UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

Facultad de Ingeniería

" SISTEMA DE TELERESERVACIONES PARA LINEAS DE TRANSPORTE TERRESTRE "

T E S I S

Que para obtener el título de:

INGENIERO EN COMPUTACION

P r e s e n t a n:

ALEJANDRA MARTINEZ GOMEZ

JUAN MARTINEZ SANCHEZ

MOISES SANTIAGO RAMIREZ

México, D. F. 1992

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

INDICE

	PAG
INTRODUCCION	2
1. CONCEPTOS E INGENIERIA DE SISTEMAS	7
1.1. Ciclo de Vida de un Sistema	7
1.2. Metodologías de Desarrollo de Sistemas	8
1.3. Redes de Computadoras	16
1.4. Lenguajes de Programación	23
2. ANALISIS Y SELECCION DE HERRAMIENTAS	32
2.1. Problemática y Requerimientos	32
2.2. Selección de Herramientas	33
3. DISEÑO	75
3.1. Introducción	75
3.1.1. Diseño de Sistemas	75
3.1.2. Interfaz con el Usuario	76
3.2. Ambito	88
3.2.1. Objetivo del Sistema	88
3.2.2. Interfaces de Software, Hardware y Humanas	88
3.2.3. Alcances del Sistema	90

3.3. Arquitectura 90
3.3.1. Módulos del Sistema 90
3.3.2. Procesos y Programas a Desarrollar 91
3.3.3. Comportamiento Dinámico 93
3.3.4. Interacciones Organizacionales 96
3.4. Descripción del Diseño 96
3.4.1. Descripción de Datos 96
3.4.2. Diagrama de Flujo de Datos 98
3.5. Detalle de Módulos 104
3.5.1. Texto Explicativo 104
3.6. Estructura de Programas 108

4. IMPLEMENTACION Y PRUEBAS 112

4.1. Implementación 112
4.2. Pruebas 135
4.3. Implantación 137

CONCLUSIONES 141

BIBLIOGRAFIA 144

APENDICES

A. Glosario de Términos 146
B. Pseudocódigos 148
C. Diccionario de Datos 160
D. Manual de Usuario 170

INTRODUCCION

INTRODUCCION

Breve Historia del Transporte en México

Durante toda la época prehispánica, el movimiento de mercancías de una ciudad a otra lo hacían trabajadores, tanto libres como esclavos, con su propia fuerza física, o bien se auxiliaban con una tira de cuero llamada mecapal, la que ataban a su frente y les permitía sujetar bultos sobre la espalda.

Fueron ellos, así como los viajeros indígenas de entonces, los que abrieron los primeros caminos en el territorio de la República. Si bien conocían el uso de la rueda, no la utilizaban en vehículos de transporte terrestre.

A la llegada de los españoles estaban ya trazadas las principales rutas de Mesoamérica, nombre genérico como se conoce a la región comprendida desde la actual Sinaloa, el área limitada al norte por los ríos Lerma y Pánuco, hasta Costa Rica. Muchas de estas rutas comunicaban con el Nayar (actualmente Nayarit) y con la zona purépecha (en Michoacán).

Los colonizadores españoles trajeron del Viejo Mundo bestias de carga y arrastre, así como diversos tipos de ganado inexistentes hasta ese entonces en el Continente Americano, de los que destaca el caballo. Inicialmente ampliaron las veredas para convertirlas en caminos de herradura por donde pudieran transitar los animales en los que se transportaban, y más adelante los adaptaron para el paso de carros y carretas.

Se abrió así la etapa en que la apertura de caminos y nuevas vías respondía a un marcado interés económico. Así, durante los tres siglos de dominación española, se construyeron un total de 26 mil 107 kilómetros de rutas. Ya desde 1550 y 1560, un viajero que hubiera ido del sur al norte, utilizaba por esas fechas un camino de herradura.

El invento de la rueda tuvo lugar tanto en Europa como en América, al igual que en otras partes del mundo, la rueda fué una gran ayuda como transporte para el hombre en esa época.

Así fué como fueron surgiendo medios de transporte, los cuales eran de gran ayuda al hombre para poder transportarse de un lugar a otro.

Hoy en día gracias al avance tecnológico, contamos con diferentes medios de transporte, entre los que destacan: los transportes terrestres, el transporte aéreo, etc.

Debido a que la población se ha incrementado, la demanda del autotransporte se ha incrementado, convirtiéndose en un sistema muy complejo, el cual requiere de información rápida y oportuna.

Dada la problemática que presentan día a día las líneas de autotransporte, se hace necesario un control más eficaz y rápido, por lo que es necesario automatizar el control de las actividades que se llevan a cabo en las centrales de autobuses.

En este trabajo tratamos de un tema que involucra la automatización de actividades de una línea de autotransporte, Tres Estrellas de Oro S.A. de C.V.

Descripción del sistema

El sistema de telereservaciones para la línea Tres Estrellas de Oro S.A. de C.V., deberá estar desarrollado de tal forma que sea fácilmente implementado en las diferentes ciudades en las que opera la línea, por ejemplo: el D.F., Guadalajara, Tijuana, Mazatlán, etc. Además, deberá contar con un sistema que permita la comunicación entre las ciudades, para tal efecto se debe considerar un servicio de comunicaciones (opción TELEPAC, Microondas, comunicación vía *modem* o cualquier otra alternativa).

Por otra parte, desde el punto de vista operacional, el sistema debe contemplar una interfaz con el usuario que sea sencilla y agradable, y debe tener la característica de enviar sólo la información básica cuando se esté operando por vía remota.

El sistema de telereservaciones será implementado en una red de área local (*LAN*), ya que permite tener terminales inteligentes, las cuales hacen que el sistema tenga una buena respuesta, siendo esta última importante para un sistema de punto de venta y más si el sistema se encuentra operando las veinticuatro horas del día, los trecientos sesenta y cinco días del año y con una demanda de operaciones alta.

Por otro lado, el sistema estará organizado de acuerdo a las diferentes áreas que contempla la central de autobuses donde se encuentra ubicada la línea de transporte. Dichas áreas incluyen los siguientes módulos:

- Taquillas
- Liquidaciones
- Catálogos
- Impresión de reportes
- Mantenimiento

Taquillas

Este módulo es uno de los más importantes, ya que en él se encuentran todas las operaciones directas de venta, que son: la reservación y venta de boletos, la cancelación de reservaciones y boletos, y la venta de cupones de agencia y su cancelación.

Otro aspecto que controla este módulo es el de poder abrir corridas extras, esto es cuando existe una demanda extra de transporte a un lugar determinado. Además, en este módulo puede llevarse el control de los despachadores.

Liquidaciones

En este módulo se efectúa el control de guías de pasajeros de cada autobús y el control de operadores del viaje.

Se encuentra también el Corte de Caja, que lleva el control por despachador y agencias de viaje.

Catálogos

El registro de la información necesaria para la operación del sistema es un aspecto de vital importancia, por lo cual debe estar contemplado en este módulo. En este módulo se efectúa desde el registro de usuarios hasta el registro de operadores.

Impresión de Reportes

Las opciones indispensables en la operación de un sistema es de la información que nos arroja después de realizar las distintas operaciones durante el día, conteniendo en este punto las estadísticas de ventas de cada despachador, el registro de ventas por población, la estadística de las cancelaciones y reservaciones efectuadas, las guías de viaje, los reportes mensuales, los reportes de salida por hora, etc.

Mantenimiento

En este punto se deben considerar la generación de archivos históricos, así como las corridas del día y la afectación de los precios.

A fin de llevar un seguimiento del desarrollo del proyecto este trabajo se ha organizado de la siguiente manera:

En el capítulo 1 se abordan diversos temas relacionados con la ingeniería de sistemas, las redes de computadoras y lenguajes de programación. A manera de repaso o revisión las características importantes del desarrollo de sistemas, metodologías de desarrollo, configuraciones y topologías de redes de computadoras, así como características de los lenguajes de programación más comunmente utilizados.

El capítulo 2 contiene el análisis de los requerimientos del proyecto, así como un estudio de las diversas herramientas de desarrollo con las que se cuenta actualmente. Entre éstas herramientas se incluyen las metodologías de desarrollo de sistemas, el *hardware* existente en el mercado y el *software* de desarrollo como sistemas operativos, lenguajes de programación,

bases de datos y *software* de comunicaciones. También en este capítulo se detalla la selección de las herramientas empleadas para el desarrollo del proyecto justificando su selección con base al ámbito del mismo.

En el capítulo 3 se hace la especificación del diseño basado en la metodología de diseño orientado a la estructura de datos, se expone el *hardware* y *software* seleccionados como herramientas, así como la arquitectura del sistema, estructuras de datos, el detalle de los módulos e interfaz con el usuario.

El capítulo 4 contiene la implementación del sistema basado en una red de área local (*LAN*) y como lenguaje de programación CLIPPER, destacando algunos de los procesos del sistema planteados en pseudocódigos. También se hace mención de algunas de las características importantes de las herramientas empleadas. Se muestran las estrategias de prueba planteadas, así como los procedimientos llevados a cabo para verificar el cumplimiento de requerimientos específicos enfocados a la operación del sistema y resultados obtenidos.

En el apartado Conclusiones se hace una evaluación general del trabajo de tesis acerca del Sistema de Telereservaciones, así como también se sugieren modificaciones futuras al mismo.

Finalmente, se tiene una sección de apéndices a donde se hace referencia a lo largo del presente trabajo para el detalle de las estructuras de datos, código de la programación, términos empleados, etc.

CAPITULO 1

CONCEPTOS E INGENIERIA DE SISTEMAS

1. CONCEPTOS E INGENIERIA DE SISTEMAS

1.1 CICLO DE VIDA DE UN SISTEMA

Para definir el ciclo de vida de un sistema se requiere de un enfoque sistemático y secuencial del desarrollo del *software* y del *hardware*. El *software* progresa a través del análisis, diseño, codificación, prueba y mantenimiento, el *hardware* progresa a través del análisis, diseño, prueba y mantenimiento. El ciclo de vida de un sistema contempla las siguientes actividades:

Ingeniería y análisis del sistema. Debido a que el *software* siempre es parte de un sistema mayor, el trabajo comienza estableciendo los requerimientos de todos los elementos del sistema, y se asigna a algún subconjunto de estos requerimientos el *software*. Esta visión del sistema es esencial cuando el *software* debe interrelacionarse con otros elementos, tales como: *hardware*, bases de datos y personas. La ingeniería y análisis del sistema contempla los requerimientos globales a nivel del sistema.

Análisis de los requerimientos del software. El proceso de definición de los requerimientos se centra e intensifica especialmente en el *software*. Para comprender la naturaleza de los programas que hay que construir, el ingeniero de *software* debe conocer la información existente alrededor del mismo, así como la función, rendimiento e interfaces requeridas. Los requerimientos tanto del sistema como del *software* se documentan y revisan con el cliente.

Diseño. El diseño del *software* es realmente un proceso multipaso que se enfoca sobre tres atributos distintos del programa: estructura de datos, arquitectura del *software* y detalle de procedimientos. El proceso de diseño traduce los requerimientos en una representación del *software* que pueda ser establecida de forma que obtenga la calidad requerida antes de que comience la codificación. Como los requerimientos, el diseño se documenta y forma parte de la configuración del *software*.

Codificación. El diseño debe traducirse en una forma legible para la máquina. El paso de la codificación ejecuta esta tarea. Si el diseño se ejecuta de una manera detallada, la codificación puede realizarse mecánicamente.

Prueba. Una vez que se ha generado el código, comienza la prueba del programa. La prueba se enfoca sobre la lógica interna del *software*, asegurando que todas las sentencias se han probado, incluyendo las funciones externas, esto es, realizando pruebas para asegurar que una entrada definida producirá los resultados que realmente se requieren.

Mantenimiento. El *software* sufrirá indudablemente cambios después de que se entregue al cliente. Los cambios ocurrirán debido a que se han encontrado errores, a que el *software* debe adaptarse por cambios del entorno externo, o debido a que el cliente requiere aumentos funcionales o del rendimiento. El mantenimiento del *software* se aplica a cada uno de los pasos precedentes del ciclo de vida a un programa existente en vez de uno nuevo.

En la figura 1 se muestra el desarrollo del ciclo de vida de un sistema.

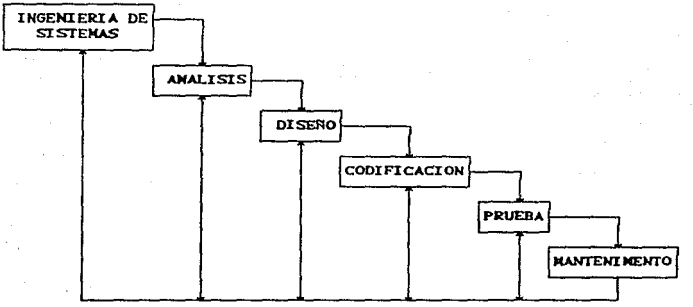


Fig. 1. Ciclo de vida de un sistema

1.2 METODOLOGIAS DE DESARROLLO DE SISTEMAS

Una metodología de desarrollo de sistemas formaliza y codifica una serie de fases para el ciclo de desarrollo del sistema, define los objetivos precisos para cada fase y los resultados que se requieren de una fase para que la siguiente pueda comenzar. Dicha metodología puede proporcionar formas especializadas para la preparación de la documentación a lo largo de cada fase.

El modelo de fases divide el ciclo de vida de un sistema en una serie de actividades sucesivas; cada fase requiere información de entrada, de los procesos a efectuar y de sus resultados, todos ellos bien definidos. Se necesitan recursos para terminar los procesos de cada fase, y cada una de ellas se efectúa mediante la aplicación de métodos explícitos, herramientas y técnicas.

Las fases que se definen dentro de una metodología de desarrollo de sistemas son las siguientes:

1. Fase de definición de requerimientos
2. Fase de diseño
3. Fase de desarrollo
4. Fase de operación y mantenimiento

Fase de definición de requerimientos

No se puede hablar de un desarrollo sistemático de programas o sistemas de cómputo sin una definición precisa de objetivos y un planteamiento claro de requerimientos.

Los objetivos deben definir las funciones generales que se esperan del producto final, y en ellos deberá basarse la estructuración del sistema.

Los requerimientos definen con precisión las características de los programas de computadora por desarrollar y establecen los alcances del sistema de cómputo. Sirven de base para el diseño del sistema y, además, permiten controlar la evolución del sistema durante sus etapas de desarrollo.

La fase de Definición de Requerimientos se realiza en tres etapas:

1. Planteamiento de objetivos
2. Análisis
3. Especificación de requerimientos

Planteamiento de objetivos

Esta etapa consiste en identificar y describir las necesidades del usuario a fin de proponer un conjunto de objetivos, que de lograrse, implicarían la satisfacción de necesidades. La claridad de los objetivos es condición necesaria para el éxito de todo proyecto. Al establecer los objetivos deberá evitarse proponer o establecer un método específico como base para el desarrollo del sistema, a menos que el empleo de dicha técnica constituya en sí uno de los objetivos del sistema. Al terminar esta etapa se aclaran las necesidades del usuario y los objetivos del proyecto.

Análisis

Esta etapa consiste en estudiar los objetivos para establecer un compromiso entre ellos, señalar prioridades, constatar su factibilidad y proponer los métodos de solución que sirvan de base para el diseño del sistema. Las actividades de esta etapa caen dentro del campo de acción de los analistas de sistemas. El grado de dificultad de esta etapa, y el tiempo requerido para realizarla, depende directamente de la complejidad del proyecto o problema a resolver. Otra condición necesaria para el éxito del proyecto la constituye un análisis correcto de los objetivos. Al terminar la etapa de análisis se conocen las funciones del sistema y los métodos de solución que deberán ser empleados para satisfacer las necesidades del usuario.

Especificación de requerimientos

En esta etapa se traducen los resultados del análisis a un idioma que permite la transferencia de ideas entre personas de distintas disciplinas. Este lenguaje está constituido por una combinación de prosa en lenguaje natural. En esta etapa también se define lo que el sistema deberá ser capaz de realizar, se establece la precisión con que deberán ser alcanzados los objetivos y se describe el sistema de cómputo desde el punto de vista del usuario, todo esto en un lenguaje que sirve de medio de comunicación entre el usuario, el analista y demás personas. Esta etapa debe ser intelegible, formal, completa y modificable.

Fase de diseño

El diseño estructural comprende la identificación de los componentes de la programación, su desacoplamiento en módulos de procesamiento y estructuras de datos conceptuales, y la especificación de las interconexiones entre componentes. El diseño detallado se refiere a detalles de cómo empacar módulos de procesamiento, y cómo instrumentar los algoritmos, las estructuras de datos y sus interconexiones. El diseño se relaciona con la adaptación de códigos existentes, modificación de algoritmos estándar, invención de nuevos algoritmos, diseño de representaciones de datos, e integración del producto final. El diseño detallado está muy influido por el lenguaje de programación.

Esta etapa se divide en tres partes:

1. Arquitectura
2. Diagrama de estructura
3. Detalle de módulos

Arquitectura

La arquitectura es la primera etapa de la fase de diseño de un sistema. Para el diseño de la arquitectura de un sistema de

programas se procede, en primera instancia, a dividir el sistema en subsistemas.

El lineamiento fundamental del diseño de sistemas recomienda subdividir los requerimientos en partes que resulten más fáciles de manejar y de entender. Lo anterior permite identificar las principales funciones que se encuentran implícitas en los requerimientos a fin de asociarlas al sistema. La arquitectura deberá incluir la definición de:

1. Los conjuntos de información
2. Los procesos o programas a desarrollar
3. El comportamiento dinámico
4. Las interacciones organizacionales
5. Las responsabilidades

Diagrama de estructura

Al nivel de la arquitectura del sistema de cómputo, los programas son definidos describiendo sus funciones y sus interfaces, pero sin mostrar su estructura interna. Para cada programa identificado en la arquitectura del sistema se procede a elaborar un diagrama de estructura, mismo que representa la estructura jerárquica de cada uno de ellos. El diagrama de estructura de un programa es una representación gráfica de la relación entre las subrutinas, el diagrama de estructura muestra, para cada subrutina o módulo, cuales son las subrutinas que lo activan y cuales subrutinas son activadas por el módulo.

Detalle de módulos

En esta etapa se detalla el proceso o función que representa cada uno de los módulos del diagrama de estructura mediante lógica estructurada. La especificación del proceso de cada módulo se describe utilizando las técnicas de programación estructurada en un lenguaje de alto nivel, denominado pseudocódigo estructurado. Es claro que la selección del lenguaje de programación deberá hacerse antes de la etapa de codificación; sin embargo, es recomendable hacer esta selección después de la etapa de análisis. Con frecuencia se selecciona un lenguaje de alto nivel para una aplicación específica con base en la experiencia individual restringida de una persona.

Fase de desarrollo

Los objetivos de la fase de desarrollo son:

1. Codificar los módulos
2. Verificar el correcto funcionamiento de cada módulo
3. Integrar los módulos para formar programas
4. Integrar programas para formar sistemas de programación

La fase de desarrollo se considera terminada cuando el usuario acepta los programas de computadora integrados a su sistema de cómputo.

La fase de desarrollo se divide en tres etapas:

1. Codificación
2. Integración
3. Pruebas de alto nivel

Durante las diferentes fases y etapas del proceso de programación se cometen errores que es necesario eliminar. Los errores que se llegan a presentar en productos de programación se pueden clasificar conforme al siguiente criterio:

1. Errores de codificación
2. Errores de diseño
3. Errores de análisis
4. Errores de especificación

Los errores de codificación son identificados durante la etapa de Integración.

Errores de diseño: un diseño imposible de implantar en el ambiente del equipo de explotación de cómputo que se pretende utilizar, o incapaz de satisfacer los requerimientos del programa, es la causa de los denominados errores de diseño.

Bajo el nombre de errores de análisis se conocen las fallas no detectadas en la fase de definición de requerimientos, que impiden satisfacer las necesidades del usuario.

Los errores de especificación son causados por especificaciones incompletas o imprecisas, dan lugar a una interpretación equivocada de las mismas.

Codificación

La etapa de Codificación de programas de computadora tiene como objetivo traducir las especificaciones de proceso de cada módulo, descritas en la fase anterior, en instrucciones ejecutables por un lenguaje de programación específico. Como se explicó en la fase de diseño, un problema complejo no se puede convertir en instrucciones de máquina o código de una manera natural y fácil. Es necesario traducir, inicialmente, la definición del problema en un lenguaje accesible al lector, como el pseudocódigo, para codificar posteriormente las operaciones descritas en un lenguaje que sea interpretable por las computadoras. Esta técnica es el corazón de varias metodologías de programación, entre ellas la Programación Estructurada.

Un requisito indispensable para el inicio de la etapa de codificación, es la terminación de la fase de diseño. La etapa de

codificación se considera terminada cuando todos los módulos han sido codificados y verificados.

Existen dos estrategias principales para la codificación de los módulos:

1. Codificación ascendente
2. Codificación descendente

La codificación ascendente permite atender inicialmente los módulos de los niveles jerárquicos más bajos. Estos contienen las operaciones más primitivas del programa y permiten sentar la base, para de ahí, construir las operaciones más complejas. La codificación descendente atiende en principio los módulos de los niveles jerárquicos superiores.

Finalmente, la etapa de la Codificación no deberá iniciarse antes de conducir una revisión formal del diseño del programa a codificar. Las normas de codificación son indispensables para el éxito de esta etapa.

Integración

El objetivo principal de esta etapa es la integración funcional de los módulos de un programa, ajustándolos a las particularidades del sistema. En esta etapa deben tomarse en cuenta dos aspectos importantes: la manera cómo se combinan los módulos para formar programas y el diseño de las pruebas que permiten identificar errores de codificación. Pueden seguirse dos estrategias:

1. Integración no incremental: validación de programas de computadora a partir de pruebas modulares independientes.

2. Integración incremental: validación de nuevos módulos agregándolos a módulos ya probados e integrados.

La integración incremental permite identificar errores en los módulos del programa de computadora adicionados más recientemente.

La integración no incremental no propicia esta identificación, ya que los errores emergerían, en este caso, hasta que todos los módulos se ensamblen.

Pruebas de alto nivel

Se denominan Pruebas de alto nivel a aquellas que tienen como objetivo identificar, no los errores de codificación ya encontrados en la etapa de Integración, sino los de análisis, especificación y diseño.

Entre estas pruebas las más importantes son las siguientes:

1. Pruebas funcionales
2. Pruebas de implantación
3. Pruebas de sistema
4. Pruebas de aceptación

Las pruebas funcionales tienen como objetivo encontrar errores de análisis, de especificación y diseño. Los casos para este tipo de pruebas son generalmente producidos mediante técnicas de análisis de entrada-salida.

Las pruebas de implantación tienen como objetivo encontrar errores de especificación, concepto y diseño en el ambiente real (equipo, sistema operativo, interfaces, etc.), en donde los programas serán finalmente instalados.

Las pruebas de sistema no consisten en volver a probar todas las funciones o programas de un sistema, sino probar mediante un juicio la compatibilidad del sistema con la documentación del usuario.

Las pruebas de aceptación tienen como objetivo comparar el producto de cómputo final con el contrato original, generalmente es el usuario el responsable de esta prueba.

Fase de operación y mantenimiento

Es en esta última fase del proceso de programación en donde se reflejan los aciertos o los errores de las fases previas. Se identifican aciertos en la medida en que los requerimientos de los programas de computadora satisfagan las necesidades del usuario, la arquitectura y los diseños se asocian a las características específicas del sistema de explotación de cómputo.

Operación

La fase de Operación de la programación se inicia con la primera instalación del sistema de programación integrado, una vez que la programación ha sido aceptada por el cliente con base a los documentos de los planes y procedimientos de prueba.

La documentación del usuario se encuentra en un documento llamado Manual de usuario, el cual es un instructivo para instalar, operar y mantener el sistema. Dependiendo de la complejidad del sistema y de la extensión de su documentación, el manual de usuario podrá ser presentado en un solo documento, ó bien, como un conjunto de guías que tratan en forma independiente los aspectos básicos de instalación, operación y mantenimiento.

La Guía de instalación define la manera y los medios de transportar el sistema de programación al equipo de cómputo.

Mantenimiento

Una vez aceptado por el cliente,el sistema de programación se entrega para operación y se inicia la etapa de mantenimiento del modelo de ciclo de vida por fases. Las actividades de mantenimiento incluyen mejoras de las capacidades, adaptación a nuevos ambientes de procesamiento, y corrección de fallas del sistema.

A continuación mencionaremos algunas Metodologías para el desarrollo de sistemas, propuestas por algunos autores.

La metodología de desarrollo según **Freeman**, se encuentra constituida por las siguientes fases:

1. Definición de requerimientos
2. Diseño
3. Desarrollo
4. Operación

La fase de definición de requerimientos se encuentra formada por las siguientes partes:

Necesidades
Análisis
Especificación

La fase de diseño está formada por:

Diseño de la arquitectura
Diseño detallado

La fase de desarrollo se encuentra formada por:

Implantación
Mantenimiento

La fase de operación la constituye:

Mantenimiento

La metodología de desarrollo según **Metzger** se encuentra constituida por las siguientes fases:

1. Definición de requerimientos
2. Diseño
3. Desarrollo
4. Operación

La fase de definición de requerimientos está constituida por una sola etapa:

Definición del sistema
La fase de diseño está formada por:

Diseño

La fase de desarrollo está constituida por:

Programación
Prueba del sistema
Aceptación

Por último la fase de operación está formada por:

Instalación y operación

La metodología de desarrollo según **Boehm** se encuentra formada por:

1. Definición de requerimientos
2. Diseño
3. Desarrollo
4. Operación

La fase de definición de requerimientos se encuentra formada por:

Requerimientos del sistema
Requerimientos de la programación

La fase de diseño la constituye:

Diseño preliminar
Diseño detallado

La fase de desarrollo la forma:

Codificación y depuración
Prueba y preoperación

La fase de operación está constituida por una etapa:

Operación y mantenimiento

1.3 REDES DE COMPUTADORAS

Introducción

La operación de los primeros sistemas de computadoras era en modo batch, es decir, la ejecución se daba sin que se interactuara para proporcionar datos o para consultar algo. Los datos y programas eran preparados mediante dispositivos mecánicos, perforadoras de

tarjetas, y eran puestos en marcha mediante un operador de computadoras. Cuando se introdujeron los sistemas operativos, éstos reemplazaron el trabajo de los operadores humanos. Los programas aún seguían siendo preparados en modo *off-line*, pero ya eran almacenados en un archivo para que el programador de actividades (*job scheduller*) los seleccionara y ejecutara. Este tipo de operación era conveniente porque las grandes computadoras que se construyeron resultaban demasiado caras y debían ser aprovechadas por varios usuarios. El esquema era una instalación central única (SITE), en donde se encontraba en operación la computadora que daba servicio a los usuarios dispersos al centro de cómputo, así como otros SITES subalternos.

Conforme fueron evolucionando los sistemas de cómputo, también fueron evolucionando los requerimientos de los usuarios. Después de existir *Main frames* como entidades de computo compartidas mediante terminales, los usuarios empezaron a demandar el tener un medio de comunicación para compartir datos, así como para hacer transferencias de información de gran volúmen. En respuesta a estas demandas se desarrollaron diversas herramientas de comunicación, para interconectar desde computadoras con una arquitectura similar hasta computadoras con arquitecturas con poca compatibilidad. Así es como surgieron las redes de computadoras, las cuales son en sí un conjunto de computadoras interconectadas por uno o por varios medios físicos de comunicación.

Ventajas de las redes de computadoras

Las redes de computadoras han tenido un gran impacto en el mundo actual y se espera que en los años sucesivos lo seguirá teniendo, debido a la gran importancia que reviste el tener comunicación de información con un tiempo de respuesta inmediato. Entre las ventajas que tienen las redes de computadoras se encuentran:

- Las organizaciones modernas suelen estar dispersas en un país o en todo el mundo. Estas organizaciones comunmente cuentan con sistemas de cómputo que requieren compartir información, requiriendo que sea lo más interactivo posible.

- La interconexión de computadoras permite que varias máquinas compartan los mismos recursos, de tal forma que se puede distribuir la carga de cómputo entre diferentes computadoras.

- En algunas redes de computadoras se cuenta con tolerancia a fallas, en donde si algunos de los CPU's conectados a la red se llegara a caer, se puede solicitar soporte de alguna otra computadora para que asuma las funciones de cómputo necesarias.

- Las redes han demostrado tener una gran flexibilidad en las empresas que requieren de automatizar muchos de sus procesos y funcionamiento en donde se requiere compartir información, así

como de observar la variación en un momento dado de algún dato como el volúmen de ventas, situación bancaria, etc.

Topologías de redes de computadoras

La topología de una red es la configuración que tiene la misma, o bien, la forma en que está conectada. Al diseñar una red es importante considerar:

- Que la recepción de datos sea correcta.
- Que el tránsito de la información se haga por la ruta más corta.
- Que el usuario espera un tiempo de respuesta inmediato.

La fiabilidad de una red está determinada por la capacidad de transportar datos correctamente de un Equipo Terminal de Datos (ETD) a otro. También intervienen la capacidad de recuperación y corrección de errores que puede haber por la falla del canal de comunicación, el ETD, el Equipo Terminal del Circuito de Datos o (ETCD), o del Equipo de Conmutación de datos (ECD).

Las topologías más importantes son: jerárquica, *bus*, estrella, anillo, malla y combinaciones de algunas de éstas.

Topología jerárquica

La topología jerárquica es una de las más comunes. 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 con jerarquía más alta es el que controla la red. En la fig. 1a se muestra una topología de este tipo, el flujo entre los distintos ETD empieza en el ETD A. Muchos fabricantes han incorporado a esta topología un carácter distribuido, dotando a los ETD subordinados de un control directo sobre los ETD situados en niveles inferiores jerárquicos, reduciendo la carga de trabajo del ETD A.

La topología jerárquica presenta ciertos problemas en cuanto a la aparición de cuellos de botella. En algunas ocasiones el ETD más alto (normalmente un *Main Frame* central) controla todo el tráfico entre los distintos ETD. Esto no sólo genera una saturación de datos sino que también genera problemas de fiabilidad. Si el ETD principal falla, la red completa deja de funcionar.

Las redes con topología jerárquica también se conocen como redes verticales o en árbol.

Topología de bus (horizontal)

La topología de *bus* se muestra en la fig. 1b. Esta estructura se utiliza frecuentemente en las redes de área local. Es fácil controlar el tráfico entre los diferentes 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 una topología horizontal está en el hecho de que suele existir un sólo canal de comunicaciones para todos los dispositivos de la red. En consecuencia, si el canal de comunicaciones falla, la red completa 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. Otro inconveniente de esta configuración estriba en la dificultad de aislar las averías de los componentes individuales conectados al *bus*. La falta de puntos de concentración complica la resolución de este tipo de problemas.

Topología en estrella

La topología en estrella (fig. 1c) es una de las más empleadas en los sistemas de comunicación de datos. La red en estrella se utilizó por los años sesenta y principios de los setenta, porque resultaba fácil de controlar; su *software* no es complicado y el flujo de tráfico es sencillo. Todo el tráfico emana del núcleo de la estrella, que en la figura es el nodo central marcado como A. El nodo A posee el control total de los ETD conectados a él.

La configuración en estrella es, por lo tanto, una estructura muy similar a la topología jerárquica, aunque su capacidad de procesamiento distribuido es limitada. El nodo A es responsable de encaminar el tráfico hacia el resto de los componentes, además se encarga 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. Sin embargo, y al igual que en la estructura jerárquica, una red en estrella puede sufrir saturaciones y problemas en caso de falla del nodo central.

Topología en anillo

La topología en anillo se muestra en la fig. 1d, y se llama así por la característica circular en el flujo de datos. En la mayoría de los casos, los datos fluyen en una sóla dirección, y cada estación recibe la señal y la retransmite a la siguiente del anillo. La organización en anillo resulta atractiva porque con ella son poco frecuentes los embotellamientos tan frecuentes en las topologías en estrella o en árbol. Además, la lógica necesaria para poner en marcha una red de este tipo es relativamente simple. Cada componente sólo ha 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. Sin embargo, como todas las redes, tiene el problema de que todos los componentes del anillo están unidos por un mismo canal y si falla el canal entre dos nodos, la red completa deja de funcionar, aunque algunos fabricantes han construido algunos diseños especiales que incluyen canales de seguridad, así como conmutadores que saltan automáticamente el nodo averiado.

Topología en malla

La topología en malla es de reciente uso, y resulta atractiva por su relativa inmunidad a los problemas de embotellamiento y averías. Esto es gracias a la multiplicidad de rutas que ofrece a través de los distintos ETD y ECD, pudiéndose orientar el tráfico por trayectorias alternativas en caso de que algún nodo esté averiado u ocupado. A pesar de que la realización de éste método es compleja y cara, un gran porcentaje prefiere esta topología por la fiabilidad que ofrece.

Las diversas topologías tienen una implementación práctica dada por los fabricantes de los diversos dispositivos que permiten tener alguna configuración en particular, pero asumiendo las características generales de las topologías de redes de computadoras.

Redes WAN

Muchos ETCD, ETD y ECD suelen estar conectados entre sí, a través de un canal telefónico, esto es porque en ocasiones se requiere tener una red de computadoras con distancias entre sí lo suficientemente grandes como para que se justifique emplear el canal telefónico como medio de comunicación. A este tipo de redes de computadoras se les ha denominado como redes de Area amplia (*Wide Area Network*).

Cuando se desea tener una red amplia WAN (*Wide Area Network*) el principal problema se encuentra en el medio o canal de comunicación a emplear, en este caso nos centraremos a analizar el canal de comunicación telefónico.

Para disponer de un enlace permanente entre un punto y otro a través de la red telefónica, se puede escoger entre una línea privada o una línea conmutada. (las líneas privadas también pueden conmutarse, a través de centros privados de conmutación). Las líneas privadas suelen ser de gran utilidad cuando se requiere rapidez para establecer una conexión y también cuando el tipo de aplicación requiere de estar en enlace varias horas. A continuación se listan las ventajas y desventajas de cada una de estas líneas:

- Línea conmutada

Ventajas:

Flexibilidad
Economía si el volúmen de datos es pequeño
Desventajas:
Tiempo de respuesta lento
Posibilidad de bloqueo en la comunicación
Baja calidad en la comunicación
Costo elevado si el tráfico es intenso

- Línea privada

Ventajas:

Soporta un mayor volúmen de tráfico
Mayor calidad en la comunicación
Libre de bloqueos en la conexión

Desventajas:

Costo elevado si el tráfico es pequeño
Escasa flexibilidad en conexiones físicas

El modem

El *modem* es la interfaz entre una computadora (que maneja una señal digital) y el canal de comunicación telefónico (por el cual se transmiten señales analógicas). Para conseguir representar los *bits* como señales analógicas, el *modem* puede modificar amplitudes, frecuencias o fases.

El modem es un claro ejemplo de un ETCD. Sirve de interfaz entre las señales digitales y las señales analógicas, además de permitir que un ETD digital transmita datos a otro ETD receptor a través de un canal analógico.

El modem se encarga de modificar la señal portadora (ya sea en su amplitud, en su frecuencia o en su fase) para poder transportar la señal en banda base.

Transmisión síncrona y asíncrona

Muchas computadoras y terminales se comunican entre sí y con los ETCD mediante códigos sin retorno a cero (NRZ). Por consiguiente, en este tipo de dispositivos la sincronía es muy importante. Para conseguirla se emplean dos convenios de formateo de los datos, los formatos asíncrono y síncrono. El primer método es el formateo asíncrono, en el cual cada *byte* de datos incluye señales de sincronía de arranque o de parada, al principio y al final, señal de datos, y señales de verificación de la recepción adecuada de los datos. La misión de estas señales consiste, en primer lugar, en avisar al receptor de que está llegando un dato y, en segundo lugar, darle tiempo suficiente para realizar algunas funciones de sincronismo antes de que llegue el siguiente *byte*. Los *bits* de arranque y de parada en realidad no son otra cosa que señales específicas y únicas que el dispositivo receptor es capáz de reconocer.

La transmisión asíncrona se emplea bastante, ya que las interfaces de este tipo entre los ETD y los ETCD son sencillas y económicas. Como la sincronización entre el emisor y el receptor tiene lugar *byte* a *byte*, es admisible cierto grado de error, lo cual puede ser corregido antes de que llegue el siguiente *byte*. Tal relajación en las exigencias de sincronismo se traduce en un ahorro en los componentes.

Existe otro procedimiento más refinado, conocido como transmisión síncrona, en lo que se emplean canales separados de reloj, o bien códigos autosincronizados. En los formatos síncronos se suprimen las señales intermitentes de arranque/parada que acompañana cada *byte*. Las primeras señales se llaman bytes de sincronización o banderas (*flags*). Su función principal consiste en alertar al receptor de la llegada de datos. Este proceso se conoce como entramado (*framing*). Se ha comprobado que los mensajes síncronos de gran longitud sin *bits* intermitentes de arranque/parada pueden presentar problemas, ya que puede suceder que el receptor se desplace con respecto a la señal.

El tipo de transmisión utilizado en una red es importante, debido a que depende del tipo de aplicación que se requiera tener; por ejemplo, una transmisión *byte* a *byte* es útil en aplicaciones interactivas que requieren tener una respuesta en línea, en cambio la transmisión hecha por medio de entramado (*framing*) es útil en aplicaciones que intercambian grandes volúmenes de información.

Las comunicaciones entre computadoras han venido a ser de gran importancia, debido a la aparición de computadoras personales, ya que éstas se utilizan ampliamente en diversas empresas por la flexibilidad que ofrecen, pero a su vez han venido a dar lugar a paquetes de información aislada y duplicada, por lo que ahora grandes compañías que se han saturado de computadoras independientes ven en las comunicaciones la alternativa para generar una infraestructura de cómputo, en donde la información se comparta entre las diferentes áreas de la empresa.

Redes corporativas

Por mucho tiempo las computadoras han sido utilizadas en los negocios, en la administración del proceso de datos, etc., teniéndose la espectativa de interconectar todas éstas dentro de una gran red que comtempla toda una compañía. Originalmente la visión de un sistema análogo era tener un gran *Main Frame* compartiendo datos con otro *Main Frame* dentro de la compañía. Pero, debido a que los *Main Frames* prácticamente han venido desapareciendo, debido al avance de la tecnología, y la aparición de las computadoras personales, se observa otra evolución hacia las redes de computadoras personales.

El enfoque de las computadoras personales

Aunque la introducción de computadoras personales fue precisamente una manera de automatizarse en forma individual, recientemente se tiene el concepto del punto de vista corporativo. Las computadoras personales han probado su gran productividad, pero al mismo tiempo se ha observado que se tiene una cierta pérdida de control al tener la información en cada una de éstas computadoras.

