30
2j

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

TESIS

SISTEMA DE CONTROL Y FACTURACION DE SOBREPESO EN LA PAQUETERIA DE ESTAFETA MEXICANA

PARA OBTENER EL TITULO DE

INGENIERO EN COMPUTACION

P R E S E N T A N :

CASTILLO PERALTA MA. ALEJANDRA

ENRIQUEZ CABRERA JOSE LUIS

ESCOBAR HERNANDEZ OSCAR

MELCHOR LOPEZ SALVADOR

NOLASCO OCAMPO MONICA VIRGINIA

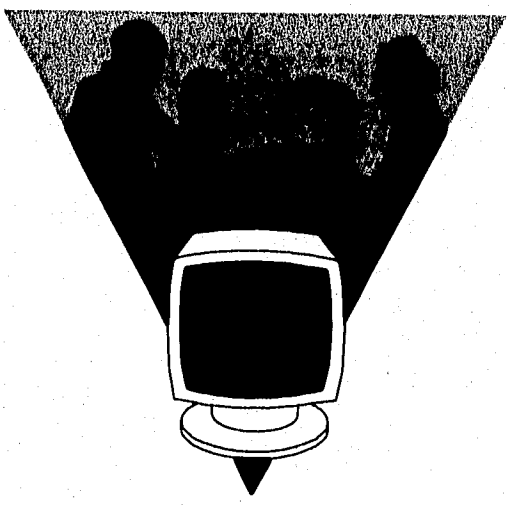
ASESOR: M. en I. LAURO SANTIAGO CRUZ OCTUBRE 1996

En agradecimiento:

A la Universidad Nacional Autónoma de México

A la Facultad de Ingeniería

A nuestros Profesores y Asesor



CAPITULO I

INTRODUCCION

Introducción

En los últimos diez años México ha sufrido una acelerada transformación en los ámbitos económico, político, social. Esta transformación conlleva nuevas necesidades que deben ser cubiertas a corto plazo, pues de lo contrario, el costo económico, social y político puede ser muy grande.

El sector de servicios debe ofrecer más y mejores servicios de una calidad antes no necesaria. Por otra parte, la calidad en los servicios que anteriormente no era un factor determinante a considerar, es hoy en día imprescindible para la supervivencia de este sector.

Con el advenimiento de la automatización de procesos podemos lograr entre otras cosas:

- Ahorro de recursos materiales, humanos y financieros.
- Producir más servicios en menos tiempo.
- Un nivel de calidad mayor.
- Mayor control en el proceso.

La ingeniería de software y las computadoras son algunas de las herramientas utilizadas en la automatización de procesos.

Un ejemplo del requerimiento de automatización es el proceso de pesaje de paquetería en Estafeta Mexicana, donde se detectaron irregularidades en el peso de la paquetería que se recibe por los diferentes servicios de mensajería que la empresa ofrece. Esto llevo a tener que comunicarle al usuario el cargo adicional de sobrepeso en su factura, la cual emitían conjuntando todas las guías que pertenecían a un cliente.

Este proceso de detección de guías con sobrepeso fue en un principio manualmente, dado que eran pocas guías que reflejaban sobrepeso. Pero debido al auge que tuvo, resultaba imposible tener todas las facturas a tiempo para su cobro. Por lo que el servicio en cuanto a este rubro deja mucho que desear, amen de las perdidas que la empresa reporta por no realizarse correctamente el trabajo.

Actualmente el proceso se lleva a cabo en un sistema desarrollado en lenguaje FOXPLUS bajo ambiente DOS, el cual no sigue ningún esquema de diseño ya que se comenzó haciendo programas por separado según las necesidades. Después se conjuntaron toda esta serie de programas y junto con otros formaron el sistema de sobrepeso. Esto llevó a diversos problemas en el área de sistemas, ya que aquí se capturaban de manera incorrecta los datos de la paquetería o fuera de tiempo, por lo que el área de facturación de sobrepeso tuvo que absorber esta tarea.

La empresa con el fin de ir a la vanguardia tecnológica y de manejar de manera más eficaz su información, realiza una fuerte inversión en la adquisición de equipos servidores más poderosos así como de un manejador de bases de datos capaz de manejar la gran cantidad de información generada a nivel nacional. El resultado fue la utilería canadiense llamada SENTAI, que tiene como manejador de bases de datos a PROGRESS, cuyas características de Lenguaje de Cuarta Generación hacen posible el manejo de información más rápido y confiable. Cabe mencionar que una vez que se adquirió PROGRESS fue utilizado para desarrollar aplicaciones en algunas áreas de la

empresa, no contemplando así al área de facturación, que sigue utilizando su sistema original desarrollado en lenguaje FOXPLUS, con una base de datos de clientes independiente a la utilizada en el corporativo de Estafeta, ocasionando que existan incongruencias en los datos de los clientes. Una consecuencia más grave aún de esto, es el hecho de que debido a que existe una base de clientes en corporativo (Aeropuerto) y una base de clientes en facturación (Ceylán), es frecuente que no se sepa a que cliente pertenece determinada guía, haciendo imposible el cobro del sobrepeso.

Asimismo, cada mes se tiene la necesidad de realizar una exportación de información referente a facturas del sistema que maneja facturación, para posteriormente importarlas al sistema que se tiene en el corporativo. Esto como consecuencia de las diferencias en ambos sistemas.

Aunado a esto, se tiene que el sistema actual de facturación está ya muy rezagado. No posee conceptos tales como bases de datos relaciónales, consistencia de datos, etc.

En caso de no realizar este cambio de sistema, la empresa podría seguir reflejando pérdidas en cuanto al cobro del sobrepeso, debido a la inconsistencia de datos entre las diferentes áreas de la empresa. Actualmente se encuentra el respectivo cliente de aproximadamente el 50 % de las guías, y mensualmente se factura por montos cercanos a los $ 300,000.00 pesos.

En el presente trabajo se pretende realizar un nuevo sistema con la utilería del corporativo (SENTAI), cuyo manejador sea PROGRESS y que además utilice la misma base de clientes que tiene el corporativo. De igual forma que la estructura de sus facturas sea la misma que la manejada en corporativo, para que al ser enviadas, su integración a la base sea automática. Se pretende que este nuevo sistema sea instalado en los tres principales centros operativos de la empresa que son: México, Guadalajara y Monterrey, existiendo un flujo e intercambio de información entre éstos.

Se pretende que con el nuevo sistema el ingreso por concepto de estas facturas se incremente en un 50%, es decir, $ 450,000.00 pesos mensuales.

El presente trabajo tiene como OBJETIVO proporcionar un sistema automatizado que agilice y controle el sobrepeso de la paquetería en Estafeta Mexicana, disminuyendo costos para la empresa y contestando rápidamente la demanda del usuario de una manera confiable y segura.

El presente trabajo está basado en ocho capítulos, en los cuales desarrollaremos una metodología para la realización del sistema antes descrito.

El **Capítulo I** corresponde a una introducción del sistema que actualmente se maneja en el área de facturación de sobrepeso de Estafeta Mexicana. Para esto hacemos un estudio de la empresa, abarcando desde su fundación, su crecimiento, las problemáticas que se generaron con este crecimiento, hasta el surgimiento del área de facturación de sobrepeso y la problemática que se tiene en la elaboración de sus procesos.

En el **Capítulo II** se muestra la metodología que puede ser utilizada para el desarrollo de software, analizando sus diversos conceptos y concluyendo la metodología a utilizar en el desarrollo del sistema propuesto.

El **Capítulo III** se refiere al análisis, enfocándonos en especial, a la detección de la problemática actual, al estudio de la organización de la empresa y de las áreas relacionadas con el cobro del sobrepeso y al sistema actual. Así mismo, se estudia el flujo de información entre las áreas. Por último realizamos una descripción del sistema propuesto.

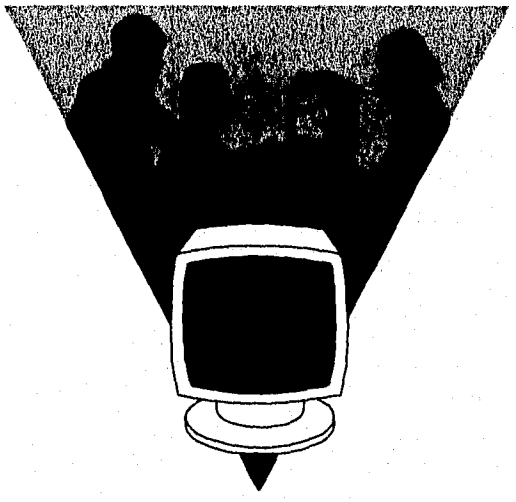
El **Capítulo IV** abarca lo que es el diseño del sistema propuesto, mediante la aplicación de técnicas y herramientas con el propósito de definir el sistema a detalle para su realización, es decir, elaboramos cartas estructuradas, base de datos, tablas a utilizar y diccionario de datos.

El **Capítulo V** contempla el desarrollo, es decir, la traducción del diseño a un lenguaje de programación.

En el **Capítulo VI** se describen el plan de pruebas, instalación y mantenimiento para nuestro sistema, incluyendo su calendarización y los manuales de usuario y operador.

Por último, se presentaran los resultados obtenidos, comentarios y conclusiones sobre el trabajo de tesis.

Además el trabajo constará de una sección para apéndices y una para bibliografía, las cuales se incluirán al final del mismo.



