

El primer tipo de descomposición que se practicó es la descomposición algorítmica. Cada módulo en el sistema denota un paso significativo en el proceso general. Los elementos funcionales del sistema se asocian a módulos. Es el paradigma del diseño estructurado descendente (top-down), o de Yourdon. En este enfoque los equipos de desarrollo principian sus proyectos utilizando diagramas de flujo de datos para desarrollar una descomposición funcional.

Al principio se tiene un paradigma por descomposición funcional junto con un análisis del flujo de datos (data flow analysis). Estas son dos de las herramientas que se utilizaron principalmente en los años setenta y principios de los ochenta.

III.4.2.1.2 Descomposición Funcional

La descomposición funcional es la división sucesiva de un sistema complejo en componentes más simples y de fácil manejo. Es decir, un sistema está definido en términos de las funciones de alto nivel que realiza. Cada una de estas funciones de alto nivel, a su vez puede ser definida en términos de otras funciones de más bajo nivel y así sucesivamente hasta llegar a un nivel de funciones que el sistema operativo o la base de datos ofrezca.

En la descomposición funcional se tienen agrupadas un conjunto de funciones en un módulo, las cuales se asocian para construir subsistemas, y éstas formarán el sistema. Es decir, que un sistema está compuesto por subsistemas, los cuales están formados por módulos, que son agrupamientos de funciones. El análisis del flujo de datos considera cómo los datos fluyen de un lugar hacia otro dentro de éste.

III.4.2.1.3 Descomposición de las Entidades de Datos

También conocido como el paradigma basado por los datos o de Jackson. En este enfoque los equipos de desarrollo principian enfatizando la información que el sistema necesita para realizar su trabajo y después construyendo un modelo de información. Este modelo en ocasiones se llama Entidad-Relación.

III.4.2.1.4 Descomposición por Objetos

En éste se descompone un sistema de acuerdo a las abstracciones fundamentales del dominio del problema y se piensan los sistemas como un conjunto de agentes autónomos cada uno actuando por si mismo, que colaboran para efectuar alguna tarea conjunta. Cada agente tiene su propio comportamiento, y cada uno de estos agentes modela alguna entidad del problema real.

III.4.3 Complejidad de un Sistema

La complejidad de un sistema en escala mayor resulta de varias causas: de la complejidad del dominio del problema; la dificultad de llevar a cabo el proceso de desarrollo; el problema de modelar sistemas continuos del dominio del problema por medio de sistemas discretos del dominio de la solución; el problema de la caracterización del comportamiento de modelos discretos, los cuales a pequeños cambios no tienen forzosamente pequeñas transformaciones como los sistemas continuos.

La complejidad del dominio del problema también concierne el problema de comunicación entre el universo de los usuarios y el de los desarrolladores. Cada uno de estos universos tiene su propio lenguaje de expresión.

Por una parte los usuarios, en ocasiones, encuentran dificultades para expresar sus necesidades en forma tal, que sean precisas y comprensibles para los desarrolladores. Muchas veces porque sólo tienen una vaga idea de lo que requieren, o porque los términos que expresan son incomprensibles para los desarrolladores que no conocen el dominio del problema. Es decir, entre usuarios y desarrolladores existen diferentes perspectivas sobre la naturaleza del problema y cada uno maneja un lenguaje diferente.

Por la otra, los requerimientos de un sistema cambian conforme se va avanzando en su análisis y desarrollo. Por lo que es necesario poder construir estos grandes sistemas en forma que los cambios, evolución y mantenimiento se puedan realizar fácilmente.

La complejidad del problema también se relaciona con la dificultad de llevar a cabo el desarrollo del sistema. En este aspecto se tiene un avance importante con el desarrollo de herramientas. Este es el tema que se denomina **CASE**.

III.4.3.1 El Modelo de Yourdon o del Flujo de Datos

El paradigma denominado Modelo de Yourdon dominó durante varios años las escuelas de informática (y aún sigue vigente). Es un modelo basado en el flujo de datos, el cual tiene sus orígenes en la modularidad, el diseño descendente, la programación estructurada. El enfoque del diseño basado en el flujo de datos extendió estas técnicas e integró el flujo de la información en el proceso del diseño y así poder derivar la estructura del sistema. Entre los principales contribuyentes tenemos los nombres de los clásicos: Constantine, De Marco, Gane, Myers, Sarsons, Stevens, Yourdon.

Una herramienta surgida de esta corriente es la técnica DFD (Data Flow Diagram) llamada Diagramación de Flujo de Datos. Esta es una técnica gráfica que

describe el flujo de información y las transformaciones que se aplican a los datos, según su movimiento desde la entrada al sistema hasta su salida.

La simbología básica de un DFD consiste de rectángulos, círculos, flechas, líneas dobles y etiquetas o títulos sobre los elementos. Tenemos los siguientes símbolos (ilustración III.1):

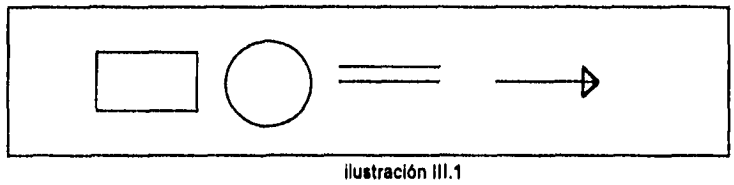


Ilustración III.1
Simbología utilizada para el paradigma de Yourdon.

a) Un rectángulo nos indica una entidad externa. Una fuente de entradas al sistema o fuente de salidas del sistema. Un rectángulo representa una entidad externa al software en construcción u otro sistema que produce información la cual se transformará o que será recibida para su uso.

b) Un círculo denota un proceso. Ejecuta alguna transformación de sus datos de entrada produciendo sus datos de salida. Un círculo representa un proceso o transformación que se aplica a los datos y que los modifica de alguna manera.

c) Una flecha nos indica el flujo de datos. Se usa para conectar los procesos entre sí a las fuentes u orígenes. La cabeza de las flechas indican la dirección de la transferencia de datos. Una flecha representa uno o más elementos de datos. Todas las flechas deben tener etiqueta que las identifique.

d) Una línea doble denota un almacén de datos. Es un depósito de datos. La cabeza de las flechas indican las entradas y las salidas del almacén de datos. La doble línea representa un almacenamiento de información generada por el sistema.

El método basado en el flujo de datos tiene un gran soporte por herramientas CASE.

> "Una herramienta CASE emplea representaciones gráficas en la pantalla para ayudar a automatizar la planeación, análisis, diseño y generación de software." [5]

Cabe comentar que el paradigma de Yourdon es una contribución interesante que ha sido retomada en parte por el paradigma basado en objetos.

III.4.3.2 El Modelo de Jackson o Basado en los Datos

El desarrollo de sistemas evolucionó surgiendo otro enfoque donde lo más importante son las estructuras de datos y no los procedimientos o el flujo de datos como podemos resumir el modelo de Yourdon:

El modelo JSD (Jackson System Development) [6] se desarrolla en los siguientes pasos:

1. Definición de las acciones y entidades. Se identifican las entidades (personas, objetos, organizaciones, sistemas) y acciones. Las entidades son quienes necesitan un sistema para producir o usar información. Las acciones son los sucesos que ocurren en el mundo real que afectan a las entidades. Las entidades se seleccionan examinando todos los nombres de la descripción del sistema. Las acciones se seleccionan examinando todos los verbos de la descripción.

2. Estructuración de las entidades. Las acciones que afectan a cada entidad son ordenadas en el tiempo y representadas mediante diagramas llamados diagramas de Jackson. Esta es una notación o diagramación para expresar relaciones entre los elementos del modelo. La estructura de una entidad describe la historia de la entidad considerando el impacto de las acciones en el tiempo. Las acciones se aplican a una entidad como secuencia, como parte de una selección o como una iteración.

3. Modelo inicial. Se construye una especificación del sistema como un modelo del mundo real. Las entidades y acciones se representan como un modelo del proceso; se definen las conexiones entre el modelo y el mundo real.

4. Definición de las funciones. Se especifican las funciones que corresponden a las acciones definidas.

5. Temporización del sistema. Se establecen y especifican las características de planificación del proceso.

6. Implantación. Se especifica el hardware y software como un diseño.

El paradigma de Jackson, también es una contribución interesante que ha sido retomada en parte por el paradigma basado en objetos.

III.4.3.3 El Paradigma Basado en Objetos (P.B.O.)

En esta parte se dará una introducción al paradigma basado en objetos, sus fundamentos y conceptos, métodos, herramientas.

> "Las técnicas del P.B.O. simplifican el diseño de sistemas complejos".[5]

El enfoque Basado en Objetos es un conjunto de principios, métodos y herramientas los cuales pueden ser instrumentos para la producción de software de mejor calidad. Es un estilo que permite una mejor organización y modularidad de grandes programas de aplicación.

Este enfoque es una evolución de los paradigmas de Yourdon y de Jackson o paradigmas basado en procesos y datos, respectivamente. El P.B.O. se basa en ambos elementos, es decir se basa en los datos y procesos que pueden ocurrir sobre esos datos. Se introduce una entidad que envuelve a los datos y procesos, el OBJETO.

Notemos que el término Object Oriented Design puede también traducirse al español como Diseño Basado en los Objetos y no solamente como Diseño Orientado a Objetos. Sin embargo, pensamos que Orientado a Objetos no refleja correctamente su semántica. ¿Por qué? Cuando un padre de familia reflexiona sobre la educación de sus hijos piensa: "Esta educación que ahora reciben está orientada a formar hombres del futuro, ... y está basada en una disciplina, rigor y libertad". Es decir, que construiremos hombres del futuro basándonos en disciplina, rigor y libertad. En informática, Object Oriented, nos indica que basándonos en objetos construiremos sistemas. Por esto en adelante utilizaremos el término basado en objetos.

La idea fundamental es que para la construcción de un sistema informático, pensemos en la misma forma como construimos los sistemas que nos rodean en la vida diaria. Así por ejemplo, supongamos que a algún ingeniero se le pide construir una casa, y empieza a pensar en los pisos, recámaras, baños, jardín; piensa en los componentes de la casa, no en su funcionalidad, no piensa que al llegar a la casa, el usuario abrirá la puerta caminará por la sala y llegará a la cocina, es decir por las funciones del sistema. Lo ideal es que piense en ambos, componentes y funcionalidad.

En la construcción de un software, un profesional del área de sistemas debería pensar de la misma forma. Cuando se le pide la construcción de un sistema de información debería pensar inicialmente en términos de los componentes. Por ejemplo, de ventas, ratones, bases de datos, etc., y después desarrollar cada uno de esos componentes por separado. Al unir todos esos componentes el sistema deberá realizar las funciones requeridas.

Actualmente, la mayoría de los estudiantes en informática han crecido en ambientes donde piensan inicialmente para la realización de una aplicación, siempre de manera funcional. Es decir, sus diseños siempre están guiados por la funcionalidad del sistema sin tomar en cuenta los componentes de dicho sistema. Cambiar esta forma en el diseño, es lo que se propone la metodología basada en objetos.

Pero, ¿qué es un objeto?. Para poder definirlo primero hay que entender el concepto de abstracción de datos.

> "El proceso de representar entidades reales con elementos internos a un programa recibe el nombre de abstracción de datos." [7]

Una abstracción surge del reconocimiento de las similitudes entre ciertas situaciones o procesos, y del concentrarse en estas similitudes ignorando las diferencias. Una abstracción denota las características esenciales de una entidad que las distingue de las demás entidades, por ejemplo, un empleado es diferente a la entidad llamada empresa.

La abstracción es el principio de ignorar aquellos aspectos de un tema (o de una entidad) que no son relevantes para el propósito actual con el objetivo de concentrarse en aquellos que si lo son.

Un objeto es la representación de una entidad de la vida real, es el resultado del proceso de abstracción. A grandes rasgos, es una variable consistente de almacenamiento (una estructura) y de todos los procedimientos o rutinas relacionados con éste, los cuales determinan el comportamiento del objeto.

"Los objetos son entidades que existen en el tiempo." [8]

El enfoque basado en objetos tiene un similar en la matemática. La idea de construcción de entidades por sus propiedades y no por su representación es algo tradicional en matemáticas y sobre todo en el álgebra. Estas entidades matemáticas son por ejemplo: las matrices, los cuerpos geométricos, los números reales, los números complejos. En todos estos sistemas la herencia y el polimorfismo son dos conceptos que se entienden perfectamente.

III.4.3.3.1 Elementos del P.B.O.

El modelo basado en objetos es un medio por el cual podemos manejar la complejidad inherente en el software. Los elementos fundamentales del modelo basado en objetos son: abstracción, encapsulamiento, modularidad, identidad y jerarquía.

En el diseño basado en objetos la identificación de las clases y objetos son las actividades clave. Esto es un proceso de abstracción.

Cuando en nuestra vida real utilizamos la abstracción admitimos que el fenómeno que tenemos en frente es complejo, y en vez de tratar de comprender todo el fenómeno en su totalidad, seleccionamos algunos aspectos de éste.

En informática sucede algo similar. La abstracción se ha utilizado en los procedimientos y los datos.

a) *La abstracción de procedimientos* es una forma de abstracción que se utiliza en forma extensiva en el análisis de requerimientos, en el diseño y programación como función-subfunción.

b) *La abstracción de datos* es una forma de abstracción que define atributos y define servicios que manipulan estos atributos. Este es el principio de definir un tipo de datos en términos de las operaciones que se aplican a estas entidades del tipo, con la restricción que los valores de tales entidades puedan ser modificadas y observadas solamente por el uso de las operaciones.

c) *Los tipos de datos abstracto*, son tipos de datos que describen un conjunto de objetos con la misma representación. Un tipo de datos abstracto extiende la noción de tipo de datos e impone la condición de esconder la implantación de las operaciones (mensajes) asociadas con el tipo de datos.

e) *Encapsulamiento u ocultación* de información. Este es el principio, utilizado cuando se desarrolla una estructura de un programa, en el cual cada componente debe encapsular o englobar una sola decisión o concepto de diseño. La interfaz de cada módulo está definida de tal forma que, no deba revelar los detalles de su funcionamiento interno a aquellos otros elementos del sistema que lo utilicen.

f) *La modularidad* permite descomponer un sistema en un conjunto de elementos interrelacionados con poca relación con otros conjuntos o módulos.

g) *La herencia* permite compartir código entre varios módulos, esto permite la reutilización de componentes. Comparte la estructura entre varios objetos. Es un mecanismo para expresar similitudes entre clases, simplificando la definición de clases similares a algunas ya definidas.

h) *La identidad* es la propiedad de un objeto que le permite distinguirse de todos los demás.

i) *El polimorfismo* es la propiedad de definir en entidades diferentes un mismo nombre de procedimiento u operación. El caso mas usual en matemáticas de polimorfismo es el operador + (suma) que en entidades diferentes (por ejemplo, números reales y números complejos) se denota por el mismo símbolo.

En un enfoque basado en objetos, las entidades fundamentales son los datos y las operaciones que se pueden hacer sobre éstos y no sólo los procedimientos o subrutinas o los datos como en el enfoque tradicional.

En este enfoque los objetos se comunican con otros objetos por medio del envío de mensajes entre ellos. Las colecciones de objetos que responden al mismo tipo de mensajes se implantan por clases.

III.4.3.3.1.1 Clase

Una clase describe e implanta todos los métodos que encapsulan o capturan el comportamiento de sus instancias. Los detalles de la implantación son totalmente internos y se encuentran escondidos o encapsulados dentro de la clase. Por lo que, las implantaciones pueden extenderse y modificarse sin afectar a los usuarios de la clase.

Cada clase tiene un esqueleto que describe el estado o estructura de sus instancias.

Los sistemas informáticos realizan ciertas acciones en ciertos objetos, y para lograr sistemas flexibles y reutilizables, es más conveniente basar la estructura del software en los objetos y no las acciones.

La idea con la programación por objetos es realizar un modelo del mundo real más cercano de su implantación en un sistema informático.

La metodología basada en objetos modela un sistema en términos de abstracciones de objetos.

Entre las muchas características de un objeto tenemos:

- Un objeto es una abstracción de alguna cosa dentro del sistema.
- Un objeto tiene ciertos atributos y ofrece ciertos servicios.
- Un objeto puede repetirse cualquier número de veces dentro del sistema.
- Un objeto puede heredar atributos y servicios de otros objetos dentro de la jerarquía.

La gran fuerza del paradigma basado en objetos es su característica de clasificación y herencia. Cuando un objeto específico es derivado de otro objeto más general, el nuevo objeto hereda todos los atributos y servicios de su padre, y solamente se requiere proveer los atributos y servicios específicos a ese objeto.

Si se aplica correctamente el paradigma basado en objetos, es factible construir aplicaciones reutilizando componentes. Estos componentes se encuentran en general reunidos en bibliotecas de objetos, o en bibliotecas de clases.

Un programa puede modificar el comportamiento de un objeto y definir otro sin tener que cambiar el código fuente del objeto original.

Un ambiente de ventanas, como Windows en el mundo **DOS** o **XWindows** en **UNIX**, es ideal para la aplicación del paradigma basado en objetos.

III.4.3.3.2 Fundamentos del P.B.O.

¿Por qué elegir el P.B.O.?

Durante mucho tiempo se han tenido que desarrollar sistemas teniendo en mente cómo "piensa la computadora", es decir, tratamos de resolver el problema desde el punto de vista que lo ve la computadora, siendo que los seres humanos tenemos una manera natural de organizar nuestro conocimiento acerca del mundo. ¡Esto debe aplicarse a la construcción del software!.

El P.B.O. permite construir mejores sistemas y hace el proceso más fácil. Mejora la comunicación con el usuario.

> "Los modelos construidos durante un análisis basado en objetos proveen una manera más natural de pensar acerca de los sistemas."[5]

> "El verdadero impacto de la programación basada en objetos está en el cambio de perspectiva del desarrollo de sistemas, que pasa de ser una actividad basada en una boutique personalizada de subrutinas a ser una industria de componentes reutilizables." [9].

Anteriormente mencionamos la diferencia entre el avance del hardware y del software, el problema de los tiempos de entrega y la dificultad del mantenimiento del software[3].

El costo de mantenimiento y la poca flexibilidad del software para hacerlo son factores importantes para tomar en cuenta la opción que ofrece el P.B.O., esta opción es la reutilización de componentes.

En la tecnología de construcción de hardware se tiene la posibilidad de intercambiar componentes. Este intercambio puede ser motivado por un fallo del componente o por un nuevo componente de mejor funcionamiento. La idea de la fabricación de software por componentes y su reutilización es similar. Si se diseña y

[3] "No es lógico que se inviertan tres pesos para la fase de análisis y diseño del sistema y siete pesos para darle mantenimiento" [12]

construye un software por componentes intercambiables se pueden lograr productos que puedan evolucionar fácilmente con el tiempo, hacia productos más fiables y robustos, productos de calidad total.

Así el componente se construye una sola vez, y es guardado en alguna biblioteca y cuando se requiere su uso posterior solamente se selecciona.

III.4.3.3.3 Malas Noticias

Introducir tecnología basada en objetos puede presentar problemas, y algunos proyectos pueden fallar si el paradigma no se aplica correctamente.

Para usar bien la tecnología de objetos se necesita mucho cuidado en el entrenamiento. Toma tiempo a los profesionales de informática pensar en términos de encapsulación, herencia, y los diagramas de análisis y diseño de objetos.

El buen uso de herencia y clases reusables requiere de cambios culturales y organizacionales. Las bibliotecas de clases deben estar bien administradas. En muchas organizaciones, construir la biblioteca de clases que tenga un alto nivel de reusabilidad puede tomar mucho tiempo.

Los beneficios de adoptar esta tecnología no se pueden tener a corto plazo, es un proceso largo que puede tomar varios meses, en lo que se construyen las bibliotecas; al llegar a este punto comienzan a desarrollarse sistemas más fácil y rápido, gracias a la reutilización de las mismas. El análisis y diseño también se vuelve más sencillo gracias a la experiencia obtenida anteriormente y el mantenimiento de los sistemas será cada vez más sencillo. El objetivo es maximizar reusabilidad y minimizar costos de mantenimiento.

"Así como las computadoras se vuelven más complejas, los humanos no tienen que pensar como computadoras; al contrario, las computadoras deberán pensar como humanos".[5]

III.4.3.3.4 Lenguajes Basados en Objetos (L.B.O.)

La selección de un lenguaje que permita utilizar el paradigma de objetos puede resultar bastante difícil.

Existen muchos libros y autores, cada uno defiende su punto de vista, dicen que un lenguaje no es bueno y otro si, hablan de lenguajes híbridos y puros, etc.. Aquí se describen las características de un lenguaje de objetos y qué tomamos en cuenta para elegir uno.