Al principio mucha gente que tenía la necesidad de cierta aplicación empezó a desarrollar sus propios programas. Muchos de

estos aficionados tenían apenas alguna preparación en programación, por lo que algunos de los programas eran fatales.

Además los usuarios empezaron a crear sus propios datos y empezaron a almacenarlos en discos duros locales. Pero muchos de estos datos no eran respaldados poniéndose en peligro de perder la información. La combinación de una programación pobre y la poca seguridad en la información causó serios problemas a los administradores y varias noches en vela. Actualmente el enfoque ha cambiado hacia las redes de tipo local (LAN: *Local Area Network*).

Para una corporación, una LAN tiene dos características atractivas: su obvia disponibilidad de comunicaciones y el compartir los archivos. Una LAN permite algún control sobre la información de la compañía, ya que la información es almacenada en un ***SITE*** central, que es el servidor (*File server*), así como también da a los administradores un grado de control en el *software* disponible. Como resultado de estas características, las redes han empezado a incursionar en las grandes compañías.

1.4 LENGUALES DE PROGRAMACION

Introducción

Para contestar a la pregunta qué es un lenguaje de programación?, tenemos que ubicarlo en el contexto de su desarrollo histórico. Muchas veces no es fácil puntualizar aquello que tiene o no importancia en este desarrollo, pues el campo a cubrir es muy amplio.

Empecemos por definir un lenguaje de programación como un conjunto de cadenas de carácteres, que son producto de la aplicación de las reglas de una gramática (serie de reglas, cuyos elementos pertenecen a un alfabeto determinado, para generar un lenguaje). Sin embargo, siempre que se piensa en un lenguaje de programación se hace pensando en una herramienta, no sólo en el sentido de la comunicación entre seres humanos, sino primordialmente como un mecanismo de comunicación entre un individuo con un problema y la computadora a ser utilizada para ayudar a resolverlo. Los lenguajes de programación surgen siempre para facilitar esta comunicación. La diversidad y cantidad de lenguajes de programación existente lleva implícito el reconocimiento de la naturaleza de cada problema, determina la complejidad de las funciones que describen breve y claramente el algoritmo para resolverlo. Si pensamos en un lenguaje de programación como un conjunto de funciones elementales y reglas para combinarlas entre sí, veremos que la naturaleza de cada problema determina la naturaleza de las funciones elementales que describen su solución.

Por otro lado, para efectos de esta tesis, sólo se mencionan algunos de los lenguajes que han tenido mayor auge en el ámbito de la computación, así como los posibles lenguajes que se pueden utilizar para el desarrollo del sistema de telereservaciones de

líneas terrestres, aplicado a una línea de transportes.

Empezaremos con el primer lenguaje de alto nivel desarrollado: Fortran. Un lenguaje de alto nivel que se orienta a la solución de problemas o procedimientos de procesamiento; las proposiciones de instrucción emplean palabras, frases, y símbolos semejantes a los empleados comunmente para describir la solución a un problema.

FORTRAN

Fortran (acrónimo de *Formula Translation*), históricamente fue desarrollado en 1957 por IBM y algunos usuarios importantes. La evolución de este lenguaje dió como resultado un lenguaje de complejidad cada vez mayor. Las versiones más importantes se denominaron Fortran, Fortran II, Fortran IV. Cada versión implicó algunas modificaciones en las instrucciones básicas y además se incluyeron características adicionales.

En 1966 se adoptó voluntariamente como Fortran estandard al de la ANS (ANS: *American National Standard*).

También se definió un Fortran estandard por la ISO (ISO: *International Standard Organization*).

Se recomienda, como base de toda programación, tanto para estudiantes así como para aplicaciones científicas, el Fortran de la ANS.

Características del Fortran:

Independiente de la máquina, muy transportable.
Permite la programación estructurada.
Orientado a aplicaciones científicas.
Manejo de números complejos.
Manejo de archivos limitados.
Entendible.

Las características de Fortran lo hacen un lenguaje poderoso, pero, su manejo de archivos es limitado. Cabe aclarar que existen versiones de Fortran con manejo de archivos muy poderosas, como Fortran Vax de DEC (*Digital Equipment Corporation*).

COBOL

Al extenderse el uso de Fortran, pero sobre todo la idea de compiladores, empiezan a surgir ideas de lenguajes para aplicaciones específicas. Muy importante es el surgimiento de Cobol (*Common Business Oriented Language*) en 1959, que va precedido por toda una familia de lenguajes orientada a dar una notación concisa y fácil de usar, para la descripción de archivos y de las distintas relaciones que guardan los datos entre sí. De la misma manera se pueden ejecutar operaciones sencillas de comparación, movimiento de información, etc. Cobol es la cima de este esfuerzo y

es dirigido principalmente por el Departamento de Defensa de los Estados Unidos. Cobol es hoy en día uno de los lenguajes más utilizados en cientos de centros de computación, con procesos comerciales, etc. Sin embargo, existe la influencia de otros lenguajes en este campo.

En 1961 apareció la versión COBOL-61, que fijó las estructuras de las subsecuentes versiones, la siguiente versión apareció en 1965, pero no fue hasta agosto de 1968 que el lenguaje se estandarizó. Este Cobol fue mejorado y evolucionó notablemente por la adición de nuevas instrucciones y la sustitución de algunas instrucciones.

Características del Cobol:

Esta última versión fue revisada en 1974 por la ANSI (ANSI: *American Standard Institute*).

Es un lenguaje autodocumentado, esto le permite al programador tener una clara idea de lo que se está realizando, aunque la autodocumentación se asemeja al idioma inglés.

Entendible: el lenguaje por la característica de ser autodocumentado, es fácil de entender.

Es un lenguaje de programación de alto nivel.

Es transportable en un 85%, esto es que en cualquier instalación de computadoras se puede correr cualquier programa en Cobol, con muy pocas modificaciones (debe ser el Cobol ANSI).

Por definición es un lenguaje orientado a los negocios, ya que puede manejar una gran cantidad de información, referente a la administración, fue hecho para usarse en las empresas.

De acuerdo a las características anteriores COBOL es un buen lenguaje para el desarrollo de sistemas administrativos., sin embargo, carece de un buen manejo de pantallas y ventanas, además es un lenguaje en el cual las deficiones de las características de variables de archivo se tienen que definir en él, siendo un lenguaje muy extenso en sus declaraciones y por lo cual se utiliza más tiempo durante el desarrollo de sistemas.

BASIC

En 1966 surge BASIC (*Beginners Allpurpose Symbolic Instruction Code*). El objetivo principal del lenguaje de programación Basic es el introducir a las personas rápida y fácilmente al mundo de la programación.

Para este lenguaje, en equipos de microcomputadoras (PC'S), existen diversas versiones (Turbo Basic) en las cuales se tiene

un buen manejo de ventanas, su desventaja principal está en su manejo de archivos ya que es limitada.

PASCAL

El lenguaje de programación Pascal fué desarrollado por Niklaus Wirth en Eidgenossishe Technishe Hochscule, Zurich, en 1960.

Diez años más tarde, fue creado el primer compilador, el cual estuvo basado en los principios del Algol-60 y del Algol-W, aunque Pascal es más poderoso y fácil de usar.

Los propósitos principales de N. Wirth estuvieron abocados a la creación de un lenguaje apropiado para enseñar programación, como disciplina sistemática, a la instrumentación de lenguajes confiables y eficientes en las computadoras actuales, y al desarrollo de programas bién estructurados y organizados.

Características del Pascal:

Las características del lenguaje Pascal son las siguientes:

Es un lenguaje completamente estructurado.

Presenta diversos métodos para el uso de datos, dado que:

Posee las estructuras: registro, (*record*), conjunto (*SET*), y apuntador (*pointer*).

Permite manipular con gran facilidad:

Listas ligadas
Arboles
Pilas
Colas

Particularizando a microcomputadoras (PC'S), existen diversas versiones de Pascal, por ejemplo, Turbo Pascal, en sus diferentes versiones, así como Turbo Profesional, el cual nos ofrece una utilería con la que se desarrollan facilmente menús.

LENGUAJE C

Por otra parte existe otra área de aplicaciones mucho muy importantes, es la de compiladores, sistemas operativos, editores, etc. Para lo anterior se requiere de un lenguaje que tenga la estructuración necesaria, pero al mismo tiempo tenga acceso a operaciones básicas a nivel de lenguaje de máquina. El lenguaje C cumple con estas características.

Particularizando, existen versiones diferentes en el lenguaje C para microcomputadoras, como son Turbo C, Quick C, este último, permite tener acceso a utilerías para el manejo de

video, lo cual es muy importante en un sistema cuando se requiere de muchos despliegues y una gran rápidez en los mismos, sin embargo, no cuenta con un buen manejo de archivos en cuanto a utilerías e instrucciones.

Diversas Utilerías para PC's

Existen diversos paquetes de aplicación para microcomputadoras, destacando entre estos los manejadores de bases de datos. Un manejador de base de datos es un lenguaje de manejo de datos, el cual permite tener relaciones con los demás datos, además emplea índices de acceso que típicamente requieren para su almacenamiento la misma capacidad que los datos del archivo al que apuntan. Una base de datos se utiliza para la definición de estructuras de archivos, ya que cuenta con un compilador (CLIPPER) que permite utilizar las definiciones hechas en dBASE, además suministra un medio flexible para desarrollo de sistemas. Este se usa definiendo comandos y funciones que se necesitan para nuestra aplicación, ya que cuenta con la integración de códigos de otro lenguaje como C, o ensamblador. Además, CLIPPER cuenta con las estructuras de la programación estructurada; por su sencillez permite el desarrollo de sistemas con gran rapidez, cuenta con instrucciones que permiten usarlo en red, y procesa eficientemente grandes volúmenes de información. Dispone de una serie de instrucciones que permiten el fácil manejo de la información.

Para el desarrollo de sistemas hoy en día, comunmente se están usando las bases de datos, es por ello que a continuación hablaremos de ellas.

Bases de Datos

Así como la ciencia de la computación ha avanzado, también los métodos de almacenamiento de datos se han vuelto más y más sofisticados. El manejo de datos ha evolucionado de simples archivos secuenciales a archivo de acceso aleatorio, organizados para integrar sistemas de bases de datos. Un manejador de bases de datos ofrece un aumento de las capacidades del usuario, pero por la naturaleza de las características agregadas, y para una operación óptima se requiere de una mayor planeación en el desarrollo de los sistemas. El tiempo que se invierte en la planeación de la base de datos se justifica una vez que ésta ha sido establecida.

Qué es una Base de Datos?

Una base de datos es una colección de archivos lógicamente relacionados, los cuales contienen datos e información estructural. Los apuntadores en una base de datos permiten al usuario accesar información relacionada y dirigirse a través de los archivos de datos. La organización de una base de datos puede adquirir diferentes formas, dos ejemplos de esto serían la estructura jerárquica y la estructura de red.

La estructura jerárquica ha sido un crecimiento natural a partir de las primeras técnicas de organización de archivos. Los datos deben ser accesados a través de niveles de calificadores. Por ejemplo, en una base de inventarios, para poder accesar la información de un producto habría que accesar primero en que almacén y a que lote pertenece. Se utilizan archivos de referencia y de enlace para lograr una asociación lógica y accesibilidad. Cuando el número de archivos de datos crece, y su interrelación con ellos, se convierte una estructura más compleja. Los requerimientos de archivos de referencia y de enlace crecen en forma exponencial por el requerimiento de ligas, esto da como resultado demasiado trabajo para accesar datos.

Las bases de datos estructuradas bajo una red funcionan con base en la premisa de que sólo cuando un grupo de datos (Archivo de datos) está relacionado con otro grupo, debe haber una liga entre ellos, de esta manera los archivos de referencia y de enlace no se requieren más. El incremento de la complejidad de la base depende directamente del número de relaciones directas existentes entre un cierto número de archivos de datos, cuando los archivos de datos se consideran como nodos, con ligas de acceso directo conectándolos, se ha formado una base de datos de estructura en red.

En una base de inventarios de cada almacén, cada uno sería un archivo diferente.

Objetivo de un Manejador de Base de Datos

Un método efectivo de organización de bases de datos implica también el mantenimiento de ésta. Sin embargo, hay muchas razones prácticas para considerar un manejador de bases de datos como la solución a ciertos problemas de almacenamiento de datos.

Los objetivos de un manejador de base de datos son el conservar la seguridad y la integridad de estos, el manejo adecuado, la inmunidad a cambios externos, y la facilidad de uso en diferentes medios. La independencia de la base de datos es lo más importante. Apartir de que los datos de la base son centralizados, se vuelve más sencillo concentrar el control de ellos. Esta centralización nos da una independencia de la base de datos y de los programas de aplicación. Las modificaciones a los programas de aplicación no afectarán a la base de datos y las modificaciones a la base de datos no afectarán a los programas de aplicación. Esta independencia provee los fundamentos para un sistema de información dinámico y con demandas de crecimiento. Un manejador de base de datos debe ser orientado al usuario, de tal forma que éste aprenda fácilmente, entendiendo y así creando una amplia gama de aplicaciones. El usuario debe ser provisto de ayuda para la seguridad y el desarrollo de sus sistemas.

Porque usar una Base de Datos

El principal beneficio por el cual utilizar un manejador de bases de datos es el ahorro de tiempo. Este ahorro se puede manifestar de las siguientes formas:

Consolidación de archivos

La mayoría de los procesos de información que se utilizan en diferentes áreas de aplicación contienen datos repetidos. Por ejemplo, el nombre de una materia prima puede aparecer en un archivo de inventario, en un archivo de fórmulas o en un archivo de tránsito. Los datos entre éstos tres archivos tal vez varíen ligeramente, causando de esta manera un desperdicio de memoria secundaria. La información redundante se vuelve un gran problema en sistemas de información grandes.

La consolidación de archivos dentro de una base de datos elimina la mayoría de las redundancias. Con el uso de apuntadores los registros de información relacionados lógicamente son ligados aunque éstos estén físicamente separados. En el caso del ejemplo anterior, sólo un archivo existiría y cada programa accesaría la información que se requiera. Al existir sólo un registro para accesar y modificar se ahorra tiempo de acceso y espacio en la memoria.

Independencia de los Programas

Las estructuras convencionales de archivos tienden a ser rígidas e inflexibles. La naturaleza de los métodos de archivos convencionales requerían que la lógica de los programas de aplicación estuviera estrechamente ligada al diseño de los archivos. Cuando es necesario alterar la estructura de un archivo, un programa debe ser escrito para cambiar el archivo, y los programas que lo accesaban deben ser modificados de tal manera que reflejen el cambio realizado en dicho archivo. Es bien conocido en sistemas, que una gran parte del tiempo se emplea en el mantenimiento de programas.

El manejador de bases de datos permitirá que la estructura de la base de datos sea independiente a los programas de aplicación, la relación entre registros será definida independientemente. Los cambios en la base de datos deberán ser incorporados únicamente en aquellos programas que manipulan el dato que cambió, los programas solo accesarán la porción de la base que sea requerida por su aplicación, lo que evita que éstos tengan que ser modificados sin accesar el dato que cambió.

Versatilidad

Las técnicas de organización de archivos convencionales permiten un acceso limitado a los datos que contienen. La mayoría de las estructuras permiten un acceso de una sola llave, con un acceso

relacional adicional, permitido solamente a través de la implementación de un programa de soporte de gran nivel.

Este manejador permitirá el acceso con varias llaves así como a través de una variedad de métodos.

Seguridad de Datos

Los manejadores convencionales contienen una seguridad para los datos muy limitada. El acceso a un dato puede ser negado al usuario sólo por la provisión de un medio físico.

Desarrollo de Programas

La estructura de la base de datos puede ser definida y construida sin el uso de programas de aplicación especiales. Partiendo de que el control de las ligas de la base de datos estará bajo el control del *software* del manejador, el programador no necesitará preocuparse por probar la estructura y podrá concentrarse en la programación de la aplicación.

Mantenimiento de Programas

Durante la vida de un sistema, los procesos requeridos definen el uso de los datos. Al igual que la organización de los archivos varía con las necesidades de la aplicación, algunos cambios pueden ser requeridos con un pequeño impacto sobre los programas de aplicación existentes. Los cambios en la estructura de una base de datos existente afectarán sólo aquellos programas que procesan los datos cambiados, ningún otro programa debe ser recompilado para reflejar el cambio de la estructura de la base.

Finalmente, las características que ofrece una base de datos, para el manejo y versatilidad en el desarrollo de programas, hacen que sea eficiente su uso y por demás adecuado.

CAPITULO 2

ANALISIS Y SELECCION DE HERRAMIENTAS

2. ANALISIS Y SELECCION DE HERRAMIENTAS

2.1 PROBLEMATICA Y REQUERIMIENTOS

Problemática

Dentro de las actividades cotidianas de una línea de transporte terrestre, se llevan a cabo diariamente salidas de autobuses a distintos destinos, así como a las diferentes poblaciones de una ruta. Esto implica tener que llevar un control muy estricto en las corridas de los autobuses, dándole a conocer a los pasajeros las rutas que ofrece dicha línea, así como, los costos y tipo de servicio. Por lo anterior, es necesario ofrecerle un servicio rápido y eficiente. También se deberá contemplar el manejo de reservaciones por vía telefónica.

La mayoría de empresas de autotransporte se han venido administrando en forma obsoleta, debido a los orígenes de las mismas, ya que para administrar un autobús no se requería de una administración especial. Hoy en día, esto ha cambiado radicalmente, debido a que existen empresas de autotransporte que han ido creciendo conforme pasa el tiempo, por lo que se ha ido aumentando la complejidad de la administración de este tipo de servicio. En estas empresas se ha tenido la necesidad de tener información compacta dentro de las diferentes centrales en donde se tienen salidas de autobuses. Una de estas empresas es el grupo Autotransportes Tres Estrellas de Oro S.A. de C.V., el cual es uno de los más grandes dentro del país. Esta empresa se ha enfrentado con la problemática de no tener un control de operaciones de venta de boletos, así como de enrolamiento de autobuses, salidas y llegadas.

Algunas de las centrales de esta empresa tienen que controlar la salida de 60 autobuses, con diferentes destinos diariamente, y sabiendo que en cada autobús viajan alrededor de 40 pasajeros, podemos estimar que el volúmen de pasajeros transportados diariamente es alrededor de 2400.

Por lo expuesto anteriormente, podemos concluir que la operación de una empresa de autotransporte, en global, representa un problema muy complejo, los ingresos del autobús durante una corrida, las reservaciones, etc., por lo que todo este trabajo realizado en forma manual, hace que dicha operación sea muy compleja e imprecisa.

Requerimientos

Otro aspecto muy importante dentro de las operaciones mencionadas anteriormente, es que diariamente se tiene poca información que sea útil para la toma de decisiones, por lo que es necesario tener un estandard de requerimientos básicos, para que la dirección de la empresa alcance su objetivo, el cual es ser una empresa líder de autotransporte en México.

Dichos requerimientos se mencionan a continuación:

- Tener un control preciso de la venta y reservación de lugares en las diferentes salidas que ofrece la empresa al pasajero.

- Determinar la plantilla de pasajeros que van en cada uno de los autobuses, así como la hora de salida o de llegada de cada uno de ellos.

- Ofrecer el servicio de venta de boletos con un mes de anticipación, considerando temporadas de vacaciones, así como incrementos en la tarifa de precios.

- Ofrecer el servicio de venta de boletos de viajes redondos (de ida y vuelta), asegurando su lugar en el autobús de regreso.

- Llevar un control automático de ingresos captados en cada una de las centrales y desglosado por cada uno de los taquilleros que realizan ventas de boletos.

2.2. SELECCION DE HERRAMIENTAS

En la década de los 80's la aparición de las microcomputadoras provocó una revolución en los sistemas de información.

Posteriormente, con la creación de los discos duros, se vió una manera de acabar con las islas de información generadas por diversas computadoras personales (*PC'S*), que tenían una capacidad de memoria limitada y que provocaban la duplicidad de información. También, se obtuvo el beneficio de almacenar grandes volúmenes de información en un solo disco.

La gran desventaja de los discos duros era su alto costo y bajo nivel de aprovechamiento, en cuanto al número de usuarios y cantidad de almacenamiento. Por lo que surgió la idea de compartir entre varios usuarios los beneficios y costos del uso de un disco duro, naciendo así LOS SISTEMAS DE REDES. Dichos sistemas se basan en compartir recursos centralizados por medio de TERMINALES. Estos sistemas permiten un alto nivel de aprovechamiento de los recursos de almacenamiento y un gran desarrollo para los sistemas de información. Ya que se pueden compartir lotes de información entre todos los usuarios de la red, pudiendo éstos, procesar la información de la manera en que ellos la requieran.

Como es lógico, con el nacimiento de los sistemas de redes, surgen nuevas ideas para hacer más provechoso y eficiente este sistema, lo cual nos lleva en nuestros días a tener diferentes modos de mandar información entre las diversas terminales conectadas a la red, además de distintos sistemas de seguridad de la información y de acceso a los mismos.

Redes

Al surgir las redes se desarrollaron diferentes maneras de comunicación de datos entre las diferentes terminales de la red (TOPOLOGIAS Y PROTOCOLOS). Esto es para hacer más eficiente dicha comunicación, tanto en velocidad de transmisión como seguridad de información, en el envío del sistema central a cualquiera de las terminales o viceversa.

Protocolos

Los protocolos son procedimientos estandar y reglas de comunicación de datos, siendo sus funciones principales:

1. - Establecer una conexión entre el emisor y el receptor.
2. - Transferencia de información.
3. - Verificación y control de errores.

Existen en la actualidad dos tipos de protocolos basicamente: ***Carrier Sense Multiple Access CSMA*** y ***Token Passing***. Ambos protocolos se basan en la idea de transmitir paquetes de información. Cada uno de estos paquetes debe tener carácteres de control, los cuales lo identifican como parte del mensaje de su terminal de origen y su terminal de destino.

Topologías

Este término se refiere a la forma en que se conectan las terminales que pertenecen a la red, tratando de optimizar rutas de transmisión y cableado de red. En la actualidad existen dos tipos de topologías principales: ***Bus*** y ***Token Ring***.

Dependiendo de las topologías y los protocolos usados, encontramos diferentes tipos de redes, entre las cuales podemos mencionar ***Ethernet*** y ***Arcnet***.

Ethernet

Su topología es de ***Bus*** en línea, cuenta con uno o varios servidores (***File Servers***), los cuales controlan el uso de todos los dispositivos que se encuentran en la red para ser compartidos entre todos los usuarios. El ***Bus*** es el único canal de comunicación entre los servidores y todas las estaciones de trabajo o terminales que se encuentran conectadas.

A través de este ***Bus***, las señales de comunicación y paquetes de información son transmitidos, y cada una de las terminales conectadas al ***Bus*** detectan si algún paquete de información viene dirigido a ellas. Una terminal sólo puede comunicarse cuando la línea se encuentra libre, si no es así tendrá que esperar, siendo ésta la característica principal del protocolo ***CSMA***.

Arcnet

La topología usada por esta red es *Token Ring*, la cual conecta a todas las terminales en forma circular, pero con ayuda de una caja repetidora de señales, a la cual se conectan tanto el *File Server* como las terminales. El objetivo de esta caja repetidora de señales es no perder la comunicación entre las diferentes terminales de la red en caso de que se atenúe la señal, así como hacer la distribución de la señal a las diferentes terminales enlazadas.

Su protocolo de comunicación es *Token Passing*. *Token*, es un paquete de información que circula entre todas las terminales de la red, si una terminal no tiene mensaje que mandar, este *Token* es pasado a la siguiente terminal. Si alguna de las terminales conectadas a la red desea enviar alguna información o mensaje, lo agregan al *Token* y éste sigue pasando a la siguiente terminal, hasta que llegue a la terminal destino, la cual recibirá el mensaje, lo tomará y volverá a dejar pasar el *Token* a la siguiente terminal de la red.

Introducción al *hardware* para conexiones *Ethernet*

La siguiente lista de componentes pueden ser usados según se requiera en configuraciones *Ethernet*:

Adaptadores: Son usados para interconectar cable grueso, cable delgado y transductores.

Conectores BNC: Son usados con cable delgado en cada extremo del cable.

Conectores DIX: Son conectores de 15 *pins*, en una tarjeta ***EtherLink*** es usado para conectar una estación de trabajo con un transductor externo (usando un cable grueso).

Tarjetas para red: Es un circuito impreso que se inserta dentro de cada estación de la red y que sirve para la comunicación.

Ethernet cable delgado: Es un cable coaxial de 0.2 pulgadas, RG-58A/U, de 50 ohms.

Conectores BNC tipo T: Los dos extremos del conector tipo T sirven como unión con el conector BNC del cable delgado; el conector sobrante sirve para conectarse con la tarjeta para red de cada estación.

Terminadores BNC: Es un conector BNC de 50 ohms, sirve para terminar el circuito de una red y es colocado en uno de los extremos del conector tipo T. También existen terminadores con un cable para tierra.

Segmento de cable: Este término se refiere al número de estaciones conectadas en el segmento de la línea. Una red puede

consistir de un segmento de cable o de varios segmentos. Se define como segmento a la longitud de cable conectada entre dos terminadores.

Repetidor: Es un dispositivo usado para conectar múltiples segmentos de cable en una red de gran longitud. Aproximadamente cada 185 mts debe conectarse un repetidor para extender a una distancia mayor el alcance de la red.

Transductores: Este es un dispositivo que se requiere cuando se usa cable grueso en una conexión ***Ethernet***. Cada transductor es conectado a una estación de trabajo por medio de un cable.

La velocidad de transmisión en redes ***Ethernet*** es de 10 Megabits por segundo.

A continuación presentaremos las características principales de los cables delgado y grueso respectivamente, utilizados en una red ***Ethernet***.

Cable delgado

1.- Máximo número de segmentos de cable, 5.
2.- Máxima longitud de un segmento de cable, 185 metros.
3.- Máxima longitud de cable de la red, 925 metros.
4.- Máximo número de estaciones conectadas a un segmento de cable, 30 (el repetidor se instala entre los segmentos de cable y cuenta como una estación).
5.- Mínima distancia entre conectores BNC tipo T, 0.5 metros.

Regla

1.- Un terminador BNC debe estar en cada extremo del segmento de cable. Uno de los dos terminadores en cada segmento de cable debe estar aterrizado.

Los elementos del *hardware* y la forma de conexión, utilizando cable delgado, se muestran en las figuras 2.1 a 2.7 de las páginas siguientes.

Cable grueso: el cable grueso es de 0.4 pulgadas de diámetro y de 50 ohms de impedancia. Los conectores serie N machos se instalan en cada extremo del cable grueso.

Terminadores serie N: Son terminadores de 50 ohms que al igual que los terminadores para cable delgado, nos sirven como el cierre del circuito de la red y van conectados en cada uno de los extremos del cable grueso, algunos cuentan con un cable para ser aterrizados.

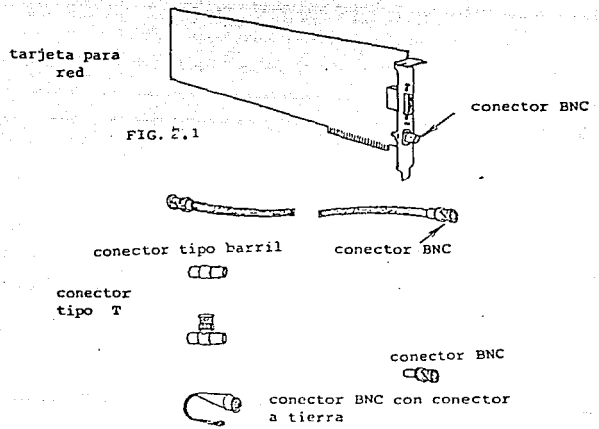
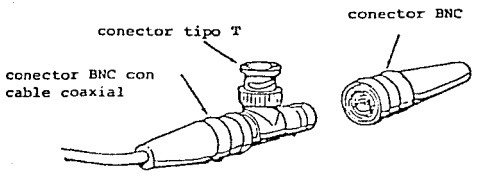


FIG. 2.1

ELEMENTOS DE HARDWARE PARA CABLE DELGADO EN CONEXIONES ETHERNET PARA LA RED NETWARE

FIG. 2.2



MUESTRA DE CONEXIONES DE ELEMENTOS UTILIZADOS EN UNA RED ETHERNET CON CABLE DELGADO.

FIG. 2.3

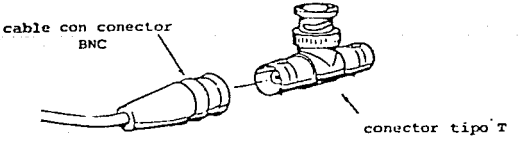
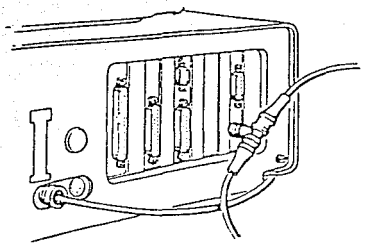
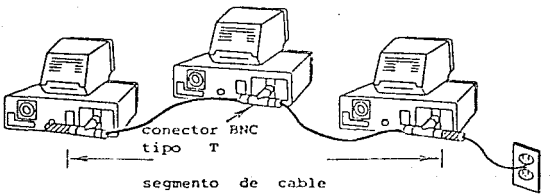


FIG. 2.4



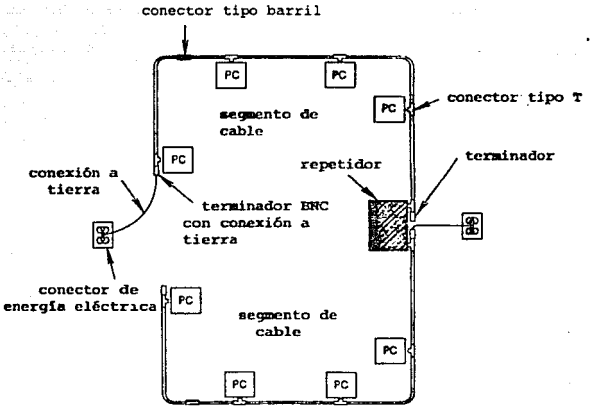
MUESTRA DE CONEXION DEL CABLE ETHERNET
CON LA TARJERA DE COMUNICACIONES

FIG. 2.5



EJEMPLO DE UNA RED ETHERNET CON CABLE DELGADO
EN UNA RED NETWARE

FIG. 2.6



EJEMPLO DE CONEXION CON CABLE DELGADO
EN UNA RED ETHERNET DE NETWARE

FIG. 2.7

Conectores tipo barril: este conector permite unir dos extremos de cable.

Cable grueso

1.- Máximo número de segmentos de cable, 5.
2.- Máxima longitud de un segmento de cable, 500 metros.
3.- Máxima longitud de cable en la red, 2500 metros.
4.- Máximo número de estaciones conectadas a un segmento de cable, 100 (el repetidor se instala entre cada segmento de cable y cuenta como una estación).
5.- Mínima distancia entre transductores, 2.5 metros.
6.- Máxima distancia del transductor a la estación, 50 metros.

Regla

1.- Un terminador debe ser conectado en cada extremo del segmento del cable. Uno de los dos terminadores debe estar aterrizado.

Los elementos de *hardware* y la forma de conexión, utilizando cable grueso, se muestran en las figuras 2.8 a la 2.11 en las páginas siguientes.

Combinación de Cable Grueso y Cable Delgado

Podemos crear una combinación de cable grueso con cable delgado en una red ***Ethernet***, usando esta combinación podemos disminuir costos, ya que si se requiere de una red que incluya una gran distancia se puede tener esta combinación.

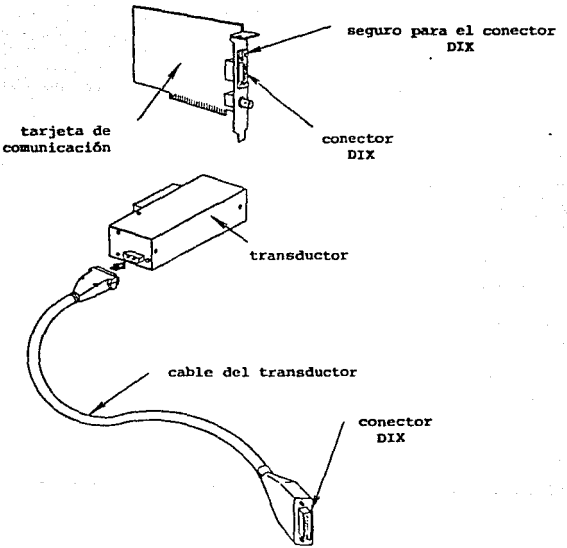
La combinación entre cable grueso y cable delgado puede darse entre los 607 pies y 1,640 pies, la mínima longitud es de 607 pies, debido a que el cable que se puede usar con esta longitud es el delgado, exclusivamente, con el cable grueso es de 1,640 pies. Cabe aclarar que las características anteriores son para un segmento de cable.

Para saber la longitud máxima del cable delgado que podemos usar en esta combinación usaremos la siguiente ecuación:

$$\frac{1{,}640 \text{ pies} - L}{3.28} = t$$

L = Longitud del segmento de cable que vamos a usar.
t = Máxima longitud del cable delgado que vamos a usar.

Un ejemplo de la forma de conexión de esta combinación se muestra en la figura 2.12.



ELEMENTOS DE HARDWARE PARA CABLE GRUESO
EN REDES ETHERNET DE NETWARE.

FIG. 2.8

CABLE GRUESO PARA LA RED ETHERNET

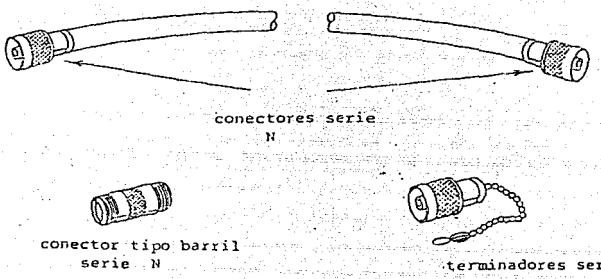
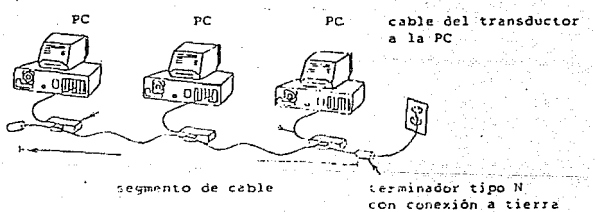
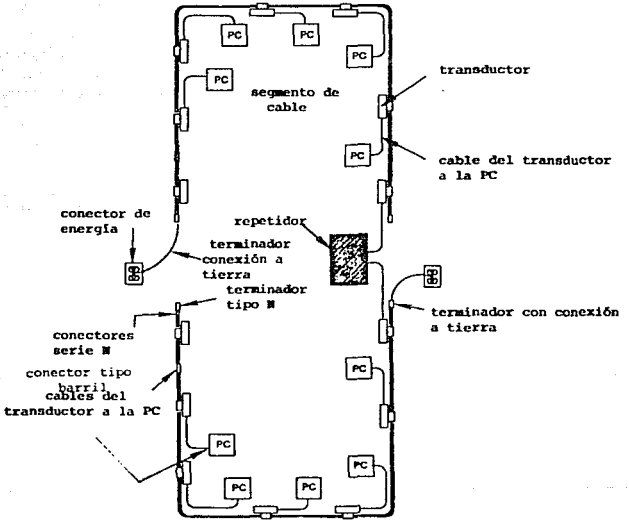


FIG. 2.9



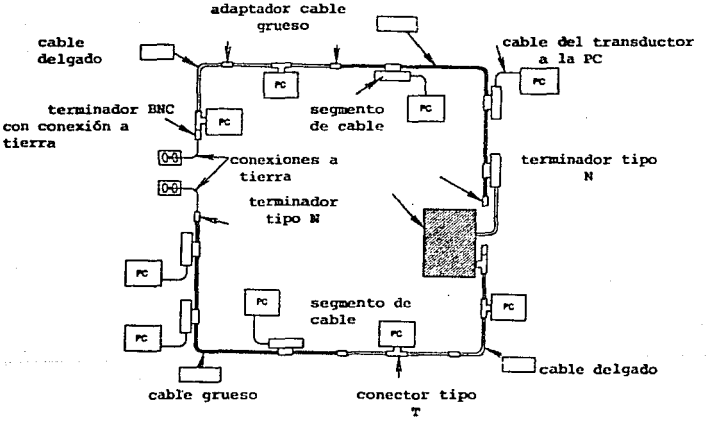
EJEMPLO DE UNA CONEXION CON CABLE GRUESO EN UNA RED ETHERNET DE NETWARE

FIG. 2.10



EJEMPLO DE UNA CONEXION CON CABLE GRUESO EN UNA RED ETHERNET DE NETWARE

FIG. 2.11



EJEMPLO DE UNA COMBINACION DE CABLE GRUESO CON CABLE DELGADO EN UNA RED ETHERNER DE NETWARE.

FIG. 2.12

Introducción al *hardware* para conexiones *Arcnet*

En una red *Arcnet* se usan los siguientes componentes de ***hardware***:

Cables

Las estaciones de trabajo están conectadas con un cable coaxial RG-62/U de 93 ohms con un conector macho BNC a cada extremo.

Repetidores

Cualquier repetidor activo tiene 8 puertos. Este repetidor se utiliza para acondicionar, amplificar y rutear la señal. El repetidor pasivo tiene cuatro puertos y se utiliza unicamente para rutear la señal.

Advertencia

Dos puertos del repetidor nunca deben estar conectados entre ellos.

Reglas para conectar estaciones de trabajo.

Para instalar la red hay que colocar las estaciones y los repetidores en los lugares asignados y conectar los cables utilizando los conectores BNC.

Además existe una serie de reglas para conectar los repetidores y las estaciones.

Reglas para utilizar los Puertos del Repetidor Activo:

Un puerto de un repetidor activo puede conectarse al puerto de otro repetidor activo o a una estación, a una distancia máxima de 2000 pies. La distancia máxima entra las estaciones ubicadas en los extremos de la red es de 20,000 pies, lo que implica que se pueden conectar hasta diez (10) repetidores en serie a lo largo de la red.

Reglas para utilizar los Puertos del Repetidor Pasivo:

Un puerto de un repetidor pasivo puede ser conectado a un puerto de otro repetidor activo, la distancia máxima entre estos repetidores es de 100 pies, siendo ésta la misma para conectar un puerto de un repetidor pasivo a una estación de trabajo.

Otra caracteristica de la red tipo *Arcnet* es que la velocidad de transmisión es de 2.5 Megabits por segundo.

Requerimientos de hardware para una Red con Sistema Operativo Netware.