CAPITULO II

METODOLOGIA

Introducción

El objetivo del presente capítulo es dar una visión general de las *metodologías* existentes y más adecuadas en el desarrollo de productos de *software*, orientadas a sistemas de información, seleccionando la *metodología* que más se ajuste a los requerimientos del sistema a desarrollar.

Hay que hacer notar que la realización de nuestro sistema, se efectuará con base en un desarrollo modular, lo cual permitirá un desarrollo fácil y en un momento determinado poder cambiar la estructura y agregar o quitar funciones a los módulos ya existentes.

Se presenta un estudio de toda la trayectoria seguida para el desarrollo partiendo desde el análisis, diseño, programación y llegando hasta las pruebas, instalación, manuales de usuario y certificación del sistema.

II.1.-Ciclo de vida de un proyecto de software

Durante las tres primeras décadas de la informática, el principal desafío era el desarrollo del hardware de las computadoras, de forma que se redujera el costo de procesamiento y almacenamiento de datos. Actualmente el problema es mejorar la calidad y reducir el costo de las soluciones basadas en computadoras, o sea soluciones que se implementan con el software.

Las enormes capacidades de procesamiento y almacenamiento de hardware moderno representan un gran potencial de cálculo. El software es el mecanismo que nos facilita utilizar y explotar este potencial. Aún en la actualidad se siguen muchos esquemas que por lo regular traen atrasos y costos muy altos en la producción de software y por ende el fracaso y la mala calidad de éste.

II.2.- Metodología para desarrollar proyectos

A continuación se presentan dos puntos de vista que consideramos relevantes para la elección de una adecuada metodología para el desarrollo de proyectos.

Punto de vista administrativo

Cabe aclarar que la presente metodología sigue un punto de vista administrativo, y se anexa sólo como una comparación enfocada por la ingeniería de software.

- Investigación preliminar
  - Requerimientos
  - Estudio de factibilidad
  - Aprobación de requerimientos
- Determinación de requerimientos
  - Qué se está haciendo?
  - Cómo se está haciendo?
  - Qué tan bien se lleva a cabo el labor?
  - Existe algún problema?
- Desarrollo de sistema propuesto (sistema fácil de modificar)
- Diseño de sistema
  - identificar alternativas y factibles a producir

Introducción

El objetivo del presente capítulo es dar una visión general de las metodologías existentes y más adecuadas en el desarrollo de productos de software, orientadas a sistemas de información, seleccionando la metodología que más se ajuste a los requerimientos del sistema a desarrollar.

Hay que hacer notar que la realización de nuestro sistema, se efectuará con base en un desarrollo modular, lo cual permitirá un desarrollo fácil y en un momento determinado poder cambiar la estructura y agregar o quitar funciones a los módulos ya existentes.

Se presenta un estudio de toda la trayectoria seguida para el desarrollo partiendo desde el análisis, diseño, programación y llegando hasta las pruebas, instalación, manuales de usuario y certificación del sistema.

II.1.-Ciclo de vida de un proyecto de software

Durante las tres primeras décadas de la informática, el principal desafío era el desarrollo del hardware de las computadoras, de forma que se redujera el costo de procesamiento y almacenamiento de datos. Actualmente el problema es mejorar la calidad y reducir el costo de las soluciones basadas en computadoras, o sea soluciones que se implementan con el software.

Las enormes capacidades de procesamiento y almacenamiento de hardware moderno representan un gran potencial de cálculo. El software es el mecanismo que nos facilita utilizar y explorar este potencial. Aún en la actualidad se siguen muchos esquemas que por lo regular traen atrasos y costos muy altos en la producción de software y por ende el fracaso y la mala calidad de éste.

II.2.- Metodología para desarrollar proyectos

A continuación se presentan dos puntos de vista que consideramos relevantes para la elección de una adecuada metodología para el desarrollo de proyectos.

Punto de vista administrativo

Cabe aclarar que la presente metodología sigue un punto de vista administrativo, y se anexa sólo como una comparación empleada por la ingeniería de software.

- Investigación preliminar
  - Requerimientos
  - Estudio de factibilidad
  - Aprobación de requerimientos
- Determinación de requerimientos
  - Qué se está haciendo?
  - Cómo se está haciendo?
  - Qué tan bien se lleva a cabo la tarea?
  - Existe algún problema?
- Desarrollo del sistema prototipo {sistema fácil de modificar
- Diseño del sistema
  - Identificar informes y salidas a producir

  - Datos específicos (dibujo de como se verá)
  - Describe los datos calculados o almacenados
  - Grupo de datos y procedimientos
  - Estructuras de archivos
  - Dispositivos de almacenamiento
  - Se usan formas para realizar diseño, diagramas, tablas, etc.
- Desarrollo del software
- Prueba de los sistemas
  - Datos especiales para localizar problemas
  - Permite usar el sistema para encontrar problemas no planteados
  - Datos Tipo
- Puesta de marcha {Mantenimiento

Punto de vista ingeniería de software

Existen diversas metodologías para el desarrollo de software, cabe destacar de entre ellas los modelos: clásico, en cascada, en espiral y en cascada estructurada. Estos modelos combinan una serie de aspectos relacionados con el desarrollo de proyectos, anteriormente manejados pero sólo en forma empírica.

También existen las técnicas de cuarta generación que se orientan a especificar un software que se aproxime al lenguaje natural o en una notación que proporcione funciones significativas. Un entorno para el desarrollo de software que soporte el paradigma 4GL puede incluir todas o algunas de las siguientes herramientas: lenguajes no procedimentales para consultas de base de datos, generación de informes, manipulación de datos, interacciones y definición de pantallas, generación de código, facilidades gráficas de alto nivel y facilidades de hoja de cálculo.

Debido a esto se busca una combinación de las diferentes metodologías para crear un diseño que se adapte a los diferentes conceptos que acentúan más la línea hacia un desarrollo de productos de software de alta calidad y un fácil mantenimiento.

- Estudio general del sistema
  - Definición del sistema
  - Diagnóstico de la situación actual
  - Análisis de factibilidad
  - Análisis del sistema
- Planeación
  - Alcances del sistema
  - Recursos
  - Estimación de costos
  - Herramientas de control de avance
- Análisis y especificaciones estructurada (Descripción de procesos)
  - Por diagramas de flujo de datos (DFD)
  - Definición de diccionario de datos
  - Minlespecificaciones
  - Arboles y tablas de decisión
  - Español estructurado

- Diseño estructurado
  - Carta estructurada
  - Características de la carta
  - Modularidad
  - Cohesión
  - Acoplamiento
  - Documentación
  - Manual de usuario
  - Manual de operación
- Pruebas y confiabilidad del sistema
  - Niveles de correctividad y pruebas
  - Generadores de datos de pruebas
  - Pruebas de validación y volumen
  - Simulación del sistema
- Instalación y mantenimiento
  - Plan de instalación
  - Capacitación
  - Carga de archivos
  - Identificar resultados y desviaciones

La presente metodología la tenemos comprendida en tres grandes bloques los cuales son la piedra angular para el desarrollo basado en prototipos.

Planeación
- Estudio de viabilidad.
- Estudio preliminar, análisis y diseño detallado.
- Determinar especificaciones funcionales.

Desarrollo
- Diseño de programas lógicos.
- Desarrollo de programación y procedimientos.
- Implantación.
- Integración de las claves y documentos de pruebas.

Mantenimiento
- Operación y prueba final.
- Desarrollo de procedimientos de operación.

II.3.- Definición y análisis del sistema

En esta etapa de la metodología se realiza un estudio breve en el cual se definen los objetivos y se evalúan costos, beneficios y necesidades para determinar que tan factible es el sistema. También se establecen las limitaciones en costos, recursos y tiempo.

En resumen es un análisis y diseño preliminar , es decir, un vistazo rápido del problema a atacar.

Análisis del sistema

1.- Introducción

2.-Descripción
- Especificaciones del sistema
  - Fuentes
    - Internas
      - Estudio directo
      - Manuales
    - Externas
      - Estudios
      - Documentación
- Análisis de requerimientos

3.- Limitantes
4.- Hardware y software utilizados y requeridos
5.- Costo
6.- Calendario
7.- Anexos (Documentación actual del sistema)

Técnicas

- Entrevistas
- Matrices
- Diagramas de Flujo
  - Lógico
  - Flujogramas
- Diagramas de bloque
- Narrativas
- Análisis Estructurado
  - Diagramas de flujo de datos
  - Pseudocódigo

Medios

- Gente
- Documentación interna
- Documentación externa
  - Manuales de organización
  - Procedimientos

II.4.- Estudio de Factibilidad

El estudio de factibilidad es un análisis comparativo entre el sistema actual y el sistema propuesto.

Del sistema actual

- Diagrama Delimitar a área organizacional
  - de flujo de datos
    - Gente
    - Medios
  - Área organizacional con el sistema
- Descripción a área organizacional estudiada
  - Objetivos y funciones de la área

2.-Descripción
- Especificaciones del sistema
  - Fuentes
    - Internas
      - Estudio directo
      - Manuales
    - Externas
      - Estudios
      - Documentacion
- Análisis de requerimientos

3.- Limitantes
4.- Hardware y software utilizados y requeridos
5.- Costo
6.- Calendario
7.- Anexos (Documentación actual del sistema)

Técnicas

- Entrevistas
- Matrices
- Diagramas de Flujo
  - Lógico
  - Flujogramas
- Diagramas de bloque
- Narrativas
- Análisis Estructurado
  - Diagramas de flujo de datos
  - Pseudocódigo

Medios

- Gente
- Documentación interna
- Documentación externa
  - Manuales de organización
  - Procedimientos

II.4.- Estudio de Factibilidad

El estudio de factibilidad es un análisis comparativo entre el sistema actual y el sistema propuesto.