III.4.3.3.4.1 Características

Para ser descrito como un lenguaje de programación orientado a objetos (LOO), debe soportar:

a) Clases y encapsulación. Cada clase contiene (encapsula) ciertos tipos de datos. Protege estos datos de uso no apropiado ofreciendo un número de operaciones permitidas.

b) Selección de métodos. Con la selección de métodos, el usuario solamente necesita especificar cuál operación debe ser aplicada a un objeto (o a uno o más objetos). Polimorfismo es una de las más comunes aplicaciones de selección de métodos.

c) Herencia. La herencia permite a los sistemas que se construyan sobre jerarquías de clases existentes. Provee mecanismos de construcción y reuso de software (no hay que reinventar la rueda). Algunos lenguajes soportan herencia múltiple, otros sólo soportan heredar de su clase padre.

III.4.3.3.4.2 Lenguajes Puros y Lenguajes Híbridos

Algunos LOO fueron diseñados específicamente para programar con objetos, encapsulación, selección de métodos, y herencia. Smalltalk fue el primer lenguaje desarrollado puramente para soportar el paradigma de objetos, le siguieron Actor y Eiffel.

Otros lenguajes usados para programar con objetos son lenguajes tradicionales que tienen la capacidad de utilizar objetos, selección de métodos y herencia. Estos se conocen como lenguajes híbridos. El más conocido es C++, una extensión de C, que actualmente es el más conocido y utilizado para programación con objetos. Otros ejemplos son Object Pascal, de PASCAL; CLOS, de LISP, y Object COBOL, de COBOL.

Los lenguajes puros son más fáciles de aprender para un programador nuevo que un lenguaje híbrido. Por ejemplo, para aprender C++ primero hay que conocer C, tomaría el mismo tiempo conocer Smalltalk que conocer C. Otras ventajas de los lenguajes puros son: reducción de complejidad, clases reusables, reusabilidad desde la herencia, fácil de hacer cambios, facilidad de experimentación.

Los lenguajes híbridos fueron hechos pensando en "la dificultad para algunos de hacer el tránsito de las viejas a las nuevas técnicas y la 'pesada' carga instalada (mental y electrónicamente) de software viejo..." [8], esta puede ser una manera menos costosa (si se tiene cuidado) de hacer una transición al paradigma de objetos. Otra ventaja es que estos lenguajes hacen chequeo estático de tipos, al

contrario de los lenguajes puros que hacen el chequeo de tipos en tiempo de ejecución, esto impone sobrecarga al ejecutarse.

Elegir un L.B.O. puro o híbrido debe hacerse con mucho cuidado. Si se trata de una empresa que no tiene problemas de costo, y que desea hacer un cambio radical en su manera de desarrollar software, elegir un LOO puro puede ser su mejor alternativa, pero si se quiere gastar lo menos posible y aprovechar los recursos humanos, de hardware y software, elegir un LOO híbrido puede ser la solución. Elegir cualquiera de los dos implica cambios en la manera de pensar, de desarrollar sistemas, requiere capacitación y conciencia de que los resultados se verán a largo plazo, requiere de paciencia y confianza en las personas involucradas así como el compromiso de conocer más y actualizarse en este nuevo paradigma por parte de analistas, desarrolladores, programadores, usuarios y hasta los jefes y directivos de la empresa. No es posible hacer un cambio sin el apoyo de todas las áreas involucradas.

Hay que hacer una consideración final sobre los LOO híbridos. Muchos programadores de C (por tomar un ejemplo) han migrado a C++, utilizan la sintaxis, hacen uso de clases, herencia y hasta de polimorfismo, pero su código carece de una aplicación real del paradigma de objetos. El código sigue utilizando trucos y recursos de la programación estructurada que nada tiene que ver con objetos, en su afán de buscar la eficiencia acostumbrada en C pierden la sencillez y elegancia del paradigma de objetos. Esto puede resultar, en lugar de benéfico, costoso en el proceso de cambio de la empresa, por lo que hay que tener mucho cuidado y conciencia de lo que se está haciendo, no perder de vista la importancia de los objetos y la reusabilidad.

III.6 Herramientas de Desarrollo

III.6.1 Herramientas para el Servidor

En esta parte del capítulo mencionamos las herramientas de software que consideramos mejor para aplicarse en un Servidor y en un Cliente, para resolver los problemas mencionados en Electropura.

III.6.1.1 INFORMIX-4GL

INFORMIX-4GL es un lenguaje de programación de 4ª generación diseñado específicamente para desarrollar aplicaciones de bases de datos. Está basado en RDSQL[4], la extensión INFORMIX del SQL producido por IBM.

Características:

Utiliza instrucciones no procedurales que permiten al programador describir lo que desea sin tener que especificar pasos detallados, debido a esto, hace posible crear aplicaciones en menos tiempo, con pocas líneas de código, haciéndolas menos susceptibles de error y más fáciles de dar mantenimiento.

Proporciona instrucciones procedurales que permiten al programador realizar características específicas en sus aplicaciones.

Es compatible con UNIX, DOS y VMS y se ejecuta en una variadad de máquinas, desde micros hasta mainframes.

[4] Rapid Development Structured Query Language. Lenguaje Estructurado de Consulta para Desarrollo Rápido,

III.5.1.2 Partes de un DBMS

Un sistema manejador de bases de datos (DBMS) puede ser dividido en dos partes:

a) Un lenguaje de datos (de definición y de manipulación), el cual es la interfaz entre el usuario y el DBMS.

b) Una ingeniería de bases de datos (database engine), la cual toma las peticiones de los lenguajes de definición y manipulación de datos y las ejecuta.

La ingeniería de bases de datos opera como un proceso o programa ejecutable y sirve a otros procesos: las aplicaciones de bases de datos las cuales contienen las instrucciones que la ingeniería de bases de datos ejecuta.

INFORMIX Software Inc. ofrece dos implementaciones diferentes del lenguaje de desarrollo de aplicaciones INFORMIX-4GL:

INFORMIX-4GL C Compiler Versión, utiliza un preprocesador para generar código fuente INFORMIX-ESQL/C. Este código es preprocesado para producir código fuente en C, el que después es compilado y ligado como código objeto en un archivo de comandos ejecutables.

INFORMIX-4GL Rapid Development System (RDS), utiliza un compilador para producir pseudocódigo (llamado "pcodigo") en un solo paso. Posteriormente, debe utilizarse un "corredor" (runner), para ejecutar la versión pcodigo de su aplicación.

Ambas implementaciones utilizan las mismas instrucciones INFORMIX-4GL y ambientes de programación casi idénticos. A pesar de que utilizan métodos

diferentes para compilar los archivos fuente en programas ejecutables, existen pocas diferencias en la interfaz con el usuario.

Realizamos el reporte que cubre las necesidades de información que la Gerencia requiere sobre las ventas de todas las plantas.

III.5.1.3 Adquisicion del Software en Electropura

Electropura tomó en cuenta las características de INFORMIX-4GL y evaluando que se podía desarrollar un sistema adecuado sus necesidades se adquirió en 1992 la versión de INFORMIX-4GL-SE para RISC y para INTEL.

Otra ventaja que se tomó en cuenta para adquirirlo fue la siguiente:

Se puede tener interfaz con el paquete administrativo MCBA de GCG (González Cortina Glender y Cía.) ya que por medio de un manejador llamado PLUS-DB, MCBA , que está realizado en COBOL, puede utilizar una base de datos de INFORMIX y así poder compartir información entre el sistema de Liquidaciones y MCBA.

III.5.2 Herramientas para el Cliente

Quedando claros los conceptos sobre lenguajes orientados a objetos, elegimos uno.

Los usuarios de cada planta utilizan computadoras personales (PC) del tipo Intel 386, utilizan Windows como herramienta de trabajo, por lo tanto la aplicación para el cliente debe desarrollarse para ese ambiente. Como lenguaje o herramienta de desarrollo podríamos utilizar Smalltalk (versión para Windows), que es un

lenguaje puro, pero el grupo tendría que comprar una copia para desarrollo y requiere un runtime por cada computadora donde se desee utilizar. Cuentan ya con un compilador de C++ para Windows, que solamente requiere estar en una computadora para hacer el desarrollo y el ejecutable se distribuye a cualquier planta. Si cuantificamos esto obtendríamos:

Smalltalk , versión Windows	US$ 3,000.00
runtime US$170.00 * 5 (plantas) =	US$ 850.00
	US$ 3,850.00
Borland C++ 4.0, versión profesional	US$ 540.00
Diferencia:	US$ 3,310.00

Tomando en cuenta que se tienen los conocimientos de C y C++, se decidió que es la mejor opción para llevar a cabo proyecto de investigación, teniendo cuidado de no caer en el problema mencionado en el capítulo anterior.

> "Sea el mejor o no, el mercado de C y los productos comerciales de software han convertido a C++ en un hecho (no obstante la falta de ortodoxia para algunos y la falta de elegancia para otros) que muchos desarrolladores han dado en llamar el lenguaje orientado a objetos 'estándar' de los 90's." [8]

III.5.3 Modelo de Desarrollo Basado en Objetos

Grady Booch [10] define ciertas etapas que pueden servir para hacer un desarrollo basado en objetos:

III.5.3.1 Etapas para el Desarrollo Basado en Objetos

1a etapa: Identificación de objetos y clases

En esta etapa se definen cuáles son los actores, agentes y servicios más importantes en el medio.

Primeramente debemos obtener una especificación informal del sistema. Es un texto que describe todo lo que el sistema debe realizar, cuales eventos influyen en su comportamiento, personas y datos involucrados, y la información resultante.

Los objetos y las clases deben buscarse dentro de la especificación informal del sistema, ¿cómo los detectamos?.

Para detectar los objetos y clases del sistema debemos seguir los siguientes pasos:

1. Definir el problema en una especificación informal.

2. Descubrir las abstracciones fundamentales en el espacio del problema, es decir objetos y clases.

   2.1. Identificar los objetos asociando dentro de la especificación informal los nombres y sustantivos.
   2.2. Identificar el comportamiento de los objetos por medio de los verbos.

3. Detectar similitudes entre los objetos para formar clases.

4. Realizar tablas de objetos similares, sustantivos o nombres y los verbos actuando sobre los sustantivos.

2a etapa: Identificación de las operaciones de los objetos y clases

Sirve para caracterizar el comportamiento de cada objeto o clase de objetos, para identificar las principales operaciones que realizarán los objetos:

-¿Cómo se crean los objetos de esta clase?

-¿Cómo pueden destruirse?

-¿Los objetos de esta clase pueden ser copiados y/o destruidos?

-¿Qué tipos de operaciones pueden efectuarse sobre los objetos de esta clase?

-¿Cómo pueden manipularse los objetos de esta clase?

-¿Cómo se define la relación de igualdad entre objetos de esta clase, componente a componente?

Las operaciones que pueden realizar los objetos serán de tres tipos:

1.Modificador. Una operación que altera el estado de un objeto.

2.Selector. Una operación que evalúa el estado actual de un objeto.

3.Iterador. Una operación que permite visitar todas las partes de un objeto sin cambiar su estado.

3a etapa: Establecer la relación entre clases y los objetos (Jerarquía)

La jerarquía es una clasificación u orden de las abstracciones. Debemos identificar la jerarquía entre las clases dentro de un sistema, aplicar herencia cuando sea necesario.

4a etapa: Implementar cada objeto

Aquí se produce un módulo de especificación. Dicha especificación también sirve como un contacto entre las aplicaciones y/o funciones de un objeto y el objeto mismo, se refiere a la descripción a detalle de los métodos de cada objeto.

Se definen los módulos y la modularidad del sistema.

Módulo, una agrupación de elementos lógicamente relacionados.

Modularidad, permite generar módulos con un alto grado de cohesión (un módulo coherente sólo debe realizar una operación) y acoplados (medida de interconexión entre modulos, entre más bajo mejor).

Hay que hacer hincapié en que estas etapas no son una receta de cocina que hay que seguir al pie de la letra y que en sólo 4 pasos tenemos un sistema desarrollado. Son 4 puntos que pueden ayudarnos dentro del *proceso en espiral* de análisis y diseño de objetos para encontrar más fácilmente cuales son los objetos y las relaciones que tienen entre ellos.

Decimos proceso en espiral porque es lo que mejor se acopla al desarrollo de sistemas con objetos, ya que el análisis, diseño y programación tienen similitudes, a diferencia de las metodologías tradicionales en donde existe una barrera conceptual entre análisis, diseño y programación (ilustración IV.1). Los programadores ven una tercera parte del mundo. Los analistas usan modelos entidad-relación,

descomposición funcional. Los diseñadores usan diagramas de flujo de datos, gráficos de estructura y diagramas de acción. Los programadores usan COBOL, FORTRAN o C.

En las técnicas de objetos, analistas, diseñadores, programadores y, particularmente importante, usuarios finales, usan todos el mismo modelo conceptual (ilustración IV.1). Todos piensan en tipos de objetos, objetos, y cómo se manipulan.

Las técnicas orientadas a objetos usan el mismo modelo conceptual para análisis, diseño y programación, tiran las paredes conceptuales entre ellos.

En el modelo de objetos no se puede decir que termina una fase de análisis y comienza la de diseño, al desarrollarse el sistema en forma de espiral, a veces hay que retomar el análisis y hacer modificaciones, de igual manera al programar un objeto podemos regresar al diseño o hasta al análisis. Los analistas, diseñadores y programadores no viven solos, todos deben estar compartiendo opiniones para lograr el mejor desarrollo del sistema. No es posible concebir grupos de trabajo donde el analista esté separado del diseñador, y estos dos, del programador.

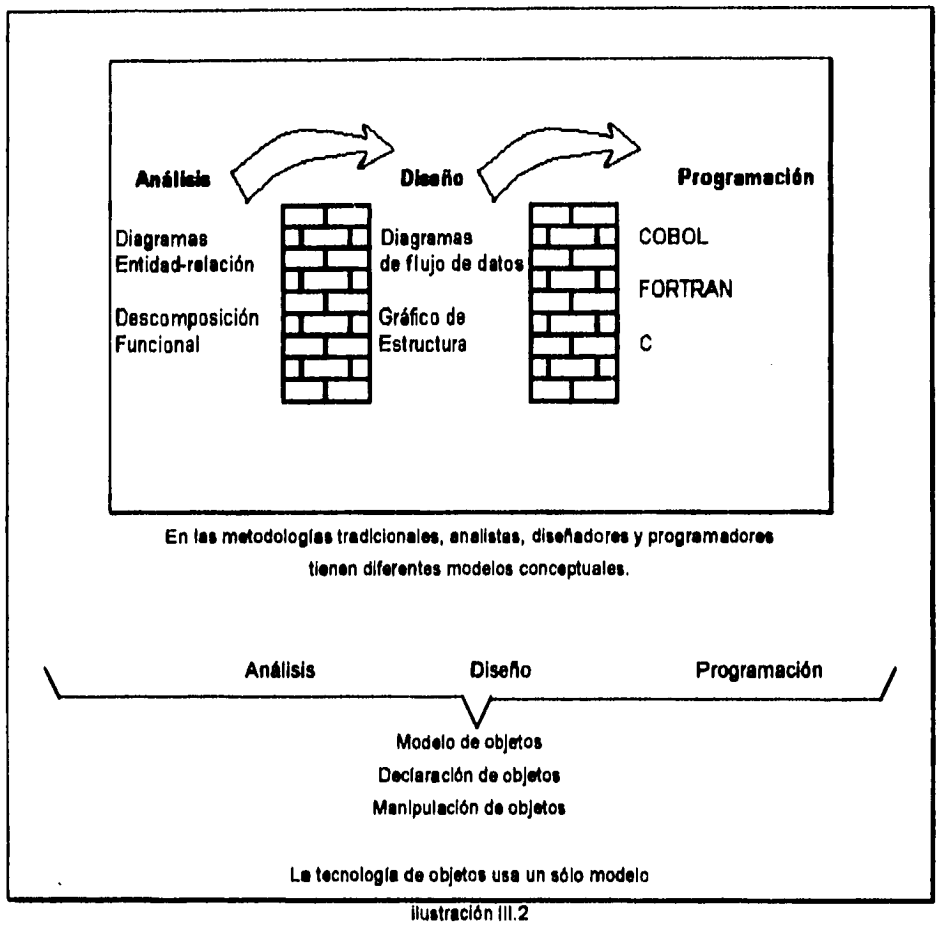


Ilustración III.2
Diferencias entre metodologías.

Por último, el usuario final también debe involucrarse en todo el desarrollo. Se hacen experimentos, se generan prototipos, y el usuario debe opinar sobre el mismo, con tal de retroalimentar y asegurar que al final del proyecto se obtenga lo que se esperaba.

IV APLICACION

IV.1 Definición de Procedimientos a Tratar

Basándonos en la metodología impartida por el Dr. Ricardo Rivera Soler, en la materia Administración de Centros de Cómputo, realizamos el análisis de procedimientos para tener una idea más clara de la problemática a tratar.

Al entrevistarnos con el personal del área de contraloría de la planta Andalucía, nos platicaron el objetivo del reporte gerencial de ventas.

Diariamente, el departamento de contraloría de cada planta se comunica a las oficinas centrales para reportar los siguientes movimientos:

- Ingresos Totales de caja
  - Efectivo
  - Documentos de crédito

- Total de productos vendidos

Esta información se obtiene de la base de datos que se controla por medio de Informix.(ilustración IV.1).

Con todos los reportes en las Oficinas Generales, se elabora un informe y se entrega al Director General de Finanzas y al Director General de la Empresa. Para llevar estadísticas de los movimientos durante la jornada de trabajo y tomar decisiones sobre implantación de políticas que permitan una mayor venta y/o distribución.

09/02/95 PURIFICADORA DE AGUA ANDALUCIA, S.A. DE C.V. 23:18:45
REPORTE GERENCIAL DE VENTAS DIARIAS
Del dia :09/02/95

ANALISIS DE MOVIMIENTOS DEL DIA

VENTA NETA DEL DIA (A,G,R)		91,238.00
EFECTIVO RECIBIDO	78,816.00	
CHEQUES	0.00	
BONIFICACIONES	9,291.98	
COMODATOS	0.00	
REPOSICIONES	0.00	
VENTA A CREDITO	6,566.02	
TOTAL ENTRADA DE CAJA		94,674.00
SOBRANTE		-3,436.00
IMPORTE MOVIMIENTO DE GARRAFON	-833	-13,328.00
SALDO DEL DIA		-16,764.00
SALDO SOBRANTE ACUMULADO DEL MES		-16,764.00
UNIDADES GARRAFON SOBRANTE DIA		-833
UNIDADES GARRAFON SOBRANTE DEL MES		-833

ANALISIS DE VENTA GRAVADA

Venta de 15,057 Aguas	Importe de Venta	Importe de Bonificaciones	Porcentaje Bonificaciones	Precio de Venta
VENTA DE PLANTA	90,342.00	9,291.98	10.285	5.383
VENTA GRAVADA	54,102.00	9,291.98	17.175	4.970

ANALISIS DE RUTAS

Tipo	Salidas Maxima	Salidas Minima	Salidas Total	Entradas Maxima	Entradas Minima	Entradas Total	Venta Maxima	Venta Minima	Venta Total	Prom.x Viaje
Viajes 80	400	0	17,420	335	0	2,363	368	-2	15,057	188
Rutas 34	1,547	210	17,420	679	1	2,363	1,141	202	15,057	442

ANALISIS POR PRODUCTO

No.Prod.	Descrip.	Zonas	Total	Importe
1	AGUA PURIFICADA (ELECTROPURA)	15,057	15,057	90,342.00
101	AGUA DESTILADA (ELECTROPURA)	0	0	.00
120	AGUA BIDESTILADA (ELECTROPURA)	0	0	.00
901	GARRAFON 19 LTS. VIDRIO	-331	-331	-5,296.00
902	GARRAFON 19 LTS. P.V.C.	-502	-502	-8,032.00
951	GARRAFON DE VIDRIO (VENTA)	16	16	256.00
952	GARRAFON P.V.C.(VENTA)	9	9	144.00
985	GARRAFON ROTO DE VIDRIO	-17	-17	272.00
986	GARRAFON ROTO DE PLASTICO	-14	-14	224.00
			Total	77,910.00

Ilustración IV.1
Reporte Gerencial de Venta Diaria.