Servidor de archivos

Las siguientes máquinas y procesadores son soportados por *Netware*, para sus diferentes versiones de sistemas operativos:

1. - IBM PC AT o compatibles con procesadores 80286 y 80386 de INTEL.
2. - IBM PS/2, modelos 30-286, 50, 55, 60, 70 y 80.

Requerimientos de Memoria

Los siguientes requerimientos son aproximados, dependiendo de la capacidad del disco o discos usados:

Server no dedicado (es aquel que se puede usar como estación principal y a su vez funcionar como terminal) 1 Mb.

Server dedicado (es aquel que sólo funciona como estación principal) 2 Mb.

LAN Drivers (*Local Area Network Drivers*): es el software que permite el manejo de las interfaces de comunicación de la red.

Los *LAN Drivers* soportados por *Netware* son los siguientes:

1. - IBM *Token Ring*
2. - *Netware Ethernet* NE1000
3. - *Netware Ethernet* NE2000
4. - *Netware* NE/2 *Ethernet*
5. - *Netware* RX-Net
6. - *Netware* RX-Net/2
7. - 3Com 3C501 *Etherlink*
8. - 3Com 3C503 *Etherlink II*
9. - SMC *or Pure Data ARCNET*
10. - Otros *drivers* certificados por Novell

Estaciones de trabajo

La siguiente lista es soportada por *Novell* como estaciones de trabajo:

1. - IBM PC AT o compatibles con procesadores 80286 y 80386 de INTEL.
2. - IBM PS/2, modelos 30-286, 50, 50z, 55, 60, 70 y 80.
3. - IBM PC XT y compatibles.

Para las estaciones de trabajo se requiere de 70 KB de memoria para el *software* y la demás memoria depende del usuario.

Lo anterior fué descrito para instalar redes de área local (LAN), sin embargo para realizar comunicaciones entre estas redes a grandes distancias (redes WAN: *Wide Aperture Netware*) se requiere, tanto ***hardware*** como ***software*** adicional como el que a continuación se describe.

Software

El ***software*** requerido para esta comunicación puede ser el recomendado por *Novell* y es *Netware Link/ Async*, el cual permite la comunicación entre redes locales por medio de una tarjeta incluye un ***Modem***, o por el puerto serial, existente en la mayoría de las máquinas, ya que controla el acceso a las diferentes redes conectadas entre sí vía remota.

Hardware para comunicaciones

Para una red de área grande (*WAN*), se tiene una tarjeta adaptadora para la comunicación entre las redes locales, llamada WNIM+, es un multipuerto de 8 *bits* para una comunicación asincrona, o un controlador de puerto serial inteligente. La WNIM+ opera conjuntamente con *Novell Netware Access Server, Netware Link/Async, y Netware Asynchronous Comunication Server (NACS)*. Esta tarjeta contiene un procesador Z-80, con 512 KB de memoria y cuatro puertos asincronos, cada uno con conector RS-232C. Cada puerto puede operar hasta un máximo de 19.2 *Kbits/s*. En la figura 2.13 se muestra la forma de conexión en este tipo de redes.

Modems

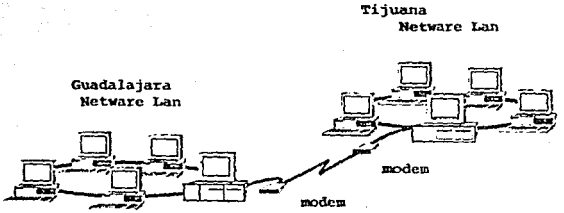
La función de transformación de señales en la rama de planeación de datos se realiza mediante dos procesos básicos.

Codificación: El tren de datos recibidos de una estación de trabajo, cuya sucesión de simbolos dependerá de la información a transmitir, se transformará en una señal compatible con la línea de transmisión utilizada. Este proceso recibe el nombre de codificación.

Modulación: Proceso mediante el cual, a partir de un tren de datos, se generan señales analógicas en la línea de transmisión, a base de modificar, en función de la señal de entrada, alguno de los parámetros que definen una onda senoidal pura (llamada portadora), de la forma $A \cos(2ft - \phi)$. Lo anterior da lugar a tres sistemas básicos de modulación:

1.- De amplitud ó ASK (***Amplitud-Shift-Keying***). A cada valor de la señal de entrada se hace corresponder otro de la amplitud (A) de la portadora.

2.- De frecuencia ó FSK (***Frecuency-Shift-Keying***), que consiste en variar la frecuencia de la portadora (F) en función de la señal de entrada.



MUESTRA DE CONEXION REMOTA ENTRE
DOS CIUDADES

FIG. 2.13

3. - De fase ó PSK (*Phase-Shift-Keying*), en cuyo caso se provocan saltos bruscos y predeterminados en la fase de la portadora, de acuerdo con la señal de entrada.

Estos tipos de modulación, descritos en forma muy elemental, tienen en la práctica matices más complejos, sobre todo, cuando se utilizan varios niveles o se producen modulaciones de tipo mixto.

En la rama de recepción, la reconversión de las señales procedentes de la línea se realiza mediante el siguiente proceso:

Demodulación: Es el proceso inverso a la modulación y, como tal, consiste en reconstruir, a partir de la señal recibida de la línea, el tren de datos que la originó. El problema estriba en que en el lugar de recepción de datos se debe decidir en qué instante se produce la transición de un estado a otro, con base en una señal (la recibida) que no es exactamente igual a la que salió del modulador distante, ya que ha sufrido los efectos nocivos de la transmisión (distorsiones, ruidos, etc.). El error que se produzca en esta decisión, respecto al instante real, determinará el grado de distorsión de la señal de datos reconstruida e influirá en la probabilidad de error en el reconocimiento final de la misma.

La modulación puede ser coherente o no coherente, según que el receptor posea o no una referencia en la onda portadora con la cual puede ponerse en fase.

Decodificación

Finalmente se ha de producir una operación inversa a la codificación que se realizó en el transmisor, con lo que se obtiene el tren de datos original.

Normalmente, los procesos en el transmisor, así como en el receptor de datos, se realizan en un único conjunto físico que recibe el nombre de *MODEM*, contracción de Modulador-Demodulador, por ser éstas las dos funciones más generalizadas.

Recomendaciones

Uno de los principales cuidados que se debe tener en cuenta, es el tipo de instalación en donde operará el equipo de cómputo, así como la selección del equipo a utilizar y más aún, si el equipo funciona las 24 horas del día y 365 días por año.

El problema de la selección de equipos de cómputo tiene matices profundamente subjetivos. A fin de poder realizar dicha selección de la forma más objetiva posible, se expone a continuación una secuencia de parámetros de cuyo análisis y posterior calificación puede derivarse una elección adecuada:

1.- Compatibilidad total con el equipo de cómputo existente en la empresa.

2.- Posibilidad de crecimiento por lo menos al doble de la capacidad solicitada sin que se degrade.

3.- Interconexión con el equipo de cómputo existente.

Otros cuidados

Verificación del Ambiente donde va a Operar el Sistema y Requerimientos de Equipos para la Energía Eléctrica.

Para que el *hardware* funcione adecuadamente en cada uno de sus componentes, es indispensable tener un ambiente propio para su funcionamiento, así como la fuente de energía apropiada, para ello es necesario verificar antes de cualquier instalación lo siguiente:

1.- Temperatura y humedad
2.- Máxima altitud
3.- Fuente de energía
4.- Frecuencia
5.- Requerimientos de energía

Se recomienda usar una línea dedicada de energía con su tierra, es decir, de tres hilos. Debido a que los componentes del ***hardware*** son muy sensibles a los cambios de energía, se pueden utilizar tres tipos de dispositivos para regular el voltaje.

Regulador.- Proteje al equipo contra las variaciones de energía.

Aislamiento.- Permite eliminar la carga estática existente en el medio ambiente y debida a factores diversos.

Fuente de poder ininterrumpible (UPS). Proteje al servidor de archivos, así como a las estaciones de trabajo, para que los archivos sean cerrados adecuadamente en el caso de que la línea comercial falle. La UPS asegura que todos los datos que se encuentran en RAM, durante una falla de energía, no se pierdan.

Hardware utilizado en la Red de la Línea de Autobuses Tres Estrellas de Oro S.A. de C.V.

Se requirió de un estudio para determinar el número de equipos a utilizar y para definir las ubicaciones de cada estación de trabajo, así como del servidor de archivos. Una vez determinados estos se pudo conocer la fuente de poder ininterrumpible adecuada, así como los requerimientos de tipo de cable utilizado y los ductos adecuados, tanto para la energía eléctrica como para el de la señal de la red. Los equipos seleccionados fueron los siguientes:

1.- Doce terminales o estaciones de trabajo AT con procesador 80286 y un servidor 80286.

Con un consumo de energía en su CPU y teclado de 0.5 amperes, y 0.3 A del monitor a 127 volts, lo cual nos da un consumo total de 101.6 volts amperes, más un 20 % de tolerancia, nos da un total aproximado de 122 volts amperes por cada equipo.

2.- Doce impresoras con un consumo de energía de 120 volts y 3 amperes, lo cual nos da un total de 360 volts amperes, más un 20 % de tolerancia, nos da un total de 432 volts amperes por cada equipo.

3.- Fuente de poder ininterrumpible requerida: se dicidió tener 2 UPS para mayor protección, ya que si alguno falla, el otro estará funcionando. La capacidad necesaria a cubrir es de 6,770 KVA.

4.- El tipo de red utilizado fué *Ethernet*, configuración del tipo *Bus*. Esta configuración permite tener una velocidad de transmisión de 10 megabits por segundo, por lo que se usaron trece tarjetas *Ethernet*.

5.- 120 metros de cable delgado. Cabe mencionar que el tipo de configuración nos permite tener un sólo cable en el cual se conectan tanto el *server* como las estaciones de trabajo, lo que permite un ahorro de cable y el uso de ductos más pequeños.

6.- Trece conectores tipo T, uno para cada estación de trabajo.

7.- Dos terminadores, uno en cada extremo del segmento de cable.

8.- Una línea de energía eléctrica dedicada para la UPS.

La primera etapa comprende la instalación del equipo mencionado anteriormente en la Central Norte de la línea de transportes Tres Estrellas de Oro S.A. de C.V.

La segunda etapa consiste en la instalación del equipo en la Central Tijuana, la cual consta de un equipo semejante al de la Central Norte.

La tercera etapa comprende la instalación del equipo en la central Guadalajara, donde los requerimientos de energía son un poco diferentes, ya que la irregularidad del servicio público de ésta puede ocasionar un mal funcionamiento en las operaciones en el sistema de cómputo. Es por eso que se requirió de una instalación de una planta eléctrica en esta central.

La cuarta etapa consiste en la interconexión de dichas centrales, mediante líneas de comunicación.

Costos aproximados por cada red instalada

Equipo	Costo
Un servidor	$ 12,000,000
Doce estaciones de trabajo .	$ 22,140,000
Trece tarjetas *Etehrnet*	$ 6,435,000
Dos terminadores de 50 ohms .	$ 18,000
Doce impresoras	$ 16,740,000
Cien metros de cable coaxial delgado	$ 300,000
Trece conectores tipo T	$ 156,000
Dos modems	$ 12,000,000
Dos fuentes de poder ininterrumpibles	$ 28,000,000
Instalación del aire acondicionado, instalación de ductos para cable coaxial e instalación eléctrica	$ 18,000,000
TOTAL	$115,789,000

Cabe hacer notar que nosotros recomendamos en lugar del equipo utilizado como servidor, un equipo vax, el cual hará las funciones de este.

El equipo que a continuación describiremos, se recomienda con base a la estructura organizacional de la empresa en donde se implementará el sistema de telereservaciones, ya que cuentan con equipo Digital en su centro de cómputo y con ello se logra una mayor homogeneidad en el equipo de la empresa. Además de que Digital cuenta con un gran respaldo a nivel internacional y entre sus cualidades esta el *software*, debido a que éste esta clasificado dentro de los mejores a nivel mundial. Además la empresa cuenta con la red más grande del mundo.

El equipo VAX se va a utilizar junto con un paquete que lo hace funcionar como servidor de archivos (PCSA), el cual trabaja en el ambiente MS-DOS (simula el ambiente DOS, pero trabaja en el VMS), por lo que el sistema VAX se utiliza como servidor con este paquete y nos puede servir como sistema multiusuario.

Para recomendar el siguiente equipo se tomaron en cuenta los siguientes puntos:

1.- Definición de las necesidades del sistema.

2.- Dividir las necesidades del sistema en:

a. Aplicación del *software*
b. Sistema Operativo y lenguajes de programación
c. *Hardware*

3.- Tener claramente identificada la aplicación del sistema.

4.- Identificar claramente el (los) lenguaje(s) requerido(s).

5.- Determinar si el sistema operativo soporta el (los) lenguaje(s) requerido(s).

Se determinó que para ello se puede utilizar perfectamente una computadora VAX de DEC (*Digital Equipment Corporation*), ya que puede cumplir con los siguientes puntos:

a. Las características de la aplicación son:

- Número de usuarios, 12
- Número de usuarios concurrentes, 12
- Respuesta esperada de los usuarios
- Tiempo real requerido
- CPU vs. I/O requeridos
- Comportamiento de datos
- Disponibilidad del equipo

b. Atributos del sistema

- Funcionabilidad
- *Performance*
- Precio
- Procesador

Atributos de la Computadora VAX

El CPU VAX exhibe administración de memoria virtual, cargador *bootstrap*, instrucciones estandard para decimales empacados, aritméticos del punto fijo y flotante, carácteres y manipulación de carácteres, dos memorias caché escritas a través del mapeo directo de 64 KB, reloj de tiempo real programable de alta precisión, reloj de tiempo año, con respaldo de batería y dos memorias de control de 16 KWORD. La CPU también incluye el control de memoria y respaldo de batería a memoria para 128 MB completos de capacidad de memoria, un puerto VAX *cluster*, un puerto *Ethernet* y dos canales VAXBI. La CPU también incluye un subsistema de consola basado sobre un conjunto de *chips* con terminales de video, disco *Winchester* de 30 MB, un *floppy* RX500 y puerto de diagnóstico remoto.

Los sistemas VAX están disponibles con un sistema operativo VMS que suministra un ambiente seguro de alto rendimiento para la ejecución concurrente de tiempo compartido multiusuario, batch y aplicaciones en tiempo real.

El sistema operativo soporta simultáneamente, tareas por lotes en tiempo real interactivas y desarrollo de programas. Además, proporciona gran variedad de lenguajes de alto nivel y programas de gran utilidad.

VAX es una familia de sistemas de computación de memoria virtual multiprogramación y 32 *bits* ofrecida por ***Digital Equipment Corporation (DEC).***

El diseño de la VAX ha sido optimizado para multiprocesamiento acelerador de punto flotante. Dispone de un CPU que tiene 64 KB de memoria caché.

Todos los procesadores VAX están implementados con una arquitectura de 32 *bits*, un conjunto de instrucciones extensivo con numerosos tipos de datos y una estructura de *bus* de 32 *bits* para una alta capacidad de direcciones virtuales de 4 Gbytes, un sistema operativo VMS, equipo de *software* de tiempo real VAXELN, un subsistema de consola, el cual permite al usuario administrar el sistema ó a los ingenieros de servicio comunicarse con el sistema a través de la terminal de consola.

Particularizando en los equipos microvax 3300/3400 son sistemas de tiempo compartido junto con el nuevo RF30 integrado al elemento de almacenamiento (ISE), permiten el manejo de información en tiempo real. Estos nuevos sistemas ofrecen más de un y medio del ***performance*** de la micro VAX II.

Como todos los sistemas VAX, estos modelos ofrecen una compatibilidad total del *software* con los demás miembros de la familia VAX. Los sistemas 3300 y 3400 en el CPU tienen la capacidad de manejo con números de punto flotante, una conexión *Ethernet* para cable grueso y delgado, un controlador de memoria de 4 a 12 Mb, Un disco duro de 150 Mb, una unidad de cinta de 270 Mb e incluyen además el *software* del sistema operativo.

Software

Uno de los componentes básicos del desarrollo del Sistema de Telereservaciones es precisamente el *software*, tanto en el que está basada la programación (plataforma de desarrollo) como el de la misma programación. En este tema veremos las diferentes características, así como las diferentes alternativas de *software* apropiado para el proyecto, destacando las ventajas y desventajas de utilizar algún sistema operativo o lenguaje de programación.

Introducción

Durante las tres primeras décadas de la computación, el principal reto era desarrollar el ***hardware*** de computadoras de forma que se redujeran los costos de procesamiento y almacenamiento de la información. Durante la década de los 80's, los avances en electrónica dieron como resultado una mayor capacidad de cálculo y una reducción de los costos.

En la actualidad esto ha cambiado. El principal reto es reducir el costo y mejorar la calidad de las diferentes aplicaciones de *software*.

La capacidad de las grandes computadoras de ayer está disponible hoy en día en un simple circuito integrado. Las impresionantes capacidades de procesamiento y almacenamiento moderno nos da un gran potencial de cálculo, pero el *software* es el elemento que nos permite explotar este potencial.

El desarrollo del *software* está fuertemente ligado a las cuatro décadas de evolución de los sistemas de cómputo. Un *performance* más alto del *hardware*, un tamaño menor y un costo más bajo han dado lugar a sistemas de cómputo más sofisticados. La tecnología se ha trasladado de procesadores con tubos de vacío a dispositivos electrónicos. En algunos libros se ha denominado a la revolución de las computadoras como la nueva revolución industrial.

Durante los primeros años del desarrollo de computadoras, el *hardware* sufrió continuos cambios mientras que el *software* se veía simplemente como un añadido. La programación de computadoras se consideraba un arte para lo que existían pocos métodos sistemáticos. El *software* se desarrollaba sin ninguna planificación aunque existieran problemas en el costo por la demanda de una planificación adecuada. En la mayoría de los sistemas se utilizaba una orientación por lotes, es decir, la ejecución de un único programa que se dedicaba a una aplicación específica.

La producción de *software* (programas desarrollados para ser vendidos a uno o más clientes) estaba en su infancia. Quien había desarrollado el *software* era el que lo utilizaba y depuraba, el diseño era un proceso implícito, ejecutado en la cabeza de alguien y normalmente no existía documentación.

La segunda era de la evolución de sistemas de cómputo va de la mitad de la década de los 60's hasta finales de los 70's. La multiprogramación, sistemas multiusuario, introdujeron nuevos conceptos de interacción hombre-máquina. Las técnicas interactivas abrieron un nuevo mundo de aplicaciones y nuevos niveles de sofisticación del *software*. Los avances en los dispositivos de almacenamiento en línea condujeron a la primera generación de sistemas de bases de datos. Esta era se caracterizó también por la llegada de las casas de *software*.

Conforme creció el número de sistemas de información comenzaron a extenderse las bibliotecas de *software* de computadoras. Pero empezaron a surgir problemas, ya que todos estos programas tenían que ser corregidos cuando se detectaran fallas, o por cambios en los requerimientos del usuario, lo cual llevó a denominar estas actividades como el mantenimiento del *software*, y a menudo estas actividades empezaron a requerir de un gran porcentaje de tiempo en comparación con otras actividades de desarrollos nuevos por lo que empezó a aparecer la crisis del *software*.

La tercera era de la evolución de los sistemas de cómputo comenzó a mediados de los 70's y continua hasta la fecha. Las redes

de área local y global, las comunicaciones digitales de alto ancho de banda, supusieron una fuerte presión sobre los desarrolladores de *software*. Esta era se caracteriza también por el amplio uso de microprocesadores y computadoras personales.

LENGUAJES DE PROGRAMACION

Dbase IV

Dbase IV es un manejador de bases de datos y lenguaje de programación muy simple y en la actualidad ha ganado un gran terreno debido a su flexibilidad para el desarrollo de aplicaciones.

Recientemente DBASE ha incorporado un método muy eficiente y rápido para manejar subconjuntos de información.

Este método se basa en el manejo de índices, los cuales son muy utilizados para solucionar ciertos procesos que requieren de un subconjunto de registros. Por ejemplo, si se desea contar el número de proveedores registrados en la ciudad de Querétaro y se tiene un índice que clasifica su información por ciudad, se puede hacer de la siguiente forma:

```
USE PROV
INDEX ON UPPER(ciudad) TAG ciudad+
SEEK 'QUER'
contador=0
DO WHILE UPPER(ciudad) $ 'QUERETARO'
  contador=contador+1
  skip
ENDDO
? 'Numero de proveedores en Queretaro: ' + ;
LTRIM(STR(contador))
```

Lo cual sería mucho más rápido que utilizar el comando COUNT ALL FOR pues se tendría que evaluar la condición para cada uno de los registros en la base de datos como si se usara el comando SET FILTER TO.

Como se mencionó anteriormente, esto sólo es perceptible si la base de datos es suficientemente grande, de no ser así prácticamente no habría diferencia entre uno y otro método.

Ahora bien, la nueva alternativa que ofrece DBASE IV 1.1 es la de crear índices condicionales, así únicamente los registros que cumplan con determinada condición serán incluidos en el índice, una vez creado el índice no será necesario volver a realizar ninguna comparación para determinar si el registro pertenece o no al subconjunto, pertenecerán aquellos que estén indexados.

Inclusive, si se añade un nuevo registro a la base de datos que no cumpla con la condición especificada en el índice, este nuevo registro existirá en la base de datos pero no será posible visualizarlo hasta no desactivar el índice.

Los índices condicionales deben ser del tipo MDX, desafortunadamente no pueden ser del tipo NDX. De cualquier forma son una herramienta muy importante y una alternativa eficiente para sustituir en ciertos procesos al SET FILTER TO.

Si se quisiera consultar dinámicamente los proveedores establecidos en Querétaro se puede usar:

```
USE PROV
INDEX ON ciudad TAG ciudad FOR UPPER(ciudad) $ ;
'QUERETARO'
BROWSE
```

De esta forma sólo serán presentados los proveedores seleccionados en la tabla del BROWSE sin ningún tipo de retraso en tiempo por haber utilizado un subconjunto de la base de datos.

Clipper

Nantucket Corporation siempre ha incluido entre las utilerías de su compilador, un ligador que encadena los diferentes objetos desarrollados para generar un archivo ejecutable.

Por muchos años el ligador incluido en Clipper era el Plink86, un ligador que en su tiempo fué bastante bueno y ofrecía como principal ventaja el poder generar *overlays* para reducir el espacio en memoria requerido por una aplicación.

Con el desarrollo de la tecnología han surgido en el mercado nuevos ligadores, los cuales superan con mucho las ventajas ofrecidas por Plink86, debido a esto, Nantucket decidió cambiar de ligador para ofrecer en la versión 5.0 de Clipper una herramienta más sofisticada y poderosa, Rtlink.

Inclusive, antes de la aparición de Clipper 5.0 muchos desarrolladores utilizaban otros ligadores diferentes a Plink86, sobre todo por cuestiones de velocidad en el ligado, ejemplos de éstos son: Tlink, Alink y Blinker, los cuales son mucho más rápidos. Los dos últimos ofrecen ventajas adicionales para el manejo de la aplicación en memoria, requiriendo menos espacio. Sin embargo, Nantucket decidió incluir en su nueva versión de Clipper un ligador diferente a los mencionados anteriormente, la razón se fundamenta en un análisis profundo de la ventaja ofrecida por todos estos productos y como resultado, según Nantucket, el ligador con las mejores características para ligar los objetos de Clipper resultó ser Rtlink.

Pocket Soft, Inc. creador de Rtlink, no es una empresa nueva en el mercado de ligadores, desde hace tiempo ha venido comercializando su producto y ahora en colaboración con Nantucket ha desarrollado una versión especial para ligar Clipper 5.0 en la forma más eficiente posible, ofreciendo en general las mayores ventajas en comparación con los demás ligadores.

La versión de Rtlink incluida en Clipper 5.0, es diferente a la versión comercial, aunque recientemente Pocket Soft ha liberado al mercado una nueva versión denominada Rtlink plus 4.1, incluyendo las características ofrecidas en la versión de Clipper 5.0 y algunas otras mejoras.

La característica más sobresaliente de este ligader, es el manejo de *overlays* dinámicos, lo cual significa que ya no es necesario definir en forma manual los *overlays*, pues el mismo ligador se encarga de hacerlo en la forma más eficiente posible, consiguiendo desarrollar aplicaciones de gran tamaño, posibles de ejecutarse en espacios limitados de memoria.

El número de *overlays* generados en forma dinámica puede ser muy grande y estos pueden ser tan pequeños (4 Kbytes) que en la práctica las aplicaciones pueden correr en ambientes de memoria muy restringidos.

Existe una desventaja en este tipo de *overlays*, debido al constante intercambio de información entre la memoria y el disco (carga y descarga de *overlays*) , la velocidad de ejecución de la aplicación se deteriora en un porcentaje, aunque mínimo, perceptible.

Esto quiere decir que una aplicación compilada con summer'87 puede ejecutarse en forma más rápida que con Clipper 5.0, aunque la misma aplicación en summer'87 podría requerir mayor espacio en memoria que la compilada con Clipper 5.0, en otras palabras, es preciso pagar un precio (velocidad) por espacio en memoria.

Existe siempre la posibilidad de ligar una aplicación sin utilizar *overlays* dinámicos, pudiendo definir estos en forma manual y por lo tanto aumentar la velocidad de ejecución, este tipo de flexibilidad es una de las características por las cuales Rtlink fué elegido por Nantucket.

Quizás una de las características más importantes en un ligador es la velocidad para generar el ejecutable, Plink86 es extremadamente lento y por lo mismo nunca fué el favorito entre los programadores, sobre todo en los periodos de desarrollo cuando se liga con mucha frecuencia.

Para solucionar este problema, Rtlink ofrece dos características de gran importancia, las cuales dan como resultado tiempos de ligado bastante aceptable (sin llegar a ser iguales a Tlink) pero muy superiores a Plink86.

La primera de estas caracteristicas es lo que se conoce como librerias preligadas, como su nombre lo indica, son una recolección de rutinas generales que serán utilizadas por todas las aplicaciones.

Estas rutinas generales son las mismas librerias de Clipper: Clipper.lib y Extend.lib (además de algunas otras nuevas librerias generales incluidas en Clipper 5.0) las cuales se requieren para cualquiera de sus aplicaciones. Siempre que se liga una aplicación, se debe especificar las librerias en donde están las rutinas generales de la aplicación. Lo cual significa que el ligador debe procesar esta información cada vez que se realice un ligado y por lo tanto el tiempo de proceso aumenta.

Esta recolección de rutinas generales se liga una sola vez con un procedimiento especial, obteniendo como resultado un archivo denominado PLL (pre-linked library), del cual cualquier aplicación (basada en esta libreria) puede extraer el código ejecutable necesario para realizar determinado proceso.

Así solamente será necesario ligar los programas que se hayan desarrollado sin incluir las librerias generales, lo cual significa ahorro de tiempo en el proceso de ligado.

Inclusive, en las librerias pre-ligadas se pueden incluir rutinas generales, desarrolladas por el programador para el control de errores, verificación de impresoras, etc. Las cuales son utilizadas generalmente por todas las aplicaciones.

Entre las peticiones solicitadas para ser incorporadas en futuras versiones de CLIPPER, sobresalió en forma muy significativa la de incorporar arreglos multidimensionales.

Nantucket incorporó los mencionados arreglos y además reestructuró en forma muy significativa el manejo de los mismos, convirtiéndolos en estructuras muy poderosas para el manejo de información, y ahora ofrece posibilidades ilimitadas para el desarrollo de aplicaciones.

En la versión CLIPPER Summer 87, los arreglos eran vectores y el tamaño de éstos vectores debía ser fijado al momento de la declaración, en cualquiera de las siguientes formas:

```
PUBLIC arreglo[10]
PRIVATE arreglo[10]
DECLARE arreglo[10]
```

Cualquiera de estas tres formas crean un arreglo con diez elementos llamado arreglo. La diferencia entre utilizar la primera o las dos siguientes es la visibilidad que tendrá el arreglo. En el primer caso el arreglo podrá ser visible en cualquier programa, procedimiento o función de la aplicación, mientras que en los otros

dos casos, sólo será visible en el procedimiento donde fué declarado y en los que sean llamados de éste.

PRIVATE y DECLARE son declaraciones sinónimas y se recomienda emplear PRIVATE en vez de DECLARE.

Si durante el desarrollo de la aplicación es necesario modificar el tamaño de este arreglo, Summer 87 no ofrece ninguna función o alternativa directa para poder conseguirlo es necesario desarrollar una rutina que en determinado momento podría resultar un tanto enredosa.

En CLIPPER 5.0, las cosas son muy diferentes, en primer lugar, los arreglos son multidimensionales, permitiendo el manejo de estructuras complejas, y además son dinámicos, pues se puede modificar su tamaño en cualquier momento. La forma de declararlos es muy similar a como se venía haciendo en la versión anterior, con la diferencia de que también se pueden declarar como locales y estáticos, ya sea internos o externos. Nuevamente la diferencia radica, fundamentalmente, en la visibilidad que tendrán estos según la forma en que sean declarados, consideraremos el siguiente ejemplo:

```
* Compilar con switch /n
# define n_lon1 10
STATIC a_arr1[n_lon1]

PROCEDURE X1
STATIC a_arr2[8]
.
RETURN

PROCEDURE X2
# define n_lon2 5
LOCAL a_arr3[n_lon2]
.
RETURN

FUNCTION X3
LOCAL a_arr4
.
a_arr4=ARRAY(3)
.
RETURN
```

El arreglo a_arr1, será visible para todos los procedimientos y funciones incluídos en el PRG, en este caso X1, X2 y X3. En el caso de los procedimientos X1 y X2 los arreglos a_arr2 y a_arr3 respectivamente sólo existirán mientras esté activo el procedimiento, en otras palabras, para cuando esté activo el procedimiento X2 o la función X3, el arreglo a_arr2 no existirá.

La función X3 crea una variable local a_arr4, la cual no ha sido inicializada a ningún valor ni tampoco como un arreglo, por lo cual se le asigna automáticamente el valor de NIL y si se utiliza la función VALTYPE(a_arr4) se obtiene como resultado 'U' (inicializada a NIL). Posteriormente, a esta variable se le asigna el resultado de la función ARRAY(n), la cual crea un arreglo con dimensión 'n', y ahora sí se aplica la función VALTYPE(a_arr4) el resultado es 'A' (arreglo).

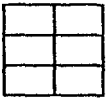
Los arreglos tienen la limitación de poder contener sólo 4096 elementos por dimensión y tantas dimensiones como sea posible mantener en memoria.

CLIPPER 5.0 maneja las dimensiones como arreglos anidados, es decir, que cualquiera de los elementos de un arreglo puede contener un valor o una referencia a otro arreglo. Consideremos el siguiente ejemplo:

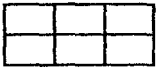
```
LOCAL a_arr1[3,2]
LOCAL a_arr2 [2][3]
```

Ambos casos tratan de un arreglo de dos dimensiones, la forma de declararlos permite utilizar cualquiera de las dos sintaxis mostradas, de cualquier forma, las estructuras declaradas son como sigue:

A_ARR1



A_ARR2



En este caso todos los elementos están inicializados a un valor de NIL, otras alternativas para la creación de estos arreglos pueden ser las siguientes:

```
LOCAL a_arr1[3,2]
a_arr1 = ARRAY(3,2)
a_arr1 = {{NIL,NIL},{NIL,NIL},{NIL,NIL}}
```

SOFTWARE DE COMUNICACIONES

Netware NOVELL

A continuación presentaremos la forma en que se configura e instala una de las versiones más utilizadas de sistemas operativos para redes, la cual es Novell Netware 286 Advanced Ver. 2.15.

El haber elegido este modelo (probable a cambiar en un futuro inmediato) se debe a que representa la metodología fundamental de instalación NetWare.

En los siguientes párrafos NOS o NetWare significará Sistema Operativo, Sistema Operativo de Red o Advanced NetWare 286; NIC significará tarjeta de Red; FS será el File Server y WS se nombrará para las estaciones de trabajo.

Dada la existencia de muchas alternativas para configurar el NetWare se debe realizar una serie de pasos previos, exigidos por el mismo NOS. Partiremos del hecho de que el disco del servidor ha sido formateado con MS-DOS y posteriormente preparado para el NetWare.

Preparar un disco para el NetWare significa darle el formato "especial" que requiere. Esto puede hacerse directamente con los discos flexibles de Novell, mediante la utilería llamada COMPSURF (COM-Prehensive SURFace analysis) o con una utilería fabricada por Ontrac, llamada 'Disk Manager NetWare'. Esta última es más recomendable por su sencillez. Si se ejecuta COMPSURF o Disk Manager NetWare se deberá obtener la misma lista de sectores defectuosos del disco duro. Si el disco duro del servidor no está preparado para NetWare será imposible cargar el NOS en él.

Existen varios métodos para configurar el Advanced NetWare 286. Por su simplicidad, versatilidad y rapidez sólo describiremos el método del disco duro.

Para este método se deberá contar con una microcomputadora con su disco duro y cuando menos 5 Mbytes de espacio libre.

Para su instalación y configuración NetWare presenta pantallas con menús de los cuales habrá que elegir las opciones adecuadas.

En cada pantalla de menús las opciones a elegir serán las indicadas en letras 'negrillas y subrayadas', si en el mismo menú aparece más de una opción en 'negrillas y subrayadas' significa que se deberán elegir en el orden en que aparecen en pantalla, una después de otra.

Preparación

Las NIC's deberán configurarse siguiendo las instrucciones de

los manuales. Generalmente los fabricantes de NIC's entregan sus tarjetas preconfiguradas, con condiciones que para la mayoría de los casos son correctas.

Si se cuenta con NIC's ARCnet, se requiere que cada una de ellas tenga un número de nodo diferente (ID), pues de haber dos o más ID iguales, habrá problemas al localizar las direcciones de los nodos y la red se caerá. Para hacer esto, en la NIC, al lado del conector BNC, se encuentran unos microinterruptores en donde se configura el ID. Algunas tarjetas se pueden configurar por software.

Una vez que se han configurado las NIC's hay que tomar nota de las condiciones de cada NIC (IRQ, Memory Address, ID, etc.), ya que estos datos se requieren en pasos posteriores.

Existen una serie de pasos a ejecutar para completar la instalación del NetWare, para los cuales se requieren conocer algunas de las características del equipo en donde se instalará, así como las características que queremos que tenga la red.

Por ejemplo, para instalar la red hay que haber definido de antemano en qué modo trabajará el File Server: Dedicado o No dedicado.

El modo dedicado (Dedicated) significa que el servidor (FS) no podrá utilizarse como estación de trabajo, sólo atenderá las peticiones de las estaciones de trabajo. En este modo se obtiene el mayor rendimiento de la red y es recomendable aún cuando se tengan pocas estaciones de trabajo. Para muchos, esto significa perder un equipo sacrificándolo como servidor, pero en realidad se gana un servidor poderoso que atenderá mejor a los usuarios y permitirá trabajar con mayor seguridad y tranquilidad.

El modo No dedicado (Nondedicated) significa que el servidor podrá utilizarse como estación de trabajo, pero para procesos no muy complejos que demanden de él demasiada atención, ya que si obligamos al FS a ser servidor y estación para trabajos pesados, la red se degradará considerablemente.

En un mercado tan competido como el de Sistemas Operativos de Redes (NOS), donde los líderes mejoran sus productos para hacerlos más atractivos y poderosos; ofreciendo mayor seguridad, eficiencia, capacidad de conexión con otras redes y ambientes operativos de minis y mainframes, se está haciendo a un lado a los usuarios cuyas necesidades no son tan avanzadas, ni requieren productos tan poderosos y sofisticados, ni están dispuestos a pagar grandes cantidades de dinero por la red.

Los pequeños y medianos negocios requieren medios económicos y flexibles para cubrir sus necesidades de procesamiento multitarea, para compartir sus recursos y comunicarse eficientemente. Tal vez los beneficios de la alta seguridad y rendimiento que requieren las grandes empresas, bancos, casas de bolsa, grupos industriales y

negocios con grandes necesidades de captura, cálculos, comunicaciones y reportes, sean complicaciones para usuarios con conocimientos elementales sobre redes, cuyas aplicaciones para administración, contabilidad, estadísticas de ventas, cálculos de impuestos, etc., no necesitan plataformas sobradas y altamente eficientes.

No hay duda que productos como NetWare de Novell, LAN Manager de Microsoft, LAN Server de IBM, VINES de Banyan, UNIX y otros, pueden cubrir esas necesidades básicas y más. La duda surge cuando el usuario piensa que sus necesidades no son tan complicadas como para pagar los precios de los productos anteriores, aún con sus versiones elementales (entry-level), y cuando ellas mismas exigen un nivel de conocimiento adicional, no tan complicado ni difícil pero requieren capacitación en forma, para no depender totalmente de su vendedor. Toda dependencia es un lastre que a la larga termina por cansar haciéndonos improductivos.

Existen más de 20 fabricantes de NOS ofreciendo soluciones a pequeños y medianos negocios, algunas de ellas son tan particulares que a pesar de ser buenas hacen que el usuario se pregunte si no dependerá después de su proveedor o qué pasará si más adelante desea conectarse a ... en fin.

Un pequeño grupo de estos fabricantes tuvieron la idea de hacer redes basadas en el DOS de Microsoft llamándoles redes DOS. Estas, para funcionar, requieren DOS ver 3.1 o una versión superior en todos los equipos a conectarse en red, ya sean servidores o estaciones de trabajo.

Sus principales ventajas son la sencillez, capacidad y bajo costo, sin cambiar en gran medida el ambiente de trabajo acostumbrado con el MS-DOS y sin tener en los Servidores discos duros con formato especial e incomprensible.

Estas redes fueron diseñadas para trabajar en modo peer-to-peer y algunas soportan también el modo Servidor. Peer-to-peer significa que cualquier equipo puede darse de alta como Servidor y compartir sus recursos siendo WS a la vez, pudiendo haber tantos Servidores como equipos haya en la red, haciendo más eficiente el uso de dichos recursos (impresoras, modems, discos duros, unidades de disco flexible etc.), aunque complique de alguna manera la administración en redes grandes.

El modo servidor significa que sólo habrá un equipo dedicado exclusivamente a las funciones de red y atendiendo las peticiones de las estaciones de trabajo. Cabe señalar que la mayoría de las estaciones de trabajo permiten ser utilizadas sin unidades de discos flexibles ni disco duro (Deskless Workstations), esto se debe gracias a que la mayoría de las tarjetas de comunicación para red, permiten agregar un circuito integrado, el cual almacena la información necesaria para realizar el boot y programas necesarios para la conexión con la red.