Del sistema actual

- Diagrama Delimitar el área organizacional
  - de flujo de datos.
    - General.
    - Detallado.
  - Ver áreas relacionadas con el sistema.
- Describir el área organizacional relacionada
  - Objetivos y funciones de las áreas.

- ♦ Complementar DFD, con descripciones narrativas de los procesos y estudiar las entradas y salidas del sistema.

- Analizar problemas y oportunidades
  - ♦ Forma de resolverlos y los elementos con que contamos.
- Seleccionar y definir los subsistemas prioritarios para su estudio y desarrollo
- De los subsistemas relacionados, definir y mejorar potenciales
  - ♦ Definir un cuestionario para poder dar la mejor solución.
- Coordinación y objetivos
  - ♦ Mejoras reales, que proporciona el sistema sin afectar los intereses de ningún área.
- Analizar las decisiones y sus requerimientos de información
  - ♦ Formalizar los requerimientos del sistema.
- Describir los requerimientos de información y definir los sistemas de información
- Evaluación y definición de alternativas de solución

Del sistema propuesto

- Introducción
  - ♦ Una breve descripción del problema y medio ambiente en el cual el sistema se encuentra y las limitantes generales del proyecto.
- Exposición razonada del sistema
  - Definición y descripción detallada del problema.
  - Breve explicación de las soluciones que se pretenden para el problema dado.
  - Formas de resumen de costo.
  - Restricciones al diseño del sistema.
- Descripción del sistema actual
  - ♦ Una revisión resumida y general del sistema ENTRADAS, PROCESOS Y SALIDAS, así como la filosofía de control actual.
- Descripción del sistema propuesto
  - ♦ Una revisión resumida y general del sistema ENTRADAS, PROCESOS Y SALIDAS, así como la filosofía de control propuesta.
- Análisis de costo beneficio
  - ♦ Es el justificante económico: (beneficio / costo) > 1.
- Evaluación con técnicas de factibilidad
- Otros aspectos relacionados con el sistema o procedimientos de la organización
- Conclusiones

II.5.- Técnicas de estimación de costos

La programación del tiempo y la estimación de costos del software están íntimamente relacionadas. La mayor parte de los costos de un proyecto de software grande son sólo aquellos relativos al pago de la gente que desarrolla el software. El costo del proyecto es directamente proporcional al número de personas-mes requeridas para terminar el trabajo. Hay, desde luego, otros costos, como los de hardware, viajes, capacitación, etc., pero éstos son más fáciles de calcular. No se basan en imponderables como la productividad del programador y la estimación del tamaño del programa fuente.
La técnica más utilizada para calcular los costos del software es estimar el tamaño del sistema de programación que se va a entregar y de ahí calcular el número de programadores-mes requeridos para construir el sistema, por medio de datos históricos

de productividad o la intuición. El costo del sistema total se basa en esa cifra más los gastos varios. Si el software se desarrolla para una agencia externa, se añade una cantidad de utilidades a esta estimación.

El tamaño del sistema se calcula por medio de un estudio del sistema preliminar para establecer las partes que componen el sistema. Se hace una estimación del tamaño de cada unidad. Tal estimación suele basarse en la experiencia y la intuición. Después se suman las estimaciones para obtener el tamaño calculado del sistema total. Este método se basa en la suposición de que se puede predecir la productividad del programador para una parte dada del sistema. También supone que el diseño preliminar no está simplificado en demasía y que refleja con exactitud el sistema que se va a producir.

Ya se ha visto que la productividad del programador se ve afectada por múltiples factores y no se puede estimar con precisión, considerando el tamaño de cada unidad que se va a desarrollar. Otros factores, como la experiencia del programador, la novedad, la complejidad y las restricciones de la aplicación también se deben tener en cuenta.

II.6.- Diseño

Proceso de aplicar técnicas y herramientas con el propósito de definir un dispositivo, proceso o sistema en suficiente detalle para permitir su realización.

El diseño de software es un proceso a través del cual los requerimientos se traducen en una representación de software.

Documento de diseño

El diseño del software es un proceso creativo que requiere del diseñador ciertas cualidades y el diseño final suele ser una repetición de varios diseños preliminares. El diseño no se puede aprender en un libro; debe practicarse y aprenderse mediante la experiencia y el estudio de sistemas ya existentes. Un buen diseño es la clave de una ingeniería de software efectiva. Un sistema de software bien diseñado es fácil de aplicar y mantener, además de ser comprensible y confiable. Los sistemas mal diseñados, aunque puedan funcionar, pueden ser caros de mantener, difíciles de probar y poco confiables. La etapa de diseño es, por tanto, la parte más importante del proceso de desarrollo de software.

Hasta hace no mucho, el diseño del software era un proceso destinado a un fin determinado. Dado un conjunto de requisitos, por lo general en lenguaje natural, se preparaba un diseño informal, a menudo en forma de organigrama. Entonces se empezaba la codificación y se modificaba el diseño a medida que se aplicaba el sistema. Al acabar la etapa de aplicación, el diseño solía diferir tanto de las especificaciones iniciales que el documento del diseño original resultaba una descripción totalmente inadecuada del sistema.

No hay una manera definida de establecer lo que se entiende por un "buen diseño". Dependiendo de la aplicación y los requisitos del proyecto particular, el buen diseño puede ser uno que permita producir una codificación muy eficiente, puede ser un diseño mínimo donde la aplicación sea lo más compacta posible o puede ser el diseño de más

fácil mantenimiento. Un diseño mantenible implica minimizar el costo de los cambios del sistema, y eso significa que el diseño tiene que ser comprensible y que las modificaciones deben tener un efecto local. Ambas cosas se logran si el diseño del software es muy coherente y poco acoplado.

Técnicas de diseño

El diseño efectivo del software se logra mejor utilizando una metodología consistente de diseño. Hay una gran cantidad de metodologías de diseño desarrolladas y que se utilizan en diferentes aplicaciones. En esencia, la mayoría de estas metodologías se pueden clasificar en una de las tres áreas siguientes:

1. *Diseño funcional descendente.* El sistema se diseña desde un punto de vista funcional empezando con una visión de alto nivel y refinándola de manera progresiva hasta llegar a un diseño más detallado.
2. *Diseño orientado al objeto.* El sistema se ve más como una colección de objetos que como funciones que pasan mensajes de un objeto a otro. Cada objeto tiene su propio conjunto de operaciones aplicadas.
3. *Diseño controlado por los datos.* Esta metodología plantea que la estructura de un sistema de software debe reflejar la estructura de los datos que este procesa. Por tanto, el diseño del software se obtiene de un análisis de los datos del sistema de entrada y salida.

Diseño Funcional descendente

El diseño descendente se basa en la noción de que la estructura del problema debe determinar la estructura de la solución del software. Esta metodología utiliza la característica humana más fundamental para la solución de problemas: la abstracción.

Después de haber formulado y descrito una solución inicial de alto nivel, el proceso de solución del problema debe repetirse para cada abstracción utilizada. Este proceso de refinamiento continua hasta que se ha preparado una especificación de bajo nivel para cada abstracción. Aunque la lista anterior significa que hay etapas de diseño claras y distintas, los límites entre ellas son variables. En algunos sistemas, el mejor enfoque puede ser seguir el orden de la lista anterior, para otros, el diseñador puede abandonar al principio la etapa del diagrama de estructura, regresar y hacerla después de describir el diseño en un lenguaje de descripción de programas.

Es muy importante que la representación de cada etapa del diseño sea clara y concisa. Una regla empírica útil que puede adoptarse es expresar el diseño de modo que cada parte de la especificación pueda describirse sin problemas en una hoja de papel de tamaño normal.

Diseño orientado al objeto

En esta metodología en vez de desarrollar el diseño de un sistema de software por medio de una descomposición funcional descendente, la metodología se orienta al objeto. En este diseño los componentes del software se ven más como objetos que como funciones. Cada objeto tiene un conjunto asociado de operaciones permitidas, y los objetos se comunican mediante el paso de mensajes que, por lo general, incluyen una instrucción para activar una función determinada.

El diseño orientado a objetos se basa en la idea de utilizar el ocultamiento de información como principal criterio de descomposición y en la noción de los tipos de datos abstractos. Esta metodología ha sido adoptada de manera entusiasta por algunos desarrolladores y educadores.

II.7.- Codificación y lenguajes de programación

El paso de codificación traduce una representación del software, dada por un diseño detallado, a una realización en un lenguaje de programación. El proceso de traducción continúa cuando un compilador acepta el código fuente como entrada y produce como salida un código objeto dependiente de la máquina. Más tarde, la salida del compilador es traducida a código máquina, que son las instrucciones reales que dirigen la lógica cableada o microprogramada de la unidad central de proceso.

La programación sistemática

La programación estructurada no emplea **goto**, implica una metodología de diseño descendente y limita las construcciones de control de la programación a ciclos **while** y proposiciones **if**. Es una técnica de construcción de programas que utiliza al máximo los recursos del lenguaje y presenta una serie de reglas que coordinan adecuadamente el desarrollo de las diferentes fases del programa.

La programación estructurada se basa en el Teorema de la Estructura o Antiguo Teorema de Bohn y Jacopini (1966), en el cual se demostró que cualquier programa con un solo punto de entrada y un solo punto de salida puede resolverse con tres únicos tipos de estructuras de control, a las cuales se les denominó secuencial, alternativa o ramificación y repetitiva.