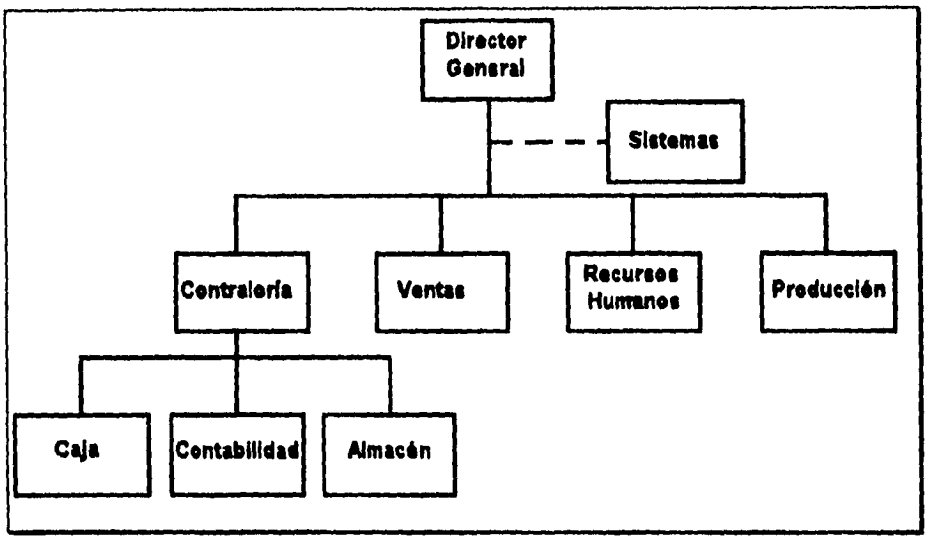
El reporte se integra de 4 partes:

- Análisis de movimientos del día. Se refiere al corte de caja y al total de movimientos de garrafón.
- Análisis de venta gravada. Esta parte menciona la venta total de aguas, su importe en dinero, el porcentaje que representa la bonificación que se dá a los aguacentros y el precio de venta real al que se está vendiendo el agua, esto significa, que debido a la bonificación que se dá por una cantidad de garrafones vendidos, el precio va a variar entre 10% y 20% menor, al precio oficial.
- Análisis de rutas. Aquí se observa el número de viajes que realizaron las rutas de cada planta, tomando como datos estadísticos el viaje con mayor número de garrafones y el de menor número de garrafones, y el promedio por cada viaje.
- Análisis por producto. Se observa la cantidad de productos vendidos.

Al hacer nuestro análisis recopilamos la siguiente información:

PURIFICADORA DE AGUA
ANDALUCIA, S.A. de C.V.
ORGANIGRAMA GENERAL

MANUAL DE INVESTIGACION	Hoja No. 1 de 1
	Fecha de Emisión 21-8-93 Vigencia Actual
Elaboró:SICE.	Autorizó:ACG.



PURIFICADORA DE AGUA
ANDALUCIA, S.A. de C.V.
DEPARTAMENTO DE CONTRALORIA

MANUAL DE INVESTIGACION	Hoja No. 1 de 1
SUBMANUAL DE OBJETIVOS	Fecha de Emisión 21-6-93 Vigencia Actual
Elaboró:SICE.	Autorizó:ACG.

Objetivo:

Llevar a cabo la verificación de los movimientos financieros, realizados por cualquier departamento y comunicarlos continuamente al área corporativa.

Razón de ser:

Es importante la vigilancia de los recursos financieros de la empresa, ya que de esta manera se evitará desperdicio y mal uso de recursos.

Tipo:

General.

PURIFICADORA DE AGUA
ANDALUCIA, S.A. de C.V.
DEPARTAMENTO DE CONTRALORIA
SECCION DE CAJA

MANUAL DE INVESTIGACION	Hoja No 1 de 1
SUBMANUAL DE PROCEDIMIENTOS GENERICOS	Fecha de Emisión 21-8-93 Vigencia Actual
Elaboró:SICE	Autorizó:ACG.

1. LIQUIDACION DIARIA DE VENTAS

1.1 Elaboración de la forma de entrada de caja y entrega de los documentos generados en el dia.

1.2 Captura en el sistema de liquidaciones de los documentos recibidos.

1.3 Entrega de efectivo junto con la entrada de caja sellada.

1.4 Revisión de efectivo contra documentos.

1.5 Transmisión de la liquidación.

1.8 Ejecutar proceso de cierre de caja.

1.7 Emisión de reportes de liquidaciones.

PURIFICADORA DE AGUA
ANDALUCIA, S.A. de C.V.
DEPARTAMENTO DE CONTRALORIA
SECCION DE CAJA

MANUAL DE INVESTIGACION	Hoja No 1 de 2
SUBMANUAL DE PROCEDIMIENTOS ESPECIFICOS	Fecha de Emisión 21-8-93 Vigencia Actual
Elaboró:SICE	Autorizó:ACG.

1. LIQUIDACION DIARIA DE VENTAS

1.1 Elaborar forma de Entrada de Caja.

1) Llenado de forma.

2) Entrega de los documentos del día.

1.2 Capturar en el Sistema de liquidaciones los documentos recibidos.

1) Captura de los datos generales del vendedor en el sistema.

2) Captura de los conceptos de venta.

3) Impresión de la liquidación.

4) Entrega de la liquidación al vendedor.

1.3 Entregar efectivo junto con la Entrada de Caja sellada.

1) Revisar total a pagar.

2) Dar al cajero dinero en morralla y billetes.

3) Regresar hoja de liquidación firmada.

1.4 Revisión de efectivo contra documentos.

1) Contar dinero recibido.

2) Aceptar cantidad correcta o rectificar.

1.5 Transmisión de la liquidación.

1) Captura del total de dinero recibido.

2) Aceptar saldo diario.

3) Actualizar saldo del vendedor y clientes.

PURIFICADORA DE AGUA
ANDALUCIA, S.A. de C.V.
DEPARTAMENTO DE CONTRALORIA
SECCION DE INGRESOS

MANUAL DE INVESTIGACION	Hoja No 2 de 2
SUBMANUAL DE PROCEDIMIENTOS ESPECIFICOS	Fecha de Emisión 21-8-93 Vigencia Actual
Elaboró:SICE	Autorizó:ACG.

1.6 Ejecutar proceso de Cierre de Caja.

1) Introducir fecha de día de corte.
2) Calcular acumulados por cliente.
3) Calcular acumulados por vendedor.
4) Calcular total de documentos.
5) Calcular total de efectivo.

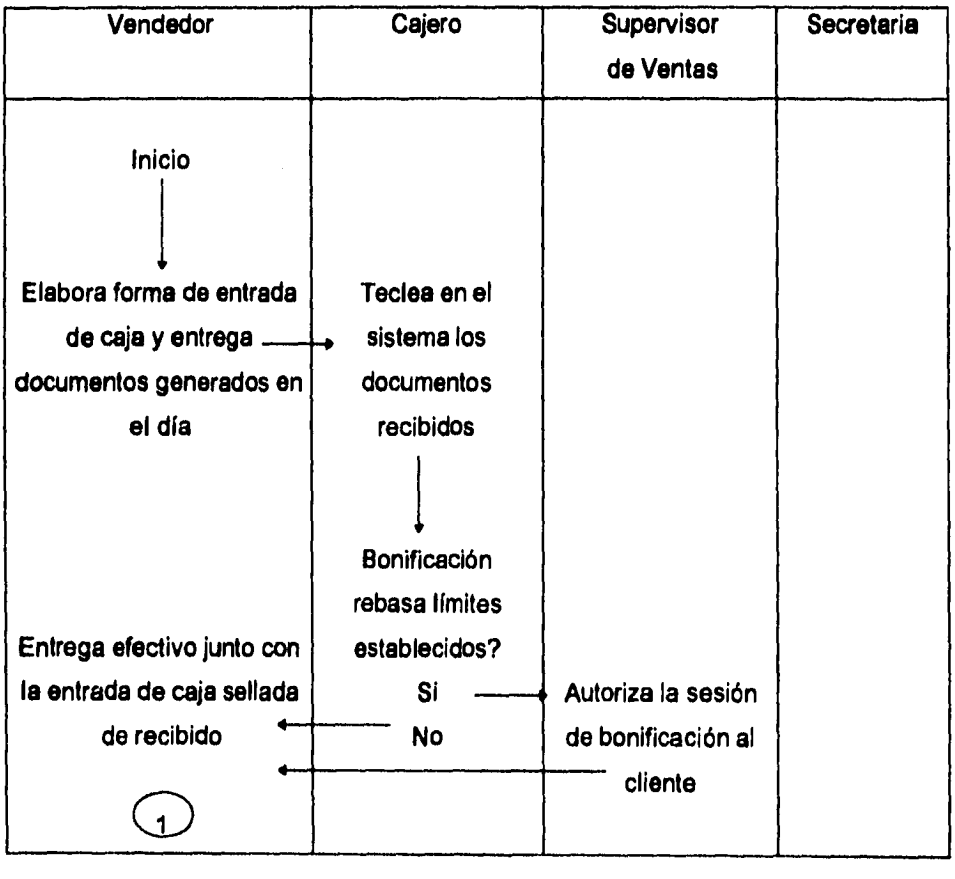
1.7 Emisión de reportes de liquidaciones.

1) Impresión de reporte de faltantes de efectivo.
2) Impresión de saldo de clientes.
3) Impresión de resumen de movimientos.
4) Impresión de reporte gerencial.

PURIFICADORA DE AGUA ANDALUCIA, S.A. de C.V.
DEPARTAMENTO DE CONTRALORIA
SECCION DE CAJA

MANUAL DE INVESTIGACION	Hoja No. 1 de 3
SUBMANUAL DE DIAGRAMAS DE FLUJO	Fecha de Emisión 21-8-93 Vigencia Actual
Elaboró:SICE	Autorizó:ACG.

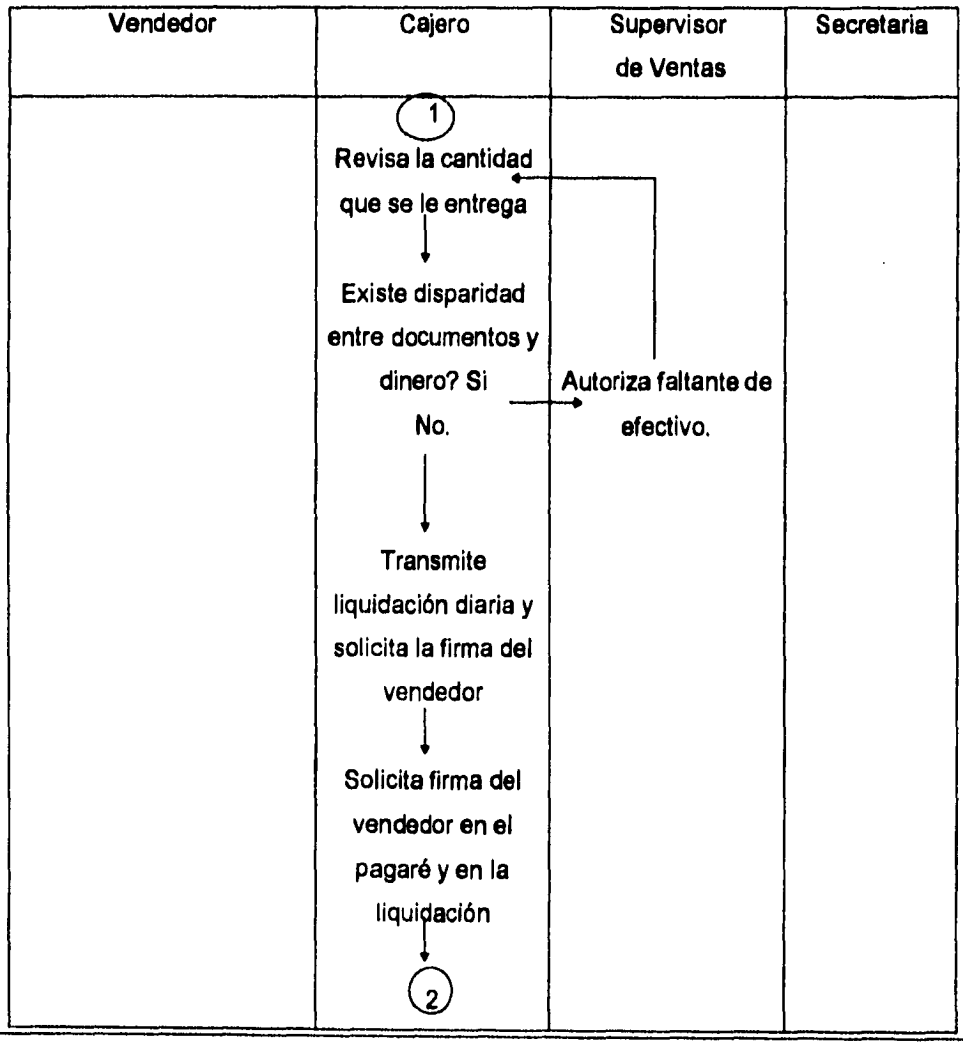
1. LIQUIDACIÓN DIARIA DE VENTAS.



PURIFICADORA DE AGUA ANDALUCIA, S.A. de C.V.
DEPARTAMENTO DE CONTRALORIA
SECCION DE CAJA

MANUAL DE INVESTIGACION	Hoja No. 2 de 3
SUBMANUAL DE DIAGRAMAS DE FLUJO	Fecha de Emisión 21-8-93 Vigencia Actual
Elaboró:SICE	Autorizó:ACG.

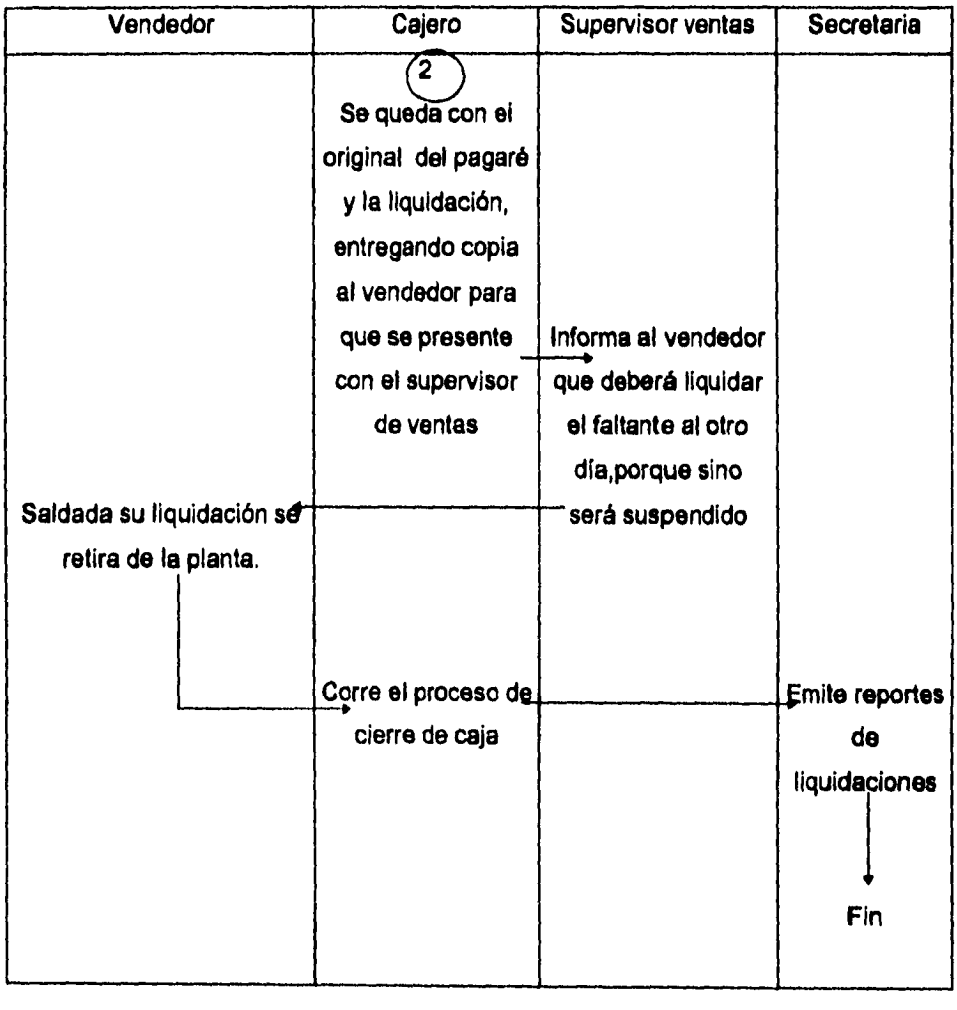
1. LIQUIDACION DIARIA DE VENTAS.



PURIFICADORA DE AGUA ANDALUCIA, S.A. de C.V.
DEPARTAMENTO DE CONTRALORIA
SECCION DE CAJA

MANUAL DE INVESTIGACION	Hoja No. 3 de 3
SUBMANUAL DE DIAGRAMAS DE FLUJO	Fecha de Emisión 21-8-93 Vigencia Actual
Elaboró:SICE	Autorizó:ACG.

1. LIQUIDACION DIARIA DE VENTAS.



PURIFICADORA DE AGUA
ANDALUCIA, S.A. de C.V.
DEPARTAMENTO DE CONTRALORIA
SECCION DE INGRESOS

MANUAL DE INVESTIGACION	Hoja No. 1 de 1
SUBMANUAL DE PROCEDIMIENTOS GENERICO	Fecha de Emisión 21-8-93 Vigencia Actual
Elaboró:SICE.	Autorizó:ACG.

1.INFORMAR AL CORPORATIVO LOS MOVIMIENTOS DE VENTA GENERADOS EN LA PLANTA.

1.1 Imprimir reporte gerencial de ventas.

Utilizar el sistema de cómputo para la impresión del reporte del día anterior de ventas.

1.2 Comunicar a las oficinas generales el reporte.

Realizar llamada telefónica para dictar la información del reporte gerencial de ventas.

PURIFICADORA DE AGUA
ANDALUCIA, S.A. de C.V.
DEPARTAMENTO DE CONTRALORIA
SECCION DE INGRESOS

MANUAL DE INVESTIGACION	Hoja No. 1 de 1
SUBMANUAL DE PROCEDIMIENTOS ESPECIFICOS	Fecha de Emisión 21-8-93 Vigencia Actual
Elaboró:SICE.	Autorizó:ACG.

1.INFORMAR AL CORPORATIVO LOS MOVIMIENTOS DE VENTA GENERADOS EN LA PLANTA.

1.1 Imprimir reporte gerencial de ventas.

1) Introducir clave de acceso.
2) Seleccionar opción del menü de reportes.
3) Capturar fecha del reporte.
4) Calcular totales, promedios, mínimos y máximos de ventas e ingresos.
5) Emitir resultados de cálculos.
6) Cortar papel de impresión.

1.2 Comunicar a las oficinas generales el reporte.

1) Llamar por teléfono a la planta.
2) Solicitar comunicación con la secretaria de la contraloría.
3) Esperar respuesta.
4) Dictar datos.
5) Solicitar verificación de datos.
6) Terminar la llamada.

PURIFICADORA DE AGUA ANDALUCIA, S.A. de C.V.
DEPARTAMENTO DE CONTRALORIA
SECCION DE INGRESOS

MANUAL DE INVESTIGACION	Hoja No. 1 de 3
SUBMANUAL DE PROCEDIMIENTOS ESPECIFICOS	Fecha de Emisión 21-8-93 Vigencia Actual
Elaboró:SICE.	Autorizó:ACG.

1.INFORMAR AL CORPORATIVO LOS MOVIMIENTOS DE VENTA GENERADOS EN LA PLANTA.

Secretaria de contraloría	Secretaria de gerencia	Director General
Inicio ↓ Introducir clave de acceso ↓ Seleccionar opción del menú de reportes. ↓ Capturar fecha del reporte. ↓ Calcular totales, promedios, mínimos y máximos de ventas y otros ingresos. ↓ Emitir resultados de cálculos. (1)		

PURIFICADORA DE AGUA ANDALUCIA, S.A. de C.V.
DEPARTAMENTO DE CONTRALORIA
SECCION DE INGRESOS

MANUAL DE INVESTIGACION	Hoja No. 2 de 3
SUBMANUAL DE PROCEDIMIENTOS ESPECIFICOS	Fecha de Emisión 21-8-93 Vigencia Actual
Elaboró:SICE.	Autorizó:ACG.