Casi todas las redes DOS incluyen una colección de NetBIOS para soportar las marcas de tarjetas de red (NIC) más populares, sin llegar a la diversidad de manejadores para NIC's de Novell. No obstante, se deberá checar que la NIC que se adquiera está incluida en dicha colección, de otra manera el fabricante de la NIC deberá proporcionar el NetBIOS adecuado.

Como entre NetBIOS hay comunicación, es posible mezclar marcas de NIC's en la misma red, conservando siempre la misma arquitectura. Es posible hacer 'Bridges' entre redes DOS, con la misma o diferente arquitectura (por ejemplo un Bridge entre una Red Ethernet con una Red ARCnet) mediante un software adicional que proporciona el mismo fabricante de la red.

El Servidor no efectúa el 'bridging', como lo hace Novell, para ello deberá utilizarse una de las estaciones de trabajo que servirá como enlace entre dos o más redes, y dicha estación trabajará en modo no dedicado, esto es, a la vez que es un 'Bridge' también es estación de trabajo.

En general, estas redes son fáciles de instalar, actualizar, mantener y son suficientemente rápidas debido a sus protocolos. Su principal función es administrar los recursos de la red para compartirlos entre todos los usuarios, y siendo los servicios de archivos y de impresiones las demandas más frecuentes de una red pequeña, es fácil comprender por qué se espera que éstas tengan una gran demanda en los próximos años.

La solución ofrecida para las redes DOS es sencilla pero útil y poderosa para negocios pequeños y medianos, la gran ventaja es que no cambian el ambiente de trabajo conocido y requieren esfuerzos de capacitación mucho menores. No obstante, siempre habrá quien les critique que en materia de seguridad siguen siendo muy pobres.

La seguridad implica muchas cosas. Estas redes han cubierto los aspectos básicos y algunas de ellas han ido más lejos, permitiendo duplicidad de disco, directorios públicos y privados, atributos de archivos dentro de cada directorio, etc.

En su mayoría, los fabricantes de estas redes no intentan competir con fabricantes de soluciones poderosas como Novell y Microsoft. Sin embargo, por mencionar sólo dos de ellos, CBIS y Performance Technology de alguna manera han querido estar presentes en ambos mercados.

A continuación se resumirán las características más importantes de algunas de estas redes.

POWERLan, de Performance Technology

Es una red que trabaja en el ambiente DOS 100 % compatible con el NetBIOS de IBM, que puede interoperar con LAN Manager directamente y, mediante productos del mismo fabricante, con UNIX y

XENIX. Soporta estaciones DOS, OS/2 y Windows 3.0, con el cual permite administrar los recursos compartidos (discos duros, floppys, impresoras, modems, graficadores, etc.) y los accesos a ellos.

POWERLan trabaja en modo peer-to-peer, por lo que cualquier computadora (XT, AT, 386SX y 386) puede declararse como Servidor de uno o más recursos, siendo estación a la vez. Esto es, los Servidores son No Dedicados. No es recomendable declarar XT's como Servidores por las limitantes propias del equipo.

Los dos esquemas de seguridad de POWERLan son: (1) asignar claves de acceso a los recursos compartidos (uno de los esquemas de seguridad de LAN Manager) y (2) asignar privilegios a los archivos dentro de cada directorio, para leer, escribir, borrar, renombrar, ejecutar y crear. Performance Technology está estudiando la posibilidad de agregar nombre y clave de acceso a la red, además de privilegios para crear y borrar subdirectorios.

POWERLan soporta diversas NIC's (ARCnet, Ethernet y Token Ring) compatibles con NetBIOS, algunas marcas y modelos de NIC's ya vienen cargados inicialmente y sólo deberá elegirse del menú la adecuada. Además, soporta adaptadores de red externos (los que se conectan al puerto paralelo), muy útiles para LapTops o equipos sin ranuras de expansión.

El consumo de memoria de POWERLan es mediano en el servidor, ocupando casi 110 Kbytes, y bajo en las estaciones, con memoria de 50 Kbytes. Es posible disminuir considerablemente el consumo de memoria base mediante el uso de memoria extendida (donde se cargan ejecutables de POWERLan) y expandida EMS 4.0 (donde se carga el manejo de memoria 'cache').

Las utilerías que lo acompañan hacen de POWERLan una red fácil de instalar, utilizar y administrar; con menús que permiten administrar las colas de impresión, para visualizar recursos y conectarse a ellos, visualizar el trabajo de los usuarios, etc. Además, cuenta con un manejador de memoria cache, haciendo los accesos al disco duro mucho más eficientes y por lo tanto la red es más rápida.

Dar de alta o baja recursos es tan fácil como teclear una instrucción de una sóla línea o mediante menús. Los recursos se declaran compartidos y se ponen en uso para cada usuario que se desee.

El rendimiento de POWERLan es mediano/alto y su mercado son las redes con pocas WS, tal vez no más de 20, y carga de trabajo mediana. No obstante, existen versiones para 2, 5, 10 y 255 usuarios y en las tres primeras pueden agregarse usuarios de uno en uno o actualizarse a la siguiente versión.

Para redes que requieren mayor rendimiento debido a su carga de trabajo y/o número de WS, Performance Technology utiliza un software llamado POWERServe. Este se monta sobre POWERLan y hace del servidor un equipo Dedicado exclusivamente a la atención de los usuarios, soporta archivos abiertos, memoria y capacidad de disco duro casi ilimitados. POWERServe también mejora la seguridad de acceso y utilización de la red.

El mismo fabricante ha diseñado el siguiente software: (1) para manejar espejos de disco POWERMirror, que permite mantener el espejo del disco en el mismo Servidor o en otro equipo, representando una ventaja considerable porque agrega a la red la tolerancia a fallas; (2) MultiVol, para manejar volúmenes múltiples; (3) POWERBridge, para establecer puentes (bridges) entre redes POWERLan con la misma o diferentes arquitecturas;(4) POWERFusion, para interconectarse a UNIX permitiendo explotar su poder y muchos productos más. Además, POWERLan viene acompañado por un correo electrónico.

Una de las críticas más fuertes hacia POWERLan, y de hecho hacia casi todas las redes DOS, es que en el servidor no hay seguridad. Desde cualquier estación sólo se puede acceder a lo que se tiene derecho, de acuerdo a la clave de acceso, pero en el servidor se tiene derecho a todo pues el disco tiene formato DOS.

Una de las características de POWERLan es que el servidor necesita dar primero el boot con ms-dos, para después entrar en operación, por lo que cualquier usuario mal intencionado podría causar destrozos. Esta última aparente desventaja en contra de la seguridad es también una ventaja, pues el mantenimiento del disco de red podrá efectuarse con utilerías poderosas y populares como Norton, PCTools u otras herramientas.

Performance Technology es un fabricante prestigiado por sus NetBIOS desarrollados para DOS, OS/2 y XENIX. Además por sus utilerías y productos desarrollados para NetWare, Vines, 3Com, 10Net, Network-OS y otras redes basadas en el MS-DOS.

LANtastic de Artisoft

En 1989 Artisoft sorprendió al mundo de redes con su producto LANtastic, ganando pruebas de comparación entre un sin número de fabricantes de NOS basados en el MS-DOS. La revista PC-Magazine ha declarado con pruebas porqué ha sido la elección predilecta.

LANtastic es un NOS peer-to-peer que soporta su propia tarjeta de red (NIC) Ethernet de 2 Mbps o cualquier NIC compatible con NetBIOS. No es compatible con el protocolo SMB ni cuenta con protocolos ISO (International Standard Organization), por lo que su intercomunicación con otras redes es prácticamente imposible.

Por ejemplo, LANtastic no es el escalón inicial de una red con potencial de crecimiento hacia las redes de Microsoft o Novell; sin

embargo, puede crecer con tantos servidores LANtastic como hagan falta, para satisfacer las demandas de 100 o más usuarios y lograr buen rendimiento a un precio razonable.

El consumo de memoria de LANtastic es muy bajo. El servidor requiere 40 Kbytes y las estaciones casi 13 Kbytes permitiendo que las aplicaciones corran sin problema alguno. Además, es posible cargar el NOS en memoria expandida para dar más espacio a aplicaciones que demandan mayor memoria base (aquellas que ocupan el espacio comprendido entre los primeros 640 Kbytes).

Para la seguridad de acceso, uso y navegación en la red LANtastic es muy bueno. Requiere del usuario su nombre y su clave para darle acceso a la red, además permite que a los usuarios se les asignen diferentes privilegios: para escritura, lectura, crear, modificar, borrar y renombrar archivos; crear, borrar y moverse en directorios; ejecutar programas y cambiar atributos de archivos.

Se ha hecho eficiente el acceso al disco duro, mediante el uso de memoria cache, donde se almacenan temporalmente los datos hasta que sean escritos o enviados a su destino, evitando así las frecuentes lecturas y escrituras y consiguiendo mayor velocidad de operación de red.

Con LANtastic viene integrado un correo electrónico fácil de usar, mediante menús permite enviar y recibir mensajes, enviar mensajes para lectura posterior y notifica al destinatario la llegada de un mensaje para que lo atienda de inmediato o lo almacene.

Además LANtastic tiene, opcionalmente, tarjetas que permiten comunicaciones con voz, como si fuera conversación telefónica, o graba mensajes, digitalizándolos, para reproducirse posteriormente. Esto es único en LANtastic y se debe reconocer que fué el primero en hacer posible (económicamente) la comunicación con voz dentro de la red.

A opinión de muchos, LANtastic no es puramente una red DOS, sino una solución particular; sin embargo, si es una red DOS, que en aras de la eficiencia y la minimización de código, sacrificó la conexión a otros ambientes y el crecimiento a redes poderosas. LANtastic es sin duda un innovador, su línea de productos es muy interesante: para transmisión de voz, para permitir accesos remotos, para comunicación vía puertos paralelos o seriales, para evitar NIC's costosas, y cuenta con utilerías para administrar y ofrecer soporte remoto, etc.; su NOS es muy atractivo y sus precios bastante razonables.

Network-OS Plus, de CBIS

CBIS es un fabricante exitoso de NOS, su red la vende a través de su canal de distribución y además ha licenciado su NOS con otros fabricantes para que lo modifiquen a su gusto y ofrezcan productos

diferentes que interoperen entre sí (tal es el caso de Performance Technology, por mencionar uno de ellos). Desde luego Network-OS es una red 100 % compatible con NetBIOS de IBM que interopera con LAN Manager y otras redes compatibles con NetBIOS.

En mucho, lo dicho de POWERLan es cierto para Network-OS, con las siguientes características propias de éste.

Para liberar memoria y dejar el espacio para las aplicaciones, Networks-OS puede acomodarse arriba de los 640K, por lo que su consumo de memoria base es muy bajo.

El servidor (llamado Netserver por CBIS) puede operar como un equipo completamente dedicado a la atención de las peticiones de los usuarios, incrementando el rendimiento de la red.

El control de impresiones de red es muy eficiente, permitiendo mover trabajos de una cola a otra, alterar el órden de los trabajos en la cola, manejar archivos de configuración de impresoras que almacenen las condiciones de tipo de letra, tamaño, etc.

En materia de seguridad Network-OS ha añadido protección de la clave de acceso y cifrado. Cuenta con un sistema de monitoreo, llamado RADAR, que permite visualizar las condiciones de operación de la red, administrar y modificar recursos compartidos (locales y remotos), despliega estadísticas de NIC's locales o remotas, etc.

Opcionalmente, Network-OS cuenta con Network-OS POWERServe Plus, software para incrementar el rendimiento de su red mediante un Netserver Dedicado; con Network-OS Plus Bridging, software para establecer bridges entre redes Network-OS; con OutBound Plus, para poder compartir múltiples modems; LANBound FAX, para compartir un equipo fax; InBound Remote, para permitir accesos remotos a la red, etc.

CBIS es un fabricante ya consolidado, con una línea de productos muy completa para casi cualquier necesidad de red, desde pocos usuarios hasta un máximo de 255, con interfaces para hacer más amigable el uso de la red, con la posibilidad de compartir CD-ROM, modems, fax, 3 impresoras paralelas y 2 seriales por servidor, etc., haciendo que Network-OS sea un NOS muy atractivo para futuros usuarios de redes.

Existen muchos más fabricantes de NOS que basan su diseño en el sistema operativo DOS que sería imposible mencionarlos a todos, sólo por hacer referencia a algunos de ellos, sin que esto signifique que éstos son peores a los mencionados, existen: 10Net Plus, PORT Lite, LANsmart (que permite correr su red concurrentemente con Novell o el protocolo TCP/IP), TOPS, etc.

Todos ellos son más o menos equivalentes en sus precios, no en rendimiento y en productos adicionales que las hacen más flexibles, seguras, eficientes, con posibilidades de crecimiento y conexión con otras redes y ambientes operativos.

Una red es para usuarios maduros que saben de la importancia del trabajo conjunto, que saben trabajar en equipo, que saben que los daños y que las experimentaciones sin conocimiento afectan a todos. Las redes DOS son para usuarios con necesidades medianas o un poco más,que cuidan, conservan y comparten su trabajo.

SISTEMAS OPERATIVOS

Un sistema operativo es, en primer lugar, un administrador de recursos, y el recurso primario que administra es el *hardware* del computador, contiene varias características:

- Define la interfaz del usuario
- Comparte el *hardware* entre usuarios
- Permite a los usuarios compartir los datos entre ellos
- Planifica los recursos entre usuarios
- Facilita la entrada/salida
- Recupera los errores

Los recursos clave que un sistema operativo administra son:

- Los procesadores
- El almacenamiento
- Los dispositivos de entrada/salida
- Los datos

Mencionaremos dos sistemas operativos de los más conocidos, los cuales son VMS y MS-DOS.

VMS

El sistema operativo VAX/VMS comprende dos componentes principales en la administración del almacenamiento: el paginador y el intercambiador. El paginador mueve las páginas individuales adelante y atrás entre los almacemanientos primario y secundario, el intercambiador mueve los procesos enteros. Cada proceso tiene un límite de conjunto residente, es decir, el número máximo de páginas que puede tener al mismo tiempo en el almacenamiento primario. Cuando un proceso ha alcanzado su límite del conjunto residente e intenta paginar una nueva página, debe abandonar una de sus páginas residentes del almacenamiento primario. Esta es una política de reposición local de páginas.

Las páginas se seleccionan para su reposición según la técnica de primero en entrar-primero en ser servido. Se añade una página reemplazada a la lista de páginas modificadas si se ha cambiado, y a la lista de páginas libres, si no lo ha sido. Las páginas modificadas son paginadas en el almacenamiento secundario solamente cuando es necesario.

En general, las páginas modificadas son reclamadas por sus procesos antes de que puedan ser paginadas hacia fuera. Las páginas libres suelen ser reclamadas antes de ser reasignadas. VMS ajusta

dinámicamente el tamaño del conjunto residente de un proceso basado en la tasa de fallos que experimenta el proceso. VMS intenta igualar la tasa de fallos en todos los procesos. VMS reduce la paginación de entrada/salida de disco al paginar los conjuntos residentes en grupos de y hacia el disco.

VMS utiliza 32 niveles de prioridad para los procesos de planificación. Los procesos de tiempo real son planificados estrictamente según su prioridad. Cada vez que se planifica un proceso normal, su prioridad decrece, pero nunca por debajo de cierta prioridad base.

Los procesos realizan varias transiciones de estado a lo largo de su existencia. Notable en VMS es la importancia del intercambio hacia adentro y hacia afuera. Los estados tienen lugar de forma especular dependiendo de si el proceso esta en el almacenamiento primario, o ha sido intercambiado hacia fuera.

La entrada/salida VMS suele ser asincrona, una vez que un proceso emite una petición de entrada/salida, puede continuar su ejecución mientras se sirve la petición de entrada/salida. Una estructura de datos describe el estado del sistema de entrada/salida en cualquier momento. Esta estructura de datos consiste en bloques de datos de dispositivos, bloques de control de unidades, paquetes de peticiones de entrada/salida, bloques de peticiones de canales, bloques de despacho de interrupciones y bloques de control de adaptadores.

Los servicios de administración de registros VMS (RMS) suelen usarse para peticiones de entrada/salida a dispositivos de almacenamiento masivo. RMS puede crear archivos secuenciales, archivos relativos y archivos indexados. Se proporcionan tres modos de acceso: acceso secuencial, acceso con clave y acceso con dirección del archivo de registros.

VMS proporciona diversos mecanismos de comunicación entre procesos para sincronizar la ejecución, enviar mensajes y compartir los datos comunes entre procesos cooperativos. Estos son las banderas de eventos comunes, buzones, áreas compartidas de almacenamiento y archivos compartidos.

MS-DOS

El sistema operativo MS-DOS pertenece al grupo de sistemas operativos mono-usuarios, lo que significa que sólo tiene capacidad de atender a un sólo usuario a la vez, muy aplicable al grupo de computadoras de tipo personal.

UNIX

UNIX, como cualquier sistema operativo es una serie de comandos que se apropia de un computador para hacerlo más accesible al

usuario. Las características que distinguen a UNIX de otros sistemas operativos son:

- Multiusuario, multitarea
- Abierto e independiente a tipos y marcas de equipo
- Gran facilidad para comunicaciones

Ventajas del UNIX

- Es independiente al tipo de tecnología y tipo de procesador.
- Su aplicación es vertical, corre desde una AT (80286) hasta un mainframe, esto es, desde 6 pantallas hasta más de 400.
- El hecho de dominarlo implica tener un control y versatilidad total de la computadora.
- Es el estandar, en la industria multiusuario.
- El UNIX naturalmente permite establecer redes o enlaces realmente funcionales entre configuraciones de equipo.
- Bajo procesadores INTEL, ofrece la capacidad de operar programas y comandos de MSDOS dentro del UNIX.
- El UNIX permite utilizar pantallas de bajo costo y/o computadoras personales, resultando las configuraciones muy económicas.

Desventajas del UNIX

- Complejidad en su uso para el usuario final.
- Difícil de instalar aún para alguien con conocimientos y esperiencia previos.
- La velocidad de respuesta de las pantallas se degrada conforme se aumenta la cantidad de usuarios al sistema.

CAPITULO 3

DISEÑO

3. DISEÑO

3.1. Introducción

3.1.1. Diseño de Sistemas

Diseño Orientado al Flujo de Datos.

Cada metodología de diseño de software tiene sus puntos fuertes y débiles. Un factor importante de selección de un método de diseño es la amplitud de áreas a las que puede aplicarse.

Un método de diseño orientado al flujo de datos es particularmente útil cuando la información se procesa secuencialmente y no existe una estructura de datos jerárquica. Por ejemplo:

1. Aplicaciones de control con microprocesadores.
2. En procesamiento de datos y pueden, además, aplicarse incluso cuando existen estructuras de datos jerárquicas.

Sin embargo, existen casos en los que una consideración de flujo de datos no sirve. En aplicaciones de bases de datos, sistemas expertos e interfaces orientadas al objeto, puede ser mejor emplear los métodos de diseño orientado a la estructura de datos o diseño orientado al objeto.

Diseño Orientado a la Estructura de Datos.

El diseño orientado a la estructura de datos puede aplicarse con éxito a aplicaciones que tengan una estructura jerárquica y bien definida de la información. Ejemplos típicos incluyen:

1. Aplicaciones en sistemas de información comerciales. La entrada y salida tienen distinta estructura (archivos de entrada, informes de salida), el uso de una base de datos jerárquica es frecuente.

2. Aplicaciones de sistemas administrativos. La estructura de datos para sistemas operativos comprende muchas tablas, archivos y listas que tienen una estructura bien definida.

3. Aplicaciones CAD/CAM/CAE. Los sistemas de diseño, fabricación, e ingeniería, asistidos por computadora, requieren estructuras de datos sofisticadas para el almacenamiento, traducción y procesamiento de información.

El diseño orientado a la estructura de datos es más fácil de aprender y más complicado de aplicar que las técnicas orientadas al flujo de datos. Sin embargo, la escuela de diseño orientada a la estructura de datos ofrece un enfoque más rico y, potencialmente, más poderoso para el diseño de *software*.

Diseño Orientado al Objeto.

Las representaciónes del diseño orientado al objeto son más propensas que otras a una dependencia del lenguaje de programación.

3.1.2. Interfaz con el Usuario

La interfaz es la sección del programa que se encarga de llevar el control de la comunicación de la computadora hacia el usuario y viceversa.

Actualmente se destina tan poca atención al diseño de la interfaz, que no es raro encontrar sistemas en donde el usuario debe recordar una gran cantidad de comandos poco claros y donde tiene que oprimir secuencias de teclas sin saber su significado. Sucede frecuentemente también que, una vez dado el comando, el sistema no responde inmediatamente y cuando así lo hace, el usuario se sorprende o confunde por el resultado. Es común ver cuando el usuario,por algún descuido insignificante, le provoca al programa reacciones desastrosas, teniendo que comenzar desde el principio otra vez.

Las frustraciones y la disminución de la productividad del usuario son culpa principalmente de un mal diseño de la interfaz. Los diseñadores del sistema pocas veces se preocupan en considerar las características del usuario promedio que utilizará su programa, y es por esto que elaboran sistemas útiles sólo para ellos mismos.

Frecuentemente sucede que todos aquellos que diseñan un sistema, cuando se enfrentan a la necesidad de construir su interfaz, toman muchas decisiones basadas en la experiencia y en las reglas que dicta el sentido común. No obstante, no son la experiencia ni el sentido común quienes deben guiar al diseñador. Gran cantidad de expertos han estudiado y determinado qué factores provocan que un sistema sea del agrado o no del usuario y los han denominado Factores Humanos.

Los Factores Humanos son aquellos que deben incorporarse en la elaboración de cualquier producto. Este término nació en Inglaterra durante la Primera Guerra Mundial, pero no es sino hasta ahora que ha tomado fuerza bajo el nombre de *Ergonomics* en Estados Unidos, y de Factores Humanos en el resto del mundo.

Análisis del Usuario

Lo primero que debe hacer el diseñador de sistemas es realizar un análisis del usuario promedio que empleará su sistema. De este análisis el diseñador obtendrá información muy valiosa, y eventualmente, logrará conocer como el programa debe comunicarse con el usuario. Existen dos puntos en que el diseñador debe concentrar su atención:

Con Relación a las Habilidades del Usuario

Dos factores, el conocimiento semántico y el conocimiento sintáctico, son los que determinan las habilidades de un usuario. Escencialmente giran alrededor del conocimiento y dominio que un usuario tiene sobre un programa en particular.

Mientras el conocimiento semántico radica en comprender qué hace el sistema, es decir, el grado de comprensión del funcionamiento general de éste, el conocimiento sintático trata sobre las acciones que debe realizar el usuario para ejecutar una tarea en particular del sistema.

De acuerdo a las dos categorías cognoscitivas, un usuario puede clasificarse en cuatro niveles:

1. Usuario Improvisado.

2. Usuario Principiante o Novato.

3. Usuario Competente.

4. Usuario Experto.

Forma en que el Usuario Interactúa con el Sistema

Además de considerar las habilidades de un usuario, el diseñador de sistemas debe también concentrar su atención en determinar la forma en que un usuario interactúa con el sistema. Fundamentalmente, trata los aspectos de frecuencia y duración de uso del sistema, así como la libertad que tiene el usuario para emplearlo o no. Con base en estos criterios el usuario recibe un nombre distinto.

1. Usuario casual.

2. Usuario continuo.

3. Usuario cautivo.

Análisis de las Funciones

Además del análisis del usuario, el diseñador debe realizar un análisis de las funciones del sistema. Básicamente, consiste en definir el objetivo general del sistema, las funciones que debe tener para cumplir con el objetivo y los pasos, o acciones, que el usuario debe realizar para llevar a cabo dichas funciones. En otras palabras, es un desgloce de los pasos desde que se inicia el sistema hasta que termina la ejecución del mismo.

De este análisis se desprenden características fundamentales que debe tener el sistema si se desea que incorpore buenos Factores Humanos, como son capacidad de recobrarse de un error, de revisar

los datos que se han introducido, etc. En forma paralela, también se determina como son los datos que manejará el sistema.

Una vez que se ha caracterizado a los usuarios, de acuerdo a las clasificaciones mencionadas, y se han determinado las funciones que debe realizar el sistema, es posible entonces, aplicar cinco lineamientos que el sistema deberá cumplir para incorporar buenos factores humanos. Los cinco lineamientos, o principios de los Factores Humanos, se establecen en el siguiente punto.

Principios de los Factores Humanos

Son cinco los principios en donde se establecen los criterios generales que los Factores Humanos dictan para todo sistema, el diseñador deberá intentar aplicarlos en la medida que le sea posible:

1. El sistema debe incluir una serie de funciones que cubran las necesidades del usuario y así éste pueda realizar su trabajo.

2. Adecuar las características del sistema a las características del usuario.

3. El sistema debe concordar con el modelo interno del usuario.

4. El flujo operacional del sistema debe concordar con el proceso de pensamiento del usuario.

5. El sistema debe evitar incomodidades físicas al usuario.

Primer Principio de los Factores Humanos

''El sistema debe incluir una serie de funciones que cubran las necesidades del usuario y así éste pueda realizar su trabajo''

Además de las funciones esenciales requeridas para que el sistema realice su objetivo fundamental, debe incluir una serie de funciones adicionales que cubran las necesidades particulares del usuario. Por ejemplo, es común que un usuario desee corregir un error en los datos que acaba de introducir. Si para lograrlo debe terminar la ejecución del sistema e introducir todos los datos nuevamente, el usuario terminará por frustrarse. Funciones como la de edición de datos, que no son parte del objetivo básico del sistema, engrosan la lista de funciones edicionales, útiles en la ejecución del sistema. La lista completa es la siguiente:

1. Desplegado del Estado del Sistema

Es común que al operar sistemas más o menos complejos, el usuario pierda la visión de dónde se encuentra el comando que está ejecutando, dentro del contexto de la estructura global del sistema. Por este motivo, es indispensable indicarle al

usuario qué parte del programa, o comando, se encuentra ejecutando. Y así, cuando el usuario seleccione el comando de edición o impresión del documento en un procesador de palabras, el programa deberá mostrar el estado, edición o impresión del comando que está ejecutando.

El estado del sistema debe desplegarse en un lugar de la pantalla, procurando que sea siempre en la misma zona. Por ejemplo, en la parte superior. Lo esencial consiste en que el usuario entienda, con el mensaje, qué parte de la estructura global del sistema se está ejecutando.

Un aspecto importante en el diseño de la organización de la pantalla consiste en presentar la información por páginas y no por movimiento vertical ascendente de líneas (*Scrolling* en Inglés). Esto es, la pantalla se debe borrar antes de desplegar nueva información.

2. Desplegado de Datos

El usuario debe poder indicarle al sistema que desea desplegar datos que previamente haya introducido, para poder revisarlos. Es muy común que se realice un cambio de pantalla, es decir, cambio de información desplegada en la pantalla, cuando se teclean nuevos datos. Es conveniente que la información se despliegue con un orden, para que el usuario pueda localizarla fácilmente. El orden puede ser alfabético, numérico o cronológico.

3.- Edición de Datos

El usuario frecuentemente comete errores cuando está proporcionando o introduciendo datos al sistema. El programa debe proveer un comando que le permita al usuario editar, es decir, modificar datos previamente introducidos. En la mayoría de los casos la edición se puede realizar en tres etapas. La primera sucede en el preciso momento que el usuario está introduciendo el dato, con las teclas de ''espacio para atrás'' o de edición. La segunda corresponde a la edición después de que se ha introducido un bloque de datos. El sistema preguntará si se desea corregir algún dato, posteriormente pedirá el número de dato a corregir y finalmente el usuario deberá reescribir el dato erróneo. La tercer y última etapa, ocurre cuando la edición se realiza después que los datos han ingresado ya a la base de datos, o conjunto de archivos.

4. Validación en la Introducción de Datos

Algunas veces se requiere que el dato que escribe el usuario cumpla con características especiales de validación. Tal y como sería, por ejemplo, con el nombre de un derechohabiente de un servicio médico. Se requiere que el nombre que proporciona el usuario concuerde, letra a letra, con el nombre como se dió de

alta al derechohabiente. Ante esto se pueden tomar cuatro alternativas:

a) Exigir que el nombre que da el usuario concuerde perfectamente con el almacenado en el archivo.

b) Si no hay concordancia con el dato almacenado, el sistema puede intentar predecir cual debería haber sido el dato de entrada.

c) Una técnica modificada de la predicción del dato consiste en mostrar al usuario dicha predicción y pedirle que confirme si está correcta.

d) Otra técnica es verificar primero la concordancia.

5. Recuperación del Error

Siempre existen errores al operar un sistema y es indispensable, si no se desea que el usuario deba poner extremado cuidado en su manejo, que tenga buenos mecanismos para recuperar el error. Existen dos tipos de error:

a) Errores de Control. Son aquellos que llevan al sistema un estado no deseado tal y como sucede cuando se elige erróneamente una opción en un menú, el sistema debe permitir salirse de ese estado erróneo. Cuando se emplea un esquema de menú de opciones, simplemente consiste en colocar una opción de salida.

b) Error de Datos. Sucede cuando los datos se escriben mal o o se seleccionan erróneamente. El sistema debe, mediante la petición del usuario, regresar al punto exacto de entrada para ese dato específico y así pueda reescribirse el dato.

6. Función de Ayuda o Asistencia

Las funciones de ayuda o asistencia, son líneas de texto integradas al sistema que se le muestran al usuario para ayudarlo en su operación. Existen varios aspectos importantes involucrados en el diseño de ayudas. Estos se detallan en los siguientes párrafos.

La función de ayuda se debe diseñar acorde al nivel de experiencia del usuario. Así, si el usuario es Improvisado, requerirá que la función de ayuda provea de gran cantidad de ayudas semánticas y sintácticas, donde se expliquen las funciones del sistema y las acciones para operarlo. En caso de que el usuario sea Principiante, la función de asistencia deberá tener ayudas semánticas como simples recordatorios breves, en donde se muestren los lineamientos funcionales

básicos del sistema. Pero también deberá tener ayudas sintácticas en forma de listas de comandos con argumentos.

Si el tipo de usuario es Competente, el nivel de asistencia deberá diseñarse de modo que sólo se muestre una ayuda sintáctica en la forma de una lista sintetizada de comandos, sin argumentos. En caso de que el usuario sea Experto, lo más recomendable es no proporcionarle una función de ayuda, pues esto lo distraería.

La función de ayuda debe también diseñarse dependiendo de su frecuencia de uso. Así, las funciones con menos uso deben disponer de un mayor número de ayudas. Cuando existen usuarios con distintos niveles de experiencia, la función de ayuda debe estar orientada al de menor experiencia. Para no distraer a los usuarios con mayor nivel, la función de ayuda puede estructurarse de modo que muestre primero una lista abreviada de comandos para usuarios Competentes. Si el usuario lo demanda, mostrará un segundo nivel, en donde se desplegará una lista más detallada de los comandos, acompañados de argumentos y organizada de tal manera que proporcione una idea vaga del funcionamiento del sistema. Un tercer nivel también estará a disposición del usuario, para explicar ampliamente las funciones del sistema. El usuario entonces, irá avanzando por cada nivel hasta encontrar el que satisfaga sus necesidades.

El diseñador debe tener en mente que la función de ayuda se diseña de acuerdo al estado en el que se encuentre el sistema. Así, la ayuda desplegada está relacionada, única y exclusivamente, con el comando del sistema que se está ejecutando en ese momento. Una vez que la ayuda ha terminado, el sistema debe retomar el control a partir del mismo punto de operación, antes de que la ayuda se desplegara.

Segundo Principio de los Factores Humanos

> ''Adecuar las características del sistema a las características del usuario''

Son seis los aspectos que cubren este principio. Básicamente tratan sobre las características que debe tener el sistema, en función de las limitaciones cognoscitivas y del nivel de experiencia del usuario promedio, esto es, Improvisado, Principiante, Competente o Experto. A continuación se detallan estos aspectos.

1. Fácil de Entender o Fácil de Usar.

Lo primero que debe realizar el diseñador de sistemas es determinar si el sistema debe ser ''fácil de usar'' o ''fácil de entender''. Un sistema ''fácil de entender'' es aquel en donde el usuario puede comprender qué hace el sistema y qué tiene que hacer para operarlo. Tiene ayudas para su

comprensión donde se enfatiza la asistencia semántica y sintáctica. A diferencia de éste, un sistema ''fácil de usar'' centra su atención en buscar la menor actividad del usuario, ya sea cognoscitiva o física, para utilizar el sistema. En general, un sistema ''fácil de usar'' se traduce en última instancia en teclear lo menos posible.

Después de haber determinado si el sistema debe ser ''fácil de usar'' o ''fácil de entender'', la selección del formato de comandos, para que el usuario ejecute acciones, es automática. Así se tiene que, los comandos de tipo menú de opciones y pregunta-respuesta son ''fáciles de entender'', mientras que los comandos del tipo órdenes directas son ''fáciles de usar''.

2. De acuerdo al Nivel Cognoscitivo del Usuario

Además del criterio fácil de usar o fácil de entender para determinar qué tipo de formato de comandos elegir, también se puede seleccionar de acuerdo al nivel de experiencia del usuario.

3. Mensajes del Sistema

Además de que el formato de comandos debe de adecuarse al nivel del usuario, los mensajes que el sistema envía también deben estar de acuerdo a las características del usuario. Así, los mensajes a los usuarios Improvisados o Principiantes deben ser extensos, de tal suerte que en el propio texto se indique que hacer. Los mensajes a los usuarios Competentes o Expertos deben ser cortos pero nemotécnicos, pues sólo sirven como simples recordatorios.

Nunca deben emplearse mensajes codificados en forma numérica, sin importar el nivel del usuario. Ya quesu significado es difícil de recordar aún para los usuarios Expertos.

4. Amplitud de las Decisiones

Una vez que se ha elegido el formato de comandos, es vital centrar la atención en las limitaciones cognoscitivas del usuario. Así por ejemplo, es responsabilidad del diseñador del sistema decidir que tantas opciones deben de asignarse a cada menú. O si es el caso, cuántos comandos deben de integrar el repertorio de comandos directos.

La amplitud de las decisiones es el número de elementos entre los que hay que elegir la función a realizar. Así, en el formato de menú, es el número de opciones que lo conforman. Para el formato de comandos directos, es el número de comandos individuales que integran el repertorio. Una práctica sana indica limitar la amplitud de las decisiones a diez. Si se

requiere tener una amplitudd mayor, se deberá tener asistencias integradas, tal y como una función de ayuda.

5. Memoria de Corto Plazo

La Memoria de corto plazo es otra limitación de tipo cognoscitivo y se describe como el proceso de leer o escuchar un dato y retenerlo en la memoria el tiempo suficiente para teclearlo. La memoria de corto plazo del usuario está limitada, generalmente a cinco o siete elementos. Por ejemplo, un número de 12 dígitos, es decir 12 elementos, normalmente excede la capacidad de la memoria de corto plazo de la mayoría de las personas. Esto provoca que algunos dígitos sean mal recordados o completamente olvidados.

Una forma de recordar secuencias largas de elementos, consiste en agruparlos en conjuntos de tamaño más manejable. Es así como un número de larga distancia, que tiene diez dígitos, es demasiado largo si no se agrupa en el código de lada, la clave de la ciudad y el número. Este método resulta altamente recomendable cuando el sistema requiere leer secuencias largas de caracteres o números, debe permitir que el usuario escriba pequeños conjuntos de elementos, pues así le será más fácil recordar la cadena completa.

6. Consistencia y Estandarización

Sin importar la longitud de los mensajes del sistema y el formato de comandos, ya sea menú, pregunta-respuesta o comandos directos, es esencial que observen una consistencia en su sintaxis. Es decir, los comandos que hacen lo mismo en distintas partes del sistema deben ser los mismos. Así, por ejemplo, los comandos u opciones del menú: fin, salida y termina, generalmente significan lo mismo. Sin embargo, sólo uno de ellos puede usarse en todo el sistema. Por extensión, funciones que son completamente diferentes no deben tener comandos iguales o aún similares, en distintas partes del sistema.

Debe procurarse que el sistema siempre envíe los mensajes de una forma similar, o al mismo lugar. Tal y como se mencionó en la organización de la pantalla, del primer principio de los factores humanos.

Algunos sistemas son inconsistentes cuando piden al usuario que introduzca un dato. Esto sucede, generalmente, cuando en casi todo el programa se espera que el usuario oprima una tecla, como return, para indicarle que el dato ha finalizado. Sin embargo, en otras secciones, el sistema no lo hace así. Cuando el usuario termina de escribir el último carácter, o dígito, por ejemplo del registro federal de causantes, automáticamente el sistema transita al siguiente

dato, sin esperar que el usuario oprima ''Return''. Esto puede desconcertar al usuario y debe emplearse sólo en contados casos.

Tercer Principio de los Factores Humanos

''El sistema debe concordar con el modelo interno del usuario''

No sólo las características del sistema deben hacer juego con características del usuario sino también el sistema debe concordar con el modelo interno del usuario. Cada persona tiene su modelo del sistema, este modelo ayuda al usuario a decidir qué comandos debe ejecutar para realizar una acción específica. De ahí la importancia de la exactitud del modelo interno para operar el sistema correctamente. Así pues, el modelo interno del sistema esta intrínsecamente relacionado con el conocimiento semántico del mismo.

1. El Modelo Interno del Usuario

El modelo interno del usuario se desarrolla con base en la experiencia y por el uso del sistema en particular, u otros sistemas semejantes, ayudado obviamente por los cursos que este haya adquirido y los manuales. Sin embargo, frecuentemente el modelo interno del usuario radica en el entendimiento de un proceso natural, el cual se apoya en el empleo de conceptos familiares. Tal es el caso de una secretaria, a quien le es familiar abrir y cerrar archivos de empleados.

Será entonces, labor del diseñador del sistema deducir cómo es el modelo interno esperado del usuario y diseñar el sistema lo más apegado al modelo. A pesar de todo, no es tan sencillo; para los usuarios Improvisados y Principiantes, el diseñador del sistema puede suponer que el modelo interno del usuario no estará bien desarrollado. Por este motivo, el sistema debe diseñarse para que se presente en pequeños paquetes fáciles de entender. Y así, el usuario emplee sólo una pequeña parte del sistema para realizar una actividad completa. Con esta técnica, conforme su modelo interno se desarrolla, el usuario podrá emplear nuevas secciones del sistema.

Para los usuarios Competentes y Expertos, el diseñador puede esperar que tengan desarrollado un buen modelo interno. Por este motivo, el sistema podrá mostrarse como un paquete integrado, que el usuario podrá emplear con todas sus funciones. Esto implica que a diferencia del caso anterior, el sistema se diseñara para ''navegar'' fácilmente por todas sus funciones y no para entenderlas fácilmente.