Clases de lenguajes de programación

Se ha estado escribiendo lenguajes de programación desde que se desarrollaron las computadoras de propósito general, hace 40 años. Es conveniente agrupar los distintos lenguajes de programación en cuatro generaciones distintas :

1. *Lenguajes de primera generación.* Se codificaba a nivel de máquina. El código máquina y su equivalente más humanamente legible, el lenguaje ensamblador, representan la primera generación de lenguajes.
2. *Lenguajes de segunda generación.* Han servido como base para todos los lenguajes de programación modernos (tercera generación). Está caracterizada por su amplio uso, la enorme cantidad de bibliotecas de software y la gran familiaridad y aceptación.
3. *Lenguajes de tercera generación.* También denominados *lenguajes de programación moderna o estructurada*, están caracterizados por sus potentes posibilidades procedimentales y de estructuración de datos. Estos lenguajes se pueden dividir en tres amplias categorías, que son :
    - *Lenguajes de alto nivel de propósito general.*
    - *Lenguajes de alto nivel orientados a los objetos.*
    - *Lenguajes especializados.*

4. *Lenguajes de cuarta generación.* Contienen una sintaxis distinta para la representación del control y para la representación de las estructuras de datos. Un 4GL representa estas estructuras en un mayor nivel de abstracción, eliminando la necesidad de especificar los detalles algorítmicos. Los lenguajes de cuarta generación combinan características procedimentales y no procedimentales. Hay varios tipos de 4GL, que son :
    - *Lenguajes de petición.*
    - *Generadores de programas.*
    - *Otros 4GL.* Aunque los dos anteriores son los más comunes, existen otras categorías, que son :
        - *Lenguajes de soporte a la toma de decisiones.*
        - *Lenguajes de prototipos.*
        - *Lenguajes de especificación formal.*

Criterios para escoger un lenguaje de programación

Los lenguajes de programación son un vehículo de comunicación entre los humanos y las computadoras. El proceso de codificación - comunicación mediante un lenguaje de programación - es una actividad humana. Es por ello que las características psicológicas del lenguaje afectan directamente a la calidad de la comunicación. Las características de ingeniería de un lenguaje tienen un impacto importante sobre el éxito de un proyecto de desarrollo de software. Finalmente, las características técnicas de un lenguaje pueden influenciar la calidad del diseño, por lo cual éstas pueden afectar tanto a los aspectos humanos como a los de ingeniería de software.

Características psicológicas

Las características psicológicas aparecen como resultado del diseño de un lenguaje de programación. Aunque estas características no se pueden medir de forma cuantitativa, se reconoce su manifestación en todos los lenguajes de programación. Estas características se describen a continuación :
Uniformidad
Ambigüedad
Compacto
Localización
Linealidad
Tradición

Las características psicológicas de los lenguajes de programación afectan de forma importante a nuestra capacidad de aprenderlos, aplicarlos y mantenerlos.

Características de ingeniería

Un planteamiento de ingeniería del software sobre las características de los lenguajes se centra en las necesidades que puede tener un proyecto específico de desarrollo de software. Un conjunto general de características de ingeniería son:

1. *Facilidad de traducción del diseño al código.*
2. *Eficiencia del compilador.*

3. *Portabilidad del código fuente.*
4. *Disponibilidad de herramientas de desarrollo.*
5. *Facilidad de mantenimiento del código fuente.*

Características técnicas

Los fundamentos de los lenguajes de programación se presentan dentro del contexto de cuatro grandes áreas. Todos los lenguajes de programación se pueden caracterizar de acuerdo con estos aspectos y se puede juzgar la calidad de un lenguaje de programación específico evaluando la potencia o debilidad de cada uno de ellos.

Estas cuatro áreas son :

1. *Tipos de datos y tipificación de datos.*
2. *Subprogramas.*
3. *Estructuras de control.*
4. *Soporte para el enfoque orientado a los objetos.*

Evaluación y elección de un lenguaje de programación

Todas las características juntas deben definir qué es lo que hace a un lenguaje "bueno" en un sentido preciso. Estas características también incluyen los diferentes puntos de vista del programador, el diseñador del lenguaje y el implementador del lenguaje.

Algunos criterios para la evaluación y comparación de lenguajes son :

1. *Expresividad*
2. *Bien-definido*
3. Tipos y estructuras de datos
4. Modularidad
5. Facilidades de entrada-salida
6. Transportabilidad
7. Eficiencia
8. Pedagogía
9. Generalidad

Estos nueve criterios nos permiten evaluar cada lenguaje; no todos son cuantificables y concluyentes como el criterio de eficiencia, el cual puede ser medido en nanosegundos. El área de aplicación de un proyecto es el criterio que más se aplica durante el proceso de selección del lenguaje.

La elección de un lenguaje de programación para un proyecto específico debe tener en cuenta tanto las características de ingeniería como las psicológicas. Sin embargo, el problema asociado con la elección puede desaparecer si sólo se dispone de un lenguaje o si el cliente demanda uno en particular. Esta elección reduce al mínimo las dificultades de codificar un diseño, reduce la cantidad de pruebas de programas necesarias y hace al programa más legible y, por tanto, más fácil de mantener.

El lenguaje debe tener características de control y estructuración de los datos que permitan producir un programa legible.

Algunos criterios para la elección son :

1. *Los requisitos del contratista del sistema.*
2. *La disponibilidad de compiladores del lenguaje.*
3. *El tamaño del proyecto.*
4. *El conocimiento del personal de programación existente.*
5. *El lenguaje de programación utilizado en proyectos previos.*
6. *La necesidad de transportar el software.*
7. *La aplicación que se está programando.*
8. *Entorno en el que se ejecutará el software.*
9. *Consideraciones de rendimiento.*

Idealmente, los requisitos del software habrán de precipitar la elección de un lenguaje que mejor se ajuste al procesamiento que se ha de llevar a cabo.

Si un lenguaje se ve como "excelente" en todos (o la mayoría) de los criterios será probablemente un lenguaje de programación superior.

Relación de los lenguajes y el área de aplicación

	Areas	Lenguajes
1.-	Comerciales	Cobol, Rpg.
2.-	Científicas o Ingeniería	Fortran, Pascal, C, Algol.

Implementación

El objetivo es codificar en un lenguaje de programación de acuerdo a las especificaciones de diseño.

Características de un buen y mal programa

Sencillez, claridad y elegancia son los sellos de los buenos programas; oscuridad, ingeniosidad y complejidad son indicaciones de un diseño inadecuado y un pensamiento mal orientado.

La claridad del código fuente se mejora mediante técnicas de codificación estructurada, buen estilo de codificación, documentos adecuados de apoyo, buenos comentarios internos, y por las características que proporcionan los lenguajes de programación modernos.

Es deseable que todos los programadores de un proyecto adopten un estilo de codificación similar, de modo que se produzca un código de calidad uniforme.

II.8.- Documentación del software

Todos los grandes sistemas, con independencia de su aplicación, tienen una cantidad enorme de documentación asociada. Esta puede clasificarse como documentación del usuario o del sistema. La documentación del usuario se compone de aquellos

documentos relacionados con las funciones del sistema, sin referirse a la forma de aplicarlas. La documentación del sistema, por otra parte, describe todos los aspectos del diseño, implantación y pruebas del sistema.

La información proporcionada junto con el sistema debe satisfacer varios requisitos, por ejemplo:

1. *Cómo usar el sistema. Sin esto aún el sistema más simple resulta inútil.*
2. *Cómo instalar y operar el sistema.*
3. *Requisitos y diseño de todo el sistema.*
4. *La aplicación del sistema y los procedimientos de prueba, para poderle dar mantenimiento.*

La documentación proporcionada con un sistema puede ser útil en cualquier etapa del tiempo de vida de éste. No necesariamente debe producirse en el mismo orden que el sistema mismo. De hecho, a menudo se requiere durante la especificación del sistema tener la información del usuario disponible, de modo que el especificador conozca las restricciones dentro de las que debe operar.

El documento de instalación debe proporcionar detalles completos sobre la manera de instalar el sistema en un ambiente particular. En primer lugar, debe contener una descripción del medio legible para la máquina en la que se proporcione el sistema: su formato, los códigos de caracteres utilizados, cómo está escrita la información y los archivos que componen el sistema. Después, debe describir la configuración mínima de hardware que se requiere para ejecutar el sistema, los archivos permanentes que deben estsblecerse, cómo iniciar el sistema y los archivos dependientes de la configuración que se deben modificar para adecuar el sistema a una aplicación particular.

II.9.- Pruebas y Confiabilidad de los sistemas

La prueba del software es un elemento crítico para la garantía de calidad del software y representa la revisión final de las especificaciones, del diseño y de la codificación.

La creciente inclusión del software como un elemento más de muchos sistemas y la importancia de los "costos" asociados a un fallo del mismo están motivando la creación de pruebas minuciosas y bien planificadas. En casos extremos, la prueba del software para actividades críticas puede costar ¡de tres a cinco veces más que el resto de los pasos de la ingeniería del software juntos!

Pruebas

La prueba de los programas es la parte del proceso de confirmación que suele realizarse durante la aplicación y también, en una forma distinta, cuando éste se ha terminado. La prueba consiste en ejercitar el programa utilizando datos similares a los datos reales que habrán de ser ejecutados por el programa, observar los resultados y deducir la existencia de errores o insuficiencias del programa a partir de las anomalías de ese resultado.