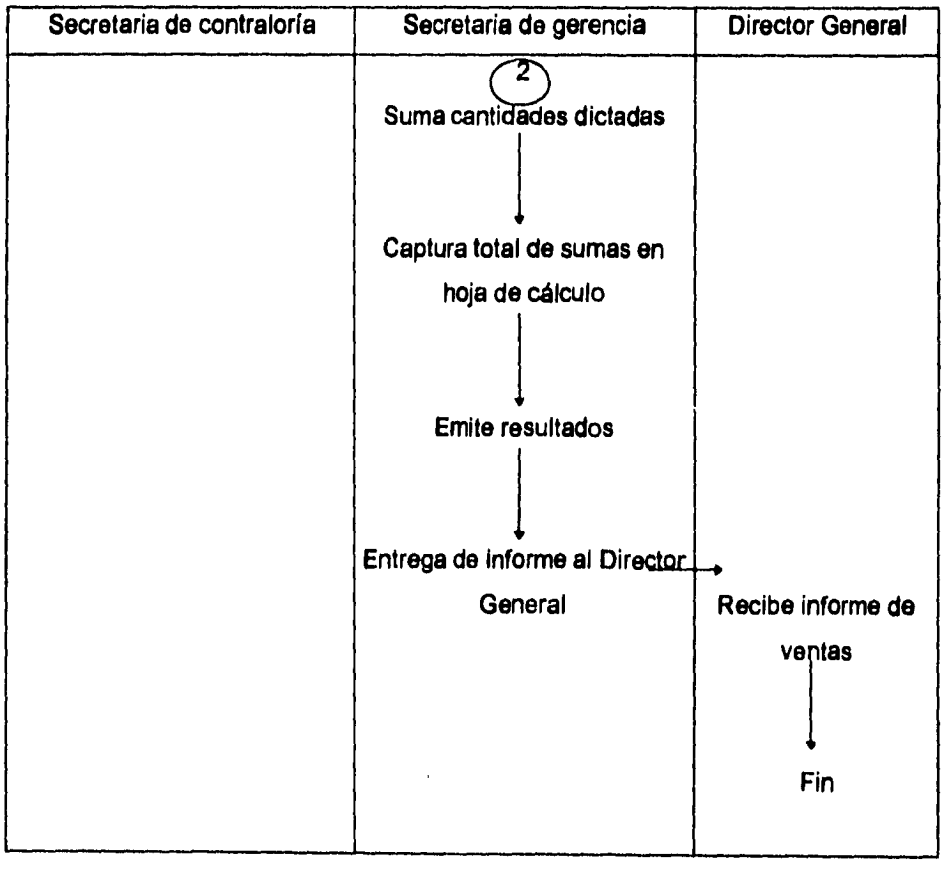
1.INFORMAR AL CORPORATIVO LOS MOVIMIENTOS DE VENTA GENERADOS EN LA PLANTA.

Secretaria de contraloría	Secretaria de gerencia	Director General
(1) Cortar papel de impresión. → Llamar por teléfono a la planta → Solicita comunicación con secretaria de la gerencia	Toma llamada telefónica → Recibe dictado de cantidades → Solicita verificación de datos → Termina la llamada (2)	

PURIFICADORA DE AGUA ANDALUCIA, S.A. de C.V.
DEPARTAMENTO DE CONTRALORIA
SECCION DE INGRESOS

MANUAL DE INVESTIGACION	Hoja No. 3 de 3
SUBMANUAL DE PROCEDIMIENTOS ESPECIFICOS	Fecha de Emisión 21-8-93 Vigencia Actual
Elaboró:SICE.	Autorizó:ACG.

1.INFORMAR AL CORPORATIVO LOS MOVIMIENTOS DE VENTA GENERADOS EN LA PLANTA.



IV.2 Uso de Metodología, Herramientas y Lenguajes

Tomando en cuenta lo descrito en el Capítulo I, situamos a SICE en el modelo de Presentación Remota (ilustración 1.2, página 7), donde el frontend realiza el Servicio de presentación y la Lógica de presentación; y el backend realiza las operaciones restantes (lógica del negocio, lógica de datos, servicio de datos y servicio de archivos), cada planta cuenta con una base de datos que se utiliza para desarrollar en el servidor la aplicación que genere el reporte, y en el cliente realizar el programa que solicite y presente el reporte obtenido. La siguiente ilustración representa lo anterior:

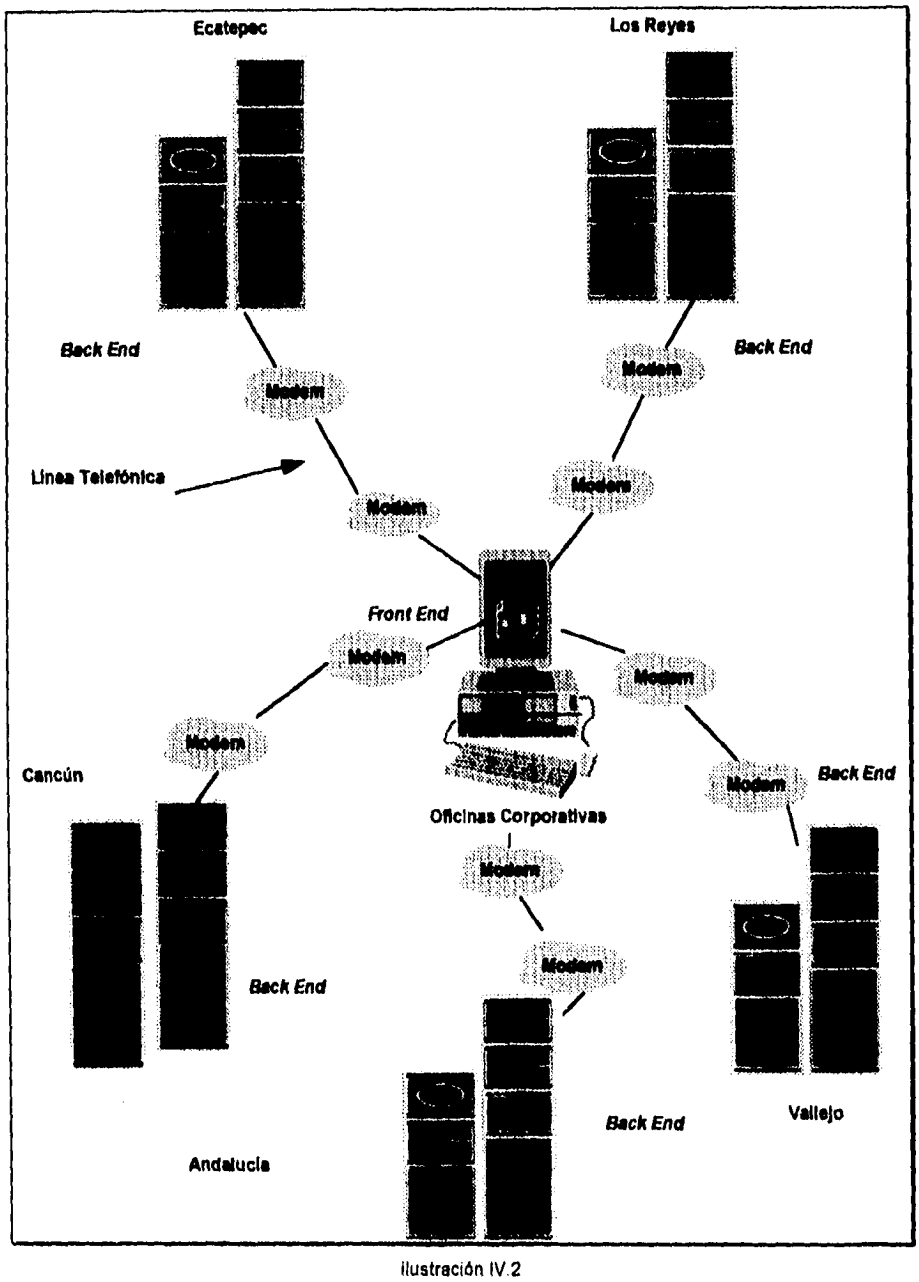


Ilustración IV.2
Esquema general de las plantas

Basándonos en el análisis de procedimientos identificamos la lógica del negocio, que es la siguiente:

1. Contar las salidas de productos por trabajador.
2. Cobrar el saldo por trabajador al final del día.
3. Realizar corte de caja separando, cuánto se obtuvo en efectivo y documentos de crédito.
4. Realizar inventario de garrafones que se movió en el día.

Con base en esta lógica del negocio, desarrollamos los programas en informix que permiten registrar información de los movimientos de venta y hacer los cálculos correspondientes para generar el reporte denominado Gerencial de Ventas Diarias. Cabe hacer notar que al trabajar con el área de sistemas de Electropura se nos dió la oportunidad de desarrollar éste módulo. En el Apéndice C se muestra el algoritmo para el programa del servidor , asi como los diagramas relacionados con la base de datos.

IV.2.1 Construcción de un Modelo Cliente-Servidor para Electropura

El diseño de la lógica del negocio se basa en la forma de operar de la planta. La lógica de la comunicación entre el cliente y el servidor se realiza tomando en cuenta las características de los equipos que se conectan.

> "Se debe tener un conjunto de ideas para planear y desarrollar proyectos cliente-servidor. Utilizando lo que es familiar, es decir, aproximaciones basadas en mainframes para desarrollar proyectos cliente-servidor puede parecer que incrementa el nivel de comodidad, pero usualmente se pierde tiempo y dinero, si no es que se pierde mucho más." [11].

Hay que recalcar que uno de los principales beneficios de cliente-servidor es su potencial para liberar sistemas rápidamente. Este beneficio es posible por que las

aplicaciones que se desarrollan con cliente-servidor se hacen con un proceso iteractivo.

Nuevas funciones se liberan en cada fase del proyecto. Dentro de un plan, se debe planear, analizar, diseñar y desarrollar rápido para cada fase. ¿Significa esto que cliente-servidor no es para perfeccionistas?. No necesariamente. Un perfeccionista con un conjunto de ideas de negocios puede manejar esto, pero un perfeccionista que tiene solamente un conjunto de ideas de tecnologías puede ser agobiado.

La planeación correcta es importante para una exitosa implementación cliente-servidor, pero no se debe permitir que esto tome mucho tiempo. Cada una de las grandes empresas tiene sus analistas de planeación quienes se retiran del trabajo y dicen que todavía forman parte de ésta. Esta gente no es la que va a ayudar a liberar exitosos sistemas cliente-servidor. Así que, ¿cómo se puede tener éxito?

Un elemento clave para adquirir el conjunto de ideas correcto cliente-servidor es trabajar con un buen conjunto de guías.

Las guías pueden ayudar a pensar el curso de la acción, así se mantiene control desde el principio. En el mundo de los negocios de hoy en día, no hay tiempo para sentarse al final del proyecto y pensar que es lo qué se va hacer diferente la próxima vez. Además, los sistemas en ambientes cliente-servidor difieren mucho, así que probablemente nunca se utilizará el mismo camino. Por lo tanto, las guías pueden ayudar a realizar el sistema correcto desde la primera vez.

También pueden ayudar a mantener unidas a las personas que trabajan en el proyecto, estas guías no deben de ser usadas como un decreto, si no deben ser pensadas desde un punto de vista común que ayude a los que se encuentran involucrados en el proyecto - desde los usuarios hasta el staff de técnicos - a

encontrar una agenda común. Una vez que el proyecto está en marcha, las guías sirven como un "chequeo" para ayudar a recordar que actualmente hay un bosque de ideas claras cuando todo lo que se ve es un montón de árboles. A través del proyecto, las guías dan una perspectiva de "afuera para adentro" así que se puede evitar ver de adentro para afuera.

IV.2.2 Aplicación de 12 Guías para lograr una Implementación Cliente-Servidor Exitosa

Algunos autores han identificado 12 guías (según [11]) para lograr una implementación cliente-servidor exitosa. Sin embargo, en algunas empresas se puede encontrar que solamente 10 guías son necesarias, o se pueden necesitar algunas adicionales. A continuación se describe cada una de éstas y su aplicación en la presente tesis.

1.- Uso del lenguaje no técnico, listar las funciones que serán incluidas en el nuevo sistema. La terminología es útil cuando hace que las cosas se entiendan, pero es dañina cuando es usada para confundir e impresionar a la gente. Una y otra vez, se escucha a usuarios quejándose de no entender lo que los técnicos están hablando. El usuario pregunta, "¿por qué no puedo hacer esto?" y las personas expertas vacilan por unos minutos, en vez de simplemente admitir, "no lo sé". ¿Por qué los usuarios deben entender a alguien que no se toma la molestia de comunicarse con ellos?. Por lo tanto, cuando se identifican las funciones que serán cubiertas por el proyecto, se debe ser simple y claro. En este caso, menos es más.

2.- Especificar las metas y beneficios del proyecto, función por función. Establecer los objetivos que definen el incremento en la productividad o los beneficios (u otra cosa que se pueda medir) que se están buscando. Definir cada objetivo, función por función, y ser específico. No ponerse en el plan de "dar el mejor

servicio", porque esta clase de lenguaje confuso hace imposible medir el éxito y ayuda a promover la falta de medición, lo cual hace que se señalen entre usuarios y técnicos. Las metas funcionales deben ser más tangibles. En nuestro caso, por ejemplo, especificar que las liquidaciones serán procesadas en 5 minutos, en vez de 30 minutos.

3.- Partir el proyecto en fases y asignar las funciones que serán discutidas en cada fase. Desarrollar proyectos cliente-servidor es un proceso iteractivo, y la funcionalidad es liberada en cada fase. Por lo tanto, después de haber desarrollado una lista de funciones, se deben asignar a diferentes fases del proyecto. En otras palabras, la fase uno traerá como resultado funciones tales como inicio de la aplicación de comunicación, y la habilidad de generar automáticamente estadísticas de ventas; la fase dos será dar otras funciones tales como habilidades de revisión de saldos rojos de empleados, y acceso remoto a los miembros de la dirección para trabajar en su oficina; y así seguir. Esta aproximación da el escenario para tener discusiones más amplias con los usuarios finales, porque a ellos les gustaría liberar ciertas funciones en un orden diferente. Para una comunicación efectiva, tales discusiones deben tener lugar en reuniones (tan cortas como sea posible), más que por medio de memorándums.

La siguiente tabla presenta la división de las fases para la elaboración de SICE.

Fase	Descripción
1	Lógica del negocio
2	Pruebas de comunicación
3	Consulta y presentación del reporte

Tabla IV.1
Fases del proyecto

4.- Determinar el nuevo sistema del negocio, manejo, y objetivos técnicos, identificar similaridades y diferencias entre estos objetivos y buscar concientizar cada uno. Los objetivos no son siempre los mismos para cada grupo. Un objetivo del negocio puede ser incrementar el número de ventas en un diez por ciento. Un objetivo del manejo puede ser cargar un camión en 15 minutos. Y un objetivo técnico por desarrollarse puede ser procesar cada liquidación en 5 minutos.

Hay que tomar en cuenta que aunque los objetivos parezcan trabajar juntos, son diferentes. Por ejemplo, suponga que se tiene un objetivo técnico que llama para procesar un pedido en 1 hora, pero el objetivo del manejo dice que el pedido debe estar liberado para el cliente en diez minutos. Obviamente existe un conflicto entre estos dos objetivos. También pueden darse los casos en los cuales los mismos objetivos son definidos usando diferentes terminologías o términos. En algún evento, se necesitará explorar las similitudes y diferencias y llegar a algunos acuerdos.

Hasta aquí la discusión se ha enfocado sólo al trabajo orientado al negocio, no al trabajo orientado a la tecnología. La habilidad de encontrar los objetivos del negocio juega el papel más importante en los sistemas cliente-servidor. Un recordatorio para la gente de cómputo:

> "Las computadoras están para ayudar al negocio. Aún si el negocio es de computadoras, se debe de recordar que los objetivos del negocio son primero." [11]

5.- Desarrollar la mejor estimación del costo en cada fase - después doblar el número. Inicialmente se deben tratar las metas del negocio antes de deslumbrarse con las herramientas y las técnicas. En el mundo de hoy en día, las herramientas están cambiando constantemente y evolucionando; sin embargo, los objetivos del negocio no van a cambiar tan radicalmente (eso se espera). Por lo tanto, hay que desarrollar los costos para cada fase - y después doblarlos-. Doblar el costo puede

parecer excesivo, pero siempre cuesta más de lo que se piensa desarrollar un nuevo sistema. No hay que ser tan ingenuo como para calcular una propuesta inicial y ponerla como objetivo. Siempre se lleva más tiempo y dinero del que se estima (Tabla IV.2).

Fase	Descripción	Tiempo estimado de elaboración
1	Lógica del negocio	3 semanas
2	Pruebas de comunicación	4 semanas
3	Consulta y presentación del reporte	12 semanas

Tabla IV.2
Tiempo estimado del proyecto

6.- Determinar el negocio y las barreras técnicas para encontrar los requerimientos. Como las restricciones están fuera de control, no puede ser técnicamente factible encontrar todos los requerimientos. Por ejemplo, quizás un mecanismo deliberado hace imposible encontrar la meta de procesamiento de 3 horas. Trate de determinar cuáles son las barreras que se tendrán antes de desarrollar el sistema.

En nuestro caso las barreras que se reconocieron antes del inicio del sistema son:

1. Conocimiento de los conceptos y programación de la comunicación entre computadoras por medio del modem.
2. Conocimiento de la metodología y programación de objetos.
3. Conocimiento de la infraestructura, función y objetivos de cliente-servidor.

7.- Determinar cuáles procesos estarán en el servidor y cuáles procesos estarán en el cliente; después desarrollar un modelo de trabajo de lo que el sistema

hará. El modelo del sistema resultante deberá ser un frontend, pero deberá ser sólo un prototipo. Hay que recordar que este prototipo no deberá ser desechado; en efecto, nada deberá ser desechado en los proyectos cliente-servidor. Los clientes frecuentemente quieren prototipos que puedan ser guardados y usados. Muchos usuarios saben qué quieren solamente después de ver un modelo de lo que ellos solicitaron. De esta manera construimos la siguiente tabla describiendo los procesos para cada fase.

Fase	Procesos	Realizado por
1	-Contar las salidas de productos por trabajador	Servidor
	-Contar el saldo por trabajador al final del día.	Servidor
	-Identificar cuánto se obtuvo en efectivo y cuánto en documentos.	Servidor
	-Elaborar estadística de comportamiento de venta.	Servidor
2	-Conectar las dos computadoras por el modem.	Cliente
3	-Hacer consulta.	Cliente
	-Presentar informe.	Cliente
	-Imprimir informe.	Cliente
	-Guardar informe.	Cliente

Tabla IV.3
Procesos del sistema SICE

8.- Decidir cuáles componentes del software deben ser comprados y cuáles componentes deben ser desarrollados; después encontrar las herramientas correctas. El software para los ambientes cliente-servidor se incrementará

considerablemente en el futuro. En efecto, se verán más aplicaciones para bancos, aseguradoras, finanzas, manufactura, y así se irá desarrollando en los siguientes años. Muchos vendedores tienen estas aplicaciones en sus pizarrones, en sus laboratorios o en pruebas alfa. Y hay unos pocos de recursos humanos y otros paquetes en el mercado. Un número de vendedores de software para mainframes están tratando de mover el ambiente cliente-servidor a éstos, mientras que otros vendedores están migrando desde la computadora de escritorio para suplir los grupos de trabajo o el software departamental.

Por lo tanto, hay que observar cuidadosamente el mercado del software y ser precavido en el desarrollo de las aplicaciones. Se debe tener cuidado en que la opción que se elija cumpla con los requerimientos porque un sistema adquirido probablemente no cumpla con todos.

Es importante seleccionar y usar herramientas apropiadas. No hay que dejarse deslumbrar por las herramientas o frontend, observe qué sucede detrás de las escenas: Un frontend puede parecer simple como un pato que se desliza sobre el agua; debajo del agua, sin embargo el pato puede estar furiosamente agitando sus patas.

9.- *Definir el impacto del proyecto en los recursos de la información.* Recuerde, que ya se tienen muchos sistemas desarrollados en mainframe, en computadoras de rango medio, o en cualquier otro lado. ¿La misma gente va a manejar el ambiente cliente-servidor?. No se asuma que los programadores que manejan COBOL, PASCAL, o RPG se harán expertos en una noche. Algunos procesos deben de cambiar para desarrollar una exitosa aplicación cliente-servidor. Recuerde qué sucedió cuando un programador de COBOL aprendió un 4GL: La primera aplicación en 4GL pareció híbrida, sin intención por supuesto, incorporó las más indeseables prácticas de programación de ambos lenguajes.

Para hacer aplicaciones cliente-servidor se requiere un cambio en la forma de pensar. Cambiar será necesario porque cliente-servidor nos dá la pauta para obtener ambientes basados en objetos. Se necesita alguna clase de programador que se encarge de crear módulos para incrementar la productividad con bibliotecas de clases de objetos. Por lo tanto las herramientas cliente-servidor y el medio ambiente hacen que se pueda mover más efectivamente la programación basada en objetos y a bases de datos.