2. Manuales

Con respecto a la documentación, o manuales, se puede mencionar que son esenciales para ayudar al desarrollo interno

del modelo del usuario (conocimiento semántico), y para proveer de los procedimientos para operar el sistema (conocimiento sintáctico). La documentación para usuarios Improvisados y Principiantes debe contener información semántica intensa, mientras que la documentación para usarios Competentes y Expertos debe tener menos información semántica y enfocar el material a simples recordatorios.

3. Cursos

Otro elemento fundamental en el desarrollo del modelo interno del usuario son los cursos. Para los usuarios improvisados y principiantes, los cursos deben concentrarse en desarrollar el modelo interno (conocimiento semántico). Esto se logra enseñando a realizar, poco a poco, funciones completas pequeñas, para después intentar funciones adicionales. Los cursos para los usuarios competentes y expertos no requieren de tanto esfuerzo, pues necesitan exclusivamente ajustar el modelo interno del usuario. Esto se logra estableciendo las diferencias entre el nuevo sistema y sistemas anteriores. El tiempo disponible se puede concentrar a impartir el conocimiento sintáctico para operar el sistema.

Como se pude apreciar hasta este punto, la selección del nivel del usuario, de acuerdo a sus características cognoscitivas, dicta la pauta en el diseño de sistemas. El desarrollo del modelo interno del usuario está muy ligado a si el sistema debe ser fácil de usar o fácil de entender y esto, eventualmente, dicta el tipo del formato de comandos.

Cuarto Principio de los Factores Humanos

''El flujo operacional del sistema debe concordar con el proceso de pensamiento del usuario''

Hasta el momento se han expuesto criterios relacionados con las capacidades cognoscitivas del usuario y de las características que el sistema debe cumplir, si se desea que esté acorde con el usuario. Pero de la respuesta del sistema, otro elemento funcional de gran importancia, nada se ha mencionado. El cuarto principio dicta precisamente lineamientos básicos para este elemento.

Cuando el usuario desee resolver un problema, desarrolla un modelo mental correspondiente a los pasos que lo solucionan. Si la operación del sistema no está de acuerdo al modelo de solución, interrumpirá frecuentemente al proceso pensante del usuario. Así es que tendrá que detenerse a pensar, primero en el problema que desea solucionar y luego en cómo funciona el sistema. Esto crea distracción, y si es muy frecuente, termina por decepcionar al usuario. Es por esto que la secuencia de operación del sistema debe concordar lo más posible con la secuencia natural de solución del problema del usuario. Tal es el caso del mecanismo de introducción de datos para almacenar una forma. Las formas son

generalmente documentos que tienen una serie de espacios, que el usuario debe llenar a mano. De estas formas se extrae información, la cual se almacena en la computadora y posteriormente se procesa. Ejemplos clásicos son las formas para el censo poblacional o para la declaración de impuestos. El mecanismo de captura o introducción de datos de la forma, debe vigilar que la secuencia de datos pedida por el sistema, tenga la misma secuencia que los datos originales de la forma. Lo cual evita que el usuario tenga que buscar constantemente los datos por toda la forma y así, disminuir la posibilidad de error.

Por otro lado, la respuesta del sistema debe adecuarse al ritmo mental del usuario. Un usuario es capaz de aplicar toda su atención al problema hasta llegar a un punto de terminación. La terminación se entiende como el estado en el cual se cubre un objetivo. Al llegar a su punto de terminación, el usuario puede liberar la atención del sistema. Teniendo esto en mente, un usuario puede tolerar retardos inherentes al tiempo de respuesta del sistema, hasta de un segundo sin perder la atención. Es más, a pesar de lo que los retardos de cinco segundos se registran con interrupciones definitivas, el usuario normalmente logra conservar la atención. Sin embargo, esto no es cierto para retardos mayores a diez segundos, en donde el flujo operacional del sistema rompe totalmente el flujo natural del usuario. Cuando esto sucede, se dice que ha ocurrido una interrupción antes de un punto de terminación normal, lo cual requiere de un esfuerzo mental adicional por parte del usuario para retomar el problema.

Es responsabilidad del diseñador, cuando realiza el análisis de las funciones del sistema, fijar el alcance del poder de atención del usuario, identificar los puntos de terminación y estructurar el sistema para evitar distraer la atención del usuario.

Algunas veces no es posible evitar los retardos inherentes del sistema. Bajo estas circunstancias, el sistema deberá informar al usuario si la respuesta tardará más de diez segundos y deberá proporcionar, si es posible, una estimación de la duración del retardo. Esto permite al usuario fijar la atención en cualquier otra cosa, sin vigilar constantemente el sistema. Es imperativo entonces que el sistema informe al usuario también cuando ha finalizado el retardo.

En otras ocasiones, el sistema debe interrumpir forzosamente el flujo operacional, pues se requiere que el usuario tome una acción en particular. Esto es, cuando un dato no es válido y el usuario debe corregirlo, o bien, cuando existe un error del sistema, por ejemplo, en el momento en que el usuario desea cargar en la memoria un archivo que no existe.

No importa si el motivo es la terminación de un retardo, o si ha ocurrido un error del sistema, la forma como el sistema debe llamar la atención del usuario depende de la manera en cómo el usuario interactúa con la pantalla. Si el usuario tiene toda la atención a

la pantalla, entonces, un simple mensaje o aviso es suficiente. Pero si el usuario mira ocasionalmente al video, las letras del mensaje, más brillantes, intermitentes, o de distinto color, son apropiadas para llamar su atención. Cuando el usuario opera el sistema en forma automática (usuario Experto), una alarma sonora es el único medio capaz de interrumpir la atención del usuario. También hay que tomar en cuenta que la manera en como se llama la atención del usuario depende del estado normal de la pantalla. La señal audible, que podría ser una campana, no es adecuada si el programa emite bajo condiciones normales tonos audibles. Algo similar sucede con mensajes intermitentes, en distinto color o brillantes. Resumiendo, las señales de alarma deben elegirse de acuerdo a estados excepcionales del programa, para que sean factores que llamen efectivamente la atención del usuario.

Quinto Principio de los Factores Humanos

''El sistema debe evitar incomodidades físicas al usuario''

Finalmente, este factor trata los aspectos relacionados con el medio ambiente que rodea al usuario y especialmente enfoca su estudio a la terminal de trabajo. Existen dos aspectos principales que determinan la comodidad en una terminal, las malas posturas del cuerpo y la fatiga visual.

1. Las malas posturas corporales

Las malas postura provocan dolor muscular y eventualmente causan fatiga o incomodidad. También, emplear mobiliario inadecuado propicia una deficiente circulación de la sangre, que en última instancia se traduce en un cansancio. La comodidad física se logra con equipo y mobiliario ajustable a las necesidades del usuario. La silla de trabajo debe permitir ajustar la altura del respaldo y la inclinación de la espalda. Pero jamás de tal forma que los pies estén despegados del suelo. Algunas sillas solucionan esto proporcionando un soporte especial para descansar los pies. La altura de la mesa debe colocar en una posición natural para las manos del usuario. Los teclados no empotrados al video, permiten que el usuario se siente a una distancia conveniente del video y que adecúe la lejanía del teclado.

2. La Fatiga Visual

La fatiga visual se debe escencialmente al reflejo de fuentes de luz incidentes sobre el video. Existen tres métodos simples para reducir el reflejo de la luz:

a) Emplear un video que permita fijar su inclinación horizontal. Esto evita al máximo la luz incidental reflejada.

b) Usar un recubrimiento antireflejante sobre la pantalla. Esto permite reducir notablemente la fatiga visual. Sin embargo,

debe cuidarse de mantener el contraste adecuado entre la intensidad de los carácteres y el resto de la pantalla.

c) Orientar las luces tomando en cuenta la posición del video, tratando que no se refleje la luz sobre la pantalla. La línea de vista de un usuario, que mira directamente el centro del monitor, puede tener una inclinación entre 15 y 20 grados de inclinación abajo de la horizontal. La pantalla debe colocarse a una distancia de 40 a 60 centímetros del ojo, medida desde el ojo del usuario a lo largo de la línea de vista. Las luces incidentes deben desviarse más allá de 20 grados de la línea de vista.

Cuando se tienen largos períodos de trabajo, es fundamental considerar lapsos de reposo que permitan relajar el cuerpo. Se recomiendan 5 minutos por cada hora, durante largas jornadas de trabajo.

Resúmen

El diseñador de sistema que desea incorporar buenos factores humanos debe, primero, considerar el número de personas que emplearán el sistema y el tipo de usuario desde el punto de vista cognoscitivo, experiencia, etc. Posteriormente, requiere determinar la secuencia de pasos para operar el sistema y, finalmente, con base en las características del usuario y del sistema arrojadas en los dos primeros pasos, intentar incluir en el diseño cada uno de los principios de los factores humanos.

Es importante tomar en cuenta que el proceso de incorporación de factores humanos no es riguroso ni exahustivo. Es decir, que no siempre es posible aplicar todos los principios de los factores humanos. Queda pues a criterio del diseñador determinar cuando es factible, técnicamente, aplicar un factor.

3.2. Ambito

3.2.1. Objetivo del sistema

El objetivo principal del sistema es automatizar los procesos relacionados con la venta de boletos, así como las demás operaciones realizadas en la Central de autobuses donde opera la línea de Autotransportes Tres Estrellas de Oro S. A. de C. V.

También se pretende dar una mejor imagen de la empresa, en su servicio a los usuarios, e internamente un mejor manejo de la información así como del control de los ingresos.

3.2.2. Interfaces de *Software, Hardware* y Humanas

Una de las secciones importantes del sistema es sin duda la correspondiente al manejo del diálogo usuario-sistema. La filosofía de la interfaz radica primordialmente en integrar un sistema

''amigable'' al usuario y que además no requiera de un gran esfuerzo para comprender lo que hace.

Un factor humano importante de tomar en consideración consiste en la destreza que tiene el usuario para operar el sistema. Podemos afirmar que la gran mayoría de los usuarios del sistema serán del tipo ''competentes'', y en algunos casos llegarán a ser catalogados dentro del nivel de ''novatos'', los cuales son deficientes en sus niveles cognoscitivos semánticos y sintácticos. Debemos diseñar por lo tanto un sistema ''fácil de entender'', esto implica que la selección de un lenguaje de comandos de tipo menús de opciones y diálogos pregunta-respuesta es fundamental.

El auxilio será provisto a través del uso de texto explicativo (ayudas), en donde se proporcionará una visión de las funciones o datos requeridos por el sistema. Este texto estará relacionado con la opción elegida en ese momento, esto es, sólo proporcionará ayuda para las opciones activas en un instante dado.

Otro tipo de ayuda es acerca de los datos que manejará el sistema, los cuales serán mostrados por medio de una ventana, en la cual se podrá elegir un dato deseado, si así lo requiere el usuario. Por ejemplo, elegir una corrida determinada.

En cuanto a los datos requeridos por el sistema, la gran mayoría se mostrarán con valores predefinidos, los cuales podrán ser modificados por el usuario de acuerdo a sus necesidades. Con esta característica se permitirá que el usuario no tenga que proporcionar muchos datos, con lo cual se incrementará la eficacia del sistema en sus tiempos de respuesta.

Sin embargo, existen algunos datos que forzosamente se deben de proporcionar al sistema, como el nombre del pasajero.

Con la finalidad de evitar errores o conflictos en la ejecución del sistema, en la interfaz se validará toda la información que se especifique, como horas de salida, lugares de destino, etc.

En lo que respecta al *hardware*, se utilizará para la entrada de información teclados y para la salida de información pantallas e impresoras de matriz de puntos; estas últimas se utilizarán para la obtención de boletos, reportes y todo tipo de salidas que requiera ser impresa.

En cuanto a interfaces humanas se cuenta con un grupo de ''taquilleros'', los cuales se encargarán de capturar la información necesaria para la emisión de boletos y la emisión de la relación de pasajeros que viajará en el autobús. Se contará con una persona responsable de la operación del sistema.

3.2.3. Alcances del sistema

Entre los alcances tenemos:

- Atender a los pasajeros en un tiempo no mayor a 5 minutos.
- Controlar las salidas de las unidades en cada viaje.
- Control de los ingresos por despachador y eficiencia en su trabajo.
- Producción de la información de reportes, los cuales muestran el comportamiento de los viajes.
- Dar un servicio de reservación por anticipado tanto de ida y vuelta en las ciudades más importantes.

Con este sistema se tendrá actualizada la información y se evitará, hasta donde sea posible, el gran manejo de la misma por parte del personal.

Con lo anterior se podrá proporcionar cualquier información respecto a los viajeros , se refinará el servicio a los viajeros, se dará solución a diversos problemas a que se enfrenta la compañía, sobre todo en el retardo de la información para la toma de decisiones.

3.3. Arquitectura

3.3.1. Módulos del Sistema

Los modulos que componen al sistema son los que se muestran en las figuras 3.1 y 3.2.

Taquillas.- Las operaciones más importantes del funcionamiento de La Central de Autobuses se realizan en este módulo, como son: la venta de boletos, reservación de boletos, venta de cupones de agencia, apertura de corridas extras, cambio y asignación de autobuses, y asignación y desasigación de despachadores.

Mantenimiento.- Se refiere al cambio de precios de tarifas, la generación de corridas del día, y el proceso para mandar información a archivos históricos.

Reportes.- Son los procesos que se refieren a la impresión de datos, como son las ventas, catálogos, reservaciones, cancelaciones y reportes mensuales.

Catálogos.- Este módulo trata de la actualización de la información necesaria para el funcionamiento del sistema, como son: itinerarios, autobuses, poblaciones, usuarios del sistema, etc.

Liquidaciones.- Aquí se controlan las salidas de los autobuses, emitiendo la guía de viaje. Los cortes de caja son emitidos en este módulo, así como los cortes del día, y de los ingresos y egresos del autobús para poder realizar su salida.

Utilerías: Este módulo contempla el proceso de *Indexación*, que permite construir los índices de las bases de datos, los cuales son utilizados en el sistema; este proceso se hace manual y no en forma automática como se hace en bases de datos de otra índole, como lo es Oracle, ya que cuando sucede una falla en el sistema puede dar una reconstrucción de índices en forma automática.

Otro proceso es el de recuperación de la información de los archivos históricos, para la emisión de reportes que se requieran.

En las figuras 3.1 y 3.2 de las siguientes páginas se muestran los módulos mencionados anteriormente.

3.3.2. Procesos y Programas a desarrollar

En el sistema de telereservaciones se necesita de varios procesos que permitan un funcionamiento eficiente del mismo, como son:

a).- Un proceso que permita desplegar la información de cualquier archivo en pantalla, es decir una ayuda, la cual se maneje através de una ventana en donde se mostrará la información solicitada por el usuario.

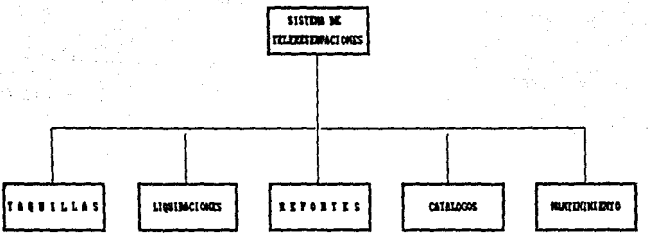
b).- En el diseño del sistema se contempla la filosofía de manejar las operaciones más comunes en una biblioteca de funciones, por ejemplo los encabezados de los reportes: el encabezado en cada pantalla de despliegue de información, la realización de la validación de la hora de captura, el borrado de una zona de la pantalla, el despliegue del estado del autobús, los mensajes de error, los textos de ayuda, etc.

c).- Otro proceso importante es el de la conexión entre redes, aquí nos auxiliaremos de las utilerías del sistema operativo *Netware*, el cuál permite configurar las características del equipo empleado para comunicación de datos en forma lejana.

d).- Entre otras cosas se requiere de un proceso que permita la entrada al sistema de telereservaciones en forma automática.

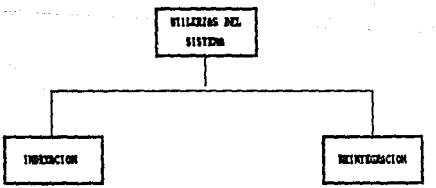
e).- En todo sistema en que se requiere guardar la información por largo tiempo, se debe tener un proceso que permita guardar dicha información en un dispositivo de almacenamiento secundario, como lo es una unidad de cinta.

f).- Otro proceso importante en el sistema de telereservaciones es el empleo de la información requerida por el usuario, es decir la información de otros lugares, que no se generan en la red local.



MODULOS DEL SISTEMA

FIGURA 3.1



UTILERIAS DEL SISTEMA

FIGURA 3.2

Los procesos anteriores son los que se requieren en el sistema de telereservaciones, adicionalmente de los programas que se muestran en la figura 3.3

3.3.3. Comportamiento Dinámico

En este punto hablaremos un poco acerca del manejo de la información.

a).-El primer paso es el registro de los datos ''primarios''que requiere el sistema para funcionar, como son: las poblaciones en donde realiza una parada el autobús, así como el kilometraje que existe entre la población de salida del autobús y las poblaciones por donde realiza su recorrido, esto es fundamental para el cálculo del precio del viaje.

Otro dato fundamental es el registro de las salidas que existen durante el día en la Central de Autobuses, así como la indicación de los lugares por donde pasa cada autobús. El registro de las series de los boletos y cupones también es indispensable, así como el de los factores de precio por kilómetro recorrido y el número máximo de estudiantes por autobús, descuentos que se pueden realizar por autobús, etc.

Otros datos necesarios son los usuarios del sistema, los posibles autobuses que puedan salir, los operadores, las agencias de viajes que puedan vender boletos de la línea de transportes (TEO), etc.

b).-El segundo paso es el manejo de la información anterior para su explotación, así como la demás información que requiere el sistema. Aquí empezaremos con lo que nosotros llamaremos ''La Generación de Dias'', que consiste en crear registros de las salidas de autobuses por día, esto es que se tenga registrado el autobús vacío para después llenarlo con cada asiento ocupado por algún pasajero.

Una vez que se realizó el proceso anterior, se tiene disponible la información para la venta, reservación de boletos o cupones. Cuando se realiza la venta de un boleto o cupón, se ocupa un lugar del autobús, además de registrarse los datos del pasajero, el tipo de pasajero, etc., por ejemplo: adulto, menor, estudiante, etc. Se registra también el importe del boleto o cupón y quién lo vendió, a que hora y en que taquilla o agencia lo realizaron. Cuando llega la hora de salida del autobús se toman los datos de quién viaja, tipo de viajero, importe y lugar de destino del pasajero, para ser emitidas en su relación llamada Guía de Viaje, donde además se agregan datos como autobús que realiza el viaje, los operadores del autobús, la dotación que se les da a los operadores para gastos del viaje y el ingreso por la venta de boletos. Una vez emitida la Guía de Viaje, no se puede vender ningún boleto en forma automática. Los datos que contiene la Guía de Viaje son

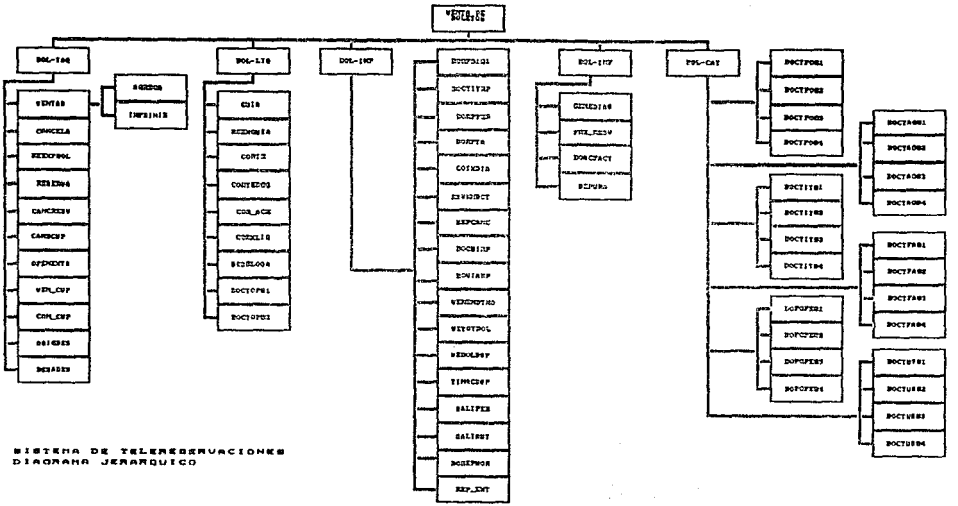


FIG. 3.3

almacenados para posteriormente ser explotados, como son la venta de boletos para la realización de cortes de caja y estadísticas que emite el sistema.

El sistema cuenta con una plantilla en donde se muestran los lugares disponibles en el autobús. Cuando se requiere de una reservación, sólo es necesario marcar el lugar seleccionado por el pasajero, para la reservación no se requiere de un boleto impreso.

Existe en la operación del sistema una opción en la cual se asignan los autobuses para cada viaje, el registro del autobús será impreso en el boleto.

El sistema puede ser ''monitoreado'' para saber el estado de las corridas y así tomar decisiones inmediatas, como ocurre cuando existe una sobredemanda de algún viaje. Adicionalmente, el sistema contempla una parte en la cual se pueden registrar las corridas extras, para cubrir la sobredemanda.

En síntesis, el sistema realizará diariamente los siguientes puntos:

a).- ''Generación de días'', apertura de la información de las corridas diarias de la línea de transportes.

b).- Venta de boletos, reservaciones, venta de cupones, remisión de boletos, remisión de cupones, cancelación de boletos, cancelación de reservaciones, cancelación de cupones.

c).- Apertura de corridas extras

d).- Asignación de Autobuses

d).- Cortes de Caja

e).- Reportes de Ventas, cancelaciones y reservaciones.

f).- Emisión de Guías de Viaje

g).- Reportes de Salidas, Guías Emitidas

h).- Depuración de la información, es decir pasar a archivos históricos la información.

i).- Regeneración de índices del sistema.

j).- Respaldo de información.

Estos son los procesos que normalmente se realizan diariamente en el sistema de telereservaciones.

3.3.4. Interacciones Organizacionales

En este punto hablaremos sobre la influencia del sistema en la empresa:

A partir de la información generada con la venta de los boletos (nos referimos a ésta como todo movimiento que implique un ingreso a la empresa), y con los cortes de caja, se conocen los ingresos de la empresa.

Adicionalmente, los cortes de caja sirven al departamento de auditorías de la empresa para sus operaciones.

Por otra parte, las Guías son un comprobante de ingresos para el autobús, dichas Guías son presentadas al final del viaje, para posteriormente emitir los cheques correspondientes (no se detalla más este proceso, debido a que no está dentro de los alcances de esta tesis).

Con los cortes de caja emitidos en el día, se realizan los depósitos de los ingresos por la venta de boletos.

Para saber si existe sobredemanda de viajes, lo que implica tener unidades disponibles para la realización de los mismos, se monitorea continuamente el sistema, con esto se podrán tomar las medidas necesarias.

Por otra parte, con los reportes mensuales por autobús, se puede observar el rendimiento de cada una de las unidades. De igual forma, mediante el reporte del operador se puede saber la eficiencia de éste.

Con la emisión de reportes mensuales, como es el de los ingresos por población, permite conocer si es rentable o no realizar paradas en dicha población. Otro reporte para la toma de decisiones es el de ventas por despachador, que junto con el de inactividad por despachador sirve para conocer el rendimiento de cada uno de éstos.

En el caso de un accidente durante el transcurso del viaje, la Guía de pasaje sirve como comprobante a la compañía de seguros de los pasajeros que viajaban en el autobús que llevaba ese viaje.

Con los puntos mencionados anteriormente, podemos conocer la importancia para la empresa del sistema de telereservaciones.

3.4. DESCRIPCION DEL DISEÑO

3.4.1. DESCRIPCION DE DATOS

Estructuras de Datos

El sistema de telereservaciones está basado en las siguientes estructuras de datos principales, (ver figura 3.4):

1. BOCTPSWD.DBF (Catálogo de usuarios)
2. BOCTAGEN.DBF (Catálogo de agencias de viaje)
3. BOCTPOB.DBF (Catálogo de poblaciones)
4. BOCTOPER.DBF (Catálogo de operadores)
5. BOCTPER.DBF (Catálogo de permisionarios de autobuses)
6. BOCTUNI.DBF (Catálogo de unidades o autobuses)
7. BOCTROLL.DBF (Catálogo de roles de viaje)
8. BOCTFAC.DBF (Catálogo de factores por kilometraje)
9. BOCTITIN.DBF (Catálogo de corridas del itinerario)
10. BOCTCORR.DBF (Catálogo de corridas de autobuses)
11. BOCTPARA.DBF (Poblaciones por las que pasa la corrida)
12. BOCTVACA.DBF (Número de descuentos por autobús)
13. BOSERIES.DBF (Series de boletos)
14. BOFOLCTE.DBF (Folio de los cortes por despachador)
15. BOFOLGIA.DBF (Folio de las guías de viaje)
16. BOREDESP.DBF (Registro de Despachadores)
17. BOMTVTAS.DBF (Maestro de movimientos de ventas)
18. BOCODE.DBF (Archivo de cortes por despachador)
19. BOGUEX.DBF (Archivo de guías de viaje expedidas)
20. BOACMCAM.DBF (Acumulado mensual por autobús)
21. BOACMDSP.DBF (Acumulado mensual por despachador)
22. BOACMDNO.DBF (Ventas acumuladas por destino)
23. BOHTRDSP.DBF (Histórico de registros de despachadores)
24. BOHTVTAS.DBF (Histórico de movimientos de ventas)
25. BOHTCODE.DBF (Histórico de cortes por despachador)
26. BOHTGUEX.DBF (Histórico de guías de viaje expedidas)
27. BOHTCORR.DBF (Histórico de corridas de autobuses)

1. BOCTPSWD.DBF (Catálogo de usuarios)

Este archivo está orientado a tener especificados los usuarios que tendrán acceso al sistema de información. La estructura de datos está diseñada para que un usuario tenga asignado un número único o *login* con el que serán identificados los movimientos que realice durante el diálogo con el sistema. Además, a cada usuario se le asignará un *password* o contraseña por medio de la cual se busca proteger o dar una característica de seguridad al sistema, ya que dicho *password* es secreto.

A cada usuario se le identifica con un nombre o descripción y un grupo al que pertenece, esto es, por ejemplo, a los usuarios que están dedicados a la venta de boletos se les identifica como que pertenecen al grupo de Despachadores, y así a los demás usuarios se les tiene clasificados por grupos. La finalidad de estos grupos de usuarios es el permitir que un solo usuario realice aquellas funciones necesarias para su trabajo.

2. BOCTAGEN.DBF (Catálogo de agencias de viaje)

Así como existen agencias de viajes de aviones, también existen agencias de viajes por autobús, por lo que el sistema en uno de sus

SISTEMA DE TELERESERVACIONES DE LINEAS DE TRANSPORTE TERRESTRE

DIAGRAMA DE FLUJO DE DATOS

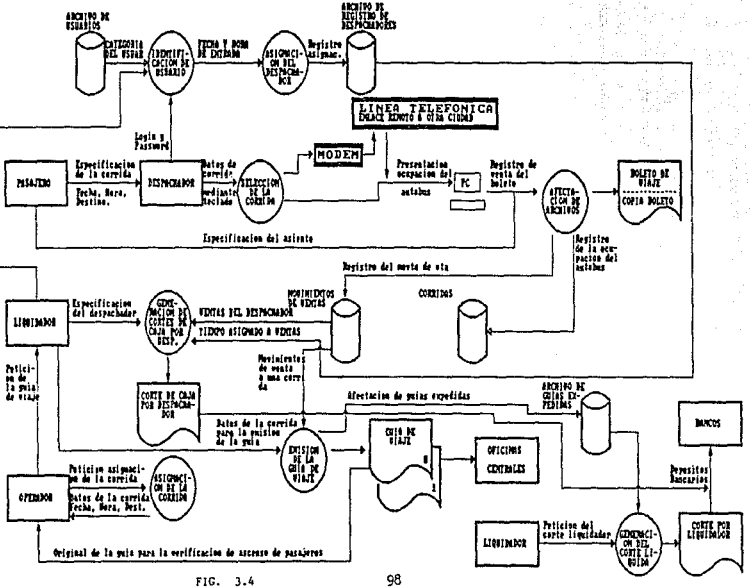


FIG. 3.4

módulos contempla la venta de cupones de agencia, los cuales al registrarse van integrando la información de ventas por agencia de viaje.

La estructura de este archivo contempla el registro de las agencias de viaje que hacen venta de boletos para la línea TEO. Entre las características de las agencias de viaje es que trabajan de acuerdo a un porcentaje de comisión sobre el importe del boleto de viaje.

3. BOCTPOB.DBF (Catálogo de poblaciones)

Esta estructura de datos contiene el catálogo de poblaciones en las que la empresa proporciona el servicio de autotransporte, la estructura contempla la clave de la población con la que se identifica, la descripción de la misma y una serie de tramos de distancia identificados con un factor por kilometraje. Para calcular el precio del boleto, el sistema hace un cálculo de cada tramo de distancia por el factor de kilometraje correspondiente.

4. BOCTOPER.DBF (Catálogo de operadores)

En esta estructura de datos se contempla la plantilla de operadores de la empresa, con sus datos: clave y nombre, ya que las guías de pasaje deben tener inscrito el ó los nombres de los operadores que realizan el viaje, por lo que este catálogo de operadores es de vital importancia en el momento de generar las guías de pasajeros por parte del liquidador.

5. BOCTPER.DBF (Catálogo de permisionarios de autobuses)

Esta estructura de datos contempla tener especificados los permisionarios de autobuses, haciendo una relación con el archivo de unidades o autobuses para aquellos reportes que requieren hacer una clasificación por permisionario y por autobús.

El archivo contiene la clave del permisionario y su descripción o nombre, mediante esta clave identifica que autobuses le pertenecen.

6. BOCTUNI.DBF (Catálogo de unidades o autobuses)

Esta estructura está diseñada para tener especificado el catálogo de autobuses que comprende la línea de autobuses, con la finalidad de hacer la asignación de la guía de viaje a un autobús específico que realizará el viaje.

La estructura contiene la clave del permisionario (dueño del autobús) y la clave del autobús. La finalidad de dicha estructura es poder presentar los ingresos por autobús y por permisionario.

7. BOCTROLL.DBF (Catálogo de roles de viaje)

Las corridas de cada central de autobuses son relativamente fijas y un autobús que sale de la central del norte con un destino tiene que cubrir un conjunto de corridas establecidas en un rol establecido por la línea de autobuses. De tal forma que un autobús que sale con destino a Guadalajara debe cubrir el recorrido después de Guadalajara a Tijuana y de Tijuana a México, pero este recorrido está establecido en un número de viaje de un determinado rol.

Así, los diversos roles están compuestos de un cierto número de viajes y cada viaje comprende una ruta de corridas. En esta estructura se tienen los datos de los diferentes roles para detectar el cumplimiento de los mismos.

8. BOCTFAC.DBF (Catálogo de factores por kilometraje)

En esta estructura se registran los diversos factores por kilometraje que definen el precio del boleto, es decir, el precio de un boleto se basa en los kilómetros de distancia a los que está la población a la que se viaja, multiplicados por uno o varios factores que establece la SCT (Secretaría de Comunicaciones y Transportes), dependiendo de la zona geográfica.

La estructura de datos comtempla el identificar a cada uno de los factores mediante una clave, así como la utilidad de los factores, en períodos en los que hay variaciones de los factores, para incrementar los precios de los boletos. Cuando vaya a haber un incremento en los factores, a partir de una fecha dada, el sistema debe calcular el precio de los boletos antes del incremento y después del incremento, ya que, recordemos que se pueden hacer ventas de boletos hasta con un mes de anticipación. Por lo cual, el sistema debe conocer el valor de los factores antes de la fecha de incremento y después de ella.

9. BOCTITIN.DBF (Catálogo de corridas del itinerario)

La estructura de datos de este archivo permite tener la información relacionada con el itinerario de corridas de las centrales de autobuses.

Las corridas de una línea de autobuses generalmente son siempre las mismas día a día, y son las que conforman el itinerario, aunque hay algunas corridas que se improvisan, las corridas extras. En ésta estructura se contempla la posibilidad de especificar corridas que saldrán solamente durante un cierto período de tiempo y ciertos días de la semana, así como especificar otras características como son el cupo del autobús, asientos reservados, etc.

En ésta estructura se utilizó un vector de carácteres para representar la información de los asientos del autobús.

10. BOCTCORR.DBF (Catálogo de corridas de autobuses)

Este archivo contiene la estructura de datos utilizada para representar las corridas de autobuses diarias. Es este archivo se encuentran registradas todas las corridas y su estado actual, es decir, su ocupación. Aquí es donde el sistema representa, además de las corridas, la información de pasajeros en el autobús. Al igual que en el catálogo de corridas del itinerario, se contempla una estructura de un vector de carácteres para representar los asientos del autobús. También en esta estructura se tiene la especificación de las paradas o poblaciones por las que pasa la corrida, ya que el sistema es capáz de detectar cuando se quiere hacer una venta a una población que no está dentro de la ruta de la corrida.

Cuando un despachador está vendiendo para una corrida en particular, esta corrida se marca como bloqueada y se identifica quien la tiene ocupada, para que cuando otro despachador intente vender a la misma corrida pueda percatarse de que está ocupada y debe intentar la venta más tarde y no provocar colisiones de información. El archivo contempla para esto un campo que permite llevar el control de bloqueos de las corridas.

11. BOCTPARA.DBF (Poblaciones por las que pasa la corrida)

La estructura de datos diseñada para este archivo contiene la identificación de la corrida a la que se hace referencia, la clave de la población por la que pasa la corrida, un consecutivo de las diferentes paradas de la corrida y una estructura de datos de 50 carácteres para representar los asientos que no pueden ser vendidos en la población origen o anterior a la población que se está especificando como parada. Este tipo de asientos se denominan asientos exclusivos de alguna parada, ya que sólo esta población puede venderlos.

12. BOCTVACA.DBF (Número de descuentos por autobús)

En esta estructura se tiene el número de descuentos otorgados por la empresa en cada autobús, para la temporada de vacaciones y para la temporada normal. Esto es, en cada autobús el sistema permitirá como máximo dos descuentos de senectud y para la temporada de vacaciones hasta 4.

En la estructura se contempla el número de descuentos autorizados para cada uno de los tipos de descuentos como: senectud, estudiantes, profesores, etc.

13. BOSERIES.DBF (Series de boletos)

En este archivo se ubica la función de venta de boletos, en la cual los Despachadores realizan la venta de pasajes a los diferentes destinos. Los boletos que se venden están identificados con un folio consecutivo asociado o relacionado con un número de taquilla, es decir, cada taquilla está identificada con una letra

del abecedario (A, B, etc.) y para cada taquilla se lleva el control de un folio o serie consecutivo del boleto. En esta estructura de datos es donde se tienen los folios consecutivos por taquilla.

14. BOFOLCTE.DBF (Folio de los cortes por despachador)

En esta estructura de datos se lleva el folio de los cortes de despachador expedidos, esto es porque dichos cortes se archivan en caso de alguna auditoria, tanto interna como externa, con la finalidad de detectar problemas en los cortes de los cajeros.

15. BOFOLGIA.DBF (Folio de las guias de viaje)

El liquidador es el encargado de hacer la emisión de las guías de viaje y éstas deben ir foliadas consecutivamente para que en la administración de la empresa se lleve la contabilización de los ingresos del autobús.

16. BOREDESP.DBF (Registro de despachadores)

El sistema de telereservaciones lleva el control de la venta de pasajes por cada despachador, y a su vez emite un corte de caja por despachador. Para identificar cuáles son las ventas de un despachador en particular, se pide al despachador que para poder vender necesita registrar su entrada o asignación al módulo de ventas, así como su salida. De esta forma el sistema genera el corte de caja con todos los movimientos registrados en el horario comprendido entre la entrada y salida del depachador y que corresponden con los movimientos realizados en la taquilla en la que se asignó dicho despachador.

Además, en esta estructura se lleva el control de los minutos de tiempo inactivo del despachador durante su jornada de trabajo. Este tiempo inactivo está calculado a partir de una estimación de 5 min, dedicados a cada movimiento de ventas restado del total de tiempo laborado durante el turno del despachador.

17. BOMTVTAS.DBF (Maestro de movimientos de ventas)

Cada registro de ventas se encuentra registrado en este archivo, también se registran los movimientos de reservaciones, cancelaciones (de venta de boletos, reservaciones, boletos canje), descuentos (profesores, ancianos, estudiantes, especiales 50 %, etc.). De tal forma que esta estructura contempla todos los datos involucrados en alguno de estos tipos de movimientos, como son la corrida en la que se vende o cancela un boleto, el número de asiento escogido, autorización del descuento por el supervisor, etc.

18. BOCODE.DBF (Archivo de cortes por despachador)

Cada vez que se emite un corte de algún despachador, éste es

foliado y se genera un registro que indica la fecha, hora y los ingresos generados por algún despachador, esta estructura contiene dicho registro.

19. BOGUEX.DBF (Archivo de guías de viaje expedidas)

En esta estructura se tiene contemplado llevar un registro de cada una de las guías de viaje que emite el liquidador, con todos los datos de la misma, como son: Fecha, hora, destino, ingresos, dotaciones, clave del liquidador que emite, etc.

La finalidad de dicha estructura es poder generar cortes de caja para los liquidadores, en base al importe global de dotaciones entregadas a los operadores, así como también emitir un reporte de las salidas por hora.

20. BOACMCAM.DBF (Acumulado mensual por autobús)

Esta estructura permite llevar el control de viajes por cada uno de los autobuses, ya que contiene todas las corridas que ha realizado un autobús durante el mes, especificando la fecha de viaje, la hora, el destino y los ingresos de dicha corrida. Este archivo es afectado en el módulo del sistema donde se efectúa la depuración de movimientos. Esta depuración borra todos los movimientos a partir de dos días atrás de la fecha actual y los pasa a un archivo histórico a la vez que afecta los acumulados.

21. BOACMDSP.DBF (Acumulado mensual por despachador)

Esta estructura de datos tiene una función similar a la anterior, llevar los acumulados de venta de boletos por cada uno de los despachadores. Al depurar los movimientos de venta, se genera el acumulado de venta de boletos en cantidad y en importe por cada uno de los despachadores, así como también se separan los boletos por tipo de autobús (autobús de primera o autobús de clase expreso).

La estructura de datos contempla también el generar acumulados por cada uno de los tipos de boletos vendidos (para adultos, menores, ancianos, estudiantes, etc.).

22. BOACMDNO.DBF (Ventas acumuladas por destino)

Este archivo tiene la estructura necesaria para generar los acumulados por destino, es decir el acumulado de las ventas de boletos durante el mes, tanto en número como en importe, en corridas de clase de primera, en corridas de clase expreso y por cada uno de los tipos de boletos que se venden.