Objetivos de la prueba

Se establecen una serie de reglas que sirven acertadamente como objetivos de prueba:

1. *La prueba es un proceso de ejecución de un programa con la intención de descubrir un error.*
2. *Un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces.*
3. *3. Una prueba tiene éxito si descubre un error no detectado hasta entonces.*

El objetivo es diseñar pruebas que sistemáticamente saquen a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo.

Los datos que se van recogiendo a medida que se lleva a cabo la prueba proporcionan una buena indicación de la fiabilidad del software y, de alguna manera, indican la calidad del software como un todo.

Se llevan a cabo las pruebas y se evalúan los resultados. O sea, se comparan los resultados de la prueba con los esperados. Cuando se descubren datos erróneos, ello implica que hay un error y comienza la depuración. El proceso de depuración es la parte más impredecible del proceso de prueba. Un "error" que indique una discrepancia de un 0.01 por 100 entre los resultados esperados y los reales puede llevar una hora, un día o un mes de diagnóstico y corrección.

Finalmente, si la prueba no descubre errores, quedará la sospecha de que no se ha pensado cuidadosamente la configuración de prueba y de que los errores están escondidos en el software.

Verificación y Validación

La prueba es un conjunto de actividades que se pueden planificar por adelantado y llevar a cabo sistemáticamente. Los objetivos de las actividades de verificación y validación son valorar y mejorar la calidad de los productos del trabajo generados durante el desarrollo y modificación del software.

La prueba de software es un elemento de un concepto más amplio que, a menudo, se referencia como verificación y validación (V&V). La verificación se refiere al conjunto de actividades que aseguran que el software implementa correctamente una función específica. La validación se refiere a un conjunto diferente de actividades que aseguran que el software construido se ajusta a los requisitos del cliente.

Hay dos tipos de verificación : formal y del ciclo de vida. Esta última consiste en el proceso de determinar el grado en que los productos del trabajo de una fase dada del ciclo de desarrollo cumplen con las especificaciones establecidas durante las fases previas. La verificación formal es una rigurosa demostración matemática de la concordancia del código fuente con sus especificaciones.

Es importante darse cuenta de que la verificación y la validación abarcan un amplio rango de actividades de la garantía de calidad del software SQA que incluye revisiones técnicas formales, auditorías de configuración y calidad, supervisión del rendimiento, simulación, estudios de viabilidad, revisión de la documentación, revisión de la base de

datos, análisis de algoritmos, prueba de desempeño, prueba de [illegible] instalación. Aunque la prueba desempeña un papel [illegible] V&V, también son necesarias muchas otras actividades

Certificación

La certificación es un modelo planeado y sistemático de todas las [illegible] para proporcionar la confianza de que el artículo se ajusta a los [illegible] establecidos. Implica la planeación y la ejecución de pruebas [illegible] desempeño y de tensión para demostrar que el sistema [illegible] requisitos.

Tipos de pruebas

Cualquier [illegible] para un sistema de [illegible] concurrente a fondo, sistemática para [illegible] a cada [illegible] que terminar [illegible] concurrente, [illegible] asegurar que "cosas [illegible] especificaciones y que [illegible] existente. [illegible]

Caja negra

[illegible]

Caja blanca

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

datos, análisis de algoritmos, prueba de desarrollo, prueba de calificación y prueba de instalación. Aunque la prueba desempeña un papel extremadamente importante en V&V, también son necesarias muchas otras actividades.

Certificación

La certificación es un modelo planeado y sistemático de todas las acciones necesarias para proporcionar la confianza de que el artículo se ajuste a los requisitos técnicos establecidos. Implica la planeación y la ejecución de pruebas funcionales, de desempeño y de tensión para demostrar que el sistema implantado satisface los requisitos.

Tipos de pruebas

Cualquier producto de ingeniería puede ser probado de una de dos formas: (1) conociendo la función específica para la que fue diseñado el producto, se pueden llevar a cabo pruebas que demuestren que cada función es completamente operativa; (2) conociendo el funcionamiento del producto, se pueden desarrollar pruebas que aseguren que "todas las piezas encajan"; o sea, que la operación interna se ajusta a las especificaciones y que todos los componentes internos se han comprobado de forma adecuada. El primer enfoque de prueba se denomina *prueba de la caja negra* y la segunda *prueba de la caja blanca.*

Caja Negra

Cuando se considera el software de computadora, la prueba de la caja negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del software. O sea, los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de la información externa se mantiene. Una prueba de la caja negra examina algunos aspectos del modelo fundamental del sistema sin tener mucho en cuenta la estructura lógica interna del software.

Caja Blanca

La prueba de la caja blanca del software se basa en el minucioso examen de los detalles procedimentales. Se comprueban los caminos lógicos del software proponiendo casos de prueba que ejerciten conjuntos específicos de condiciones y/o bucles. Se puede examinar el "estado del programa" en varios puntos para determinar si el estado real coincide con el esperado o afirmado.

Prueba del Camino Básico

La prueba del camino básico es una técnica en que los casos de prueba derivados del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa.

Prueba de la Estructura de Control

La técnica de prueba del camino básico descrita anteriormente es una de muchas técnicas para la *prueba de la estructura de control.* Aunque la prueba del camino básico es sencilla y altamente efectiva, no es suficiente por sí sola.

Prueba de Condiciones

La prueba de condiciones es un método de diseño de casos de prueba que ejercita las condiciones lógicas contenidas en el módulo de un programa. Una *condición simple* es una variable lógica o una expresión relacional.

Prueba de Bucles

La prueba de bucles es una técnica de prueba de la caja blanca que se centra exclusivamente en la validez de las construcciones de bucles. Se pueden tener cuatro clases diferentes de bucles :

Bucles simples.
Bucles anidados.
Bucles concatenados.
Bucles no estructurados.

Generadores de datos de prueba

Los generadores de datos de prueba son programas que generan automáticamente una gran cantidad de entradas de prueba para un sistema. Lamentablemente los generadores de datos de prueba no pueden producir los resultados correspondientes, pues en tal caso serían equivalentes al programa que se está probando.

Los generadores de datos de prueba son más útiles para situaciones en las cuales se debe probar el funcionamiento de un sistema en un ambiente práctico.

Analizadores estáticos.
Auditores del código.
Procesadores de asertos.
Generadores de archivos de prueba.
Verificadores de prueba.
Controles de prueba.
Comparadores de salida.
Sistemas de ejecución simbólica.
Simuladores de entorno.
Analizadores de flujo de datos.

Niveles de correctividad

La corrección aparece como una consecuencia de una prueba efectiva, o sea, cuando un caso de prueba descubre un error, es el proceso que resulta de la eliminación del error. El proceso comienza con la ejecución de un caso de prueba. Se evalúan los resultados y aparece una falta de correspondencia entre los datos esperados y los reales y lo que se pretende es corresponder el sistema con una causa, llevando así a la corrección.

Plan de pruebas

El proceso de prueba sigue estas cinco etapas:

1. Prueba de funciones. Las funciones y procedimientos que componen un módulo se prueban para asegurar su operación correcta.

2. Prueba de módulos. Las funciones se combinan para formar un módulo. Este módulo se prueba para asegurar que su desempeño cumpla con las especificaciones.

3. Prueba de integración. En ocasiones denominada prueba de subsistemas o en cadena, aquí se integran un conjunto de módulos, la prueba se concentra en las interfaces de los módulos debido a que se supone que los módulos mismos son correctos.

4. Prueba total de los programas. En ocasiones denominada prueba de sistemas, este nivel de prueba combina todos los subsistemas en un paquete total de programas. En esta etapa el proceso de prueba se concentra más en la detección de los errores de diseño y en determinar qué tan bien se cubren los requerimientos de los usuarios.
5. Prueba de aceptación. El enfoque de la prueba de aceptación está en qué tan bien funciona el programa con datos reales en el ambiente de operación, su conformidad con las especificaciones de diseño, y su capacidad para satisfacer los requerimientos de los usuarios. La prueba de aceptación normalmente es realizada por los usuarios o los representantes de los usuarios. Si el programa pasa esta prueba, entonces es aceptado por los usuarios y se lleva a su operación plena.

II.10 Instalación y Mantenimiento de los Sistemas

Plan de Instalación

La instalación de un sistema de información tiene que estar bien definida, pues abarca el equipo físico en el que se va a instalar el sistema, el espacio de suelo para el equipo físico, las necesidades del sistema operativo sobre el que va a trabajar, detalles en cuanto a la capacidad mínima de disco duro para poderse instalar, la capacidad mínima de memoria, si debe contener algunos dispositivos periféricos adicionales, como el MODEM, o tarjeta de red, si se requiere de un servidor, cuántas máquinas están consideradas como mínimo, identificar cuantos operadores van a manejar el sistema y la ubicación geográfica de la misma.

Plan de Mantenimiento

El término de mantenimiento de software se usa para describir las actividades de la ingeniería de software que ocurren después de entregar un producto al cliente. Las actividades de mantenimiento implican mejorar los productos de software, adaptarlos a nuevos ambientes y corregir problemas. Esto proporciona nuevas capacidades funcionales, mejorar los despliegues al usuario y los modos de interacción, revalorar los documentos externos y la documentación interna, o revalorar las características del desempeño de un sistema.

Como no existe un solo tipo de mantenimiento para el equipo de cómputo, consideramos pertinente mencionar cuatro categorías principales de mantenimiento:

- ***Mantenimiento correctivo*** (mantenimiento de reparación). Involucra la corrección de errores en una parte del software.