10.- Entender el impacto cultural y organizacional en la corporación y desarrollar un plan para manejar los cambios. Si el Gerente de Sistemas de Electropura, piensa que la microcomputadora es sólo una moda, entonces su carrera está terminada.

Los cambios organizacionales ocurren, y hay que ajustarse a ellos.

11.- Desarrollar un plan a seguir para el grupo de técnicos. Para desarrollar un sistema cliente-servidor exitoso, espere altos costos. Sin embargo, ¡el costo de no realizarlo es aún más alto !

Si sus programadores de cliente-servidor están escribiendo programas con un **GUI** frontend, deben aprender programación de manejo de eventos, también, programación orientada a objetos usando bibliotecas de clases y código de patrones. Y los analistas de sistemas deben capacitarse para desarrollar aplicaciones junto con los programadores en forma rápida y correcta, además de cómo particionar procesos.

También, las redes (locales y de áreas más amplias) es una parte integral del ambiente cliente-servidor, pero muchas organizaciones no tienen experiencia en problemas de redes, sobre todo, cliente-servidor no es tan intuitivo como la gente habría creído. Por lo tanto, prepararse es lo más importante para tener éxito en cliente-servidor.

12.- Desarrollar sistemas usando metodología incremental. Las aplicaciones que están en los tradicionales mainframes son proyectos típicamente implementados con el método de "cascada" (ilustración IV.3). Primero, los desarrolladores empiezan planeando. Cuando ellos terminan ese paso, hacen el análisis. Después de analizar, empiezan el diseño y el desarrollo para completar una fase. Y, si fueron listos (o así lo pensaron), los desarrolladores recibirán firmas de los usuarios al final de la planeación y otra vez al final del análisis. Desafortunadamente, para cuando ellos terminen el desarrollo, los usuarios que firmaron las especificaciones ya no estarán en la empresa. Y como este proceso toma al menos 3 o 5 años, el sistema terminado ya no cumple con las necesidades.

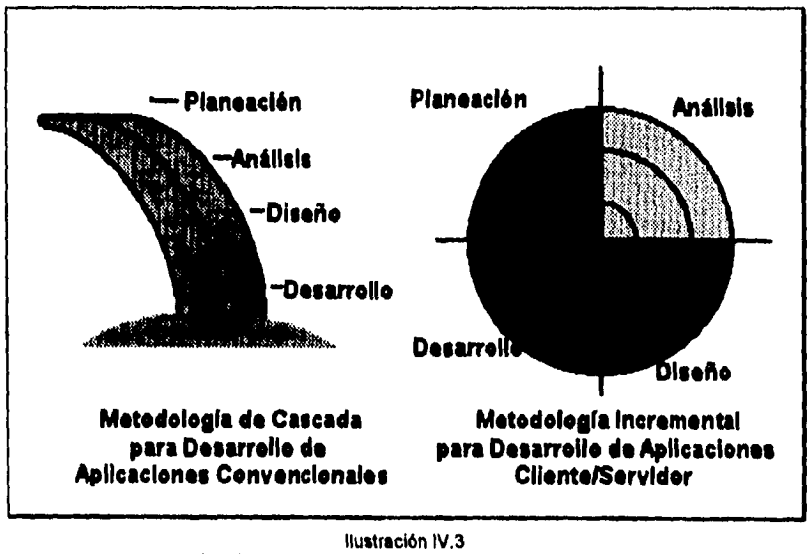


Ilustración IV.3
Metodología de cascada vs metodología incremental

Para los desarrollos cliente-servidor, se utiliza el método de espiral para construir el sistema de manera incremental. Con el método convencional de cascada, una vez que el agua a caído, no puede regresarse; esto es, no se puede regresar a un paso anterior. Con el modelo incremental, sin embargo, se puede

identificar diversas funciones, planearlas, y después llevar a cabo el análisis, diseño y desarrollo. En este punto se puede decidir si se debe continuar con el proyecto o cancelarlo antes de invertir más tiempo, dinero, análisis o diseño. Cerca del 50% de todos los proyectos son abortados durante alguna de estas fases. Usando la aproximación de espiral, se puede venir a una fase y quitar algo antes de invertir una tremenda cantidad de dinero. En la siguiente fase, se puede proceder a ampliar, agregar más funciones, y entonces planear, analizar, diseñar y desarrollar. Después decidir si se puede continuar con el proyecto.

Conforme el desarrollo avanza, permite a los usuarios ver la construcción del sistema y decir cuando algo no les parece. Frecuentemente los usuarios saben qué es lo que quieren realmente después de haber usado el sistema. Por lo tanto, el método de espiral involucra trabajo en fases y funciones, y expande el sistema tanto como el negocio necesite cambiar. Al sistema debe dársele constantemente mantenimiento y desarrollo. El método de espiral permite hacer un desarrollo incremental rápido y realizar aplicaciones cooperando con el usuario final.

Se puede proceder a través del proceso en círculos más grandes, se llegará al momeno donde los incrementos sean mucho más pequeños simplemente porque se tienen hechos los refinamientos necesarios al sistema. En este punto, se detendrá a tratar de mejorar el sistema y enfocarse al desarrollo de nuevas funciones para estar de acuerdo en los cambios en el negocio. Por lo tanto, se puede usar una aproximación modular y dejar lo que no se necesita. En efecto, el sistema cliente-servidor puede ser un medio ambiente viviente, creciente, evolucionante, más que un sistema estático extintivo.

IV.2.3 Definición de Clases y Objetos

Habiendo definido ya el problema y elegido un LOO a utilizar, pasemos a definir las clases y los objetos, apoyándonos de las etapas definidas por Booch[10]. Primero hagamos una definición informal.

La aplicación desarrollada en este seminario debe permitir, desde el corporativo, hacer una consulta, conectándose por medio del modem, al servidor de datos que se encuentra en cada una de las plantas. Los resultados de la consulta deben ser presentados en pantalla y permitir generar un reporte, conocido como Reporte Gerencial de Ventas Diarias. El reporte puede ser impreso y/o guardado en archivo para su revisión posterior.

Dentro de esta especificación del sistema encontramos como posibles objetos:

Modem
Servidor
Archivo
Consulta
Planta
Oficinas Generales
Pantalla
Reporte

Tabla IV.4
Posibles objetos para el proyecto

y como posibles métodos:

hacer consulta
conectar
presentar reporte
imprimir reporte
guardar

Tabla IV.5
Posibles métodos para el proyecto

Es importante analizar detalladamente los posibles objetos y los métodos, encontrar relaciones y similitudes. Cabe la posibilidad de que algunos candidatos de objetos no sean tomados en cuenta como tal, o que formen jerarquías de clases con herencia. Con respecto a los métodos, es posible que se agreguen más o que otros no se tomen en cuenta. También deberán aparecer atributos de cada objeto.

Tomemos primero el objeto modem. Como se menciona en el capítulo II, es un dispositivo que permite a una computadora comunicarse, por medio de la línea telefónica, con otra computadora que tenga también un modem.

Definición del objeto Serial

De esta manera vemos que primero debemos definir un objeto serial, del cual el objeto modem hace uso. Aquí podemos utilizar herencia, derivando atributos y métodos del puerto serie al modem y agregar otros. Los atributos y métodos de Serial son:

Atributos	Métodos
Baudrate	Iniciar
Bits de datos	Cerrar
Bits de paro	Leer
Paridad	Escribir
Puerto	

Tabla IV.6
Definición inicial para el objeto Serial
(La columna de métodos no corresponde
uno a uno a la columna de atributos.)

IV.2.3.1 Diagramas para Representación de Objetos

Para representar los objetos de una manera gráfica y que permita generalizar su jerarquía hacemos uso de dos tipos de diagramas:

a) Diagramas Fern: Es un tipo de diagrama que tiene semejanza a una estructura de red o de árbol. Son muy útiles para ver las categorías de los objetos, muestra claramente las rutas de herencia, aunque pueden llegar a ser complejos. Un ejemplo es:

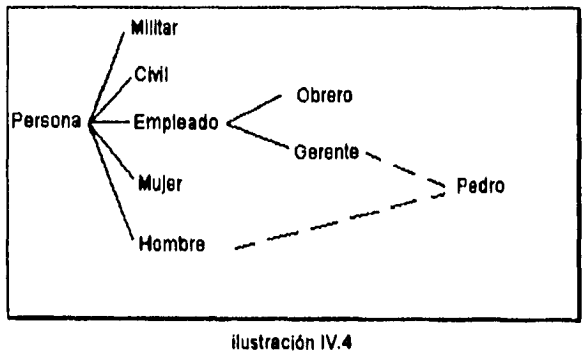


Ilustración IV.4
Ejemplo de diagrama Fern

Es una categorización de personas. Las líneas seguidas representan herencia hacia una clase, la línea punteada representa una instancia de la clase. Pedro es una instancia de la clase Gerente y de la clase Hombre, de esta manera hereda características de Hombre y de Gerente (y de Empleado).

b) Diagramas de caja: Es otra alternativa para mostrar la categorización de clases. Son usados cuando hay pocos objetos. Muestran subtipos de objetos:

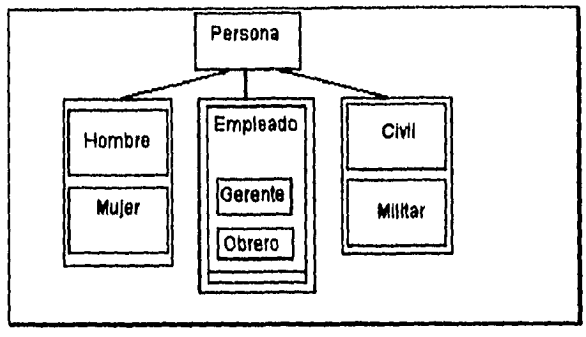


Ilustración IV.5
Ejemplo de diagrama de subtipos

Muestra la misma categorización pero agrupando por subtipos: sexo, tipo de empleado y condición cívica. La línea horizontal debajo del subtipo obrero indica que es posible tener aún más subtipos, pero que todavía no se definen.

De la misma manera podemos presentar la categorización de modem:

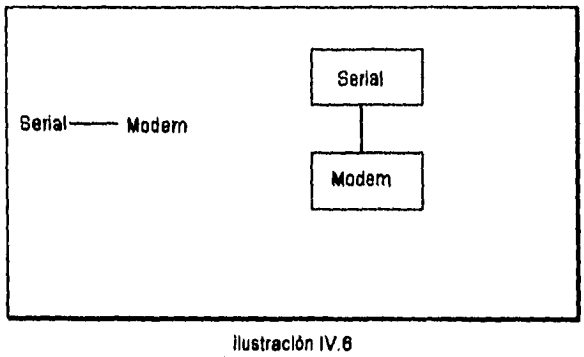


Ilustración IV.6
Categorización del objeto Modem

Pero aquí no termina la categorización de modem. Así como existe un puerto serial existe un puerto paralelo, no son iguales, pero hacen tareas similares, tales

como leer del puerto y escribir al puerto, entonces podemos definir una clase que contenga estos dos métodos comunes y heredar a cada puerto, obviamente no es lo mismo escribir al puerto serial que al paralelo, a cada clase le corresponde un método especial para escribir (este es un ejemplo claro de polimorfismo). Esta clase general la llamaremos TPuerto

Definición de la clase TPuerto

TPuerto es la clase que encapsula el comportamiento general de cualquier tipo de puerto. Su categorización es:

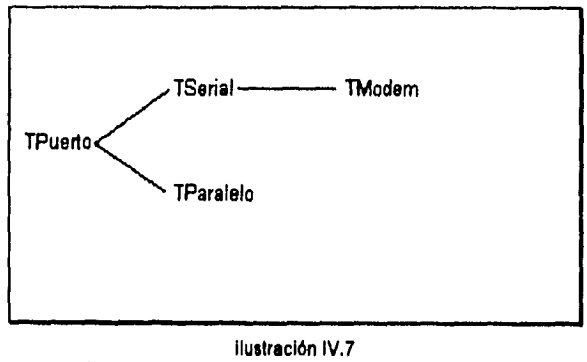


Ilustración IV.7
Categorización del objeto TPuerto en diagrama Fern

Como son pocos objetos mejor utilizamos los diagramas de caja:

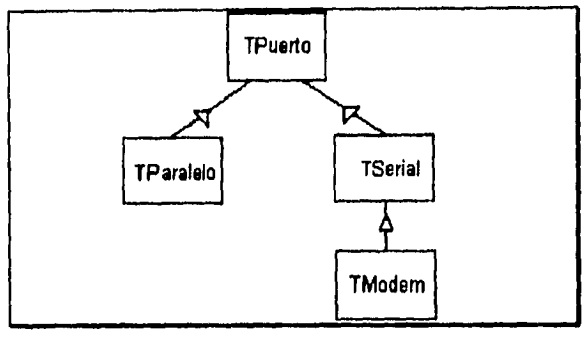


Ilustración IV.8
Categorización del objeto TPuerto en diagrama de subtipos

Las flechas indican que son una clase derivada de la que están apuntando.

Estos diagramas solamente representan las clases y sus relaciones. Para conocer sus atributos y sus métodos utilizamos diagramas como el que sigue:

TPuerto
Buffer Estatus
Leer Escribir

Tabla IV.7
Definición de la clase TPuerto

La caja representa una clase, el nombre de la clase se describe en la parte superior, en la primera mitad los atributos y en la segunda mitad los métodos.

Representación de clases a través de la herramienta utilizada

La herramienta utilizada es el compilador Borland C++ versión 4.0, para el desarrollo utiliza una representación gráfica de clases orientada a programadores en C++ (ilustración IV.9), la **V** representa una variable (o atributo) y la **F** una función (o método), muestra también si es hija y/o padre de otra(s) clase(s).

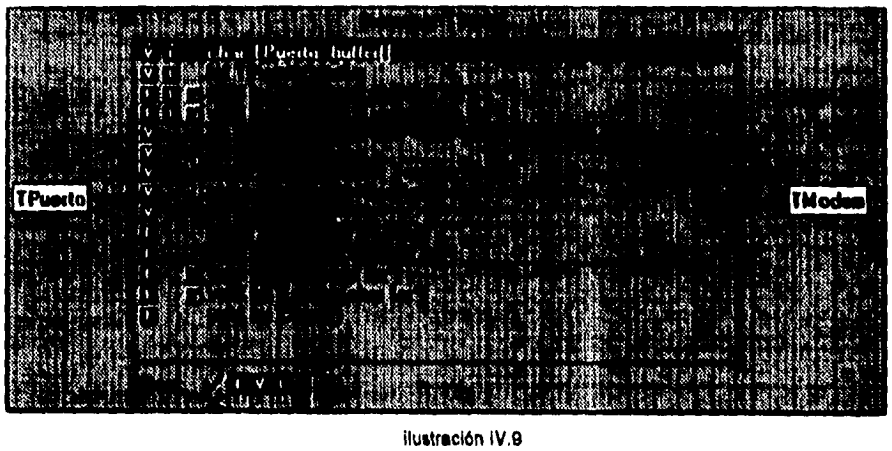


Ilustración IV.9
Representación de una clase por medio de la herramienta utilizada para el proyecto SICE.

En C++, para representar un objeto se utiliza una estructura de datos llamada clase. En ésta se describen los atributos y métodos del objeto. Una clase es la generalización del objeto.

La definición de las clases para esta categorización en código de C++ es:

```
#ifndef PUERTO_H
#define PUERTO_H

class TPuerto{
  protected:
   char buffer[1024];
  public:
   int status;
   virtual void Leer(char *)=0;
   virtual void Escribir(char *)=0;
};

#endif
```

Listado IV.1.
Archivo puerto.h, definición de la clase TPuerto.

Definición de la clase TSerial

La clase TSerial hereda atributos de la clase TPuerto, como son buffer, status, Leer y Escribir.

Los atributos y métodos añadidos para el comportamiento de la clase TSerial son los siguientes (tabla IV.8):

TSerial
Buffer
Estatus
Baud_rate
Puerto
Paridad
Bits_paro
Bits_datos
Iniciar
Cerrar
Leer
Escribir

Tabla IV.8
Definición de la clase TSerial

El código C++ para esta clase es:

```
#ifndef SERIAL_H
#define SERIAL_H

#include "puerto.h"
#include "tconfgpt.h"

class TSerial : public TPuerto{
  private:
   int ReceiveMode;
   TConfigPtoXfer TransferStruct;
   TConfigPto*   ConfigPto;
  public:
   int Puerto;
   TSerial();
   void Iniciar(TWindow*);
   void Leer(char *);
   void Escribir(char *);
   void Cerrar();};
#endif
```

Listado IV.2.
Archivo serial.h, definición de la clase TSerial.

Definición de la clase TModem

Esta clase hereda los métodos y atributos de TPuerto y TSerial. Además define nuevo comportamiento y atributos como son NumeroTelefonico, Marcar, Colgar, su definición puede observarse en la tabla IV.9.

TModem
Buffer
Estatus
Baud_rate
Puerto
Paridad
Bits_paro
Bits_datos
NumeroTelefonico
Iniciar
Cerrar
Leer
Escribir
Marcar
Colgar

Tabla IV.9
Definición de la clase TModem

El código C++ para esta clase es:

```
#ifndef TMODEM_H
#define TMODEM_H

#include "puerto.h"
#include "serial.h"

class TModem : public TSerial{
  private:
    char NumeroTelefonico[30];
public:
    void Marcar(char *);
    void Colgar(char *);
};

#endif
```

Listado IV.3.
Archivo tmodem.h, definición de la clase TModem.

Debido a que la comunicación se hace a través del puerto serial, la clase TParalelo no se consideró para describirla aquí. Las clases TProceso, TCliente y TServidor quedan de la siguiente manera:

Definición de la clase TProceso

Siguiendo con el análisis de la aplicación y de los posibles objetos, concluimos que sólo resta una categorización, que comienza con el objeto Proceso, que tiene dos subtipos Cliente y Servidor:

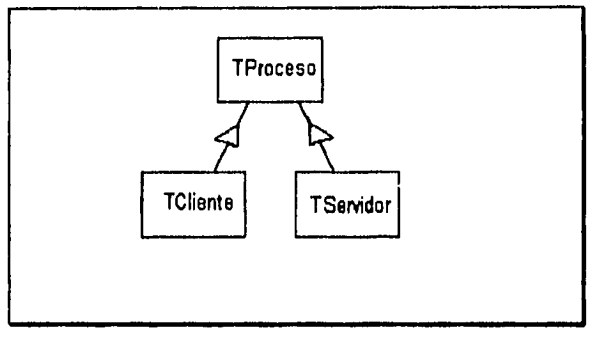


Ilustración IV.10
Categorización del objeto TProceso, TCliente y TServidor

TProceso es la clase general para el comportamiento del proceso cliente y el proceso servidor.

TProceso
Login
Password
Estatus

Tabla IV.10
Definición de la clase TProceso

El código fuente se puede observar en el listado IV.4.

La clase TProceso define tres atributos y ningún método. Servirá como clase base para otras que definimos enseguida.

Definición de la clase TCliente

TCliente
Login Password Estatus
Peticion

Tabla IV.11
Definición de la clase TCliente

La clase TCliente, como su nombre lo indica, se refiere al proceso que va a realizar la petición a un servidor. El método Peticion se encargará de enviar el Login y Password al servidor. Estatus servirá para saber si la conexión y los permisos del usuario son correctos.

El código fuente se puede observar en el listado IV.4.

Definición de la clase TServidor

TServidor
Login Password Estatus
Responder

Tabla IV.12
Definición de la clase TServidor

La clase TServidor se encarga de responder a los servicios que pide el cliente. En la aplicación el servicio es el de generar el reporte general de ventas (ilustración IV.1). El reporte se obtiene desde el servidor con Informix 4GL, con éste creamos una aplicación que genera el reporte y lo envía como respuesta a la petición del cliente. Así se aprovecha la infraestructura ya instalada en el servidor y aprovechamos el modelado de objetos.

El siguiente listado presenta el código fuente para las clases TProceso, TCliente y TServidor.

```
#if !defined(__procesos_h)
#define __procesos_h

#include <owl\owlpch.h>
#pragma hdrstop
#include <stdio.h>
#include <string.h>

struct TClaves{
  char Login[10];
  char Password[10];
};

class _OWLCLASS TProceso{
  public:
        TClaves Claves;
        int estatus;
        TProceso(){
              estatus=0;
              strcpy(Claves.Login,"");
              strcpy(Claves.Password,"");
        }
};

class TServidor : public TProceso{
  public:
        TServidor():TProceso(){};
        void Responder();
};

class TCliente : public TProceso{
  public:
        TCliente() : TProceso(){};
        void Peticion(TClaves);
};

#endif
```

Listado IV.4.
Archivo procesos.h, definición de las clases Tproceso,
TServidor y TCliente.