Este archivo se genera cuando se hace la depuración de movimientos para pasar la información a archivos históricos.

23. BOHTRDSP.DBF (Histórico de registros de despachadores)

La estructura de datos de este archivo es igual al archivo BOREDESP, se utiliza para ir llevando la información histórica del registro de los despachadores, es decir, se tiene la información de los días en que estuvo trabajando el despachador, así como la hora de entrada y de salida.

24. BOHTVTAS.DBF (Histórico de movimientos de ventas)

Esta estructura es igual a la del archivo BOMTVTAS.DBF y contiene el registro histórico de los movimientos de ventas efectuados (incluyendo, cancelaciones, reservaciones y descuentos especiales). Este archivo tiene la finalidad de no saturar los archivos de trabajo con información poco reelevante y que redundaría en hacer que los procesos de reconstrucción de índices fueran demasiado lentos.

25. BOHTCODE.DBF (Histórico de cortes por despachador)

Esta estructura es idéntica a la estructura de datos de Cortes por despachador y funciona como un histórico, en donde se pasa toda la información de cortes por despachador de 2 días atrás, a partir de la fecha de operación del sistema. La finalidad de los históricos es mantener con un performance adecuado la operación del sistema, eliminando información que no es relevante, pero respetándola en caso de ser necesaria una consulta.

26. BOHTGUEX.DBF (Histórico de guías de viaje expedidas)

Esta estructura es igual a la de guías de viaje expedidas y contiene el registro histórico de las mismas. Esta se usa en el caso de requerir de alguna consulta de guías de viaje expedidas en días anteriores.

27. BOHTCORR.DBF (Histórico de corridas de autobuses)

Esta estructura es igual a la utilizada para las corridas de autobuses y también contiene información histórica del sistema.

3.5. Detalle de Módulos

3.5.1. Texto Explicativo

El sistema está dividido en siete módulos, los cuales se encuentran divididos de la siguiente forma:

- Taquillas
- Liquidaciones
- Reportes
- Mantenimiento
- Catálogos

Existen otros dos módulos que son de utilidad:

- Indexación
- Reintegración de información

Módulo de Taquillas:

Este módulo es uno de los más importantes, ya que dentro de él se encuentran las operaciones más importantes de la empresa, como es la venta de boletos, cancelación, reexpedición, reservación, cancelación de reservaciones, cambio de cupones (los cupones son vendidos en las agencias de la empresa), aperturas de corridas extras (esto es cuando existe demanda de pasaje a distintos destinos), venta de cupones, cancelación de cupones, asignación de los despachadores y desasignación de los mismos.

El módulo de taquillas está formado por los siguientes procesos:

-VENTAS
-CANCELA
-REESPBOL
-RESERVA
-CANCRESV
-CAMBCUP
-OPENEXTS
-VEN_CUP
-CAN_CUP
-ASIGDES
-DESADES

Módulo de Liquidaciones

En este módulo se efectúa el control de guías de pasajeros de cada autobús, reexpedición de guías, el corte de caja, reimpresión de corte de caja, corte por liquidador, corte por agencia, desbloqueo de corridas, esto se usa cuando en una de las terminales se queda bloqueada una corrida, altas de catálogos de operadores y bajas de catálogos de operadores

Este módulo se encuentra formado por los siguientes procesos:

-GUIA
-REEXGUIA
-CORTE
-CORTEDOS
-COR_AGE
-CORXLIQ
-DESBLOQA
-BOCTOP01
-BOCTOP02

Módulo de Reportes:

En este módulo se encuentran todos los reportes como son: reporte de intinerarios, reporte de permisionarios, reporte de tarifas, reporte del corte del dia, reporte de reservaciones y cancelaciones de boletos, reporte de cancelaciones de boletos, reporte de guías, reporte que genera las salidas que hubo en el día, reporte de ventas mensuales por destino, reporte de venta total de boletos por mes, reporte de ventas de boletos por despachador, reporte de inactividad por despachador, reporte de salidas de autobuses por permisionarios y reporte de salidas por autobús por día.

Este módulo esta integrado de la siguiente manera:

-BORPD101
-BOCTITRP
-BORPPER
-BORPTA
-COTXDIA
-RSVKNDCT
-REPCANC
-BOGUIRP
-BOVIARP
-VEMENDTNO
-VETOTBOL
-VEBOLDSP
-TINACDSP
-SALIPER
-SALIAUT
-BOREPHOR
-REP_EXT

Módulo de Mantenimiento:

En este módulo se considera la generación de archivos históricos, así como las corridas del día y la afectación de los precios.

Este módulo se encuentra integrado de la siguiente manera:

-GENEDIAS
-PRE_RESV
-BOACFACT
-DEPURA

-Módulo de Catálogos:

En este módulo se efectúa las altas, bajas, cambios y consultas del registro de poblaciones, intinerarios, permisionarios, agencias, operadores, usuarios, etc.

Se encuentra integrado de la siguiente manera:

Poblaciones

-BOCTP001
-BOCTP002
-BOCTP003
-BOCTP004

Intinerarios

-BOCTITO1
-BOCTITO2
-BOCTITO3
-BOCTITO4

Permisionarios

-BOPGPE01
-BOPGPE02
-BOPGPE03
-BOPGPE04

Agencias

-BOCTAG01
-BOCTAG02
-BOCTAG03
-BOCTAG04

Factores

-BOCTFA01
-BOCTFA02
-BOCTFA03
-BOCTFA04

Usuarios

-BOCTUS01
-BOCTUS02
-BOCTUS03
-BOCTUS04

Por último el módulo de utilerías contempla lo siguiente:

Proceso de Indexación:

Este proceso permite crear los archivos de índices que utiliza el sistema para su funcionamiento.

Proceso de Reintegración:

Es el encargado de integrar la información de los archivos históricos que genera el sistema a los archivos maestros, de los cuales se obtienen todos los reportes del sistema.

3.6. Estructura de Programas

La estructuración de los programas se hizo basada en las características del lenguaje de programación (CLIPPER), para aprovechar las bondades que nos brinda.

CLIPPER es un lenguaje que permite hacer una programación estructurada y tiene la característica de que al compilar, genera una estructura arborescente de los programas compilados, de tal forma que se puede observar claramente como se hacen las llamadas entre los diversos programas que comprenden el sistema.

La forma de estructurar los programas se hizo clasificando la programación por medio de niveles de programación:

1) Nivel Aplicación .- En este nivel se ubican todos aquellos programas directamente relacionados con la aplicación, por ejemplo, los programas encargados de desplegar la ocupación de un autobús, registrar el movimiento de venta, etc.

2) Nivel Rutinas de aplicación.- En este nivel se ubican aquellas rutinas que son empleadas por los diversos programas que se encuentran al nivel aplicación, por ejemplo, el despliegue de los tipos de pasajeros que van en el autobús: senectud, adultos, menores, etc.

3) Nivel Rutinas de uso general.- Aquí se ubican todas aquellas rutinas que son de uso completamente general, es decir, que incluso no dependen del sistema a desarrollar, ejemplos de estas rutinas son: rutinas de manipulación de cadenas, rutinas estandares de formateo de reportes, mensajes de error, etc.

4) Nivel Rutinas en lenguaje C.- Al desarrollar el sistema, observamos que para lograr una interfaz sencilla con el usuario, era necesario de hacer uso de una presentación de pantallas, con toda aquella información que resulta útil para la venta de boletos, y la forma de presentarla debe ser lo más natural posible. CLIPPER, aunque es un lenguaje muy versátil y sencillo, no permite hacer accesos directos a la memoria de video, por lo que hubo la necesidad de crear una interfaz con lenguaje C e implementar algunas rutinas en dicho lenguaje, éstas se clasificaron en este nivel, aunque también se clasificaron en este nivel por requerir de una programación más compleja.

La programación a nivel de la aplicación se encuentra contenida en archivos ASCII, nombrados mediante 8 carácteres y una extensión para facilitar la identificación de programas, estos 8 carácteres están estructurados de la siguiente manera:

a) 2 carácteres que identifican al sistema o la función principal del mismo:

p. ej. BO --> BOLETOS (VENTA DE BOLETOS)

b) 2 carácteres que indican el tipo de archivo de datos con el que trabaja principalmente el programa.

p. ej. CT --> CATALOGO
MT --> MAESTRO
HT --> HISTORICO
.
.

c) 2 carácteres que indican el archivo de datos principal que ocupa el programa.

p. ej. US --> USUARIOS
IT --> ITINERARIOS
.
.

d) 2 carácteres que especifican una función en particular.

p. ej. 01 --> ALTAS
02 --> BAJAS
03 --> CAMBIOS
.
.

Por ejemplo: BOCTUS01.PRG

De acuerdo con la estructura planteada, podemos decir que este archivo es un programa que utiliza un archivo de usuarios, el cual es un catálogo y que el programa tiene la función 01 (Altas).

Los demás niveles de programación tienen una estructura similar, con la finalidad de que el posible mantenimiento futuro del sistema sea lo más ágil posible.

Es importante destacar una característica importante del sistema en cuanto a las ayudas que se proporcionan en línea para auxiliar al usuario en datos que debe proporcionar al mismo. El sistema contemplará un archivo HELP.PRG, el cual es un archivo de default de CLIPPER para proporcionar ayudas en línea. La característica importante es que el sistema se estructurará con este archivo y

otro llamado AYUDA.PRG, para poder proporcionar en cada campo que se pida al usuario una consulta inmediata de la base de datos en donde se encuentra dicha información.

La estructura de esta ayuda rompe en cierta medida con el concepto de programación estructurada, porque dentro de la secuencia de ejecución de algún programa de aplicación se puede dar el caso de que se esté ejecutando una instrucción de solicitud de datos (GET, READ) y si se presiona la tecla F1 automáticamente el control de ejecución pasa al programa HELP.PRG.

Aunque no se adapte muy bien a un esquema de programación estructurada, resulta ser una herramienta muy poderosa para el usuario, para operar el sistema de una forma muy ágil. Por otra parte, la estructura de programas se sigue teniendo bien definida, sólo que hay que contemplar estas posibles interrupciones por la solicitud de ayuda en la estructuración de los programas en sí.

CAPITULO 4

IMPLEMENTACION

Y

PRUEBAS

4. IMPLEMENTACION Y PRUEBAS

4.1 IMPLEMENTACION

A pesar del paradigma de la ingeniería del *software*, el lenguaje de programación impactará a cada una de las etapas del proyecto, pues en algunos casos éste tiene un papel muy relevante incluso en los requerimientos de operación. Pero aunque ocurra lo anterior, el papel del un lenguaje de programación ha de mantenerse en perspectiva. Los lenguajes proporcionan los medios de traducción hombre-máquina; sin embargo, la calidad del resultado final se encuentra más fuertemente ligado a las actividades de la ingeniería del *software* que preceden y siguen a la codificación, es decir, a las etapas de análisis y diseño.

Durante el paso de planificación del proyecto, raramente se toman en consideración las características técnicas de un lenguaje de programación. Sin embargo, la planificación de las herramientas de soporte asociadas con la definición de recursos puede requerir que se especifique un compilador en particular (y su *software* asociado) o un entorno de programación. La estimación de costos y el plan de trabajo puede requerir de ajustes a la curva de aprendizaje debido a la inexperiencia de la plantilla de desarrollo en el lenguaje de programación especificado.

Una vez que se han establecido los requerimientos del *software*, las características técnicas de los lenguajes de programación que son candidatos, se hacen más importantes. Si se requieren complejas estructuras de datos, habrá que evaluar cuidadosamente los lenguajes que soporten sofisticadas y bien definidas estructuras de datos. Si lo importante es un alto rendimiento y posibilidades de tiempo real, se deberá especificar un lenguaje diseñado para aplicaciones en tiempo real o para eficiencia en memoria-velocidad. Si se especifican muchos reportes de salida y una fuerte manipulación de archivos, se encontrarán lenguajes como COBOL, RPG, etc. Idealmente, los requerimientos del *software* habrían de llevar a la elección de un lenguaje que mejor se ajuste al procesamiento que se ha de llevar a cabo. En la práctica, sin embargo, a menudo se selecciona un lenguaje porque... ''es el único que se tiene en el equipo de cómputo''.

La calidad del diseño del *software* está establecida por la forma en que sea independiente de las características de los lenguajes de programación. Sin embargo, los atributos del lenguaje juegan un papel muy importante en la calidad de un diseño final, y afectan profundamente a la especificación del diseño.

El diseño de datos también puede verse influenciado por las características del lenguaje. En la actualidad, existe una nueva clase de lenguajes de programación (ADA, CLU, Smalltalk y otros) que soportan el concepto de tipo de datos abstracto, obtenidos por el usuario, la implementación directa de listas enlazadas y otras

estructuras de datos. Estas posibilidades proporcionan un gran poder durante los pasos de diseño preliminar y detallado.

El efecto de las características de un lenguaje de programación en los pasos que componen la prueba del software es difícil de determinar. Los lenguajes que soportan directamente las construcciones estructurales tienden a reducir la complejidad de un programa, haciéndolo de alguna forma más fácil de probar. Los lenguajes que soportan la especificación de subprogramas y procedimientos externos hacen que la prueba de integración sea mucho menos propensa a errores. Por otro lado, algunas características técnicas de los lenguajes pueden impedir la prueba. Por ejemplo, la estructuración de bloques en ALGOL se puede especificar de tal forma que produzca la pérdida de datos intermedios cuando se produce la salida de un bloque, con lo que es mas difícil de descubrir el estado de un programa.

Al igual que con la prueba, no se comprende totalmente el efecto de las características de los lenguajes de programación sobre el mantenimiento del *software*. Sin embargo, no hay duda de que las características técnicas pueden mejorar la legibilidad del código y reducir la complejidad, lo que es importante para un mantenimiento efectivo.

La elección de un lenguaje de programación depende de una infinidad de características que no están completamente definidas o especificadas con claridad, ya que varias de ellas pueden estar directamente involucradas con el ámbito del proyecto.

En el presente trabajo se seleccionó a CLIPPER como enguaje de programación, que para el proyecto resulta ser uno de los lenguajes más apropiados desde el punto de vista del desarrollo, prueba y mantenimiento, ya que CLIPPER ofrece una gran flexibilidad en la definición de las estructuras de datos de archivos, así como también posee una lógica de control de bloqueos bastante simple y práctica de implementar en el caso de la aplicación que representa el Sistema de Telereservaciones.

Para realizar la implementación del Sistema de Telereservaciones, se llevaron a cabo las siguientes etapas o fases:

1) Planeación.- A partir del diseño del sistema, representado por un diagrama de flujo de información, así como de las estructuras de datos propuestas e interfaces mediante pantallas se prosiguió a planear la implementación, considerando las diversas entradas y salidas de información que requeriría el sistema.

2) Programación.- En ésta fase se prosiguió a identificar los módulos que compondrían al sistema así como las funciones asignadas a cada uno de ellos para así desarrollar el conjunto de programas necesarios. La programación se dividió prácticamente en niveles de

abstracción, aplicando las metodologías de la programación estructurada para facilitar la prueba y depuración del *software*.

3) Pruebas de Unidad.- En esta fase se probaron los módulos en forma independiente para verificar que ya no existían errores de programación, así como errores de semántica en los mensajes de ayuda del sistema, sintaxis de la información desplegada por pantalla y por los reportes, formateo de cantidades numéricas, alineación de claves, etc.

4) Correciones de Unidad.- Esta etapa consistió en la corrección de aquellos errores detectados en la fase anterior y que se pasaron por alto durante la etapa de programación, fué muy importante esta fase ya que en la fase de programación se olvidan muchos detalles que repercuten en *bugs* del sistema que provocan fallas de trascendencia en una operación real del sistema.

5) Pruebas de Integración.- Consistió en la prueba de los diversos módulos que interactúan entre sí, como por ejemplo, el módulo de venta de boletos afecta los archivos de trabajo que contendrán los movimientos relacionados con la venta de boletos, esta información es utilizada posteriormente por el módulo de guías de pasaje, en donde es muy importante que la información generada por el primer módulo esté perfectamente validada y a su vez se necesita que el segundo módulo haga una interpretación de la misma acorde con el primero. Esta fase tiene su importancia por los posibles *bugs* que se ocultan por la apariencia de un buen funcionamiento del *software*.

6) Correcciones de Integración.- Esta fase también consistió en corregir todos aquellos errores detectados en la fase anterior.

7) Instalación de la red.- Esta etapa fué muy importante ya que en ésta se inicializó uno de los equipos para que trabajara como *file server*, se hicieron los procesos necesarios para cargar el sistema operativo en el disco duro, así como también se configuró el *shell* de arranque de las estaciones de trabajo. Una vez instalada la red se procedió a realizar las pruebas necesarias para verificar la comunicación entre las diversas terminales con el servidor de archivos. Además se instaló el *software* de comunicacion remota por módem para interconectar los diversos *file servers* de la red *WAN* entre las diversas ciudades. Se hicieron las pruebas de comunicación remota para verificar el adecuado funcionamiento de la red.

8) Pruebas de verificación de requerimientos.- Esta fase consistió en volver a realizar las pruebas de integración de forma tal que se simulara la operación normal del sistema ya instalado en cada una de las centrales de la empresa. Esta fase fué muy importante para checar que los diversos bloqueos de los registros de datos estuviesen bien validados, de tal forma que la información fuese afectada por el usuario correcto y solucionar los problemas

de concurrencia de usuarios a un mismo registro, también sirvió para detectar problemas en tiempos de procesamiento y tiempos de respuesta adecuados, etc.

9) Modificaciones finales.- Esta última fase consistió en hacer las adecuaciones necesarias para resolver los últimos detalles encontrados en el *software*.

En este capítulo se mostrará la implementación de algunos programas del sistema en base al lenguaje de programación seleccionado, dicho lenguaje se escogió como se mencionó anteriormente por la gran versatilidad que ofrece y a la facilidad de aprendizaje para programar. La versatilidad que muestra CLIPPER se observa entre otras cosas, en el manejo de los archivos, en los cuales, se tiene la especificación de datos sin depender de los programas que los utilicen. Tal vez no ofrece todas las características de una base de datos como RDB/VMS (de ***Digital Equipment Corporation***) pero de una manera simple permite definir archivos de datos sin tener que hacer una especificación de la estructura dentro de los programas, así como también permite definir índices de acceso por medio de los cuales se puede tener un acceso a la información por medio de llaves en forma inmediata.

La etapa de implementación depende en gran medida de la especificación del diseño, en donde entre más aproximado sea al funcionamiento último del sistema resulta más útil porque hay que considerar las características particulares del lenguaje de programación. Tal vez si se especificó en el diseño la estructura de los programas, en cuanto aclasificarlos por la función que desempeñan, basados en una metodología de diseño de programas por capas, es decir, organizar la programación por la complejidad de algoritmos y por la posible reutilización del código representa ventajas importantes, si se consideran las características propias del lenguaje, o bien, del compilador y ligador. En seguida hablaremos más a detalle de la filosofía de diseño por capas empleada en el presente trabajo.

Diseño por capas

Esta filosofía de diseño se encuentra muy bien ejemplificada en los sistemas operativos. En los sistemas operativos las capas más internas las forman rutinas que manejan el ***hardware***, las capas superiores hacen uso de las capas inferiores y van desligando al ***software*** de la aplicación de tareas primitivas.

De una manera semejante, el sistema de telereservaciones está sustentado en rutinas que forman familias. Cada familia de rutinas está sujeta a funciones muy específicas. Las familias de rutinas de niveles superiores únicamente hacen uso de rutinas de familias inferiores o de su mismo nivel.

Las ventajas de este tipo de diseño son las siguientes:

1.- Se evita la repetición de código. Con el uso de diseño por capas, antes de comenzar la programación se identifican las necesidades comunes, y se crean familias de rutinas con funciones semejantes.
2.- Facilidad para programar rutinas complejas. Las rutinas que se encuentran en las capas más externas, basan sus algoritmos en el uso de rutinas de niveles internos, de esta manera no se adentran en los detalles y resulta mucho más fácil la programación.

3.- Se facilita el mantenimiento. En caso de requerir hacer cambios a los algoritmos, ya sea por mejoras o por modificaciones. Si estos cambios afectan a muchas rutinas, posiblemente con las modificaciones de un solo grupo de rutinas se logre el cambio.

Para ejemplificar la filosofía del diseño por capas, se presentará una breve explicación.

Debido a que la terminal de video es el instrumento de comunicación con el usuario. Se decidió diseñar la pantalla de forma tal que el usuario tenga zonas bien definidas para cada tipo de información presentada en la computadora, facilitando la comunicación con el usuario cuando se encuentra operando el sistema, ya que cada zona la asociará con un tipo de indicación por parte del sistema.

Zona de Encabezados

Menús

Area de trabajo

Mensajes del Sistema

En la zona de encabezados se mostrara un encabezado dependiendo del módulo en donde se encuentre el usuario.

La zona de menús y área de trabajo estará ocupada por el menú de opciones que puede utilizar el usuario. Si en alguna opción es necesario capturar bastante información, la zona se ocupará por una forma de captura. Una vez que se termine de usar la opción de captura, esta zona mostrará el menú de opciones que corresponden al nivel en donde se encuentre.

Si el usuario requiere de despliegue de información en la región de menús y captura, se abrirá una ventana en donde se mostrará la información solicitada por el usuario.

En la zona de mensajes del sistema, cuando el usuario incurra en algún error aparecerá un mensaje explicando el motivo del error,

sobre una barra iluminada que va corriendo de derecha a izquierda para hacer más natural su lectura y para facilitar el despliegue de mensajes demasiado largos.

Este diseño de la interfaz con el usuario deja ver que el diseño del *software* deberá contemplar rutinas de manejo de video que cumplan con las funciones de despliegue de mensajes informativos, errores, despliegue de datos, etc.

Para cumplir con los requerimientos de diseño, se tendrá clasificado el *software* por capas, de tal forma que tengamos rutinas de manejo de video que a su vez serán manipuladas por rutinas de niveles superiores para la visualización de un tipo particular de información en la pantalla de la computadora.

Al estar capturando un campo que es llave principal de algún archivo, el sistema proporciona una ayuda en línea, abriendo una ventana con la vista general del archivo en donde se encuentra esa llave junto con algunos datos generales. Esta ayuda tiene la finalidad de auxiliar al usuario con una consulta inmediata de datos que requiere para la operación del mismo.

Por medio de estas características de funcionamiento de cada una de las zonas se pudieron distinguir una serie de familias de rutinas.

El nivel más interno lo constituyen las familias de rutinas de detección de teclado y de manejo de video como se explicó anteriormente.

El segundo nivel lo constituyen las rutinas de utilerías y ayudas.

El tercer nivel lo constituyen los programas de aplicación, como son los menús muy particulares de la aplicación, los reportes de explotación de datos, las capturas de información de las corridas extras y normales, etc.

Una de las ventajas de las dos capas inferiores, es el uso de éstas rutinas en otros sistemas, ya que son de propósito general y se les puede dar un gran uso como herramientas para el desarrollo de sistemas.

A continuación detallaremos la implementación de algunas rutinas del sistema para aclarar el funcionamiento del software del Sistema de Telereservaciones.

Como se mencionó anteriormente, el sistema proporciona ayuda en línea sobre los campos de captura que se refieren a llaves de algún archivo. Por ejemplo, estando dentro del módulo de catálogos, en el catálogo de usuarios, si pedimos hacer una modificación, el sistema pide que digitemos el *login* de entrada del usuario que deseamos

modificar, para que por medio de este campo y por medio de un índice que se encuentra definido por este campo haga la búsqueda.

En la siguiente hoja se presenta un listado del programa BOCTUS03.PRG, el cual es el programa de modificaciones del catálogo de usuarios. En seguida explicaremos el funcionamiento del mismo indicándolo mediante las líneas de programación numeradas en el listado.

Las líneas 6 y 7 se encargan de referenciar al archivo de datos y a los índices de acceso, que son dos, *login* del usuario y el nombre del usuario.

En la línea 18 se hace la petición del *login* del usuario por medio de la instrucción MSG, en la 19 se prepara el campo de video protegido para la captura del *login* en la variable de memoria xpass_login y el READ de la línea 15 hace la lectura.

Desde la perspectiva del usuario, cuando el sistema pide el login del usuario a modificar, si él presiona la tecla F1, automáticamente el sistema abre una ventana con todo el catálogo de usuarios para que consulte el que quiere modificar, e incluso se puede traer el login que seleccione de dicha ventana.
Toda esta lógica de funcionamiento no está contenida en el programa que estamos analizando, sino que se encuentra en otro módulo objeto del sistema formado por los programas HELP.PRG y AYUDA.PRG, en la siguiente página se muestran los listados de dichos programas.

Ahora trataremos de explicar el funcionamiento de los programas, auxiliándonos del número de línea de programación al que haremos referencia.

Empezando con el programa HELP.PRG, cuando se presiona la tecla F1 estando en ejecución el sistema, CLIPPER atrapa ésta tecla y automáticamente busca el código del programa HELP.PRG para ejecutarlo, de tal forma que en este momento pasa el control a dicho programa con los siguientes tres parámetros:

1. El programa que se encontraba ejecutando en el momento en que se presionó la tecla F1.

2. El número de línea del programa que se estaba ejecutando.

3. El nombre de la variable que estaba leyendose.

La utilización de estos tres parámetros es muy importante, por ejemplo, en la línea 7 se está preguntado si el nombre del programa que se estaba ejecutando es HELP.PRG o AYUDA1.PRG, esta pregunta tiene la función de filtrar aquellas llamadas con F1 dentro de los programas para que no se dé el caso de llamadas recursivas a los programas de ayuda.

```
01 *
02 * Sistema..: Sistema de Telereservaciones para Lineas de Autotransporte
03 * Titulo..: Programa de Modificaciones al Catalogo de Usuarios.
04 * Nombre..: BOCTUSO3.PRG
05 *
06 select 9
07 set index to boctpswd,boctps01
08 borra()
09 pantausua()
10 tit('Modificaciones al Registro de Usuarios')
11 do while .t.
12   xpass_login  = space(08)
13   xpass_nom    = space(30)
14   xpass_pswd   = space(20)
15   xpassword    = space(20)
16   xpassaux     = space(20)
17   xpass_gpo    = space(01)
18   msg('Digite el Login del Usuario')
19   @ 10,32 get xpass_login picture '99999999'
20   read
21   if lastkey()=27
22     exit
23   endif
24   @ 10,32 say xpass_login
25   seek xpass_login
26   if ! eof()
27     if rec_lock(1)
28       xpass_nom  = pass_nom      && carga los datos de la parada
29       xpassaux   = pswd(pass_pswd)
30       xpassaux   = xpassaux+space(20-len(xpassaux))
31       xpass_gpo  = pass_gpo
32       desp_usua()
33       opc_e=1
34       do while .t.
> y pulse DY para elegir')
36         @ 12,12 prompt 'Nombre del Usuario:'
37         @ 14,12 prompt 'Password.....:'
38         @ 16,12 prompt 'Grupo de Usuarios.:'
39         menu to opc_e
40         @ 12,12 say 'Nombre del Usuario:'
41         @ 14,12 say 'Password.....:'
42         @ 16,12 say 'Grupo de Usuarios.:'
43         do case
44           case opc_e=0
45             replace pass_nom  with  xpass_nom   pass_pswd  with pswd(xpass_pswd) ;
46                     pass_gpo  with  xpass_gpo
47             @ 21,17 clear to 21,78
48             @ 21,17 say xpass_login+' '+substr(xpass_nom,1,15)+' '+grupofat(xpass_gpo,all_grupos)  && despliega ultima modificac
n
49             unlock
50             exit
51           case opc_e=1
52             msg('Digite el Nombre del Usuario')
53             @ 12,32 get xpass_nom picture '@!'
54             read
55             @ 12,32 say xpass_nom
56           case opc_e=2
57             xpass_pswd   = space(20)
58             xpassword    = space(20)
59             msg('Digite el Password del Usuario')
60             set color to ,n/n
```

```
61            @ 14,32 get xpass_pswd picture '@!' valid !empty(xpass_pswd)
62            read
63            set color to
64            msg('Digite nuevamente el Password del Usuario para rectificar')
65            set color to ,n/n
66            @ 14,32 get xpassword  picture '@!' valid !empty(xpassword)
67            read
68            set color to
69            if xpass_pswd<>xpassword
70              error('No es correcto el PASSWORD retecleado.')
71              xpass_pswd = xpassaux     && toma el password anterior
72              loop
73            endif
74          case opc_a=3
75            msg('Digite el Grupo al que pertenece el Usuario')
76            @ 16,32 say space(20)
77            @ 16,32 get xpass_gpo pict '@!' valid xpass_gpo $ 'DSTLJC'
78            read
79            @ 16,32 say grupo[at(xpass_gpo,all_grups)]
80        endcase
81      enddo
82      borrausua()
83      xpass_login = space(08)
84    else
85      error('Ese registro NO se puede modificar por el momento ya que esta ocupado por otro usuario')
86    endif
87  else
88    error('El Login ' + xpass_login + ' no existe en el Registro de Usuarios')
89  endif
90 enddo
91 return
```

```
001 *
002 * Sisteaa.: Sisteaa de Telereservaciones para Lineas de Autotransporte
003 * Titulo..: Programaa para desplegar una ventana de ayuda a los catalogos
004 * Nosbre..: HELP.PRG
005 *
006 parameters prog,num_lin,nom_var
007 if prog $ 'HELP,AYUDA1'
008   return
009 endif
010 if nom_var='TIPO'
011   subind=1
012   ti=0
> y pulse DY para elegir')
014   caja_tipos=savescreen(15,56,23,67)
015   set color to i,w+/i
016   @ 15,56,23,67 box 'ZD?3YD03 '
017   for ti=1 to 7
018     @ 15+ti,57 prompt tipos[ti]
019   next ti
020   menu to subind
021   set color to
022   if lastkey()#27
023     tipo=substr(all_tips,subind,1)
024     clear gets
025   endif
026   restscreen(15,56,23,67,caja_tipos)
027   return
028 endif
029 if nom_var='XPASS_GPO'
030   ti=0
031   caja_tipos=savescreen(15,56,22,73)
032   set color to i,w+/i
033   @ 15,56,22,73 box 'ZD?3YD03 '
034   for ti=1 to 6
035     @ 15+ti,57 prompt grupo[ti]
036   next ti
037   menu to subind
038   set color to
039   if lastkey()#27
040     xpass_gpo=substr(all_grups,subind,1)
041     clear gets
042   endif
043   restscreen(15,56,22,73,caja_tipos)
044   return
045 endif
046 if nom_var='XSAL_TIPO'
047   ti=0
048   caja_tipos=savescreen(15,56,19,66)
049   set color to i,w+/i
050   @ 15,56,19,66 box 'ZD?3YD03 '
051   @ 16,57 prompt 'Normal    '
052   @ 17,57 prompt 'Extra     '
053   @ 18,57 prompt 'Cancelada'
054   menu to subind
055   set color to
056   if lastkey()#27
057     xsal_tipo=if(subind=1,'N',if(subind=2,'E','C'))
058     clear gets
059   endif
060   restscreen(15,56,19,66,caja_tipos)
```

```
061   return
062 endif
063 do case
064   case nom_var=='HUEGOL'
065     if revisar()
066       return
067     endif
068     val_ini=&nom_var
069     lvar=trim(&nom_var)
070     prog='erick'
071     do ayudal with 'V'+substr(dtos(fech_vta),3)+dtno+hora+'1 +'V','Asiento',;
072                    'vtas_tmov+substr(dtos(vtas_fesal),3)+vtas_dtno+vtas_hrsal+vtas_impre+vtas_aarc+vtas_taqui+vtas_serie',17,6,;
073                    'vtas_name+" "+transform(vtas_asto,"@B 99")+" "+if(vtas_nucpn='CANCELADO',vtas_dreal+" "+';
074                    'boleto_serie,vtas_dreal)+if(vtas_equip," Equip",space(6))','!!!!!!!!','vtas_taqui+vtas_serie'
075   case nom_var=='AGENCIA'.and.prog=='CAN_CUP'
076     if revisar()
077       return
078     endif
079     val_ini=&nom_var
080     lvar=trim(&nom_var)
081     prog='Erick'
082     do ayudal with 'PP'+substr(dtos(fech_vta),3)+dtno+hora,'Agencia y Cup'n ,;
083                    'vtas_tmov+vtas_aarc+substr(dtos(vtas_fesal),3)+vtas_dtno+vtas_hrsal+vtas_agcia+vtas_nucpn';
084                    ,16,1,'vtas_nucpn+" "+vtas_name+" "+transform(vtas_asto,"@B 99") ,'999','vtas_agcia'
085     if  lastkey()#27
086         agencia  = vtas_agcia
087         cupon    = vtas_nucpn
088         keyboard chr(13)
089     endif
090   case nom_var=='CANCELADOS'
091     if revisar()
092       return
093     endif
094     val_ini=&nom_var
095     lvar=trim(&nom_var)
096     prog='Juan'
097     do ayudal with 'R'+substr(dtos(fech_vta),3)+dtno+hora+'R','Nombre y  Reservo ',;
098                    'R'+substr(dtos(fech_vta),3)+dtno+hora+'R  ; &&+vtas_agcia+vtas_nucpn';
099                    ,16,3,'vtas_name+" "+vtas_nujeff+" "+transform(vtas_asto,"@B 99")','@!'
100   case nom_var=='CUPON'
101     if revisar()
102       return
103     endif
104     val_ini=&nom_var
105     lvar=trim(&nom_var)
106     prog='moises'
107     do ayudal with  PP'+substr(dtos(fech_vta),3)+dtno+hora+ Agencia y Cup'n',;
108                    'vtas_tmov+vtas_aarc+substr(dtos(vtas_fesal),3)+vtas_dtno+vtas_hrsal+vtas_agcia+vtas_nucpn',;
109                    17,6,'vtas_name+" "+transform(vtas_asto,"@B 99")+" "+vtas_dreal',;
110                    '!!!!!!!!','vtas_nucpn'
111   case nom_var=='DTNO'.and.prog$ ;
      'VENTAS,CANCELA,RESERVA,DESBLOQA,CAMCPESV,GUIA,REESPBOL,CAMSCUPO,CAPTCAMI,EQUIPAJE,CAMCEQUI,VEN_CUP,CAN_CUP'
112     if revisar()
113       return
114     endif
115     esta=.t.
116     nom_var='otra'
117     otra=dtoc(fech_vta)
118     val_ini=&nom_var
119     lvar=''
120     nada='BLANCOS'
```

```
121     blancos=''
122     do ayuda1 with '1A'+substr(dtos(fech_vta),3),'','corr_guia+corr_edo+substr(dtos(corr_fech),3)+corr_dtno+corr_hora';
123                   ,1,6,"corr_dtno+' '+transform(corr_hora,'@R 99:99')",'',nada
124     if lastkey()#27
125       hora=corr_hora
126       dtno=corr_dtno
127       fech_vta=corr_fech
128       keyboard chr(13)
129     endif
130   case nom_var=='DTNO'.and.(prog=='GUIAEQUI'.or.prog=='REEIGUIA')
131     if revisar()
132       return
133     endif
134     esta=.t.
135     nom_var='otra'
136     otra=dtoc(fech_vta)
137     val_ini=&nom_var
138     lvar=''
139     nada='BLANCOS'
140     blancos=''
141     do ayuda1 with '2A'+substr(dtos(fech_vta),3),'','corr_guia+corr_edo+substr(dtos(corr_fech),3)+corr_dtno+corr_hora';
142                   ,1,6,"corr_dtno+' '+transform(corr_hora,'@R 99:99')",'',nada
143     if lastkey()#27
144       hora=corr_hora
145       dtno=corr_dtno
146       fech_vta=corr_fech
147       keyboard chr(13)
148     endif
149   case nom_var=='HORA'.and.prog$
      'VENTAS,CANCELA,RESERVA,DESBLOQA,CANCRESV,GUIA,REEXPBOL,CAMBCLPO,CAPTCAMI,EQUIPAJE,CANCEQUI,VEN_CUP,CAN_CUP'
150     if revisar()
151       return
152     endif
153     val_ini=&nom_var
154     esta=.t.
155     do ayuda1 with '1A'+substr(dtos(fech_vta),3)+dtno,'','corr_guia+corr_edo+substr(dtos(corr_fech),3)+corr_dtno+corr_hora';
156                   ,1,6,"corr_dtno+' '+transform(corr_hora,'@R 99:99')",'',''
157     if lastkey()#27
158       hora=corr_hora
159     endif
160   case nom_var=='HORA'.and.(prog=='GUIAEQUI'.or.prog=='REEIGUIA')
161     if revisar()
162       return
163     endif
164     val_ini=&nom_var
165     esta=.t.
166     do ayuda1 with '2A'+substr(dtos(fech_vta),3)+dtno,'','corr_guia+corr_edo+substr(dtos(corr_fech),3)+corr_dtno+corr_hora';
167                   ,1,6,"corr_dtno+' '+transform(corr_hora,'@R 99:99')",'',''
168     if lastkey()#27
169       hora=corr_hora
170     endif
171   case nom_var$'IPOB_CVE,DTNO,CVE_PARAD'
172     if revisar()
173       return
174     endif
175     val_ini=&nom_var
176     lvar=trim(&nom_var)
177     do ayuda1 with &nom_var,'Poblaci*n','POB_CVE',1,3,"POB_DES",;
178                    '!!!','POB_CVE'
179   case nom_var$'IRGL_CLAVE'
180     if revisar()
```

```
    return
  endif
  val_ini=&nom_var
  lvar=trim(&nom_var)
  do ayudal with &nom_var,'Clave Roll', ROL_CLAVE',1,2,'ROL_NOMBRE',;
                 '!!','ROL_CLAVE'
  if  lastkey()#27
      xrol_clave = rol_clave
      keyboard chr(13)
  endif
case nom_var$'IFAC_CVE'
  if revisar()
    return
  endif
  val_ini=&nom_var
  lvar=trim(&nom_var)
  do ayudal with &nom_var,'Ingo. Fact','FAC_CVE',1,1,;
  "Prim="+transform(FAC_IPA,'@z 999.99')+' '+transform(FAC_FC1,'')+' '+transform(FAC_IMJ,'@z 999.99')+' Exp='+";
  "transform(FAC_EXP,'@z 999.99')+' '+transform(FAC_FCE,'')+' '+transform(FAC_EMJ,'@z 999.99')",':','FAC_CVE'
case nom_var$'IAGEN_CVE,V_AGENCIA'
  if revisar()
    return
  endif
  val_ini=&nom_var
  lvar=trim(&nom_var)
  prog='acises'
  do ayudal with &nom_var,'Agencia','AGEN_CVE',1,3,'AGEN_DES+' '+TRABSFORM(AGEN_COM,'99')+' %',;
                 ':::','AGEN_CVE'
case nom_var=='FECH_EXTS .and.prog=='OPENEXTS'
  if revisar()
    return
  endif
  nom_var='FEEXT'
  &nom_var=dtoc(fech_exts)
  val_ini=&nom_var
  lvar=''
  nada='Blancos'
  blancos=''
  do ayudal with 'EC','Corrida','corr_ticor+corr_edo+substr(dtos(corr_fech),3)+corr_hora+corr_dtno',3,6,;
                 "substr(dtoc(corr_fech),1,2)+'-'+substr(emslaomth(corr_fech),1,3)+'-'+substr(dtoc(corr_fech),7)+' '+";
                 "transform(corr_hora,'@R 99:99')+' '+corr_dtno+' '+boctpcb -> pcb_des+' '+str(corr_cupo,2,0)",'',nada
  if lastkey()#27
    hora=corr_hora
    dtno=corr_dtno
    fech_exts=corr_fech
    keyboard chr(13)+chr(13)
  endif
case nom_var=='HORA',and.prog$'BOL-TAQ,BOL-LIQ,BOL-INF,BOL-IMP,BOL-CAT,PRE_RESV'
  if revisar()
    return
  endif
  val_ini=&nom_var
  lvar=trim(&nom_var)
  do ayudal with &nom_var,'Itinerario',;itin_hora+itin_dtno',;
                 1,4,"itin_dtno+' '+str(itin_cupo,2,0)+' '+transform(itin_data,'99/999/999')+' '+";
                 "iff(itin_clase='P','Primera','Expreso')+' '+iff(itin_ticon='N','Normal','Extra ')",'@R 99:99','itin_hora'
  if lastkey()#27
    dtno=itin_dtno
    keyboard chr(13)
  endif
case nom_var=='FECH_VTA'
```

```
241     if revisar()
242       return
243     endif
244     esta=.t.
245     nom_var='otra'
246     otra=dtoc(fech_vta)
247     val_ini=&nom_var
248     lvar=''
249     nada='BLANCOS'
250     blancos=''
251     part_one=iif(prog='GUIAEQUI'.or.prog='FEEXGUIA'),'2','1')
252     do ayudal with part_one+'A'+substr(dtos(fech_vta),3),'','corr_guia+corr_edo+substr(dtos(corr_fech),3)+corr_dtno+corr_hora'
                  ,1,6,"corr_dtno+' '+transform(corr_hora,'@R 99:99')",'',nada
253     if lastkey()#27
254       hora=corr_hora
255       dtno=corr_dtno
256       fech_vta=corr_fech
257       keyboard chr(13)+chr(13)
258     endif
259   case nom_var$'1PER_NUM'
260     if revisar()
261       return
262     endif
263     val_ini=&nom_var
264     lvar=trim(&nom_var)
265     do ayudal with &nom_var,'Pensionario','PER_NUM',1,5,"PER_NOM",'!!!!!','PER_NUM'
266   case nom_var=='ASTO'
267     if revisar()
268       return
269     endif
270     val_ini=&nom_var
271     lvar=trim(&nom_var)
272     do ayudal with taquilla+substr(dtos(fech_vta),3)+dtno+hora+'*';
273                   ,'Asiento','vtas_taqui+substr(dtos(vtas_fesal),3)+vtas_dtno+vtas_hrsal+vtas_impre+str(vtas_asto,2,0)';
274                   ,16,2,'vec_tipos[at(vtas_tipo,todos_tipos)]','99',str(vtas_asto)'
275   case nom_var$'CVE_USU'
276     if revisar()
277       return
278     endif
279     val_ini=&nom_var
280     lvar=trim(&nom_var)
281     param = trim('1'+val_ini)
282     do ayudal with descend(param),'Despachador',"descendiif(empty(desp_hocte),'1','2')+DESP_LOGIN)",2,8,;
283        "transform(desp_feent,'@E')+' '+transform(desp_hrini,'@R 99:99')+' --> '+transform(desp_fesal,'@E')+' '+";
284        "transform(desp_hrsal,'@R 99:99')",'!!!!!!!!!','DESP_LOGIN'
285     if lastkey()#27
286       cve_usu = desp_login
287     endif
288   case nom_var$'CVE_AUX'
289     if revisar()
290       return
291     endif
292     val_ini=&nom_var
293     lvar=trim(&nom_var)
294     param = trim('2'+val_ini)
295     do ayudal with descend(param),'Despachador',"descendiif(empty(desp_hocte),'1','2')+DESP_LOGIN)",2,8,;
296        "transform(desp_feent,'@E')+' '+transform(desp_hrini,'@R 99:99')+' --> '+transform(desp_fesal,'@E')+' '+";
297        "transform(desp_hrsal,'@R 99:99')",'!!!!!!!!!','DESP_LOGIN'
298     if lastkey()#27
299      cve_aux = desp_login
300       entra  = .t.
```

```
301     endif
302   case nom_var=='PASS_LOC1'
303     if revisar()
304       return
305     endif
306     val_ini=&nom_var
307     lvar=trim(&nom_var)
308     do ayuda1 with xpass_login,'Usuario','pass_login',1,8,'pass_nom*'+'pass_opo','99999999','pass_login'
309 endcase
310 return
311 *
312 * Funcion: REVISAR
313 * Revisa si hay registros en el archivo.
314 *
315 Function Revisar
316 go top
317 if eof()
318   error('El Archivo est  vac/o !)
319   return .t.
320 endif
321 return .f.
```

Las siguientes líneas preguntan por el tercer parámetro que es el nombre de la variable que se estaba capturando para identificar qué variable era ésta. En la línea 302 se pregunta si la variable es XPASS_LOGIN, que es la variable que se estaba capturando en la línea 20 del programa BOCTUS03.PRG, de tal forma que esta condición se cumple y entonces se ejecutan las líneas de la 303 a la 308 del programa HELP.PRG.