- ***Mantenimiento adaptativo.*** Involucra la alteración de un programa para traerlo a línea con los cambios en su especificación. Tales cambios pueden resultar de nuevos requerimientos del usuario o de un cambio en el ambiente del sistema operacional.

- ***Mantenimiento perfectivo*** (mantenimiento productivo). Este no altera ni la especificación ni la adherencia del software a él, pero mejora el desempeño al hacer que el software consuma menos recursos (o al menos, reducir el costo general de los recursos que consume).

- ***Mantenimiento preventivo.*** Esta categoría final dirige sus beneficios hacia el proceso general del mantenimiento mismo. Implica hacer cambios al software, que por si mismos, no mejoran ni la corrección ni el desempeño, pero provocan que las actividades futuras de mantenimiento sean más fáciles de llevar a cabo.

Capacitación

Una vez que se ha decidido implantar el sistema, se debe planear cómo se dará a conocer a la administración y a todo el personal de la organización que se involucren con el mismo. Es importante señalar que cuando damos a conocer un sistema debe ser en forma sencilla, puntualizando la naturaleza, las metas y los beneficios, la forma de operación y los programas propuestos de capacitación y adiestramiento del personal.

Los programas de capacitación variarán en tiempo y contenido, dependiendo de a quién vayan dirigidos, así pues tenemos que para los gerentes de área o para algunos elementos de la administración el programa de capacitación será corto, mientras que el personal operario de los sistemas tomará cursos más prolongados donde se explicarán detalladamente todas las características del sistema, sus entradas, sus salidas, la relación con otros sistemas, el mantenimiento de archivos, etc.

Carga de Archivos

Lo constituye el preparar datos y disponer de los archivos maestros del sistema. La creación de estos archivos constituye una tarea que consume mucho tiempo. No sólo es necesario que los datos se conviertan a un formato aceptable al nuevo sistema, sino también que los analistas garanticen que esto se lleva a cabo sin perder detalles o precisión.

Aprobación Final

Una vez que el sistema ha pasado las pruebas y exámenes que se le han aplicado y se ha comprobado que satisface eficientemente los requerimientos de los usuarios, se puede decir entonces que el sistema está posibilitado para poder ser implantado en el equipo de cómputo para su operación y manejo. Solo resta aclarar a los usuarios los cambios que ha sufrido el sistema y el efecto que tendrán en el uso de la información.

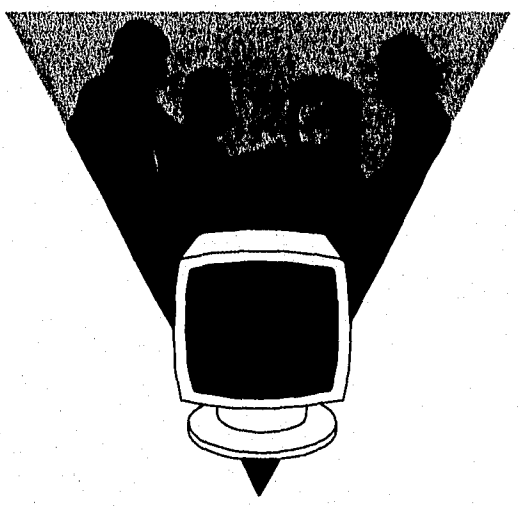
Identificación de Resultados y Desviaciones

Los errores típicos que se presentan una vez concluido el programa son en cuestiones de diseño, en aspectos lógicos y en el formato en el que fueron capturados los datos.

La corrección de errores normalmente envuelve la modificación de una o más declaraciones en el código del programa. Por lo tanto, si existen modificaciones en el módulo de origen (programa fuente), éste debe ser recompilado, religado y reejecutado para verificar si efectivamente se corrigieron los errores y los resultados obtenidos son los adecuados.

En muchas ocasiones suele suceder que los errores en un programa se encuentran en cierto módulo, que a su vez producen un resultado que sirve de entrada a otros módulos y esto va acarreando un error a lo largo de la corrida de un programa. En este caso el correr el programa por pasos nos permite realizar un análisis meticuloso del funcionamiento del mismo en cada instrucción, pudiendo observar el procesamiento de un dato y su variación dentro de un módulo.

Es muy importante utilizar datos en los procesos del programa, en los cuales el resultado ya se sabe de antemano, para poder verificar si efectivamente el programa está funcionado de la forma para la que fue diseñado y si no es así para realizar un nuevo análisis de diseño. El usuario puede necesitar prestaciones adicionales al sistema, de tal forma que deben realizarse modificaciones.



CAPITULO III

ANALISIS

En el capítulo anterior se describieron varias metodologías que se podrían utilizar en el desarrollo de nuestro sistema. Entre ellas hemos seleccionado la que se aplicará a nuestro problema. También estamos considerando herramientas de los lenguajes de cuarta generación, por lo que es posible aplicar la metodología de una manera mucho más sencilla.
La etapa de análisis empieza estableciendo los requisitos de todos los elementos del sistema. El proceso de recopilación de la información se centra e intensifica especialmente para el software, puesto que se debe comprender la naturaleza de los programas a construir, así como su función, rendimiento e interfaces necesarias.

De acuerdo a lo anterior, se desarrollan los siguientes puntos:

1. Descripción de la empresa
   - Organigrama general de la empresa.
   - Giro y Actividades de la empresa.
2. Delimitación del área organizacional
   - Organigrama del área organizacional.
   - Objetivos y funciones de las áreas.
   - Diagrama de flujo de datos.
   - Descripción de los procesos del diagrama de flujo de datos.
   - Areas relacionadas con el sistema.
3. Análisis del sistema actual
4. Análisis de problemas y oportunidades
5. Selección y definición de los subsistemas prioritarios
6. Descripción del sistema propuesto
7. Análisis costo / beneficio
8. Características principales de los manejadores de bases de datos
9. Programa de actividades

III.1.- Descripción de la Empresa Estafeta Mexicana

En este punto procederemos a hacer un breve análisis de la estructura organizacional de la empresa como sus objetivos y funciones, además de describir los servicios que ofrece la misma.

Organigrama general de la empresa

Estafeta Mexicana es una sociedad anónima perteneciente a un grupo de varios accionistas mexicanos y un accionista mayoritario de nacionalidad alemana. En el escalafón más alto tenemos ubicado al presidente de la empresa, respaldado a su vez por su consejo administrativo, y al Director General, ver figura 3.1. Todas las decisiones referentes a la empresa las toman el presidente, el consejo administrativo y los directores.

Dada la extensión que alcanzan los servicios de Estafeta y las dimensiones que tiene actualmente la empresa, ésta se encuentra organizada en cinco distritos, que abarcan todo el país de la siguiente manera:

- Distrito 1.- Nuevo León, Chihuahua, Durango, Tamaulipas, San Luis Potosí, Zacatecas, Aguascalientes, Guanajuato, Querétaro, Michoacán, Colima, Nayarit, Coahuila, Hidalgo.
- Distrito 2.- D.F. , Edo. de Méx., Morelos.
- Distrito 3.- Puebla, Guerrero, Oaxaca, Chiapas, Veracruz, Tabasco, Campeche, Yucatán, Quintana Roo, Tlaxcala.
- Distrito 4.- Corporativo Administrativo (No refleja ventas).
- Distrito 5.- Jalisco, Sinaloa, Sonora, Baja California Norte, Baja California Sur.

El organigrama correspondiente a Estafeta Mexicana se presenta en la siguiente figura:

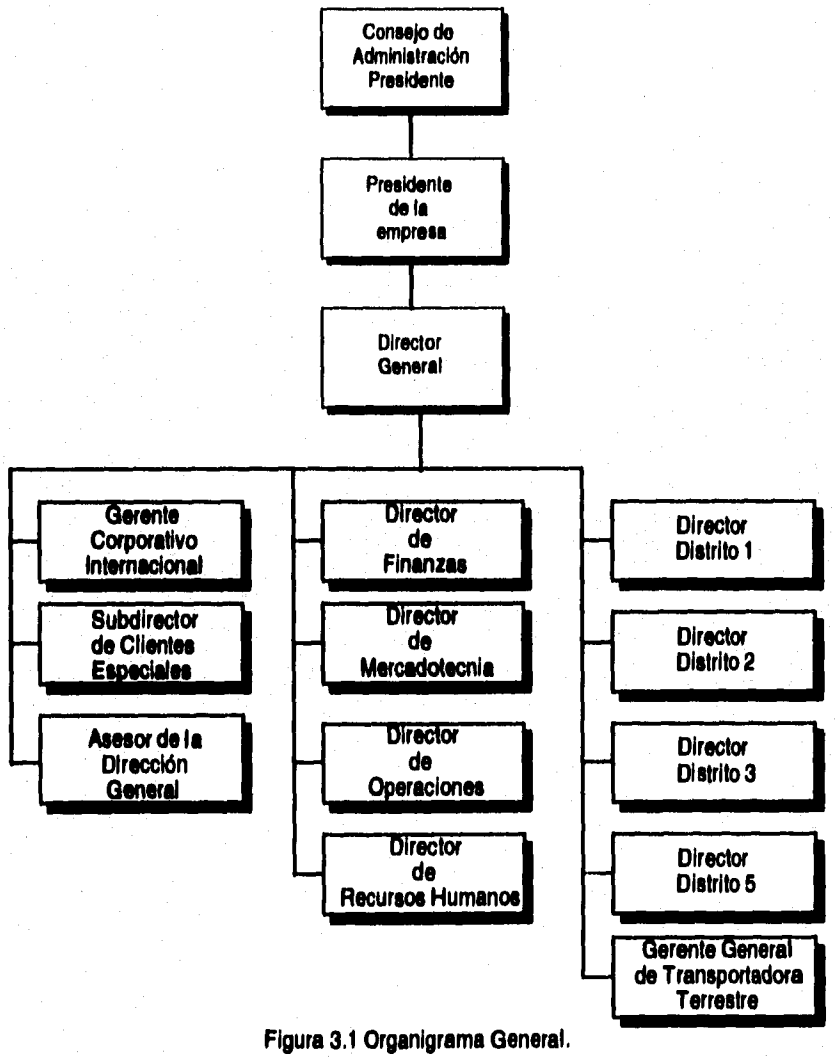


Figura 3.1 Organigrama General.