Todas estas clases sirven para resolver la parte de la comunicación con el modem (TPuerto, TSerial, TModem) y para el control de los procesos cliente y servidor (TProceso, TCliente, TServidor).

Quizá en un futuro clases como TPuerto o TModem se incluyan en esta herramienta como parte de su biblioteca estándar, y si no, nosotros ya creamos las clases necesarias para hacerlo, sólo hay que utilizarlas.

Clases de interfaz con el usuario (frontend)

Una de las clases más importantes de la biblioteca de ventanas que ofrece la herramienta se llama TApplication. Esta clase a su vez es derivada de TModule, TApplication actúa como un controlador basado en objetos para una aplicación Windows, cabe mencionar que Windows aunque es un ambiente basado en objetos no fue desarrollado con un L.B.O., y su **API** ofrece muy poco basado en objetos. TApplication y TModule sustituyen el comportamiento básico requerido por una aplicación Windows. Sus funciones miembro crean: instancias de la clase, la ventana principal y procesan los mensajes (todos éstos son conceptos de programación Windows que no serán tratados aquí).

Creamos una clase que incluyera el comportamiento de TCliente y de TApplication, esta clase es TModemApp, la cual hereda todos los atributos y métodos de ambas clases, su definición es:

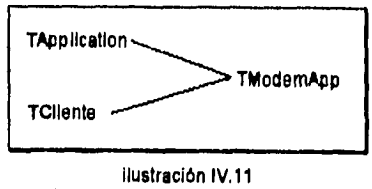


Ilustración IV.11
Categorización del objeto TModemApp

TModemApp
Client
Printer
Estatus
Login
...
SpeedBar
OpenFile
InitMainWindow
InstalaTimer
CmFileNew
...

Tabla IV.13
Definición de la clase TModemApp

Los puntos suspensivos indican que todavía hay más atributos y métodos, pero es una lista muy larga que no consideramos importante incluir, puede ver los atributos completos en el listado IV.5.a, IV.5.b y IV.5.c.

De la clase TModemApp heredamos a una nueva clase, TPlanta, y agregamos un método llamado Iniciar:

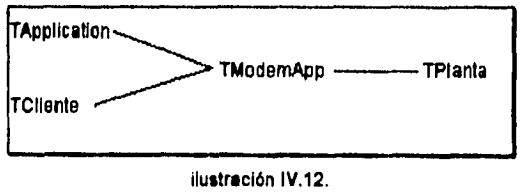


Ilustración IV.12.
Categorización del Objeto TPlanta

TPlanta
Client ...
SpeedBar Iniciar ...

Tabla IV.14
Definición parcial de la clase TPlanta

TPlanta es la clase que inicia la ejecución de la aplicación. Además de TApplication existen otras 100 clases más (aproximadamente) que la herramienta ofrece para el manejo de controles y servicios de Windows.

Para el acceso de datos por medio de cajas de diálogo de Windows es necesario heredar clases de TDialog (parte de la biblioteca de Borland C++), estas clases son TClavesAcceso (para solicitar login y password), TConfigPto (para configurar los parámetros del modem), TInputDialog (para pedir cualquier tipo de dato, como el número telefónico) y TModemAbout (para mostrar el nombre de los que desarrollamos el programa). Puede observarse la jerarquía de estas clases en la ilustración IV.13.

```
#if !defined(__modem_h)
#define __modem_h

#include <owl\owlpch.h>
#pragma hdrstop
#include <owl\statusba.h>
#include <owl\controlb.h>
#include <owl\buttonga.h>
#include <owl\printer.h>
#include <owl\editfile.h>
#include <owl\opensave.h>
#include <owl\edit.h>
#include <owl\listbox.h>
#include "procesos.h"
#include "apxprint.h"
#include "modem.rh"            // Definición de todos los recursos.

class TEditFileFont : public TEditFile{
 private:
  TFont *Letras;
 public:
  TEditFileFont(TWindow *t=0,int Id=0,
          const char far*txt=0):TEditFile(t,Id,txt){
   Letras=new TFont("PCPlus SS",10,7);
  }
  ~TEditFileFont(){
   delete Letras;
  }
  void SetupWindow(void){
   TEditFile::SetupWindow();
   ::SendMessage(HWindow,WM_SETFONT,
                 (WPARAM) HFONT(*Letras),MAKELONG(1,0));
  }
};
```

Listado IV.5.a.
Archivo modem.h, definición de TModemApp y TPlanta.

```
//{{TApplication = TModemApp}}
class TModemApp : public TApplication, public TCliente {
  private:
    TEditFileFont  *Client;
    TOpenSaveDialog::TData FileData;
    BOOL           HelpState;
    BOOL           ContextHelp;
    HCURSOR        HelpCursor;
  private:
    void SetupSpeedBar (TDecoratedFrame *frame);
  public:
    TModemApp ();
    virtual ~TModemApp ();
    void OpenFile (const char *fileName = 0);
    TPrinter       *Printer;
    BOOL           Printing;

//{{TModemAppVIRTUAL_BEGIN}}
  public:
    virtual void InitMainWindow();
    BOOL   InstalaTimer();
    virtual BOOL CanClose ();
    virtual BOOL ProcessAppMsg (MSG& msg);
//{{TModemAppVIRTUAL_END}}

//{{TModemAppRSP_TBL_BEGIN}}
protected:
    void CmFileNew ();
    void CmFileOpen ();
    void CmFileClose ();
    void CmFilePrint ();
    void CmFilePrintSetup ();
    void CmPrintEnable (TCommandEnabler &tce);
    void CmHelpContents ();
    void CmHelpUsing ();
    void CmHelpAbout ();
```

Listado IV.6.b.
Continuación del archivo modem.h, definición de TModemApp y TPlanta.

```
    void EvWinIniChange (char far* section);
    void CmModemConfig ();
    void CmModemMarcar ();
    void CmModemColgar ();
    void CmConsultarVentas ();
    void EvTimer (UINT timerId);
    void CmConsultaClaves ();
    void CmConsultaLogout ();
    void CmConsultaVentasEnable (TCommandEnabler &tce);
    void CmConsultaLogoutEnable (TCommandEnabler &tce);
    void CmConsultaClavesEnable (TCommandEnabler &tce);
//{{TModemAppRSP_TBL_END}}
  DECLARE_RESPONSE_TABLE(TModemApp);
};   //{{TModemApp}}

class TPlanta : public TModemApp{
  public:
    int Iniciar(){return Run();}
};

#endif                          // __modem_h
```

Listado IV.5.c.
Continuación del archivo modem.h, definición de TModemApp y TPlanta.

La herramienta permite observar en diagramas Fern la jerarquía de objetos, la que mostraremos, por partes, señalando los objetos utilizados para la aplicación:

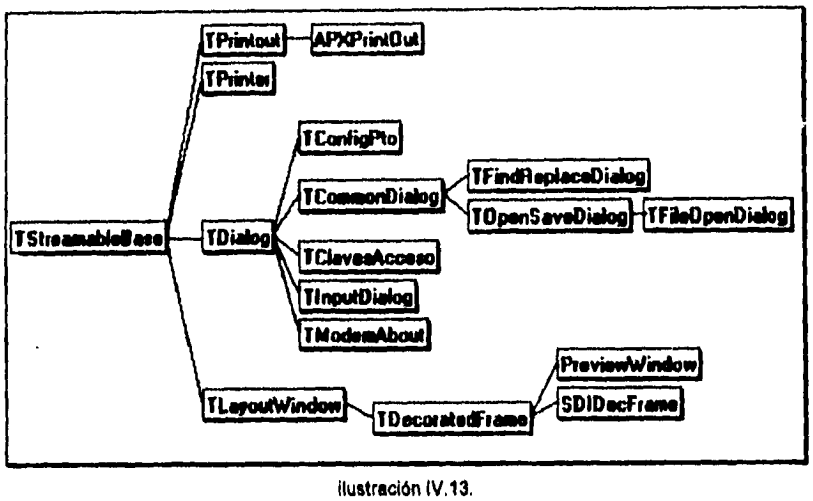


Ilustración IV.13.
Arbol de objetos de interfaz, representado con la herramienta

TStreamableBase junto con TEventHandler son las clases principales, de éstas se derivan bastantes clases, por lo que presentaremos sólo las más representativas. El siguiente diagrama muestra la jerarquía de TPlanta:

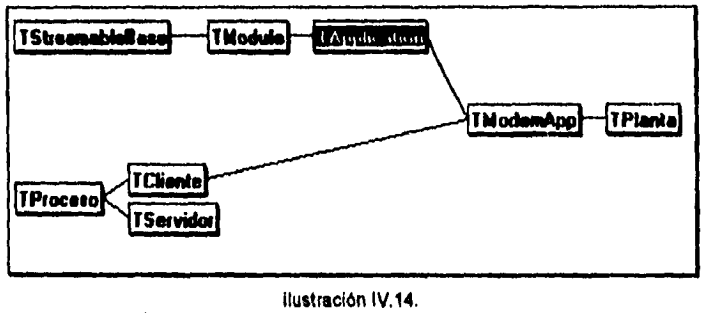


Ilustración IV.14.
Arbol de objetos de interfaz, representado con la herramienta

TEventHandler se encarga de capturar los mensajes de la aplicación y Windows, y los despacha a la función correspondiente. Esta clase es la base para la clase TWindow, que maneja cualquier ventana:

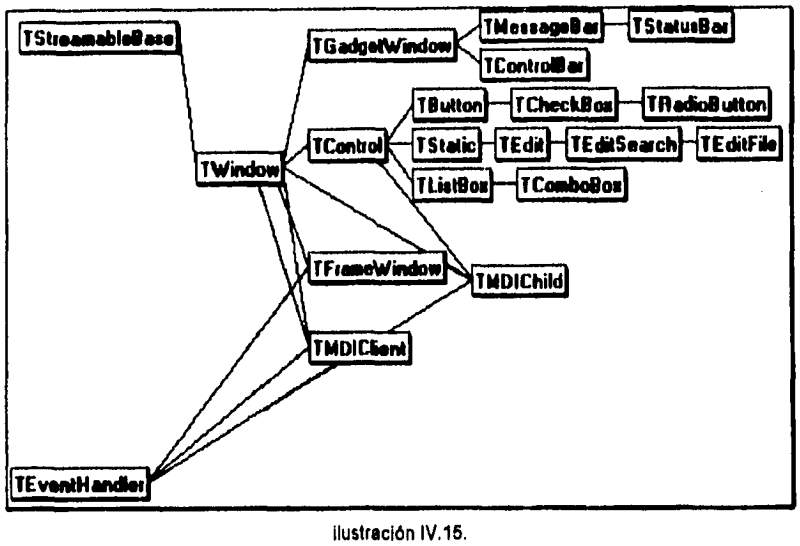


Ilustración IV.15.
Arbol de objetos de interfaz, representado con la herramienta

Hasta aquí se cubre el modelo de objetos. Las clases que creamos (TPuerto, TModem, TCliente, etc.) son el resultado del proceso en espiral para el desarrollo de sistemas, conforme se fue construyendo la aplicación hubo necesidad de retomar el diseño para añadir y quitar tanto métodos como atributos.

La siguiente tabla resume las clases utilizadas, haciendo una división por el tipo de problema que resuelven:

Tipo	Clases
Clases de comunicación	TPuerto TSerial TModem
Clases de control de procesos	TProceso TCliente TServidor
Clases de interfaz	Biblioteca OWL TModemApp TPlanta

Tabla IV.15
Clases del proyecto divididas por tipo de problema que resuelven

Como puede observarse con estos tres tipos de clases se ha implementado un cliente basado en objetos. Actualmente el sistema (SICE) está distribuido en las diferentes plantas, logrando la entrega oportuna del reporte de ventas, y demostrando su utilidad para una mejor toma de decisiones.

El manual de procedimientos está revisándose para ser actualizado de acuerdo a los resultados obtenidos.

El área de sistemas planea continuar con la metodología de objetos para seguir desarrollando aplicaciones para Electropura, aprovechando la experiencia obtenida y mejorando lo realizado.

CONCLUSIONES

1. La arquitectura cliente-servidor es una opción a las empresas que requieran una mejor distribución de sus procesos de información tomando en cuenta los recursos de cómputo con los que cuentan.

2. Se debe de llevar a cabo un análisis profundo en la empresa donde se quiera implantar cliente-servidor y tener bases sólidas para la realización de un proyecto con esta arquitectura.

3. El beneficio que se obtiene de llevar a cabo un desarrollo con la arquitectura cliente-servidor se refleja gradualmente con el transcurso del tiempo a mediano plazo.

4. Cliente-servidor es una arquitectura que se está utilizando cada vez más en las empresas, no por ser una moda sino porque se está reflejando en los resultados de operación de las mismas.

5. Cliente-servidor ayuda a la comunicación y transferencia de información en lugares distintos de una misma empresa.

6. La programación basada en objetos facilita la reutilización de código, que permite el desarrollo de sistemas más rápido.

7. Para implementar el paradigma basado en objetos se requiere una mayor difusión del mismo en todo el ámbito informático.

8. El desarrollo con el paradigma basado en objetos junto con la arquitectura cliente-servidor permite la mejor utilización de ambas herramientas.

9. Como puede apreciarse, en el desarrollo de la tesis los objetivos planteados desde el inicio fueron cumplidos satisfactoriamente.

10. Los procedimientos administrativos son base para la elaboración de un sistema que automatice parte o en su totalidad los procesos de la empresa.

11. La comunicación cliente-servidor no solamente puede realizarse por medio de línea telefónica, pues existen tecnlogías como radio, microondas, etcétera, que no se utilizó en la presente tesis por razones de costo e infraestructura de la empresa.

12. La realización de sistemas a través de cliente-servidor y el paradigma de objetos, facilita el desarrollo de aplicaciones que dan solución al procesamiento de información.

Apéndice A. Mapa de la Infraestructura Cliente-Servidor

Durante nuestra investigación encontramos diferentes productos que cumplen con ciertas características de los elementos de la arquitectura cliente-servidor, discutida en el capítulo I, a continuación presentamos la división de éstos:

A.1 Productos para el cliente

INTERFACES GRAFICAS

• Windows	• Motif
• Macintosh	• Open Look
• Presentation Manager	• NeXTSTEP
	• Wabi

SISTEMAS OPERATIVOS

• MS-DOS	• AIX
• Windows NT	• A/UX
• Windows Chicago	• HP-UX
• Windows NT Cairo	• Mach
• OS/2	• OSF/1
• Workplace OS	• SCO-ODT
• Taligent	• Solaris
• Macintosh	• SVR4
• DR-DOS	• Ultrix
	• UnixWare

HARDWARE

• IX86, Pentium	• Alpha
• PowerPC	• 68xx
• SPARC	• MIPS
	• PA-RISC

PROCESADORES DE PALABRAS

• Ami Pro	• MacWrite
• Microsoft Word	• DeScribe
• WordPerfect	• JustWrite
• WordStar	• Professional Write Plus

HOJAS DE CALCULO

• Lotus 1-2-3	• Improv
• Excel	• Wingz
• Quattro Pro	• SpreadBase

DIBUJO Y PINTURA

• CorelDRAW	• Aldus Freehand
• Harvard Graphics	• CorelPAINT
• Micrographics Designer	• Adobe Photoshop
• Adobe Ilustrator	• Canvas
•	• PhotoStyler

PRESENTADORES GRAFICOS Y DESKTOP PUBLISHING

• Frame Maker	• Quark Xpress
• PageMaker	• Freelance Graphics
• Corel Ventura	• Persuasion
• Microsoft Publish	• Power Point
•	• Compel

COMUNICACIONES

• Crosstalk	• ProComm
• LapLink	• HyperACCESS
• Relay Gold	• Carbon Copy
•	• Norton pcANYWHERE

A.2 Productos para la red

MIDDLEWARE

• Remote Procedural Call (RPC)
• Message-Oriented Middleware
• Object Request Broker (ORB)
• Conversational
• Remote Database Access
• Message Routing Services
• Directory Services
• Time Services
• Terminal Services
• IDAPI
• ODBC
• Security Services

TRABAJO DE LA RED

• Protocolos
• Networks
• Wiring
• Inter-Network

MANEJADORES DE LA RED

• Sistemas Administradores
• Protocolos de Sistemas Administradores
• Manejadores de Red

A.3 Productos para el Servidor

SISTEMAS OPERATIVOS DE LA RED

• NetWare	• VINES
• LAN Manager	• Pathworks
• LAN Server	• Apple Share
•	• PC-NFS

SISTEMAS OPERATIVOS

• NetWare	• Macintosh
• Windows NT	• AIX
• OS/2	• HP-UX
• OS/400	• OSF/1
• MVS	• SCO Unix
• VM	• Solaris
• VMS	• SVR4
• MPE\IX	• Ultrix
•	• UnixWare\

HARDWARE

• iX86, Pentium	• AS/400
• 68xx	• PowerPC, Power
• VAX	• SPARC
• HP 3000	• Alpha
• System/390	• PA-RISC
•	• MIPS

A.3 Productos para el Servidor

SISTEMAS OPERATIVOS DE LA RED

• NetWare	• VINES
• LAN Manager	• Pathworks
• LAN Server	• Apple Share
•	• PC-NFS

SISTEMAS OPERATIVOS

• NetWare	• Macintosh
• Windows NT	• AIX
• OS/2	• HP-UX
• OS/400	• OSF/1
• MVS	• SCO Unix
• VM	• Solaris
• VMS	• SVR4
• MPE\iX	• Ultrix
•	• UnixWare\

HARDWARE

• iX86, Pentium	• AS/400
• 68xx	• PowerPC, Power
• VAX	• SPARC
• HP 3000	• Alpha
• System/390	• PA-RISC
•	• MIPS

RDBMS

• Oracle	• DB2, DB2/2, DB2/6000
• Microsoft SQL Server	• SQL/DS
• Sybase SQL Server	• OS/400 DBM
• Informix	• XDB
• Ingres	• Watcom SQL
• SQL Base	• Rdb
• InterBase	• Allbase/SQL
• CA-DB, IDMS, DATACOM	• Teradata DBC
• Progress	• Supra
•	• ADABAS

ODBMS

• ObjectStore	• Servio Gemstone
• Objectivity/DB	• HP OpenODB
• Versant	• UniSQL/M
• Ontos	• OS
• Matisse	• Ilustra
•	• Poet

MANEJADORES DE TRANSACCIONES

• Tuxedo	• CICS
• Encima	• Integris UniKix
• Top End	• VIS/TP
•	• ACMS

WORK FLOW

• WorkFlo	• Flowlogic
• Plexus Floware	• FloWareSIS Route Builder
• IBM Image Plus	• Wang OPEN/image
• Folio Views	• Reach Workman
• Image Enabled NetWare	• ViewStar
•	• Xorox in Concert

A.4 Herramientas de desarrollo profesional

GUI 4GLs

• Microsoft Visual Basic	• Oracle Forms	• INM VisualAge
• Powersoft PowerBuilder CODE	• Sybase Gain Momentum	• Sapiens Vision
• Gupta SQL Windows	• Ingres Windows4GL	• JYACC JAM
• KnowledgeWare ObjectView	• Unify Vision	• Magic
• Uniface	• Wang Labs PACE	• Gradient Visual DCE
• Progress	• Smaltalk/V	• ESL Workbench
• Mozart	• Bachman Ellipse	• Dynasty
• KnowledgeWare Flashpoint	• Magna X	• Forte
• Kaseworks KASE:VIP	• Seer HPS	• Cognos Axiant, Power House
• Blyth Omnis 7	• VisualWorks	• Intelligent Environments Aplication Manager

CASE

• Andersen Foundation for Cooperative Processing	• HP SoftBench	• Seer HPS
• Texas Instruments IEF	• IBM AD/Platform SDE Workbench	• Softlab
• KnowledgeWare	• Bachman Model Driven Development	• SYNON
• Oracle CDE CASE	• LBMS System Engineering	• Objectoy
• Sybase Enterprise Momentum	• Cadre Teamwork Object TEAM	• Performance Architek
• CGI PACBASE	• Intellicorp Object Management Workbench	• Visible Systems Workbench
• Intersolv APS	• Logic Works ERwin/SQL	• SES/Workbench
• DEC Cohesion, FUSE	• Popkin System Architect	•