La línea 303 pregunta por una condición que lleva a ejecutar la función REVISAR, la cual se encuentra codificada de la línea 315 a la 321 y cuyo objetivo es revisar si el archivo seleccionado (que es precisamente el catálogo de usuarios) tiene por lo menos un registro.

Después de verificar si hay registros en el archivo continúa ejecutando en la línea 306 en donde se salva temporalmente el valor de la variable que se estaba capturando, así como también se obtiene la longitud de la misma en la línea 307.

En la línea 308 se hace una llamada al programa AYUDA1.PRG, al cual se le pasan como parámetros los datos con los cuales trabajará la ventana de ayuda, los cuales son:

1. La variable que se estaba capturando y en la cual deberá quedar el valor que indiquemos en la ventana de ayuda.

2. La cadena que indicará lo que debemos seleccionar, en este caso es ''Usuario'', ya que lo que debemos seleccionar en la ventana de ayuda es precisamente un usuario de los que se desplegarán.

3. El nombre del campo llave del archivo de datos, en este caso es PASS_LOGIN.

4. La posición a partir de la cual se encuentra el campo llave.

5. El número de carácteres que componen a la llave, en éste caso 8 porque ésta es la longitud del campo PASS_LOGIN que es llave.

6. La cadena que tiene la expresión a desplegar como datos adicionales en la ventana de ayuda por cada uno de los registros del archivo de datos, en este caso la expresión consiste de el nombre del usuario y el grupo de usuarios al que pertenece: PASS_NOM y PASS_GPO respectivamente.

7. La máscara que se deberá aplicar al campo llave para armar el picture que indica el formato de despliegue.

8. Y finalmente la expresión a desplegar como campo llave en la ventana de ayuda.

En la siguiente página se muestra un listado del programa AYUDA1.PRG, el cual recibe los parámetros que explicamos anteriormente mediante la instrucción PARAMETERS en la línea 06, la línea 07 declara un arreglo que se utiliza para cargar los renglones que se desplegarán en la ventana de ayuda, y en la línea 08 se declara un apuntador al arreglo.

De la línea 09 a la 18 se tiene la lógica que revisa si existen registros en el archivo de datos con la característica que se haya especificado en la variable que se estaba capturando, es decir, el funcionamiento de esta ayuda es de la siguiente manera: si el usuario pulsa F1 teniendo en blanco la variable XPASS_LOGIN, el programa proporciona ayuda de todo el archivo de usuarios, en caso de que el usuario digite en la variable XPASS_LOGIN algunos carácteres, se tomará como la característica a buscar dentro del catálogo de usuarios y se desplegarán aquellos registros en los que el campo PASS_LOGIN empiecen con lo que contiene la variable de memoria XPASS_LOGIN.

En la línea 19 se asigna el número de registro a partir del cual se encuentran los registros que cumplen con la característica en una variable NUREG. En la línea 20 se inicializa la variable OP en 1, la cual será utilizada para dejar en ella el número de opción que se haya elegido en la ventana de ayuda, esto es, cuando se proporciona la ventana de ayuda, se hace proporcionando control sobre los registros del archivo como si cada registro fuese una opción de un menú HIGHLIGHT, proporcionando al usuario una barra de color intenso que puede desplazar por cada uno de los registros mostrados. Si el usuario presiona *enter* en alguno de ellos entonces se toma como la opción elegida y su valor queda precisamente en ésta variable.

Más adelante, en las líneas siguientes se hacen algunos cálculos y se pregunta por el nombre del programa que se estaba ejecutando al momento de presionar F1, en este caso el nombre del programa es BOCTUS03.PRG, por lo que el CASE que se está ejecutando entra en la opción OTHERWISE que está en la línea 36 en donde la línea 37 salva la pantalla hace una llamada a la función PROM1 y después restablece nuevamente la pantalla. Estas tres líneas de programación son las que dan el efecto de que aparezca una ventana encima de los datos que había en pantalla, así como da la sensación de que al salir de la ventana, hemos quitado esa ventana de ayuda de encima de nuestros datos de pantalla.

Ahora analicemos la función PROM1 que se encuentra en el listado del programa SC1.PRG:

La función PROM1 se encuentra en las líneas de la 084 a la 263 y funciona de la siguiente manera.

En las líneas 088 a la 095 se prepara el recuadro que define a la ventana de ayuda, así como también se despliegan algunos

```
01 *
02 * Sistema..: Sistema de Telereservaciones para Lineas de Autotransporte
03 * Titulo...: Programa de Ayuda de catalogos en una ventana
04 * Nombre...: AYUDA1.PRG
05 *
06 parameters cve,msg_top,campo,po_ini,cuantos,des,mascara,campo2
07 declare sed[18]
08 p=1
09 if len(trim(cve))=0
10   go top
11 else
12   seek trim(cve)
13   if eof()
14     error('No existe ningún registro con esas caracteristicas')
15     &nom_var=val_ini
16     return
17   endif
18 endif
19 nuregs=recno()
20 op=1
21 set message to
22 la=len(mascara)
23 posspra=(' ',mascara)
24 lcasc=if(posspra>0,la,la-posspra)
25 area=' '
26 do case
27   case (prog$'GUIA').and.(nom_var='APOL_CLAVE')
28     save screen to pantalla
29     op=proc1(sed)
30     restore screen from pantalla
31   case prog$;
32   'VENTAS,CANCELA,RESERVA,DESBLOQA,CANCRESV,GUIA,REEXPBOL,CAMBCUPO,CAPTCAMI,EQUIPAJE,GUIAEQUI,REEYGUIA,CANCEQUI,VEN_CUP,CAN_CUP'
33     area=savescreen(1,59,12,79)
34     op=ch_corr(sed)
35     restscreen(1,59,12,79,area)
36   otherwise
37     save screen to pantalla
38     op=proc1(sed)
39     restore screen from pantalla
40 endcase
41 do case
42   case op=0                          && no hacer nada
43     return
44   otherwise                          && Elegir una opcion
45     skip op-c1-1
46     &nom_var=substr(&campo,po_ini,cuantos)
47     clear gets
48     return
49 endcase
50 return
```

```
001 *
002 * Sistema..: Sistema de Telereservaciones para Lineas de Autotransporte
003 * Titulo..: Procedimientos auxiliares de manipulacion de video para las ayudas
004 * Nombre..: SCI.PRG
005 *
006 procedure mover
007 DO WHILE .t.
008   DO WHILE .t.
009     y=ltrim(str(inkey()))
010     if  '"+y+"' $ #1#,#4#,#5#,#6#,#19#,#23#,#24#,#27#,#29#
011       exit
012     endif
013   ENDDO
014   if y='27'
015     exit
016   endif
017   DO CASE
018     case y=='1'                    && Home.
019       part1()
020       c2=c2-c1
021       c1=0
022       part2()
023     case y=='6'                    && End.
024       part1()
025       desp=79-c2
026       c1=c1+desp
027       c2=79
028       part2()
029     case y=='29'                   && Ctrl-Home.
030       part1()
031       r2=r2-r1
032       r1=0
033       part2()
034     case y=='23'                   && Ctrl-End.
035       part1()
036       desp=24-r2
037       r1=r1+desp
038       r2=24
039       part2()
040     CASE Y=='24'                   && Mover Hacia abajo          (Flecha abajo)
041       IF R2<24
042         part1()
043         r1=r1+1
044         r2=r2+1
045         part2()
046       ENDIF
047     CASE Y=='19'                   && Mover Hacia la izq.        (flecha Izq.)
048       IF c1>0
049         part1()
050         c1=c1-1
051         c2=c2-1
052         part2()
053       ENDIF
054     CASE Y=='4'                   && Mover Hacia la Derecha      (Flecha der.)
055       IF c2<79
056         part1()
057         c1=c1+1
058         c2=c2+1
059         part2()
060       ENDIF
```

```
061     CASE Y=='5'                 && Mover Hacia Arriba           <Flecha arriba>
062       IF R1>0
063         part1()
064         r1=r1-1
065         r2=r2-1
066         part2()
067       ENDIF
068     ENDCASE
069   ENDDO
070   return
071   *
072   function part1
073   panta=savescreen(r1,c1,r2,c2)
074   restscreen(r1,c1,r2,c2,first)
075   return .t.
076   *
077   function part2
078   first=savescreen(r1,c1,r2,c2)
079   restscreen(r1,c1,r2,c2,panta)
080   return .t.
081   *
082   *
083   *
084   function proal
085   parameters arreglo
086   set cursor off
087   first=savescreen(R1,C1,R2,C2)
088   @ r1,c1,r2,c2 box '22222222 '
089   set color to i+
090   @ r1+1,c2,r2-1,c2 box '3'
091   @ r1,c1+1 say 'Elegiri'+msg_top
y DY'
093   @ r2,c1+23 say '<PgDn> Avanzar '

095   @ r1,c1+45 say '<Esc> Salir'
096   set color to
097   p=1
098   movto =.f.
099   trimcve=trim(cve)
100   lentrim=len(trimcve)
101   do while .t.
102     if ! movto
103       ci=0
104       bini=.f.
105       do while (! eof()) .and. trimcve=substr(&campo,1,lentrim).and. ci<rx(+ry
106         if ! bini
107           bini=.t.
108           regini=recno()
109         endif
110         valor_des=&des
111         len_valor_des=len(valor_des)
112         valor_campo2=&campo2
113         npos=if(cy-cx-1masc-2)len_valor_des,len_valor_des,cy-cx-1masc-2)
114         texto=transform(valor_campo2,mascara)+if(empty(valor_campo2),'',space(1))+substr(valor_des,1,npos)+
115               space(cy-cx-1masc-npos-2)+if(empty(valor_campo2),space(1),'')
116         ren_glon=r1+ci+1
117         set color to i+
118         @ ren_glon,c1 say chr(65+ci)
119         set color to
120         @ ren_glon,c1+1 say texto
```

```
121       ci=ci+1
122       sed[ci]=texto
123       skip
124     enddo
125     end=ci
126     if ci+r1(r2-1
127       if p)ci
128         p=end
129       endif
130       @ ci+r1+1,c1+1,r2-1,c2-1 box ''
131       @ ci+r1+1,c1,r2-1,c1 box '2'
132     endif
133   else
134     movto=.f.
135   endif
136   nur=recno()
137   set color to u+
138   @ r1+p,c1+1 say arreglo[p]
139   set color to
140   go regini
141   skip-1
142   set color to i+
143   if (substr(kcampo,1,lentrim)(trimcve.and. ! empty(cve)).or. bof()
144     @ r1+1,c2 say '3'
145   else

147   endif
148   go nur
149   if substr(kcampo,1,lentrim)>trimcve.or.eof()
150     @ r2-1,c2 say '3'
151   else
152     @ r2-1,c2 say ''
153   endif
154   set color to
155   do while .t.
156     inkey(0)
157     ti=lastkey()
158     t=asc(upper(chr(ti)))
159     if ltrim(str(ti))$'1,3,4,5,6,13,18,19,24,27,306'.or.(t)=65.and.t(=64+c1).or.ti=306
160       if ti#306
161         @ r1+p,c1+1 say arreglo[p]
162       endif
163       exit
164     endif
165   enddo
166   do case
167     case t)=65.and.t(=64+c1    && +asc(vec_opic1])
168       p=t-64
169       movto=.t.
170     case t=1                              && ir al tope          (flecha arriba)
171       p=1
172       movto=.t.
173     case t=6                              && ir al fin           (flecha abajo )
174       p=end
175       movto=.t.
176     case t=19.or.t=5                      && ir hacia arriba     (flecha der. o flecha arriba)
177       if p=1
178         if nureq#regini
179           go regini
180           skip -1
```

```
181         if ! (substr(&campo,1,lentrim(tricave)
182           skroll(ri+1,ci+1,rf-2,cf-1,1,tipovideo)
183           valor_campo2=&campo2
184           xyz=transform(valor_campo2,mascara)+if(empty(valor_campo2),'',space(1))+substr(&des,1,npos)+
185               space(cy-cx-lmasc-npos-2)+if(empty(valor_campo2),space(1),'')
186           @ ri+1,ci+1 say xyz
187           regini=recno()
188           if end<ry-rx+1
189             end=end+1
190             ci=ci+1
191             go nur
192             set color to i+
193             @ ri+ci,ci say chr(64+ci)
194             set color to
195           else
196             go nur
197             skip -1
198           endif
199           aims(sed,1)
200           sed[1]=xyz
201           movto=.t.
202         endif
203       else
204         p=end
205         movto=.t.
206       endif
207     else
208       p=p-1
209       movto=.t.
210     endif
211   case t=24.or.t=4                        && ir hacia abajo     (flecha abajo o flecha izq.)
212     if p=end
213       if p=ry-rx+1
214         if substr(&campo,1,lentrim))tricave.or.eof()
215           p=1
216           movto=.t.
217         else
218           skroll(ri+2,ci+1,rf-1,cf-1,-1,tipovideo)
219           valor_campo2=&campo2
220           texto=transform(valor_campo2,mascara)+if(empty(valor_campo2),'',space(1))+substr(&des,1,npos)+
221                space(cy-cx-lmasc-npos-2)+if(empty(valor_campo2),space(1),'')
222           @ rf-1,ci+1 say texto
223           adel(sed,1)
224           sed[18]=texto
225           go regini
226           skip
227           regini=recno()
228           go nur
229           skip
230           movto=.t.
231         endif
232       else
233         p=1                             &&& linea nueva
234         movto=.t.
235       endif
236     else
237       p=p+1
238       movto=.t.
239     endif
240   case t=27                               && no elegir nada    ( Esc )
```

```
241       set cursor on
242       return 0
243     case ti=306                          && empezar movto.     ( Alt-M )
244       do mover
245       movto=.t.
246     case t=13                            && eleccion           ( Enter )
247       set cursor on
248       return p
249     case t=18                            && pantalla anterior  ( PgUp  )
250       if regini=nureg
251         movto=.t.
252       else
253         skip rx-ry-ci-1
254         if substr(&campo,1,lentrim)<trimcve
255           go nureg
256         endif
257       endif
258     case t=3                             && pantalla siguiente (PgDn)
259       if eof().or.(trimcve>substr(&campo,1,lentrim))
260         movto=.t.
261       endif
262   endcase
263 enddo
```

mensajes informativos que se encuentran contenidos en el mismo contorno de la ventana.

En la línea 101 se tiene una estructura WHILE que se cierra en la línea 263. Este ciclo principal es el que controla o permite que podamos movernos en la ventana de ayuda por medio de las teclas de control PG UP (Pantalla arriba), PG DWN (Pantalla abajo), etc.

En la línea 105 se tiene otra estructura WHILE que cierra en la línea 124 y es la encargada de hacer el despliegue en la ventana de ayuda de la sección de archivo que se puede mostrar en los 18 renglones posibles de la ventana de ayuda. También tiene la función de ir generando en una variable de memoria llamada TEXTO, la cadena que se desplegará como un renglón de la ventana de ayuda por cada registro del archivo de datos, esto se hace en la línea 114.

En la línea 155 se tiene nuevamente otra estructura WHILE que cierra en la línea 165 y es la encargada de capturar una tecla de control así como de identificar si se presiona una letra de selección.

En la línea 166 se tiene la estructura CASE, encargada de identificar la tecla que se haya pulsado para ejecutar la acción correspondiente. En caso de presionarse una letra, se recalcula en base al código ASCII su correspondiente posición en la ventana de ayuda, esto se hace en la línea 168.

Más adelante, de manera similar se pregunta por las demás teclas de control de la ventana como son: Flecha hacia arriba, flecha hacia abajo, flecha hacia la derecha, flecha hacia la izquierda, tecla de escape, teclas ALT-M para activar movimiento de la ventana, pantalla anterior y pantalla siguiente.

Es importante destacar que se implementó la función ALT-M en la ventana de ayuda, la cual permite mover de lugar la ventana en caso de que estorbe en la posición de *default* la visualización de datos. También se tiene implementado todo el control del menú HIGHLIGTH en donde se lee alguna tecla de control, se identifica y se ejecuta la petición correspondiente como pudiera ser el desplazarse por cada uno de los registros mostrados en la ventana de ayuda.

4.2. PRUEBAS

1. Pruebas funcionales
2. Pruebas de implantación
3. Pruebas de sistemas
4. Pruebas de aceptación

Pruebas funcionales

Las pruebas funcionales tienen como objetivo encontrar errores de análisis, de especificación y de diseño.

Este tipo de pruebas se realizarón minuciosamente en cada una de las subrutinas y funciones utilizadas en el sistema, así como en cada uno de los programas realizados.

Conforme se iban desarrollando las utilerías del sistema se probaban en forma independiente hasta llegar a tener todo el conjunto de utilerías necesarias para el sistema. Los módulos que conforman el sistema de Telereservaciones se integraron con cada uno de los programas desarrollados y se probaban conforme a esa integración.

Una vez terminado el módulo se realizaban pruebas de integración de la información para cada uno de ellos y así cada uno de los módulos. Cuando ya se realizaron todos los módulos se procedió a una simulación de entrada y salida de datos con el personal del centro de cómputo, como usuarios del sistema, y con ello conocer los posibles errores que pudiera tener el sistema. Finalmente se efectuaron las correcciones necesarias, y el sistema quedo libre de estos errores.

Pruebas de Implantación

Tiene como objetivo encontrar errores de especificación, concepto y diseño, en el ambiente real (equipo, sistema operativo, interfaces, etc.) en donde los programas serán finalmente instalados.

Una de las primeras pruebas de implantación fue el número de ***buffers*** y archivos, ya que estos influyen en la respuesta del sistema, llegando a la conclusión de que el número de ***buffers*** ideal para el sistema en particular es de 15 con un número de archivos abiertos de 60, por cada estación de trabajo.

Con respecto al ***hardware*** se realizarón pruebas en cada una de las redes instaladas como son: chequeo de cada una de las estaciones de trabajo conectadas a ellas, ya que de esta forma se prueba cada una de las tarjetas de comunicación, checar el buen funcionamiento de las unidades de energía ininterrumpible, cortando la corriente eléctrica suministrada y con ello lograr conocer el tiempo máximo de duración de las baterías de las UPS, para tomar las medidas adecuadas mientras dure la carga de las baterías.

Durante las pruebas de implantación, del sistema de telereservaciones en la red, surgieron detalles como duplicidad de boletos. Mediante estudios del sistema se vió la posibilidad de que existieran errores de programación, sin embargo, una vez checados los algoritmos, no se encontraron errores en éstos, por lo que se llegó a concluir que el problema se debía al manejo de memoria del sistema operativo Netware y con tal motivo se tuvieron que agregar algunos campos en la base de datos utilizada, para tener controlados los accesos a los registros y con ello evitar el problema de duplicidad de boletos.

Otra prueba realizada fue utilizando el equipo VAX 3100 y su *software* (PCSA), utilizado como servidor. Primeramente se instaló el sistema en dicho equipo y empezó a funcionar con tres estaciones de trabajo. Los resultados obtenidos, después de realizar las modificaciones necesarias en los parámetros que utiliza este equipo son:

- Una mejor respuesta del sistema.
- Realiza respaldos del sistema, por ejemplo cada 5 minutos, lo que no se puede realizar con Novell.
- Tiene comunicaciones remotas sin compartir con un equipo adicional estas funciones como en Novell. Cabe aclarar que también Novell permite usar el servidor de archivos para realizar las comunicaciones remotas, sólo que éste se degrada en forma considerable, lo cual no es conveniente en nuestro sistema.

Con las pruebas realizadas con el equipo VAX, se llegó a la conclusión que es uno de los mejores equipo para ser utilizado en este sistema. Sin embargo, debido al presupuesto de la empresa no se pudieron realizar las implantaciones de estos equipos, por lo que se tuvo que seguir con Novell.

Pruebas del sistema

Las pruebas del sistema no consisten en volver a probar todas las funciones o programas de un sistema, sino probar mediante un juicio la compatibilidad del sistema con la documentación del usuario.

Para realizar estas pruebas nos auxiliamos del departamento de auditoría de la empresa, en donde fue instalado el sistema, para chechar cada uno de los reportes emitidos por el sistema y con ello comprobar el buen funcionamiento del mismo. Para más tarde realizar la implantación.

Pruebas de Aceptación

Las pruebas de aceptación consistieron en la revisión minusiosa por parte del usuario del sistema, para ver si realmente era agradable a éste, y si cumplía con todos sus requerimientos planteados al inicio del desarrollo del mismo.

4.3 IMPLANTACION

Durante esta etapa se procedió de la siguiente manera:

El primer paso fue la instalación de una red provisional en las instalaciones de la empresa en donde se implantó el sistema de telereservaciones. Lo anterior se realizó con el objeto de dar capacitación a las personas que manejan el sistema, dicha capacitación consistió del siguiente temario:

1. Configuración de una computadora

   - Definición de una Computadora
   - Hardware
   - Unidad Central de Proceso (CPU)
   - Unidad de Memoria
   - Dispositivos de Entrada y de Salida
   - Clasificación de Memorias
   - Software
   - Aplicaciones de la Computadora
   - Recomendaciones sobre el cuidado del equipo
   - Definición de un sistema de información
   - Definición de un usuario de un sistema de información
   - Definición de un Ingeniero de Software
   - Diferencias Máquina-Hombre

2. Conceptos Básicos

   - Catálogos, archivos
   - Boleto
   - Guía
   - Cupón
   - Itinerario, factores

3. Sistema de boletos

   - Partes del sistema de boletos
   - Tipos de usuarios del sistema de boletos

4. Práctica común

   - Venta de boletos normales
   - Cancelación de boletos normales
   - Venta de cupones (ida-vuelta)
   - Cancelación de cupones
   - Reexpedición de boletos
   - Reservaciones
   - Cancelación de reservaciones
   - Registro de cupones de agencia
   - Cancelación de cupones de agencia
   - Desbloqueo de corridas
   - Emisión de guías
   - Corte por despachador
   - Corte por agencia
   - Generación de días

5. Prácticas de acuerdo a las funciones de cada persona

   - Es decir según las prioridades de cada persona. En este punto se dividieron los usuarios del sistema en grupos de acuerdo a sus fuciones. Después de cumplir 20 hrs. de

práctica con el sistema, se llevo a cabo una práctica conjunto donde se simula el funcionamiento real de la central y con ello concluye el curso de capacitación.

El segundo paso fue la instalación de la red en el lugar donde operará el sistema. Una vez terminada la instalación se procedió a las pruebas respectivas de comunicaciones entre las estaciones de trabajo, para la verificación de su buen funcionamiento.

El tercer paso incluye la instalación del sistema de telereservaciones, la introducción de los datos iniciales que necesita para operar y la última prueba de preoperación del sistema.

El cuarto paso consistió en la liberación del sistema y el seguimiento del mismo durante una semana de operaciones.

Los pasos del uno al cuarto fueron realizados en cada una de las ciudades en donde se instaló el sistema de telereservaciones, las cuales fueron México, Guadalajara y Tijuana.

El quinto paso consitió en la interconexión de dichas ciudades, através de modems y lineas conmutadas, para lograr de esta forma la la comunicación de dichas ciudades y con ello la venta de boletos de viaje redondos.

CONCLUSIONES

CONCLUSIONES

Para determinar las aportaciones del sistema de TELERESERVACIONES y sus posibles fallas se tomó como referencia el medio ambiente en el cual opera y el conocimiento de sistemas del mismo tipo.

APORTACIONES DEL SISTEMA

La principal aportación de este trabajo es el satisfacer las necesidades para el cual fue creado. Ya que con este sistema se cuenta con un medio con el cual se lleva un buen control de los ingresos que obtiene la empresa en donde se implantó el sistema, por la venta de boletos, así como la gran ventaja de poder brindar al público la información en forma oportuna y precisa y la obtención de reportes necesarios para la toma de decisiones.

El *software* desarollado fue creado pensando en las personas que lo utilizarían, así como el público y en la importancia de la información para los ejecutivos de la empresa. Y para asegurar la aceptación de nuestros usuarios, la interfaz usuario-sistema fue diseñada tratando de aplicar lo que se ha dado en llamar ''Factores Humanos''.

El desarollo del sistema de TELERESERVACIONES fue la pauta para crear un gran número de utilerías, las cuales pueden ser utilizadas en el desarrollo de cualquier sistema ya que son de uso general.

LIMITACIONES

Entre las limitaciones tenemos:

- Sólo contempla el viaje redondo asegurado en las ciudades más importantes.
- Para realizar algunos procesos, como son respaldo de información y reconstrucción de índices,se tienen que parar las actividades, es decir, no es concurrente.
- Sólo puede ser transportable a equipos que simulenelambiente DOS.
- Cuando existe la falla de una terminal se tieneque realizar la reconstrucción de índices.

Una posible limitación del sistema es el no estar desarrollado en un lenguaje altamente transportable.

Enlo querespectaal*hardware* no setieneunsistema redundante (por falta de presupuesto), lo que ocasiona pérdidas en los ingresos de la empresa (TEO) cuando el *hardware* en sus componentes principales falla.

POSIBLES MEJORAS AL SISTEMA

a) Dentro de este punto podemos mencionar la posibilidad de desarrollar todas las rutinas de uso general en lenguaje ensamblador, así como todo el manejo de video utilizado en el sistema.

b) La transportabilidad de nuestro sistema se limita a equipos que manejen el ambiente DOS, ya que está desarrollado en Clipper, C y Pascal, esto hace que el sistema no sigue la tendencia del *software* totalmente atado a un solo equipo de una sola marca. Por otra parte el tener un sistema con elementos estandar, trae consigo la ineficiencia en el uso de las bondades del equipo de cómputo donde reside. Sin embargo, se puede desarrollar en un lenguaje que sea altamente transportable como lo es Pascal, C ó utilizando una base de datos como lo es *oracle*, *informix*, ya que permiten con un mínimo de cambios ser utilizados en diversos equipos ya sean personales o multiusuarios.

c) El sistema puede ser más eficiente si se utiliza un sistema de comunicaciones mejor que las líneas privadas, esto puede lograrse por medio de un servicio especial para este tipo de aplicaciones como lo son la red digital de teléfonos de México, Telepac o un servicio por medio de satélites,con lo que se lograría una respuesta del sistema muy aceptables cuando se requiere información vía remota.

d) Utilizar *hardware* especial para redes de computadoras y aplicaciones de uso constante, como son discos duros y computadoras fabricadas por *novell*, *ibm*.

e) Incluir en el sistema un reporte que se emita cada una o dos horas en el cual se imprima el estado de cada corrida que saldrá durante las 24 horas siguientes, esto es para poder tener información a la mano en caso de alguna falla del sistema.

R E C O M E N D A C I O N E S

En el caso específico del lugar donde opera el sistema de TELERESERVACIONES se debe tener un sistema redundante hasta donde lo permita el *software* de la red, ya que todo el *hardware* no está libre de posibles fallas. Si se cuenta con esto se pueden evitar pérdidas en la información, así como parar la venta de boletos en períodos prolongados de tiempo y con ello lograr que no se traduzca en pérdidas económicas para la empresa.

Se puede lograr con el sistema tener conectadas términales en las agencias de viaje para ofrecer un mejor servicio al público y lograr una mejor imagen de la empresa.

La utilización de un servicio especializado en la comunicación de la información es ideal para esta aplicación ya que de ello se puede tener un servicio al usuario más rápido y eficiente.

Es importante tener en cuenta que el sistema se utiliza los 365 días del año y las 24 horas del día y por tal motivo es recomendable utilizar equipos de computo especiales para tener el mínimo de fallas de *hardware*.

BLIBLIOGRAFIA

- REDES DE COMPUTADORAS. PROTOCOLOS, NORMAS E INTERFASES
  Uyless Black
  Edit. Macrobit 1987

- Publicación PC/TIPS Noviembre de 1991
  Redes. La interconectividad es todo un protocolo, Estrategias para desarrollar un proyecto de red institucional.

- Publicación BYTE Marzo de 1991
  Editorial McGraw-Hill
  State of The Art. NETWORK MANAGEMENT.

- Cobol Estructurado
  Roberto Phillipakis
  Edit. McGraw-Hill

- Pascal
  Niclaus Wirth
  Edit. McGraw-Hill

- Publicación BYTE Mayo de 1991
  Edit. McGraw-Hill
  CLIPPER 5.01

- Manual de CLIPPER SUMMER 87
  Nantucket Corporation.

- Netware Async Remote Router
  Novell Inc.

- Manuales de Novell Netware SFT /Advanced
  Installation, Administration, etc.

APENDICE A

A. GLOSARIO DE TERMINOS

CORRIDA: Se refiere al recorrido que realiza el autobús, desde su origen al final de su trayectoria incluyendo las paradas intermedias.

CORRIDA EXTRA: Cuando existe una sobredemanda de una corrida y se realiza un nuevo horario temporal, para cubrir dicha sobredemanda se le asigna dicho nombre.

ITINERARIO: Son las salidas diarias que se realizan de un origen a un destino, incluyendo corridas extras.

CUPON: Es el ''Boleto'' que se vende en las agencias de viajes.

CUPON ABIERTO: Es el cupón o boleto el cual puede ser utilizado en cualquier día para viajar.

GUIA: Es la relación de pasajeros que viajan en el autobús.

DOTACION: Es una cierta cantidad de dinero que se les proporciona en cada corrida para sus gastos que puedan tener durante el viaje como son: gastos de casetas, comidas, hospedajes, diesel, etc.

BOLETO: Es el comprobante de viaje del pasajero.

RESERVACION: Es la marca de la plantilla del autobús.

LIQUIDACION: Se refiere a los movimientos tanto de ingresos y egresos de un autobús en una corrida.

DESPACHADOR: Es la persona encargada de proporcionar información y emitir boletos.

SUPERVISOR DE TAQUILLAS: Es la persona responsable de cuidar el buen funcionamiento de las taquillas.

LIQUIDADOR: Se encarga de realizar cortes de caja para cada taquilla, emite la guia de viaje y proporciona las dotaciones.

JEFE DE SERVICIOS: Realiza funciones de asignación de autobuses a cada corrida, así como realizar el registro de corridas extras cuando éstas se requieren.

CONTROLADOR: Es la persona encargada de todas las funciones del sistema, así como el mantenimiento del mismo.

OPERADOR: Es la persona que conduce el autobús.

PERMISIONARIO: Es el dueño del autobús.

FACTOR POR KILOMETRO: Es el precio por kilómetro recorrido.