Debido a esta estructura se tienen 4 directores de distrito comandados por el director general. Cada distrito a su vez se divide en regionales, localidades y plazas, con sus consiguientes directores regionales y gerentes de plaza.

El Corporativo Administrativo de Estafeta posee cuatro Direcciones: de Finanzas, Mercadotecnia, Operaciones y Recursos Humanos. Cada una de ellas con sus respectivas subdirecciones y gerencias de área.

Dentro del distrito 2, al que pertenecen las oficinas de la ciudad de México, se encuentra ubicado el centro operativo de Av. Ceylán, que es el mayor centro de distribución de la república. Aquí se ubica la gerencia de administración, a la cual pertenece el área de facturación de sobrepeso.

Cabe mencionar que Estafeta tiene dos áreas de facturación, ya que se factura el costo del servicio y el sobrepeso que éstos generan de manera independiente, sin olvidar claro, que existe una comunicación constante entre ellas. El área de facturación de los servicios se ubica en las instalaciones de la sucursal de Aeropuerto, mientras que, como se vio anteriormente, la facturación del sobrepeso se lleva en la sucursal de Ceylán. Una vez que Ceylán genera su factura la envía a Aeropuerto, para que ahí se le de el seguimiento administrativo que le corresponde.

Giro y actividades de la empresa

Estafeta Mexicana, S.A. de C.V., funge como empresa desde el 8 de Agosto de 1979. Su giro es el servicio de mensajería y paquetería, con una cobertura total en el país, con 500 oficinas y más de 1,500 vehículos que garantizan agilizar el canal de ventas, optimizar inventarios y reducir substancialmente costos de los usuarios. Sus objetivos y metas son:

- Ofrecer el servicio líder en mensajería y paquetería nacional.
- La satisfacción del cliente.
- Que cada envío llegue bien y a tiempo.
- Confiabilidad, calidad y rapidez.

Por otra parte, Estafeta Mexicana posee una infraestructura con Centros Operativos y de Cómputo de alta tecnología, transporte, rastreo electrónico, para conocer con una llamada telefónica el status del envío incluyendo fecha, hora y nombre de la persona que lo envió. También tiene procesos totalmente automatizados de clasificación, embarque, transbordo, registro, control y seguridad en la distribución de productos.

En la Tabla 3.1 se muestran los diferentes tipos de servicios que la empresa proporciona. Estos se pueden clasificar en dos tipos, los que generan sobrepeso (facilito y programado), y los que no son susceptibles de considerarlo; esto es por algunas políticas que la compañía ha definido y que a continuación se enuncian:

a) Relación Gasto-Beneficio (monto sobrepeso menor 50.00 pesos).
b) Contratos con compañías que demanden gran cantidad de envíos.
c) Consideraciones extraordinarias a convenios con compañías (Clientes especiales).
d) Convenios entre servicios (Estafeta vs Empresas) denominados intercambios.

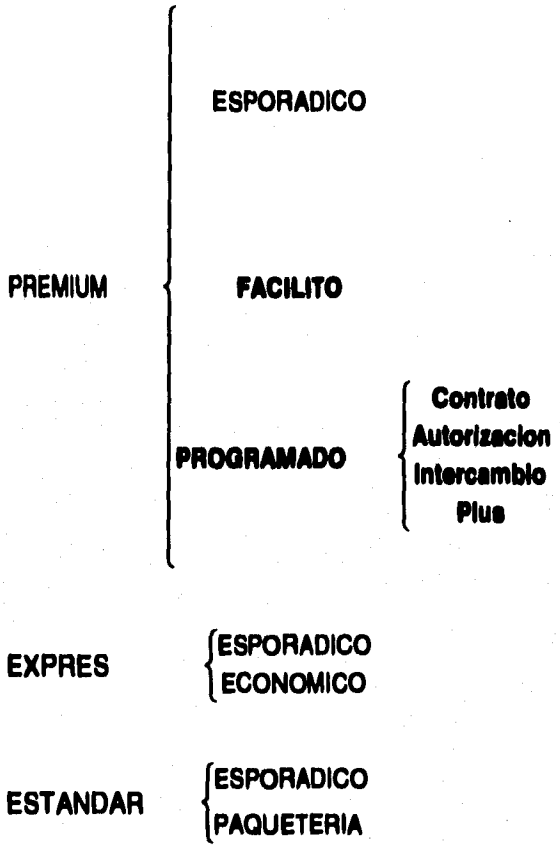


Tabla 3.1 Servicios de mensajería.

A continuación describiremos cada uno de los servicios :

Premium

Es el más rápido de los servicios que se ofrecen, con la garantía más sólida de la industria. Cuenta con servicio de recolección y ofrece tres modalidades de servicio:

Esporádico : Para necesidades eventuales.
Facilito : Para necesidades frecuentes, gozando de descuento por volumen.
Programado : Para envíos con fechas y destinos predeterminados.
Contrato : Servicio de entregas prepagadas y calendarizadas.
Intercambio : Servicio basado en convenios con empresas. Por ejemplo, empresa X (publicidad) realiza propaganda a Estafeta a cambio de servicios de mensajería.
Autorización: Servicio basado en una autorización especial de funcionarios de Estafeta, sobre convenios con empresas para el manejo de la paquetería, con servicios de rapidez y eficiencia especial.
Plus: En este servicio se cuenta con un tiempo de entrega de menos de 24 hrs. por lo cual lo hace el más rápido.

Exprés

Es el servicio con la mejor relación entre precio y rapidez de entrega en el mercado. Los envíos se pueden depositar en cualquiera de las oficinas y cuenta con dos modalidades :

Esporádico : Para necesidades eventuales.
Económico : Para necesidades frecuentes, gozando de descuento por volumen.

Estándar

Es el canal más eficiente para la distribución de productos. Permite abatir costos y aumentar las ventas, mediante la rotación óptima de sus inventarios. Los envíos se pueden depositar en cualquiera de las oficinas y los grandes volúmenes se reciben en los Centros Operativos. Ofrece dos modalidades de servicio :

Esporádico : Para necesidades eventuales.
Paquetería : Ideal para la distribución de productos.

Todos los servicios descritos anteriormente se controlan por etiquetas denominadas ***guías*** figura 3.2; en éstas se recaba la siguiente información: Datos del cliente, Datos del destinatario, Tipo de servicio, Plaza destino, Descripción y peso en Kg. del paquete, Fecha de envío, Ruta de recolección y un Número de folio que es consecutivo y único.

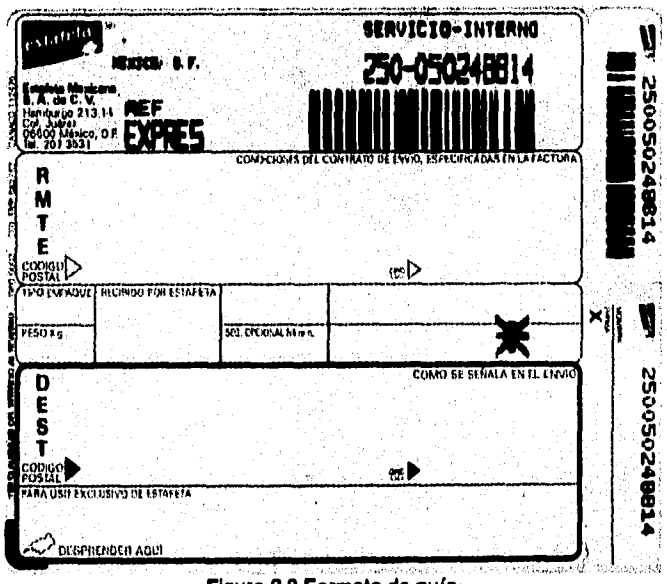


Figura 3.2 Formato de guía.

III.2 Delimitar área de facturación de sobrepeso

En este punto describiremos el área de facturación de sobrepeso en la cual, debido al estudio que realizamos, detectamos el problema que da origen a nuestro tema de tesis.

Organigrama del área de facturación de sobrepeso

En la figura 3.3 se presenta el organigrama que describe al área de facturación. La ubicación física de esta área es en la sucursal de Av. Ceylán y está compuesta por siete personas: el Director, el Gerente Administrativo, el Jefe de Facturación y 4 Operadores.