X-Windows/Portable GUI Front-Ends

• Aisys TeleUSE	• Netron/Client GUI-Builder	• V.I Corp. X-Designer
• HP Interface Architect	• Neuron Data Open Interface	• Visix Galaxy
• ICS Builder Xcessory	• NLS XFaceMaker	• Visual Edge UIM/X
• ImageSoft Object/Designer	• Open Aspect	• Zinc Designer
• Liant C++/Views	• XVT Portability Toolkit	•

A.5 Herramientas para usuarios finales

DBMS Personales

• AceFile	• Claris FileMaker Pro	• Microrim R:BASE
• Acius 4th Dimension	• DataErase Express	• Microsoft Access, FoxPRO
• Borland Paradox, ObjectVision, dBASE IV	• Kedwell Software DATABOSS	• Powersoft PowerMaker
• CA-dBFast, CA-Clipper, CA-Visual Objects	• Lotus Approach	• SPC Superbase

DSS/EIS

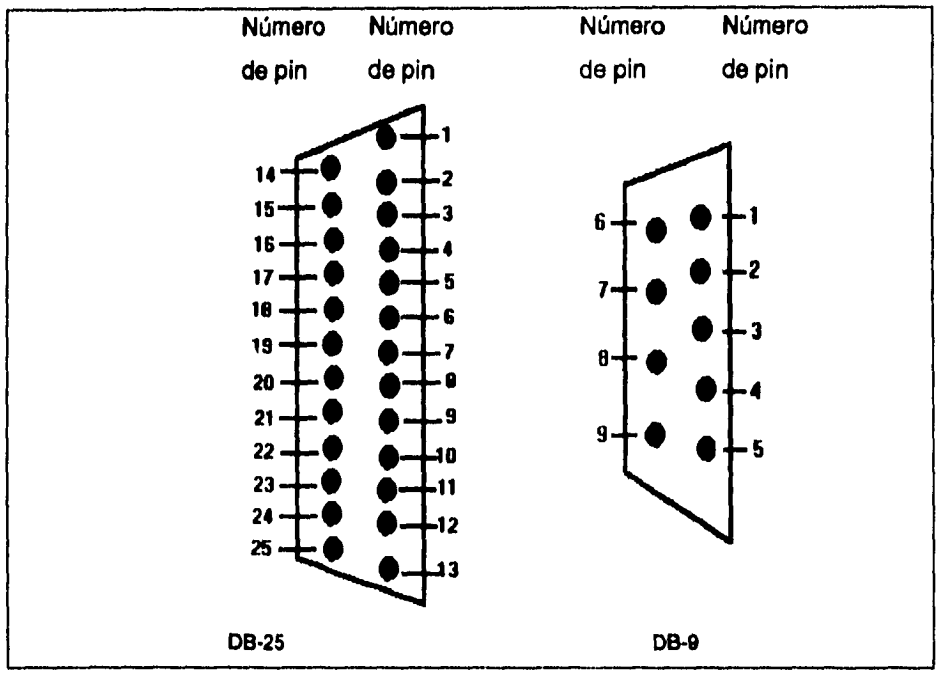
• ReportSmith	• DEC Query	• Must Nomad
• Business Objects	• Gupta Quest	• Natural Language
• Cognos PowerPlay, Impromptu	• IBI Focus, Level 5 Object	• OpenBooks
• Comshare Prism, Commander	• Iconic Query	• Oracle CDE Card, Browser, Reports, Graphics
• Anadyne GQL	• Informix ViewPoint	• Powersoft PowerViewer
• Brio DataPrism, DataPivot	• IQ Access, Intelligent Query	• Q+E
• ClearAccess	• Gentium	• SAS
• Concentric R&R Report Writer	• Arbor Essbase	• Spinnaker Personal Access
• Coromandel QBE Vision	• IRI Express MDB	• SPSS
• Cross Access	• Pilot Lightship	• ANSWER: Journey
• Crystal Reports	• Metaphor DIS	• Systat
• Trinzic Forest & Trees	• Control +	•

Workgroup

• Action Technologies Action Workflow System	• D&B SmartStream	• NCR Cooperation
• Applix, ApplixWare	• DEC ObjectWorks	• Option Technologies OptionFinder
• Borland OBEX	• Enable software Higgins	• Oracle Office
• Collaborative Technologies VisionQuest	• IBM FlowMark	• Reach Software Workman
• Corporate Memory Systems CM/1	• Lotus Notes	• Ventana GroupSystems V
• WorkPerfect Office	•	•

Apéndice B. Asignación de Pines del RS-232C

Funciones de los pines de la interfaz RS-232-C:



Pin 1 Tierra física: el conductor está conectado eléctricamente al chasis del equipo.

Pin 2 Datos transmitidos: señales de datos que se transmiten desde el DTE hasta el DCE. Estas son las señales que representan los datos de usuario propiamente dichos.

Pin 3 Datos recibidos: señales de datos de usuario que se transmiten desde el DCE hasta el DTE.

Pin 4 Petición de transmisión (RTS-Request to send); señal dirigida desde el DTE hasta el DCE. Este circuito notifica al DCE que el DTE dispone de datos para transmitir. El circuito CA se emplea también en líneas semidúplex para controlar el sentido de las transmisiones de datos.

Pin 5 Permiso para transmitir (CTS -Clear to send); señal procedente del DCE, con la que se indica al DTE que ya puede transmitir sus datos. La señal CTS puede activarse (ON) al recibir una señal portadora en línea procedente del modem remoto.

Pin 6 Equipo de datos preparado (DSR -Data Set Ready); señal procedente del DTE, con la que se indica una de las siguientes condiciones: (a) que la máquina está "descolgada", es decir, conectada al canal de una línea conmutada , (b) que el DCE está en modo de transmisión de datos, (c) que el DCE ha completado las funciones de sincronización y responde con tonos.

Pin 7 Tierra lógica, común a todos los circuitos. Establece la referencia del potencial eléctrico para el resto de los pines. En realidad no tiene nada que ver con una tierra verdadera; sólo es un circuito de referencia común.

Pin 8 Detector de recepción de señal en línea; señal procedente del DCE, con la que se indica que éste ha detectado la señal portadora generada por el modem remoto.

Pin 15 Temporización del elemento de señal de transmisor; señales procedentes del DCE que proporcionan la temporización a las señales de datos que estén siendo transmitidas hacia el DCE a través del circuito BA (Datos Transmitidos).

Pin 17 Temporización del elemento de señal del receptor; señales procedentes del DCE que proporcionan al DTE la temporización necesaria para las señales de datos que estén siendo recibidas por el circuito BB (Datos Recibidos).

Pin 20 Terminal de datos preparado (DTR -Data Terminal Ready); señal procedente del DTE, con la que se indica que la terminal u ordenador están encendidos, que no se detecta ningún indicio de mal funcionamiento y que no se encuentra en modo de pruebas.

Pin 21 Detector de calidad de la señal; señal procedente del DCE, con la que se indica que la señal recibida tiene la calidad suficiente para suponer que no ha aparecido ningún error.

Pin 22 Indicador de timbre (RI -Ring Indicator); señal procedente del DCE, con la que se indica que se está recibiendo una señal de timbre a través de un canal conmutado.

Pin 23 Selector de velocidad binaria de la señal; señales procedentes del DTE y del DCE, respectivamente, que indican la velocidad de señalización de los datos, en las máquinas dotadas de velocidad dual. Algunos dispositivos son capaces de transmitir a velocidades binarias variables.

Pin 24 Temporización del elemento de señal del transmisor; señales procedentes del DTE que proporcionan la temporización a las señales de datos que estén siendo transmitidas por el circuito BA (Datos Transmitidos) hacia el DCE. El DTE se encarga de generar esta señal; si el DCE es el que genera el sincronismo, el circuito utilizado es el DB.

Además de esos circuitos, en RS-232-C, se definen otros cinco circuitos designados como canales secundarios: SCA, SCB, SCF, SBA y SBB. Los circuitos restantes se emplean para funciones de prueba y para otras misiones que dependen del fabricante, o simplemente no se utilizan.

Apéndice C. Algoritmo del Programa Servidor y Diagrama de Base de Datos

Principal.

1 Recibir parámetro de fecha de reporte en variable dia_venta.

2 Abrir base de datos.

3 Seleccionar de nombre de la planta purificadora.

4 Llamar a función Busqueda_en_Tablas.

5 Cerrar base de datos.

Termina Principal.

Busqueda_en_Tablas.

1 Seleccionar de cantidad total de garrafones utilizados en el día.

2 Seleccionar de cantidad total de garrafones utilizados al día anterior de la venta.

3 Seleccionar de los saldos Cantidad_Pagada y Cantidad_a_Pagar.

4 Obtener diferencia de Cantidad_Pagada y Cantidad_a_Pagar.

5 Llamar a Busqueda_de_Rotos.

6 Seleccionar el detalle de la venta de todas las liquidaciones del día de la venta.

7 Mientras existan liquidaciones del día

- Separar los conceptos
- Importe de Efectivo.
- Importe de Bonificaciones.
- Importe de Comodatos.
- Importe de Créditos.
- Cantidad gravada.
- Importe en cheques.

8 Seleccionar

- Clave de producto.
- Descripción del producto.

9 Sumar de total de ventas.

10 Sumar el total de ventas por precio del artículo.

11 Llamar Salida_de_Reporte.

Termina Busqueda_en_Tablas.

Busqueda_de_Rotos

1 Seleccionar la suma total de garrafones rotos de todas las liquidaciones

2 Multiplicar el total de garrafones rotos por el precio del garrafón.

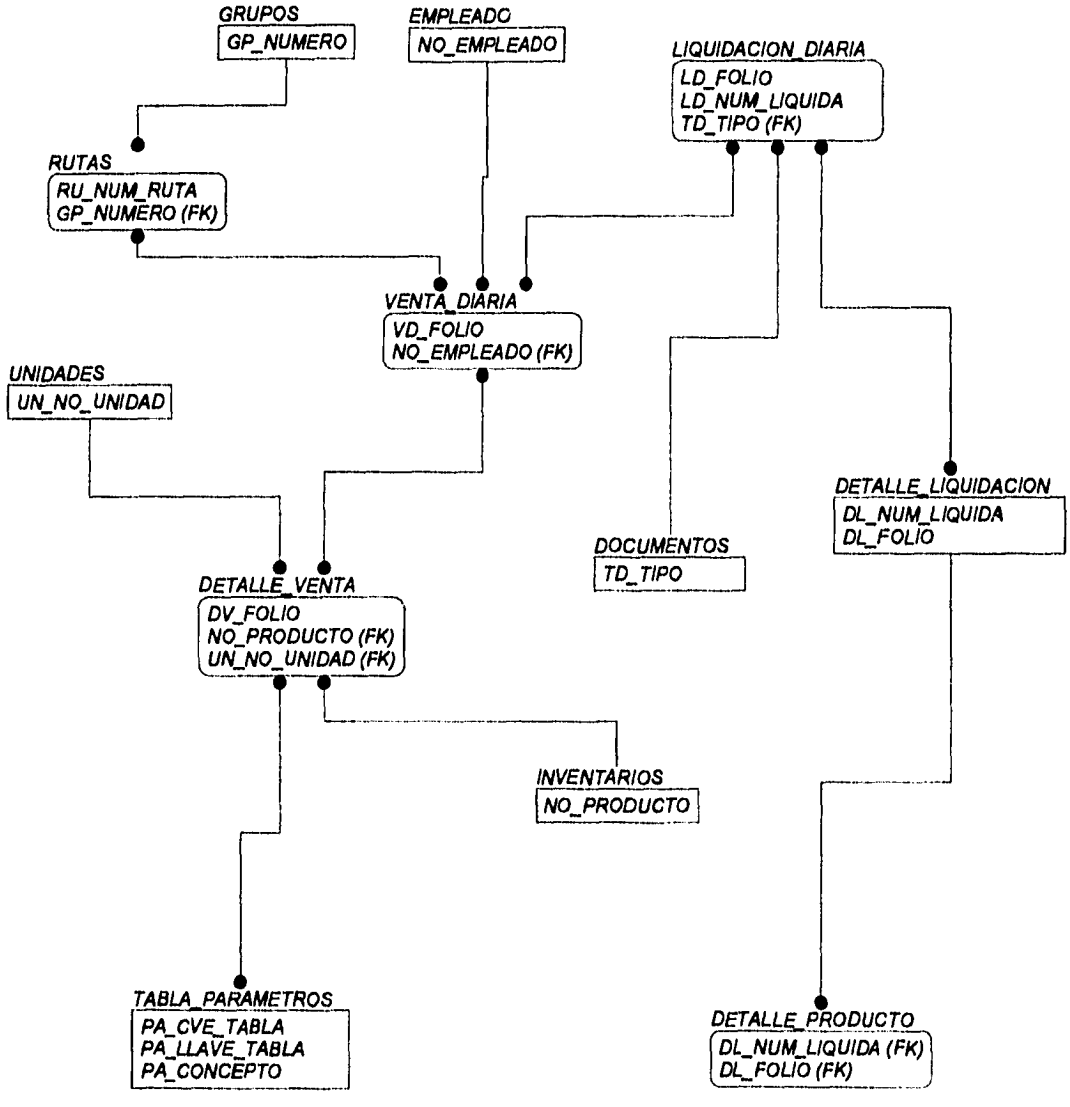
Termina Busqueda_de_Rotos.

Salida_de_Reporte

1 Imprimir

- Fecha del reporte.
- Datos de la planta.
- Cantidad vendida.
- Saldo de garrafón.
- Efectivo recibido.
- Documentos de crédito.
- Bonificaciones.
- Resumen de salidas y entradas a la planta.
- Resumen de análisis de productos e importe.

Termina Salida_de_Reporte


GRUPOS
GP_NUMERO
EMPLEADO
NO_EMPLEADO
LIQUIDACION_DIARIA
LD_FOLIO
LD_NUM_LIQUIDA
TD_TIPO (FK)
RUTAS
RU_NUM_RUTA
GP_NUMERO (FK)
VENTA_DIARIA
VD_FOLIO
NO_EMPLEADO (FK)
UNIDADES
UN_NO_UNIDAD
DETALLE_LIQUIDACION
DL_NUM_LIQUIDA
DL_FOLIO
DOCUMENTOS
TD_TIPO
DETALLE_VENTA
DV_FOLIO
NO_PRODUCTO (FK)
UN_NO_UNIDAD (FK)
INVENTARIOS
NO_PRODUCTO
TABLA_PARAMETROS
PA_CVE_TABLA
PA_LLAVE_TABLA
PA_CONCEPTO
DETALLE_PRODUCTO
DL_NUM_LIQUIDA (FK)
DL_FOLIO (FK)

A. GLOSARIO DE TERMINOS INFORMATICOS

ANSI:

Abreviación de "American National Standards Institute". Una institución voluntaria que ayuda a definir estándares, y que también representa a los E.U. en la Organización Internacional de Estándares (ver ISO).

API:

Siglas de "Application Program Interface". En general, todo el grupo de funciones o procedimientos, que se invocan desde un programa de aplicación, para utilizar un software de base. Por ejemplo: API's para OS/2, API's para un cierto gateway, etc.

ASCII:

Siglas de "American Standard Code for Information Interchange". Forma estándar de codificar los caracteres en un patrón de 7 bits. El ASCII extendido utiliza 8 bits y logra codificar 256 patrones en lugar de 128.

Asíncrona:

Forma de transmisión que no requiere que el receptor y el transmisor mantengan en "sincronía" sus relojes. Pero en cambio necesita que el transmisor "inserte" bits antes y después del caracter para que el receptor lo reconozca. Es más barata que la transmisión síncrona, pero menos eficiente.

BackEnd:

En general, software o hardware que actúa sin ser visto en una arquitectura cliente-servidor. En un manejador de Bases de Datos (DBMS) se denomina así a la parte del software, generalmente ubicada en el servidor, que se encarga de seleccionar, controlar, ordenar, indexar y administrar la información.

Batch:

Un método de procesamiento de datos en donde todos los trabajos se agrupan primero, para después enviarse en forma secuencial a la computadora para su proceso.

Baudio(baud):

Medida de velocidad de transmisión de datos. La velocidad en baudios es igual al número de veces que cambia la condición de la línea por segundo. A velocidades bajas, los baudios y los bits por segundo, son lo mismo. Sin embargo, cuando la velocidad aumenta, por cada baudio son codificados varios bits, por lo que dejan de ser sinónimos.

BIOS:

Siglas de "Basic Input/Output System". Servicios de software y/o firmware que definen la forma en que interactúan las aplicaciones y todos los puertos seriales y paralelos de entrada/salida.

Bit:

Un dígito binario; puede ser 0 o 1. Es la unidad más pequeña de información que indica dos estados "off (0), on(1)".

Bit de paridad:

Método sencillo para detectar errores en la transmisión. Se agrega un bit en 0 o 1 dependiendo del número de unos que tenga el patrón a enviar.

Boot:

Proceso de carga de los programas básicos para encender la computadora.

Bps:

Abreviación de bits por segundo. La medida de velocidad de transmisión más utilizada. En redes locales lo más frecuente es hablar de Mbps (Mega bits por segundo). Es importante hacer notar que la abreviación de bit es una b minúscula, mientras que la Byte es una B mayúscula.

Buffer:

Es un espacio donde se almacenan datos temporalmente mientras se les puede enviar a su destino final.

Bus:

Es un circuito de transmisión eléctrica que sirve para transportar información entre varios dispositivos de una computadora.

Byte:

Conjunto de 8 bits que representan un caracter en binario.

Cable coaxial:

Un tipo de cable eléctrico en el cual un alambre sólido de metal es cubierto por un aislante, todo lo cual es protegido por una malla de metal cuyo eje de curvatura coincide con el alambre de ahí el nombre de coaxial (eje común).

Cable Null Modem:

Un cable RS-232C en el cual las señales 2 y 3 están invertidas, haciendo ver a las dos computadoras a las cuales conecta, como si transmitiera a través de modems.

Canal:

Un camino físico o lógico que permite la transmisión de información. En algunos casos puede ser sinónimo de Bus.

CASE:

Siglas de "Computer Aided Software Engineering". La utilización de software para ayudar en la definición, elaboración, documentación y algunas otras áreas del desarrollo de programas.

CCITT:

Siglas de "Comité Consultivo Internacional de Telegrafía y Telefonía. Fija estándares internacionales en comunicaciones. Se encuentra ubicado en Ginebra, Suiza.

Circulto de conmutación:

Es una técnica de telecomunicaciones que dinámicamente establece una conexión física antes de un intercambio de información, presentando la conexión siguiendo el intercambio.

Conectividad:

Estado que permite la transferencia de señales eléctricas desde un origen hasta un destino.

Conector:

Es un accesorio al final de un alambre o conjunto de alambres que facilitan su conexión a un recurso.

CPU:

Siglas de "Central Processing Unit". Generalmente se utiliza este término para definir el Procesador Central de una computadora. Es la base de una computadora digital.

CRC:

Siglas de "Cyclic Redundancy Check". Código de detección de errores. Se basa en realizar una división del patrón a enviar entre un número binario de X bits

(polinomio). El residuo de la división lo pega al número. Del lado del receptor se realiza la operación contraria y se verifica si los bits han llegado correctamente.

CSA:

Siglas de "Client-Server Arquitecture". Arquitectura cliente-servidor. Modelo de trabajo donde el proceso se distribuye entre dos o entidades: la que solicita (cliente) y la que da el servicio (servidor).

Data link, Nivel de:

Nivel 2 del modelo OSI. En este nivel se arman los "frames" y se verifican errores de transmisión (usualmente a través del código CRC).

DB2:

Manejador de bases de datos de IBM para ambiente MVS (mainframes). Utiliza SQL, y define en sí mismo un dialecto estándar.

DB9:

Conector de 9 pines, utilizado para la interfaz RS-232C,. También se utiliza para cables STP en Token Ring.

DB25:

Conector de 25 pines, el más utilizado para la interfaz RS-232C, por lo que se considera un estándar de facto.

DCE:

Siglas de "Data Communications Equipment". En la terminología común es sinónimo de modem. Más formalmente DCE es el equipo que se coloca entre los dispositivos terminales (DTE) y la red.

DOS

Abreviación de Disk Operating System, sistema operativo en disco.

Downsizing

El proceso conocido como downsizing consiste en migrar aplicaciones completas o funciones de estas, de mainframe centralizados a redes o sistemas centralizados menores, buscando así acercar la aplicación al usuario final. Toma ese nombre de las técnicas de "adelgazamiento" de los sistemas modernos de producción: "todo lo que no da valor agregado a la producción debe eliminarse", esto es lo que se llama producción esbelta y que surge desde 1970.

Drive:

Dispositivo de almacenamiento en el cual los datos son escritos y leídos desde un disco o una cinta. Un drive que es físicamente agregado a una estación de trabajo es llamado drive local.