APENDICE B

B. PSEUDOCODIGOS

INTRODUCCION

Para este punto presentaremos algunos pseudocódigos de algunos de los programas que conforman el Sistema de Telereservaciones:

VENTAS MENORES

```
Verifica la existencia de cupo para menores si no existe
      despliega mensaje de no hay lugar para menores
      fin de ventas de boletos para menores
fin de verifica

Si se desea vender boletos para menores
      verifica el cupo del autobús
Fin de verifica el cupo del autobús

Si no existe cupo en el autobús
      Despliega mensaje de no existe lugar en el autobús
      fin de la rutina de venta de boletos a menores
Fin del no existe cupo en el autobús

Mientras la venta de boletos es menor que la autorizada
      Vende el boleto
      Verifica el número de asientos permitidos para menores
Fin del mientras
```

ASIGNACION A DESPACHADORES

```
Verifica que el despachador no se encuentre en el archivo
      Asigna al despachador
En caso de estar asignado
      Si el despachador no se ha registrado en la hora de salida
      Despliega mensaje de despachador asignado
En caso contrario
      Si el corte no fue hecho
      Indica que el despachador no ha realizado el corte
       En caso contrario
       Asigna al despachador
       Fin de verifica corte
       Fin de verifica despachador
Fin Verifica en el archivo
```

AYUDA PARA CAPTURA POR SELECCION

```
Si existe dato clave
        Ve al inicio del archivo
En caso contrario
        Busca el registro con la clave
        Si no encuentra el registro
           Despliega mensaje de no existe registro con esa clave
           Termina la rutina
        Fin del no encuentra
Fin de no existe dato

En caso que la llamada sea por un programa clasificado
        Salvar las coordenadas especiales
        llama a la rutina para elegir una opción de la ayuda, si se
        desea
        Restablece las coordenadas
Fin del caso
Para el caso de no elegir un dato de la ayuda
        Termina la rutina
En otro caso
        Determina el dato elegido
        Termina la llamada
Fin de las opciones
```

ACTUALIZA FACTORES

```
Mientras se requiera la rutina haz
        Despliega pantalla de factores
        Abre archivos
        Elige un campo de los factores
        Para el caso en que se elija actualizar los factores
        Selecciona el archivo de factores
        Mientras no esté ocupado el registro de factores
           Si no está ocupado el registro
               Reemplaza los valores con los proporcionados
               Termina el mientras
           Fin de ocupado
        Fin del mientras
        Para las opciones de los campos
           Despliega el mesaje del campo requerido
            Pide el dato y validalo
        Fin del caso
Fin del mientras
```

BAJAS AL CATALOGO DE AGENCIAS

```
Selecciona el archivo de agencias
Despliega pantalla de agencias
Mientras se desea dar de baja
        Pide clave de la agencia
        Busca agencia
        Si no encuentra agencia
           Despliega mensaje de no existe agencia con esa clave
        Fin de no encuentra
        Si la agencia no está ocupada por otro usuario
           Despliega los datos de la agencia
           Verifica la baja
           Si la verificación es afirmativa
              Borra la agencia
           Fin de verifica
        En caso contrario
           Manda mensaje de agencia ocupada por otro usuario
        Fin de no está ocupada
Fin del mientras
```

MODIFICACIONES AGENCIAS

```
Selecciona archivo de agencias
Despliega pantalla de agencias
Mientras se desean realizar modificaciones
        Pide clave de la agencia
        Busca agencia
        Si se encuentra registrada la agencia
           Si la agencia no se encuentra ocupada por otro usuario
                Despliega pantalla agencias
                Mientras deseamos cambiar un dato de la agencia
                        Despliega datos de la agencia
                        Pide datos a cambiar
                        En caso de que la opción sea actualizar
                            actualiza el registro de la agencia
                Termina el mientras
                Para el caso de que la  opción  sea  modificar
                        Despliega campo a modificar
                        Pide el dato y validalo
                 Fin del caso
                Fin del mientras
           En caso contrario
                Despleiga mensaje de agencia ocupada por otro usuario
           Fin de agencia ocupada
        En caso contrario
           Despliega mensaje de agencia inexistente
        Fin de agencia registrada

  Fin de modificaciones de agencias
```

EMISION DE CORTES POR AGENCIAS

```
Mientras se desea un corte
        Pide fecha de la cual se desea realizar el corte
        Mientras se desea una agencia
           Pide clave de la agencia
           Verifica si existe la agencia
           Si no existe la agencia
               Despliega mensaje de agencia inexistente
           Fin de no existe la agencia
           Verifica que existan los movimientos de agencias
           verifica que se quieran los cortes
           Si la verificación es afirmativa
               Manda mensajes para preparar la impresora y espera a
               estar lista.
               Mientras existan movimientos de venta de las agencias
                     Imprime datos
               fin del mientras existe movimientos
        Fin del mientras se desea agencia
    Fin del mientras se desea corte
Fin del mientras del corte
```

IMPRESION DE TARIFAS

```
Mientras deseamos impresión de tarifas
        Despliega mensaje de preparar la impresora
        Espera a que la impresora esté lista
        Obten fecha vigente de las tarifas
        lee registro
        Mientras existan registros
           Verifica el contador de líneas si es > 59
               Imprime encabezado
           fin de verifica
           Imprime datos de las tarifas vigentes
           lee siguiente registro
        Fin del mientras existen registros
        Verifica si la impresión es correcta
        Si la impresión es correcta
           termina el ciclo
        En caso contrario
           vuelve a repetir la impresión
        Fin de impresión correcta
Fin del mientras deseamos impresión
```

REINTEGRACION DE ARCHIVOS AL SISTEMA

```
Mientras exista reintegración haz
  Pide fecha de reintegración y valida fecha
  abre archivos
  Desplegar mensaje de reintegración de archivos
  Mientras existan datos
       Desplegar reintegración del archivo de ventas
       Seleccciona archivo de ventas
       Mientras existan datos en archivo de ventas
            Si ( la fecha de ventas = fecha reintegración ) ó
               ( fecha de ventas = fecha reintegración -1 ) ó
               ( fecha de ventas = fecha reintegración +1)
            entonces
                 asigna datos del arch. de ventas his.a variables aux.
                 asigna variables auxiliares al archivo mtr. de ventas
                 Mientras esté ocupado el registro
                      Se puede agregar un registro al arch. mtro. ven.
                           asigna los valores de las var. aux al los
                           campos del archivo de ventas.
                           Termina el ciclo del mientras
                      Fin de verifica si se puede agregar un registro
                 Fin del mientras está ocupado el reg.
                 Borra el registro del archivo histórico de ventas
                 Selecciona el archivo de acumulado mensual por desno.
                 Busca regis. con fecha de venta y el destino dado
                 Reemplaza los campos con los del arch. hist. vtas.
                 Para los diferentes tipos de pasajeros reemplaza el
                 importe adecuado.
                 Selecciona el archivo de acumulado por despachador
                 Busca el registro del despachador
                 Si no existe en el archivo
                     Agrega un registro
                 En caso contrario
                     AdueNate del registro
                 Fin del existe
                 Actualiza los campos del archivo
                 Dependiendo el tipo de pasajero asigna el precio
                 correcto.
            Fin de verifica el rango de la fecha
            lee siguiente registro
       Fin de existen datos en ventas
  fin del mientras existan datos en el archivo de ventas


  Despliega mensaje de reintegración de archivo de corridas
  Abre al archivo histórico de corridas
  Mientras existan datos en el archivo de corridas
       Si la  fecha de corrida esta en el rango de reintegración
            Utiliza variables auxiliares para dejar los campos del
            archivo histórico de corridas.
```

```
          Selecciona el arch. maestro de corridas
          Mientras no se pueda agregar un registro en el arch.
          maestro de corridas.
               Se puede agregar un registro en el arch. de corridas
                    Graba registro en el archivo maestro de corridas
                    Termina el ciclo de mientras no se pueda agregar
               Fin del se puede agregar
          Fin del mientras
          Borra del archivo histórico de corridas el registro
     Fin de valida fecha en el rango
Fin del mientras existen datos en el arch. hist. de corridas

Despliega mensaje de actualización del archivo de equipaje
Abre el archivo histórico de equipaje
Mientras existen datos en el archivo histórico de equipaje
     Si la fecha de salida está en el rango de reintegración
          Asigna a variables auxiliares los datos del registro
          Verifica si existe exceso de peso
          Si existe sobrepeso
               Busca en el archivo de acumulado mensual por destino
               Si no existe  registro
                    Agrega un registro
               En caso contrario
                    Adueñate de él
               Fin de existe registro
               Graba el registro
               Busca en el archivo boletos acumulados por
               despachador
               Si no existe el registro
                    Agrega un registro
               En caso contrario
                    Adueñate del registro
               Fin del no existe
               Actualiza el registro de despachador
               Selecciona el archivo de equipaje
          Fin del exceso
          Mientras no se pueda agregar un registro en equipaje
               Se puede agregar un registro en equipaje
                    Graba registro en equipaje
                    Fin del mientras
               Fin del se puede agregar
          Fin del mientras
          Borra del archivo histórico de equipaje el registro actual
          Lee siguiente registro
Fin del mientras existen datos en el histórico de equipaje

Despliega mensaje de Actualización del archivo de guías
Abre el archivo de guías histórico

Mientras existan datos en el archivo histórico de guías
     Si la fecha está en el rango de reintegración
```

```
                Asigna los valores de los campos a variables auxiliares y
                a los campos del archivo de guías
                Agrega un registro al archivo de guías
            Borra el registro del archivo histórico de guías
            Fin del valida fecha
        lee siguiente registro
        Fin del mientras existen datos en el histórico de guías
        Despliega mensaje del arch. de registro de despachadores
        Abre el archivo histórico despachadores
        Mientras existan datos en el archivo histórico despachadores
            Si la fecha está en el rango de reintegración
                Asigna los valores de los campos a variables auxiliares y
                a los campos del archivo maestro despachadores
                Agrega un registro al archivo despachadores
            Borra el registro del archivo histórico despachadores
            Fin del valida fecha
        lee siguiente registro
        Fin del mientras existen datos en el histórico despachadores

        Cierra archivos

    Fin de mientras se desean actualizaciones
```

CANCELACION DE CUPONES

```
        Abre archivo de boletos de agencia
        Salva pantalla anterior
        Despliega pantalla de cupones de agencia
        Mientras se desea una cancelación
            Pide fecha de salida de la corrida y validad fecha
            Busca en el archivo de corridas con la fecha anterior
            Si no existe ninguna corrida con esa fecha
                Despliega mensaje de no existen corridas con la fecha
                dada.
                repite el ciclo.
            Fin de no existe
            Pide el destino y valida destino
            Pide hora de salida y valida destino
            Pide clave de la agencia que vendio el cupón y valida
            agencia.
            Obten número de cupón a cancelar y valida cupón
            Verifica que no esté cancelado el cupón
            Si es el cupón a cancelar
                Pon marca de cancelación en el registro
                Actualiza el registro de venta de cupones de agencia
            Fin de cupón correcto
            Restablece pantalla anterior
        Fin del mientras se desea una cancelación
        Abre archivo de ventas de boletos
        Fin de la rutina cancelación de cupones de agencia
```

VENTA DE CUPONES

```
Salva pantalla anterior
Despliega pantalla de venta de cupones de agencia
Abre archivo de cupones de agencia
Mientras se desea vender cupones
     Pide fecha de salida del autobús
     Busca registro en el archivo de corridas
     Si no existe la una corrida con esa fecha
          Despliega mensaje de no existe corrida con esa fecha
     Fin de no existe
     Pide hora de salida y valida hora
     Pide destino y valida destino
     Verifica fecha para ver descuentos en caso de existir
     Cuenta descuentos y despliega descuentos disponibles
     Despliega asientos disponibles y exclusivos
     Si no existen lugares para venta
          Despliega mensaje de corrida llena
     Fin del no existen
     Pide destino de viaje
     Valida el destino al que se desea viajar
     Si no existe el destino
          Despliega mensaje destino no registrado
     En caso contrario
          Destino valido
     Fin del no existe
     Calcula precio del viaje según clase y fecha del factor
     Ve tipo de asiento a reservar
     Verifica si el número de asientos marcados no sea mayor a
     los disponibles.
     Pide datos de la agencia
     Si no es valida la agencia
          Termina el ciclo de venta
     Fin de valida
     Pide datos del cupón
     Obten datos del pasajero
     Si confirma venta
          Agrega registro de venta
     Fin confirma
     Para el caso de ser un menor de edad calcula precio del
     viaje.
     Agrega registro como venta de menor de edad
Fin de mientras venta de cupones
restablece pantalla anterior
Fin de venta de cupones
```

PRE-RESERVACIONES A CORRIDAS

```
Despliega pantalla de itinerarios
Mientras Deseamos una pre-reservación
     Obten hora de salida del camión y valida hora
     Obten clave del destino del camión
     Busca destino en corridas
     Si no existe destino
          Despliega mensaje de corrida no existe
          Pide otra vez datos
     Fin no existe
     Busca datos de la corrida en itinerarios
     Si tipo de corrida es extra
          Pide fecha de inicio y terminación de las corridas
          valida numero de días de las corridas extras
     En caso contrario
          Despliega información de días asignados
     Fin de tipo
     Actualiza registro
Fin de pre-reservación
```

GENERACION DE DIAS

```
Abre archivo de itinerarios
Verifica que el archivo de itinerarios no esté vacío
Si está vacío
     Termina generación de días
Fin de vacio
Despliega fecha de la última generación
Mientras se desea generar días
     Pide fecha de inicio de la generación de días y valida fecha
     Pide fecha de terminación y valida fecha
     Verifica que no existan días generados en ese rango
     Confirma generación de días
     Si no se verifica
          Termina el ciclo
     Fin de no se verifica
     Inicio de proceso de generación de días
     Mientras no sea fin de archivo de itinerarios
          Si el tipo de corrida es extra y fecha de
          vigencia es valida
               Borra registro de itinerarios
               Borra registro de paradas
               lee siguiente registro de itinerarios
          Fin si es corrida extra
          Asigna variables auxiliares valores del reg. de
          itinerarios.
          Si es tipo de corrida extra
               Actualiza fecha de vigencia con la de itinerarios
          Fin de tipo de corrida
```

```
        llena paradas según hora y destino
        Mientras la fecha generada < fecha de terminación de
        días generados
             Si es corrida extra
                  Valida fecha en el rango para extras
                       Agregra registro como extra
             En caso contrario
                  Agrega registro como normal
             Fin de corrida extra
             Incrementa el día generado
             lee sigiente registro de itinerarios
        Fin del mientras valida rango de días
    Fin de no existe más registros en itinerarios
    Elimina las corridas extras que nunca se abrieron
Fin de genera días
```

CANCELACIONES

```
Abre archivos
Salva la pantalla anterior
Despliega la pantalla de ventas
Busca usuario
Si es un usuario que no puede cancelar
     regresa de la rutina
Fin del valida usuario
Mientras se desea cancelar
     Pide datos generales de la cancelación y valida datos
     Pide número de boleto a cancelar
     Verifica que no este cancelado
     Verifica que existe el boleto
     Despliega datos del boleto a cancelar
     Obten datos del supervisor que autorizó
     Pide contraseña del supervisor
     Si no es un supervisor valido
          Despliega mensaje de incorrecto
          termina el ciclo
     En caso contrario
          Pide tipo de cancelación ( 50% o 100 % )
          Pon marca de cancelación
          Agrega registro en venta de boletos
          Imprime cancelación
     Fin de supervisor no valido
Fin de mientras se desea cancelar
Restablece la pantalla anterior
Fin de cancelar
```

RESERVACIONES

```
Abre archivo de ventas
Salva la pantalla anterior
Despliega pantalla de ventas
Pide datos de usuario
Si el usuario no es valido
     Regresa de la llamada
Fin del valida usuario
Mientras se desean reservaciones
     Pide fecha de salida del autobús
     Busca corrida con esa fecha
     Si no Existe corrida
          Despliega mensaje
          Termina el ciclo
     Fin del no existe corrida
     Pide datos generales de la corrida y valida datos
     Busca cupo en la corrida
     Si no existe lugar en la corrida
          Despliega mensaje
          termina el ciclo
     Fin del no existe lugar
     Busca lugar para reservaciones
     Si no existen lugares disponibles de reservaciones
          Despliega mensaje
          Termina el ciclo
     Fin de no existen lugares
     Vende el lugar
     Cuenta el número de reservaciones hechas
     Si el número de reservaciones es mayor que el permitido
          Despliega mensaje de no se pueden realizar más
          reservaciones.
          termina el ciclo
     Fin del número de reservaciones
     Pide datos de la persona que realiza la reservación
     Agrega registro de ventas como reservación
Fin de mientras se desean reservaciones
Restablece pantalla anterior
Fin de reservaciones
```

APENDICE C

C. DICCIONARIO DE DATOS

1RA Factor del precio por kilómetro para el servicio de primera clase.

LIM1RA Fecha de vigencia del factor de precios para el servicio de primera clase.

1RA_2 Factor del precio por kilómetro que rige después de la fecha indicada por LIM1RA.

EXP Factor del precio por kilómetro del servicio expreso, sólo está vigente hasta el día indicado por LIMEXP.

LIMEXP Fecha límite de vigencia para el factor de precios indicado por EXP.

DOTA Factor de dotación por kilómetro del que se llevan los autobuses.

LIMDOTA Fecha límite de vigencia del factor de dotación.

DOT_2 Factor que entra en vigencia después de la fecha indicada por LIMDOTA.

RES Límite máximo de reservaciones por corrida.

NORM_SEN Número de descuentos para ancianos en temporada normal.

NORM_EST Número de descuentos a estudiantes en periodo normal.

NORM_PROF Número de descuentos a profesores en periodo normal.

NORM_50 Número de descuentos del 50% de órdenes de servicio.

NORM_ESP Número de descuentos a socios en periodo normal (50%).

NORM_100 Número de boletos con pago completo contratado por una empresa, en el periodo normal.

INI_VACA Fecha de inicio del periodo mas próximo de vacaciones.

FIN_VACA Fecha de terminación del periodo vacacional.

VACA_SEN Número descuentos vigentes para ancianos, en periodo vacacional.

VACA_EST Número de descuentos a estudiantes en periodos vacacionales.

VACA_PROF Número de descuentos a profesores, durante las vacaciones.

VACA_50 Número de Descuento del 50% de órdenes de servicio, en periodo vacacional.

VACA_ESP Número de Descuentos a socios durante vacaciones (50%).

VACA_100 Número de boletos con pago completo contratado por una empresa, en el periodo vacacional.

PARADA Clave de una población determinada.

DESCRIPCION
Nombre de la población en donde se realiza una parada.

KMS
Distancia en kilómetros de la central a la población.

HORA
Hora de salida de una corrida

DTNO
Clave de una población determinada (igual a parada)

CUPO
Número de lugares asignados a una corrida.

CLASE

Identifica la categoria del viaje (de primera clase o servicio expreso).

ASTO1

Asiento del operador uno en el autobús.

ASTO2

Asiento del operador dos en el autobús.

INIEX

Fecha de inicio en las cuales se asignan corridas extras

FINEX

Fecha de terminación para un periodo de corridas extras.

DIAS

Dias en que salen las corridas extras.

RESV

Contiene la posición de los asientos los cuales son asignados a reservaciones en caso de existir.

CVE

Clave de una población, en donde se realiza una parada en una corrida.

ASTO

Asientos reservados para una población.

PARA_SEC

Número de la parada secuencial de una población durante el viaje.

PER_NUM

Número de permisionario.

PER_NOM

Nombre del permisionario

UNI_NUPER

Número de permisionario.

UNI_NUM Número de la unidad de un permisionario.

EDO Describe si la corrida está abierta o cerrada (corridas extras).

TIPOS

Descripción de la ocupación de lugares de la corrida, es decir quien ocupa un asiento determinado, un anciano, un niño, un operador, etc.

CORR_LOCK

Permite conocer si un usuario está utilizando el registro de la corrida.

NUMCAM

Clave del camión asignado a la corrida.

LOGIN

Identificación de un usuario del sistema de boletos.

TAQUI

Taquilla en la que se esta usando el sistema de boletos. Tiene valores como A, B, ect.

SERIE_NUM

Número de serie de boletos para una taquilla.

VTAS_FECHA

Fecha de venta de un boleto.

SERIE

Número de serie del boleto vendido.

TPUEN

Es un campo que permite realizar la conexión cuando se realiza una cancelación y quien la realizó.

IMP

Costo del boleto vendido.

VTAS_ASTO

Número de asiento vendido.

TIBOL

Tipo de boleto: reimpreso o reexpedido.

NUMJEF

Clave del jefe de servicios que autorizó la cancelación o la reexpedición del boleto.

HRVTA

Hora en que se realiza la venta del boleto.

NAME

Nombre del pasajero.

DREAL

Población a la que se dirige el pasajero.

IMPRE

Estado de la impresión del boleto, esto es si se imprimió o no.

CORIMP

Contiene el número de veces que se reimprimió el boleto, en caso de haberlo realizado.

COREXP

Para el caso de pérdida del boleto se puede tener, reexpedición de boletos.

TMOV

Permite conocer si se realizó una venta o una cancelación, reservación, venta de cupones de agencias, por lo que los valores posibles de la variable son V (venta), C (cancelación), R (reservación), P (venta de un cupón).

POCAN

Población del boleto cancelado

MARC

Cuando se realiza una cancelación su valor es de C (cancelación), cuando no se da este movimiento su valor es de V (venta).

AGCIA

En caso de ser un cupon vendido por una agencia , nos permite conocer la clave de la agencia que realizó la venta.

CORR_FECH

Fecha de salida de una corrida.

CORR_GUIA

Indica si ya se sacó la guia de esa corrida o no :

1 no se saco la guia
2 ya se saco la guia

NUMCPN

Número de cupón vendido por una agencia.

KGEXT

Si el pasajero lleva peso de exceso se registran los kilogramos de más.

EQUIP

En caso de que el pasajero registre equipaje se enciende esta bandera.

PSWD

Contraseña de un usuario del sistema.

GPO

Grupo al que pertenece el usuario, esto es para asignación de privilegios dentro del sistema.

PASS_NOM

Nombre del usuario del sistema.

DES_FEENT

Fecha en que entra el despachador al sistema.

HRINI

Hora en la que inica sus labores el despachador.

HRSAL

Hora en la que el despachador deja de utilizar el sistema.

INACT

Tiempo en que el despachador esta inactivo.

HOCTE

Hora en la que se emitió el corte de un despachador.

DES_FESAL

Fecha en la que sale un despachador del sistema, esto es por que puede trabajar durante el cambio de día.

FOLIN

Folio de inicio para el registro de equipaje.

FOLFI

Folio de terminación del registro de equipaje.

EQUI_IMPTE

Importe causado por el equipaje en exceso.

EQUI_PESO

Peso del equipaje del pasajero que lleva como exceso.

DREAL

Destino del pasajero.

EQUI_HRREG

Hora que registra su equipaje el pasajero.

EQUI_TPUEN

Permite enlazar cuando se registra una cancelación en el registro del equipaje, a la taquilla en la que se realizó el registro del equipaje.

EQUI_TMOV

Nos da a conocer si fue una alta de equipaje o una cancelación.

EQUI_FEREG

Fecha en la que registra el pasajero su equipaje.

NUPAS

Número de pasajeros por corrida.

FOL_EQUI

Folio del registro del equipaje.

TAQ_EQUI

Taquilla en la que se realiza el registro del equipaje.

EQUI_NUM

Número de equipaje registrado.

V_FECHA

Fecha del archivo historico de ventas.

DH_FECHA

Fecha del archivo histórico de despachadores.

IMEXP

Importe de boletos vendidos con servicio expreso por despachador.

IMPRI

Importe de la venta de boletos marcados con servicio de primera clase.

HRCTE

Tiempo de inactividad del despachador.

IMPEQ

Importe total de las ventas por equipaje.

GIA_FECHA

Fecha que lleva la guía.

NOGUI

Número de la guía.

NOIMP

Número de impresión de la guía.

G_LIQUI

Nombre del liquidador

G_INGRE

Ingreso de la guía de viaje

SATE

Costo de la salida del autobús. (salida de la terminal)

BOL1

Número inicial de la boletera del operador

BOL2

Número final de la boletera del operador.

OPER1

Nombre del operador 1 asigando a la corrida.

OPER2

Nombre del operador 2 asigando a la corrida.

OTROS

Diversos gastos (lavado del camión, servicio , etc.)

NOVIA

Número de viaje en el que va el autobús.

NUPER

Número del permisionario.

IMPEQ

Importe de equipaje registrado.

TIPO

Tipo del boleto vendido adulto, niño, anciano, cupon, etc..

NUMBOL

Serie de un boleto vendido.

CODE_FOLIO

Ultimo folio del corte de caja de un despachador.

MANUAL

DE

USUARIO

D. MANUAL DE USUARIO

Descripción General

El sistema de telereservaciones de boletos, está enfocado a mejorar el servicio a usuarios de la empresa, para ello se cuenta con varios módulos, los cuales son dedicados a tareas muy específicas como son: el control de boletos vendidos, la rapidez de información a usuarios, la mejora en el servicio al cliente, el control del personal (despachadores, choferes, etc.), etc.

Principales componentes del *software*

El sistema se está implementando en un equipo PC compatible utilizando un manejador de base de datos llamado Clipper, utilizado tanto para la programación como para el despliegue de las pantallas de ayuda.

El manejador de base de datos instalado en este equipo ofrece una gran versatilidad en la programación, es por ello que se opto por él.

Como herramientas secundarias de *software* se utiliza el lenguaje de programación C, para implementar las pantallas de captura de información. Además, se han desarrollado pequeñas utilerías para implementación de los menús que presenta el sistema. Esta utilería se auxilia de carácteres de control para el manejo de pantalla. Todo esto se ha venido desarrollando en el ambiente del sistema operativo DOS versión 3.3. y con el sistema operativo NOVELL SFT (*system fault tolerant*).

Interfaces de *software hardware*

Para la entrada de información se utiliza un equipo PC compatible.

Para la salida de información (generación de reportes) se utilizarán las impresoras de martillo.

La impresora se utiliza para la obtención de boletos, cupones abiertos, guías, cortes por despachador, etc., esto es, todos los reportes que emite el sistema.

Se cuenta con un servidor de archivos y una serie de microcomputadoras conectadas en red (NOVELL).

Objetivos

Se pretende evitar todo proceso manual, en lo que respecta a todo lo relacionado con la venta de boletos, así como todo lo que implica.

Con ello se pretende dar una mejor imagen de la empresa, en su servicio a los usuarios e internamente un mejor manejo de la información y en el control de los ingresos.

Por último, tener actualizada en la mejor manera posible la información y evitar el deperdicio de recursos humanos, en la manipulación de la información.

Logrando lo anterior se podrá proporcionar cualquier información al viajero en caso de ser solicitada, se refinará el servicio a los viajeros. El sistema también pretende dar solución a diversos problemas a que se enfrenta la compañía, sobre todo en el retardo de alguna información indispensable para la obtención de reportes.

Sistema de telereservaciones

Tipos de usuarios del sistema de telereservaciones

Despachador

Supervisor de taquillas

Telefonista

Liquidador

Jefe de servicios

Controlador de servicios

1) Despachador e Informes:

- Venta de boletos normales
- Cancelación de boletos normales
- Reexpedición de boletos
- Asignación de despachadores
- Desasignación de despachadores

2) Supervisor de Taquillas

- Mismos derechos del despachador
- Autorización de cancelación de boletos
- Reservaciones
- Cancelación de reservaciones
- Registro de itinerarios
- Apertura de corridas extras
- Autorización de reexpediciones de boletos
- Ampliación del cupo y asignación de un autobús

3) Telefonista

- Registro de cupones de agencia
- Cancelación de cupones de agencia

4) Liquidador

- Desbloqueo de corridas
- Ampliación del cupo y asignación de autobús
- Corte de agencia
- Corte por despachador
- Reexpedición de cortes
- Emisión de guías de pasaje
- Reexpedición de guías
- Corte por liquidador
- Generación de días
- Reportes diarios
- Reporte de poblaciones
- Movimiento de corridas
- Movimiento de permisionarios
- Movimiento de tarifas
- Reporte de reservaciones, cancelaciones y descuestos especiales
- Reporte de Cancelaciones
- Reporte diario de guías
- Reporte de viajes por hora
- Reporte de salidas por hora
- Reporte de salidas por autobús

5) Jefe de Servicios

- Mismos derechos del supervisor de taquillas
- Registro de permisionarios
- Registro de agencias
- Movimientos de corridas
- Reporte de poblaciones
- Movimiento de permisionarios
- Cambio de tarifas
- Cortes diarios
- Reporte de reservaciones, cancelaciones y descuentos especiales
- Reporte de cancelaciones
- Reporte diario de guías
- Reporte de viajes por hora
- Reportes mensuales

6) Controlador de Servicios

- Todos los derechos del Jefe de Servicios
- Generación de días
- Prereservaciones a corridas

- Actualización de factores
- Mantenimiento
- Movimiento de factores por kilómetro
- Movimiento de usuarios

En caso de falla del sistema:

1. Checar conexiones de los equipos.

   Si sólo es un equipo volver a dar reset al equipo y no realizar el paso 2.

2. Para el caso en el que todas las máquinas se queden en el proceso que estaban realizando seguir los siguientes puntos:

   a) Avisar a todos los usuarios del sistema que se dará de baja para realizar el proceso de indexación.

   b) Ir al *server*, para dar las siguientes instrucciones :

Aparece	Proporcionar
:	*Down* (oprimir la tecla de enter)

   Si existen usuarios en el sistema y no se pueden salir aparecerá un mensaje en el cual se indica que los archivos están abiertos, por lo que debemos proporcionar YES

Aparece	Proporcionar
warning open files	yes (oprimir enter)

   Tardará unos momentos en mandar un mensaje en el cual nos indica que podemos apagar el equipo.

Aparece	Proporcionar
Server shutdown reboot	Debemos apagar y endencer el equipo
Aparece	Proporcionar
F1 continue, F2 setup	F1 (es la tecla)

   Esperar un momento, para que el equipo nos indique que se encuentra listo para continuar y esto lo sabremos cuando nos de dos puntos (:) y podamos dar el comando MONITOR (enter).

- Actualización de factores
- Mantenimiento
- Movimiento de factores por kilómetro
- Movimiento de usuarios

En caso de falla del sistema:

1. Checar conexiones de los equipos.

   Si sólo es un equipo volver a dar reset al equipo y no realizar el paso 2.

2. Para el caso en el que todas las máquinas se queden en el proceso que estaban realizando seguir los siguientes puntos:

   a) Avisar a todos los usuarios del sistema que se dará de baja para realizar el proceso de indexación.

   b) Ir al *server*, para dar las siguientes instrucciones :

Aparece	Proporcionar
:	*Down* Coprimir la tecla de enter)

   Si existen usuarios en el sistema y no se pueden salir aparecerá un mensaje en el cual se indica que los archivos están abiertos, por lo que debemos proporcionar YES

Aparece	Proporcionar
warning open files	yes Coprimir enter)

   Tardará unos momentos en mandar un mensaje en el cual nos indica que podemos apagar el equipo.

Aparece	Proporcionar
Server shutdown reboot	Debemos apagar y endencer el equipo
Aparece	**Proporcionar**
F1 continue, F2 setup	F1 Ces la tecla)

   Esperar un momento, para que el equipo nos indique que se encuentra listo para continuar y esto lo sabremos cuando nos de dos puntos C:) y podamos dar el comando MONITOR Center).

Aparece :	Proporcionar monitor (dar enter) con esto aparecen los cuadritos

c) Una vez realizado el punto anterior se debe realizar el proceso de indexación:

Dar reset al equipo y esperar a que entre al menú principal. Dar la tecla de ESC

Aparece	Proporcionar
f:login>	login supervisor (dar enter)
Aparece	Proporcionar
f:system>	cd .. (dar enter)
Aparece	Proporcionar
f:>	cd boletos (dar enter)
Aparece	Proporcionar
f:boletos>	inddir (dar enter)

Aparece

Un mensaje que dice suspendase cualquier operación del sistema durante este proceso. Pulse (ESC) para continuar.

Lo anterior nos indica que por ningún motivo debe estar alguna persona utilizando el sistema, en caso contrario no podemos realizar este proceso. Este proceso se deberá efectuar en el momento en que todos los usuarios esten fuera del sistema (para tal efecto pueden estar todos en el menú principal o en su defecto tener apagados los equipos).

Proporcionar

ESC

Aparece

Aquí podemos observar varias opciones dentro de las cuales se encuentran una que indica

T o d o s

Proporcionar

Debemos ir a la opción: T o d o s
(Con la flecha que indica hacia abajo) y dar enter.

Aparece un mensaje en el cual nos indica que está indexando los archivos correspondientes.

Una vez que termina con este proceso nos presenta un mensaje para seleccionar una opción.

Aparece

Seleccione su opción

Proporcionar

ESC (la tecla)

Aparece

Un cuadrito de salir SI o NO

Proporcionar

Elegir la opción de SI (dar enter)

Aparece	Proporcionar
f:boletos>	boletos (dar enter)

d) Dar reset a todos los equipos y esperar a entrar al sistema.

MENU PRINCIPAL

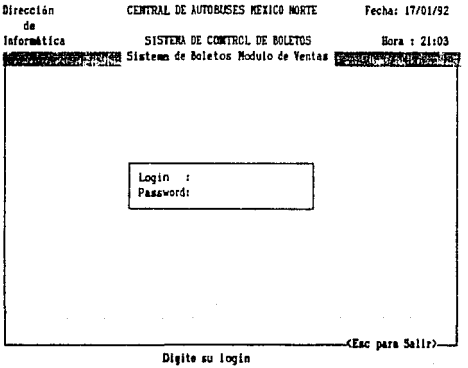
El menú principal se encuentra formado por cinco módulos, descritos en el capítulo 3 del trabajo de tesis, los cuales son:

Taquillas
Liquidaciones
Impresión de reportes
Mantenimiento
Catálogos

Para poder entrar a cualquiera de los módulos se selecciona la opción con el cursor y se pulsa enter.

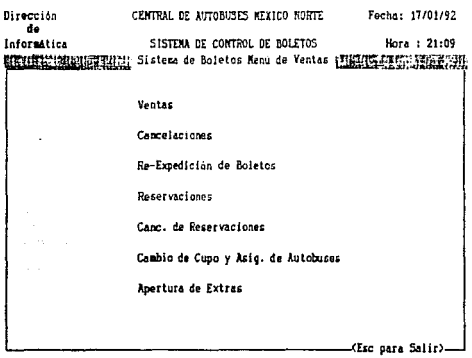
A continuación se muestran como se encuentra agrupado el sistema según el usuario.

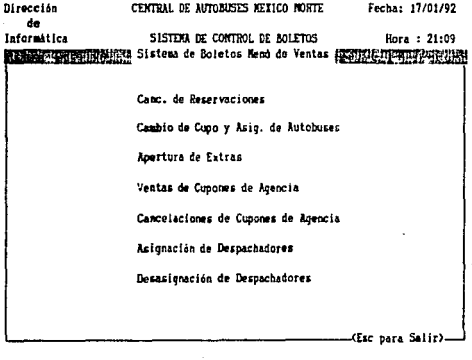
En esta pantalla se digita el *login* y enseguida el *password* del usuario para poder entrar al menú principal, al aparecer la pantalla del menú principal se selecciona la opción requerida con el cursor.



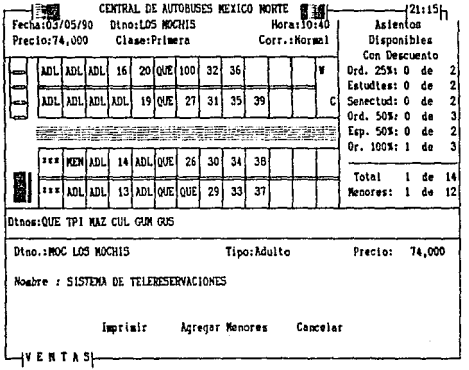
Digite su login

Esta pantalla nos muestra el menú de ventas, el cual consta de varias opciones, las cuales se seleccionan por medio de los cursores. Al seleccionar una opción se pulsa enter y de esta manera se entra a la opción seleccionada.

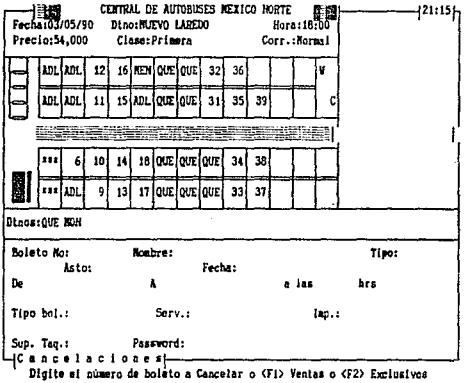




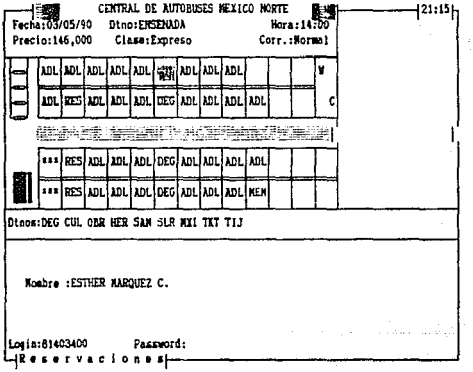
Esta pantalla nos muestra la venta de un boleto, se requiere oprimir F1 para que se desplieguen en la pantalla los horarios que existen durante el dia al destino deseado, después se elige el asiento, el tipo de persona (adulto, estudiante, senectud, etc.) y se pide el nombre del pasajero y se pulsa enter y automáticamente manda a imprimir el boleto.



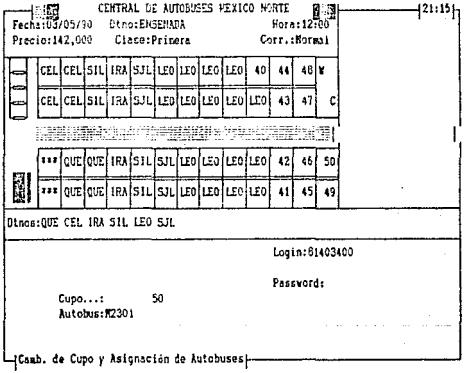
La siguiente pantalla nos muestra la cancelación de un boleto, se selecciona la opción de cancelación en el menú de ventas, se digita el número de boleto que se va a cancelar y se pulsa enter.



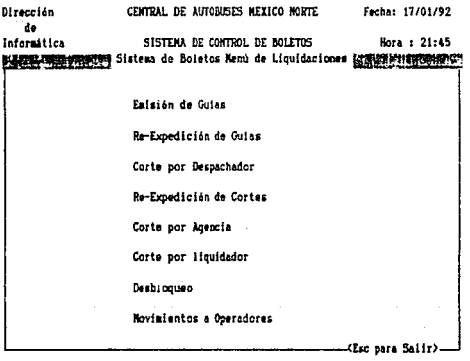
Esta pantalla nos muestra como reservar un boleto, se selecciona en el menú la opción de reservaciones, se escoge el número del asiento y se escribe el nombre de la persona a la que se le reservó, y se pulsa enter.



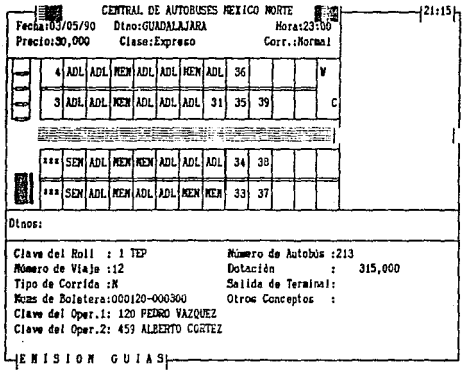
Esta pantalla nos muestra como realizar el cambio de cupo y la asignación de autobús. Se selecciona la opción en el menú principal y se pulsa enter, se pone la fecha, destino y hora deseada, después se pulsa enter, y es cuando permite ampliar el número de asientos, después se pulsa enter, enseguida con el cursor se selecciona la parte donde dice autobús, y se pulsa enter y en ese momento se escribe el número de autobús.



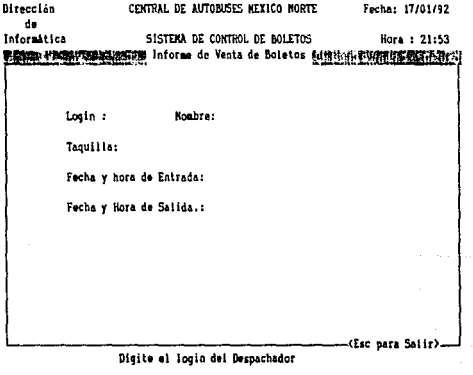
Esta pantalla nos despliega el menú de liquidaciones, el cual consta de varias opciones, las cuales se muestran a continuación, para elegir cualquier opción se selecciona con el cursor y se dá enter.



Esta pantalla nos despliega como hacer una emisión de una guía. Para poder hacer la emisión de una guía de requiere de varios datos, los cuales se muestran en la pantalla, para ir registrando los datos se usan los cursores y al terminar se pulsa enter.



Esta pantalla nos presenta como entrar a la opción de corte por despachador.



Las siguientes pantallas nos muestran el menú de mantenimiento, el menú de catálogos y el menú de reportes. Las opciones en cualquiera de los menús se seleccionan con los cursores y se pulsa enter.