DTE:

Siglas de "Data Terminal Equipment". Las PC´s y las estaciones de trabajo son ejemplos de DTE´s. Normalmente utilizadas junto con DCE´s y líneas de transmisión.

EBCDIC:

Siglas de "Extended Binary Coded Decimal Interchange Code". Método de IBM para codificar caracteres en una forma binaria.

EIA:

Siglas de "Electronics Industries Association". Institución que elaboró el estándar de comunicaciones RS-232C. Se encuentra ubicada en Washington, USA.

DOS

Abreviación de Disk Operating System, sistema operativo en disco.

Downsizing

El proceso conocido como downsizing consiste en migrar aplicaciones completas o funciones de estas, de mainframe centralizados a redes o sistemas centralizados menores, buscando así acercar la aplicación al usuario final. Toma ese nombre de las técnicas de "adelgazamiento" de los sistemas modernos de producción: "todo lo que no da valor agregado a la producción debe eliminarse", esto es lo que se llama producción esbelta y que surge desde 1970.

Drive:

Dispositivo de almacenamiento en el cual los datos son escritos y leídos desde un disco o una cinta. Un drive que es físicamente agregado a una estación de trabajo es llamado drive local.

DTE:

Siglas de "Data Terminal Equipment". Las PC's y las estaciones de trabajo son ejemplos de DTE's. Normalmente utilizadas junto con DCE's y líneas de transmisión.

EBCDIC:

Siglas de "Extended Binary Coded Decimal Interchange Code". Método de IBM para codificar caracteres en una forma binaria.

EIA:

Siglas de "Electronics Industries Association". Institución que elaboró el estándar de comunicaciones RS-232C. Se encuentra ubicada en Washington, USA.

EOT:

Siglas de "End of Transmision". Se utiliza en protocolos orientados a byte (byte-oriented).

ETCD:

Equipo de terminación de circuitos de datos. Su misión consiste en transformar las señales portadoras de la información a transmitir, utilizadas por los ETD, en otras que conteniendo aquella misma información, más alguna adicional de uso exclusivo entre ambos ETCD, sean susceptibles de ser enviadas hasta el ETD distante, mediante los medios de telecomunicación clásicos.

ETD:

Equipo terminal de datos. Cumple dos funciones básicas: ser fuente o destino final de los datos y controlar la comunicación.

FAX:

Texto o gráficas transmitidas vía líneas de comunicación a un punto remoto donde un original es reproducido. La transmisión puede ser análoga o digital. Existen tarjetas para integrar este servicio a una red local.

Firmware:

Conjunto de programas requeridos para implementar una función específica. Estos programas se encuentran almacenados en ROM. (Memoria que sólo permite leer).

Frame:

Unidad de información del nivel 2 del modelo OSI. Usualmente un frame consta de tres partes: un header (o encabezado) que trae información de control, direcciones fuente y destino, etc. Un campo de información y un campo de CRC (verificación de errores).

Frecuencia:
Número de ciclos por unidad de tiempo. Normalmente medida en Hertz (Hz), que son ciclos por segundo.

Frecuencia Modulada:
Proceso en donde se varía la frecuencia de una señal analógica para poder transportar información digital. FM es el método de modulación que más se utiliza en modems diseñados para utilizar líneas telefónicas analógicas.

FrontEnd:
En ambientes de bases de datos, el software que le presenta la información al usuario (Reside en la estación de trabajo).

FrontEnd Processor:
Dispositivo encargado de "lidiar" con todas las comunicaciones, descargando así de trabajo al procesador central (CPU). En IBM se denomina Communication Controller.

FTP:
Siglas de "File Transfer Protocol". Un servicio de alto nivel bajo ambiente TCP que permite y controla el proceso de transferencia de archivos a través de una red.

GUI:
Siglas de "Graphical User Interface". Enlace de comunicación o interfaz entre un usuario y el sistema operativo de una computadora. Generalmente utiliza pantallas diseñadas con base en íconos (figuras) que representan las funciones disponibles para el Usuario. Windows de Microsoft es un ejemplo de un GUI.

HalfDuplex:
Forma de transmisión en la que ambos extremos del sistema de comunicación pueden transmitir pero no simultáneamente.

Hardware:

Equipo físico. Todos los componentes electrónicos y mecánicos de una red, como computadoras personales, periféricos, tarjetas de red y cables.

Host:

Comúnmente sinónimo de mainframe. En la terminología es cualquier nodo en una red, capaz de brindar algún servicio.

Integridad:

Característica de la información de reflejar datos congruentes con la realidad.

Integridad de Entidad:

Regla por la que cada entidad en un archivo, debe ser reconocida de manera única. Un buen manejador de base de datos (DBMS) debe observar esta regla.

Interoperabilidad:

Proceso donde las computadoras pueden operar interactuando con otras a través de una red sin conversión de datos o intervención humana.

IPX:

Protocolo "puerto-a-puerto", propio de Novell, que actúa en el nivel 3 del Modelo OSI (Nivel de RED). Entre sus ventajas está el tener direcciones de tres campos: nodo, red y socket, que le permiten tener enlaces entre redes y varios procesos corriendo en diferentes servidores. Está basado en el protocolo de nivel 3 de XNS.

ISA:

Siglas de "International Standard Arquitecture". Tipo de canal de 8 o 16 bits en la arquitectura de motherboards, tarjetas, etc.

ISO:

Siglas de "International Standards Organization". Institución internacional que se encarga de especificar estándares en diversas áreas.

ISO 2110:

Comunicación de datos - conector de interfaz de 25 líneas entre DTE y DCE, y asignación de las líneas.

LAN

Local Area Network. Red de área local, por lo general es la red que se encuentra en un mismo edificio.

Login:

Acción de entrar a utilizar un host o un servidor de red, establecer una sesión de trabajo y ser reconocido como usuario por el Sistema Operativo.

Mainframe:

Unidad principal. Suele utilizarse para identificar una computadora "grande" (capaz de ser la estructura principal de una instalación de informática).

Microsegundo:

Una millonésima de segundo.

Milisegundo:

Una milésima parte de un segundo. Se representa por la abreviación ms.

Modem:

Yuxtaposición de Modulador/Demodulador. Dispositivo que convierte señales digitales desde una terminal (o PC) a una señal adecuada para transmitirse en un

canal telefónico (analógico). En el otro extremo, otro modem reconvierte la señal analógica en digital, y la transmite a la computadora de ese extremo.

MOS:

(Metal Oxide Semiconductor): designa una clase de transistores realizados con un metal, un óxido y un semiconductor (silicio).

MVS:

Siglas de "Multiple Virtual Storage". Sistema operativo de IBM, el cual optimiza operaciones en línea, tiempo real, multiusuario y multitarea.

Nanosegundo:

Milmillonésima de segundo. Para su representación se utiliza la nomenclatura ns.

ODBMS

Object Database Management System. Sistema manejador de bases de datos basadas en objetos.

OSI

Siglas de "Open Systems Interconnect". Estructura lógica y estándar de 7 niveles de protocolos definida por ISO para facilitar la comunicación en ambientes heterogéneos

Path:

Nombre asignado a la ruta tomada por un paquete de un punto hacia otro dentro de una red.

Protocolo:

Conjunto de reglas convencionales, utilizado para comunicar dos dispositivos de la misma naturaleza.

RAM:

Siglas de " Random Acces Memory". Memoria que puede ser escrita y leída de manera dinámica. Puede ser accesada por el usuario en cualquier punto con facilidad y sin tener que leer grabaciones anteriores.

RDBMS:

Relational Database Management System. Sistema administrador de base de datos relacionales.

Repetidor:

Dispositivo que retransmite y amplifica la señal recibida. Actúa solamente en el nivel I del modelo OSI.

Robustez

Característica que se le dá a los programas que están diseñados de manera que resisten modificaciones sin perder su escencia.

ROM:

Siglas de "Read Only Memory". Memoria no-volátil que puede ser leída pero no modificada.

RS-232C:

Interfaz estándar para conectar un DTE a un DCE. La especificación técnica ha sido publicada por la EIA. Tradicionalmente usa 25 pines (o patas).

SEND/RECEIVE

Señales de envío y recepción de datos.

Serial (Interfaz):
Interfaz electrónica entre un dispositivo receptor o transmisor y un canal de transmisión simple.

Servidor:
Dispositivo de hardware y rutina de software que provee uno o más servicios predefinidos a una población de entidades usuarias, tales como nodos en una red.

Sistema Operativo:
Conjunto homogéneo de programas, concebido para permitir la utilización racional y cómoda de un sistema de proceso de datos.

SNA:
Siglas de "Systems Network Architecture". La arquitectura de protocolos para redes creadas por IBM.

SOKET:
Son direcciones de transporte a través de las cuales ocurre la comunicación. Son definidos en la capa de transporte del modelo OSI.

SPOOL:
Siglas de "Simultaneous Peripheral Operations On Line". Comúnmente, el software encargado de controlar las colas de espera en una impresora. Utilería del Sistema Operativo.

SQL:
Siglas de "Structured Query Language". El lenguaje de consulta y acceso a base de datos más común en la actualidad, definido como estándar por IBM, ANSI e ISO.

SQL-Server:

Servidor de Bases de Datos desarrollado por Microsoft y Sybase. Posee características sumamente poderosas en manejo de transacciones, integridad de la información y control de concurrencia.

STP:

Siglas de "Shielded Twisted Pair". Cable de par torcido de alambre con protección eléctrica adicional, enrollado alrededor de conductor, normalmente en hoja de aluminio. Corresponde al tipo 2 de IBM.

TCP/IP:

Juego de protocolos creados en los 70's por Vince Cerf, profesor de Stanford, por encargo de Pentágono. El objetivo era lograr protocolos independientes del hardware. Hoy en día, son los protocolos que permiten la mayor conectividad entre los más diversos equipos (desde una Mac, hasta una Cray).

Teleproceso:

Sistemas que combinan comunicaciones y proceso de datos de tal forma que permiten ejecutar los procesos computacionales en una localidad distinta del punto de entrada o del punto de salida de información.

Transmisión Síncrona:

Forma de transmisión en la que ambos extremos deben tener un mismo pulso de reloj, y con base en éste, ambos extremos conocen en que momento puede transmitir. Aunque en la transmisión síncrona no se necesitan bits de inicio y final por cada caracter, el hardware requerido para sincronizar los pulsos de reloj, la hace más cara que la asíncrona.

TTL:

Abreviatura lógica de transistor a transistor.

UNIX:

Sistema Operativo multiusuario, desarrollado por AT&T. Es considerado muy flexible, poderoso y altamente portable. Corre en muchas plataformas de minis, y en algunas micros y mainframes.

V.28:

Características eléctricas de los circuitos de intercambio no equilibrados con doble corriente.

XMODEM:

Un protocolo asíncrono de control del nivel de data-link del Modelo OSI. Este protocolo es del dominio público.

XNS:

Siglas de "Xerox Network Services". Especificaciones de los servicios de alto nivel de una red desarrollada por Xerox Corporation para soportar su línea de productos de automatización de oficinas.

X/OPEN:

Organización dedicada a estandarizar los niveles mas altos de UNIX para poder crear un medio ambiente común de aplicaciones y mejorar la interoperabilidad de sistemas.

X/Windows:

Protocolo Cliente/Servidor orientado al desarrollo de interfaces gráficas. Desarrollado originalmente en MIT en el proyecto ATHENA.

WAN

Wide Area Network. Red de área amplia, como puede ser una red que comunica a dos o más ciudades.

B. GLOSARIO DE TERMINOS REFERENTES AL GIRO DEL NEGOCIO

Agua:

En las plantas se manejan los tres tipos de agua con que se comercializa:

Agua purificada
Agua destilada
Agua bidestilada

Aguacentro:

Centro de distribución de agua debidamente autorizado por la gerencia de ventas para vender agua electropura y tener el beneficio de la bonificación.

Bonificaciones:

Cuando un vendedor realiza una venta de contado a un aguacentro, se le dan más garrafones de agua, de acuerdo al nivel de compra que realiza y de acuerdo a un porcentaje establecido.

Ejemplo:

Cliente "La rosita"
Porcentaje de Bonificación: 10%
Venta de contado: 300 aguas
Bonificación : 300 * 10 % = 30 aguas más.

Esto tiene la finalidad de poder desplazar más producto en el mercado.

Cheques:

Este concepto refleja el monto en cheques que se reciben en caja.

Comodatos:

Cuando un cliente está en vías de establecerse como aguacentro, se le da la facilidad de prestar envase de la planta para que empiece su trabajo, para esto se hace un contrato donde se especifica que el garrafón está prestado o "comodatado" y que no pertenece al cliente sino a la empresa.

Efectivo Recibido:

Este concepto refleja la cantidad de dinero en efectivo que se recibió en la caja por la venta del día.

Faltante o Sobrante:

Este concepto refleja la diferencia de la resta de venta neta del día - total entrada de caja, si el resultado es positivo entonces se dice que hubo faltante de efectivo y si el resultado es negativo se dice que entraron más recursos de los que se vendieron.

Garrafón:

El garrafón de 19 litros se maneja en dos presentaciones, plástico de PVC y Vidrio.

Importe de bonificación:

Valor del número de aguas dadas al cliente por la compra de aguas al contado. El número de aguas está basado en cuanto al porcentaje sobre la venta a ese cliente.

Importe movimiento de garrafón:

El número de garrafones faltantes o sobrantes por su precio nos dá este concepto.

Precio de Venta:

Este concepto se obtiene de la operación siguiente:

PV = (Importe de Venta - Importe de Bonificación)/ No. total de aguas

Así como el porcentaje de Bonificación, también se obtienen dos precios de venta.

Porcentaje de Bonificaciones:

Se obtienen dos porcentajes:

El del resultado de dividir el importe de bonificación entre venta de planta y el otro es

El de dividir el importe de bonificación entre venta gravada

Reposiciones:

Se refiere al cambio de garrafones que los vendedores hacen cuando éstos están semi-abiertos, o con deficiencias, y se refleja en el costo.

Rutas:

Número de rutas que salen a distribuir en un día.

Saldo del día:

Es el resultado de la resta del concepto faltante/sobrante menos el importe de movimiento de garrafón.

Saldo sobrante acumulado del mes:

Este concepto es un acumulado de los resultados diarios del concepto saldo del día.

Total Entrada de Caja:

Es la suma de documentos y efectivo que entró a la caja en un día.

Unidades Garrafón Faltante del Día:

Concepto que se refiere al número de garrafones que no entraron a la planta, respecto a los que se sacaron en el día.

Unidades Garrafón Sobrante del Día:

Concepto que se refiere al número de garrafones que entraron de más a la planta, respecto a los que se sacaron en el día.

Venta a crédito:

Concepto que refleja el importe de la venta a crédito que se realizó a clientes debidamente autorizados para esta operación.

Venta de agua:

Este concepto es el resultado del número de aguas que se vendieron en el día. Incluye los tres tipos de agua.

Venta de planta:

Este concepto se refiere al valor que representa la venta de aguas por su importe.

Venta gravada:

Es el importe del número de aguas por las que se dió bonificación. Estas aguas se vendieron de contado lo que origina que a los aguacentros se les de una bonificación en especie.

Venta neta del Día:

Este concepto se refiere a la suma de la venta de Agua en sus tres presentaciones + garrafón + el pago del garrafón que los trabajadores rompen.

Viajes:

Concepto que muestra el número total de viajes que realizaron todos los camiones en el día.

Zonas:

Denominación que se le da a una área geográfica que recorre una ruta. Cada zona está bien delimitada respecto a otras.

Venta Foránea:

Importe de venta por vender al interior de la República.

Venta Directa:

Importe de venta por vender directamente al consumidor en la planta. Esta venta se realiza con previa autorización de la Dirección General y debe ser por una cantidad considerable.

Venta Intercos:

Importe de la venta a compañías del mismo grupo, esto se debe a traspasos de productos, los cuales no se cobran en efectivo sino con documentos.

REFERENCIAS

Publicaciones Periódicas

[1] García, Juan. *Sobre la recomendación RS-232C.* PC Semanal, año 1, volumen 2, número 41, 8 de febrero de 1993,.

[2] Dzubeck, Frank. *Cliente-servidor.* PC-Magazine en español, octubre de 1993.

[3] Edelstein, Herb. *Unraveling Client/Server Architectures.* DBMS Database & Client/Server Solutions, volumen 7, número 5, 1994.

[9] Tsichritzis, D., Niertrasz, O. y Gibbs, S. . *Beyond Objects: Objects.* International Journal of Intelligent and Cooperative Information Systems, volumen 1, número 1, 1992.

[10] Booch, Grady. *Object-Oriented Development.* IEEE Transactions on Software Engineering, volumen SE-12, número 2, febrero de 1986.

[11] Shaku, Atre. *Twelve step to succesfull Client-Server.* DBMS Database & Client/Server Solutions, volumen 7, número 5, 1994.

Libros

[4] Gordon, Davis. *Sistemas de Información Gerencial.* McGraw-Hill, 1987.

[5] Martin, James. *Principles of Object-Oriented Analysis and Design.* Englewood Cliffs, N.J., Prentice-Hall, 1993.

[6] M. Jackson. *System Development.* Englewood Cliffs, N.J., Prentice-Hall, 1993.

[7] Arrieta, Norberto. *Notas para el curso Programación Orientada a Objetos usando C++.* México, Marzo de 1991.

[8] Katrib, Mora. *Programación Orientada a Objetos en C++.* México D.F., INFOSYS, 1994.

[12] De Marco, Tom. *Structured Analysis and System Specification.* Englewood Cliffs, N.J., Prentice-Hall, 1979.

BIBLIOGRAFIA

1. Atkinson, Lee y Atkinson Mark. *Using Borland C++ 3.* Que Corporation, USA, 1992.

2. Beltrao, José y Philipe Jacques. *Redes locales de computadoras, protocolos de alto nivel y evaluación de presentaciones.* McGraw-Hill Interamericana de España, 1990.

3. Block, Uyless. *Redes de computadoras, protocolo, normas e interface.* Macrobit, 1990.

4. Borland. *ObjectWindows for C++, Programmer's Guide.* Borland International Inc., 1993.

5. Borland. *ObjectWindows for C++, Reference Guide.* Borland International Inc., 1993.

6. Borland. *Borland C++, User's Guide.* Borland International Inc., 1993.

7. Budd, Timothy. *An introduction to Object-Oriented Programming.* Addison-Wesley 1991.

8. Eckel, Bruce. *Aplique C++*. McGraw-Hill, 1991.

9. Fiedler, David y Hunter, Bruce. *UNIX System Administration*. Hayden Books, USA, 1986.

10. Guerrero, Gabriel. *Paradigmas de Software*. Saxsa S.C., México, 1992.

11. Gofton, Peter. *Mastering Unix serial communications*. Sybex, 1990.

12. Hartmeier, Michael. *Turbo C++, Acceso rápido*. Computec Editores, 1993.

13. Intel. *Intel SatisFaXtion Modem/100, Owner's Guide*. Intel Corporation, 1992.

14. Kernigham, Brian y Pike, Rob. *The Unix programming environment*. Englewood Cliffs, N.J.,Prentice-Hall, 1984.

15. Mahler, Paul. *An Informix 4GL tutorial*. Englewood Cliffs, N.J., Prentice-Hall, 1990.

16. Microsoft. *C++ Tutorial for MS-DOS and Windows Operating System*. Microsoft Corporation, 1991.

17. Rivera, Ricardo. *Apuntes del curso Administración de Centros de Cómputo.* México, 1992.

18. Rumbaugh, James; Blaha, Michael; Premerlani, William; Eddy, Frederick y Lorensen William. *Object Oriented Modeling and Design.* Englewood Cliffs, N.J., Prentice-Hall, 1991.

19. Sage, Russell. *Tricks of the UNIX Masters.* SAMS, The Waite Group, USA, 1987

20. Saldivar, Juan. *Downsizing: Para una mayor eficiencia de recursos.* Red, enero de 1992.

21. Schildt, Herbert. *Turbo C++ for Windows, Inside & Out.* McGraw-Hill, 1992.

22. Seyer, Martin . *Complete Guide to RS-232 and Parallel Connections.* Englewood Cliffs, N.J., Prentice-Hall, 1988.

23. Stevens, Richard. *UNIX network programming.* Englewood Cliffs, N.J., Prentice-Hall, 1990.

24. Stroustrup, Bjarne. *The C++ programming language.* Addison-Wesley, 1991.

25. Tore, Ramkrishna. *Procesamiento de datos en Unix con Informix SQL, ESQL/C, C-ISAM.* McGraw-Hill, 1990.