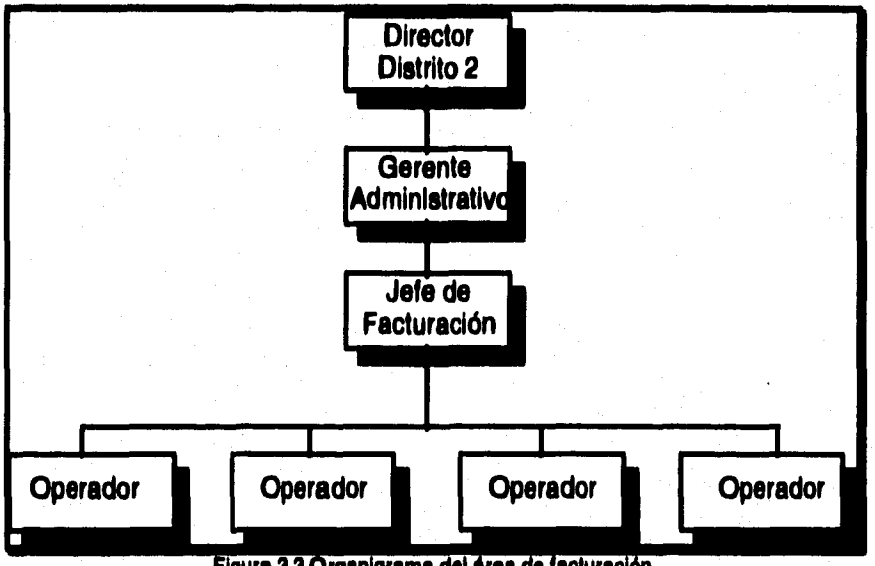


Figura 3.3 Organigrama del área de facturación.

Objetivos y funciones del área de facturación de sobrepeso

Los objetivos y funciones del área de facturación de sobrepeso son los siguientes:

Objetivo.- Facturar correcta y oportunamente el sobrepeso de todos los servicios que lo generan.

Funciones:

- Alimentar la base de clientes con datos correctos para la emisión de sus facturas (alrededor de 5,000 clientes).
- Solicitar al departamento de control de facturación el número de cliente para su captura.
- Captura de guías tanto de México como de provincia con sobrepeso que no pasan por la banda de distribución y por lo tanto no son registradas por los lectores ópticos del sistema de básculas.
- Elaboración de refacturaciones de sobrepeso por actualización de datos.

Diagrama de flujo de datos

En la figura 3.4 podemos ver como se da el flujo de datos entre las áreas participantes del sistema de facturación de sobrepeso.

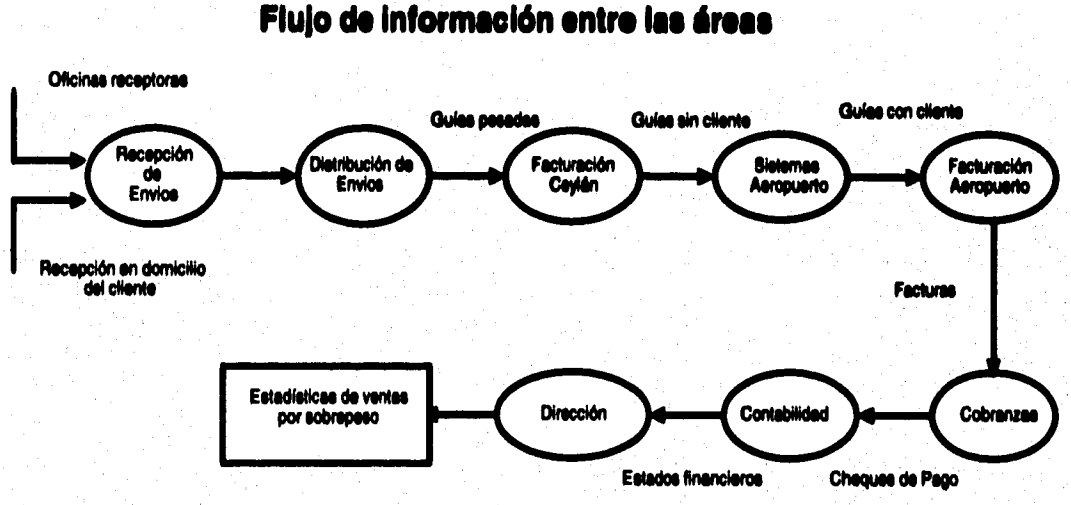


Fig. 3.4 Diagrama de flujo de datos.

Areas relacionadas con el sistema

El cobro del sobrepeso involucra varias áreas de la empresa, dado el complejo proceso administrativo que éste implica. En seguida mencionamos cada una de las áreas involucradas en el cobro del sobrepeso y a grandes rasgos, las funciones que realizan.

- Recepción de Envíos.- Se encarga de la recopilación de los paquetes del cliente. Esta recopilación puede ser en oficinas receptoras o en el domicilio mismo del cliente.
- Distribución de Envíos.- Realiza el pesado de los paquetes, recopilación de sus datos (incluyendo sobrepeso), y distribución a los diferentes centros de la república.
- Facturación Ceylán.- Ubicada en las instalaciones de Av. Ceylán, es responsable directa del control del sobrepeso y su facturación.
- Sistemas.- Realiza búsqueda de los clientes generadores de sobrepeso en la base de producción de Estafeta.
- Facturación Aeropuerto.- Lleva a cabo la recepción de todas las facturas a nivel nacional para su control y registro en la base de producción de Estafeta.
- Crédito y Cobranzas.- Da el seguimiento necesario para el cobro de las facturas.
- Contabilidad.- Lleva el control de los montos que se cobran por concepto de sobrepeso. Emite los reportes mensuales de ventas (estados financieros).
- Dirección.- Evalúa los estados financieros y controla las estadísticas de ventas por concepto de sobrepeso.

III.3.- Análisis del Sistema Actual

A continuación describiremos el proceso de cobro del sobrepeso, que incluye los siguientes pasos:

1. Adquisición de las guías. El precio de éstas es de acuerdo al servicio requerido y a la cantidad de guías. Cabe mencionar que cada guía ampara entre 1 y 3 kg., por lo que el sobrepeso varía desde 1 hasta 45 kg. dependiendo del tipo de servicio, como se muestra en la tabla 3.2. Por ejemplo, si se adquieren 100 guías del servicio Facilito en Zacatecas (distrito 1), se pagará por cada una, $44.00 M.N. y al utilizar una guía para enviar un paquete a Monterrey (distrito 1) y éste pesa 10 kg. entonces se pagará $8.00 x 9kg. de sobrepeso que dará un total de $72.00 M.N., pero si se enviara a Acapulco (distrito 3) o por vía aérea el mismo paquete, entonces se pagaría $108.00 M.N. de sobrepeso.

Servicio	Peso (kg.)	Cantidad de guía	Precio ($)	Sobrepeso por cada 1/2 kg. adicional	
				misma zona ($)	entre zona y/o aéreo ($)
Facilito	1 kg. por cada guía	20 - 50	46.00	4.00	6.00
		51 - 100	44.00	4.00	6.00
		101 - 250	43.00	4.00	6.00
		251 - 500	42.00	4.00	6.00
		501-1000	40.00	4.00	6.00
		1001-2500	37.00	4.00	6.00
		2501-5000	35.00	4.00	6.00
		5001-10000	31.00	4.00	6.00
		10001- en adelante	29.00	4.00	6.00
Programado	3 kg. por cada guía en envíos terrestres y 1 kg. por cada guía en envíos aéreos.	Volumen semanal			
		1 - 3	39.00	4.00	6.00
		4-10	37.00	4.00	6.00
		11 - 45	34.00	4.00	6.00
		46 - en adelante	32.00	4.00	6.00

Tabla 3.2 Precios de guía de acuerdo al servicio y volumen.

2. El cliente al tener su guía y paquete los deposita en los centros de recepción que se encuentran ubicados en diversos establecimientos.

3. Posteriormente son transportados al Centro Operativo de Distribución.

4. En este lugar son pesados los paquetes y se toma el registro de la guía, de donde se obtienen los siguientes datos:
   - Número de la guía
   - Origen
   - Destino
   - Sobrepeso
   - Fecha destino
   - Hora destino
5. Estos datos son enviados, al término del día, por el área de distribución al área de facturación.

6. El área de facturación almacena los datos en una base de datos auxiliar, en donde quincenalmente se hace un corte y depuración. Además se generan las siguientes actividades:

   - Envío de la base auxiliar al área de sistemas.
   - Procesamiento de la base auxiliar y obtención de archivo tipo ASCII, el cual contiene el número de la guía como dato principal.
   - Del archivo se realiza la lectura de las guías y se busca su respectivo cliente en la base de facturas.
   - Se imprimen los reportes de guías encontradas (formato 1) y guías no encontradas (formato 2).
   - Envío de reportes al área de facturación.
   - Búsqueda vía telefónica de los clientes a los que corresponden las guías no encontradas.
   - Captura de guías encontradas. Esta captura realiza el cálculo automático del monto por el sobrepeso.
   - Generación de borrador de facturas para su validación.
   - Emisión de facturas.
   - Envío de facturas al área de cobranzas.

III.4- Análisis de problemas y Oportunidades

Una vez que hemos analizado el proceso del sistema actual, es necesario entrar en la problemática que conlleva, ya que es ésta la que da origen a nuestro proyecto de tesis.

Para el año de 1993, con el fin de manejar de manera más eficaz su información, Estafeta Mexicana realiza una fuerte inversión en la adquisición de equipos servidores, así como de un manejador de bases de datos capaz de manejar gran cantidad de información generada a nivel nacional. El resultado fue la utilería canadiense llamada SENTAI, que tiene como manejador de bases de datos a PROGRESS, cuyas características de Lenguaje de Cuarta Generación hicieron posible el manejo de información más rápida y confiable en otras áreas de la compañía, no así en el área de facturación.

Problemas

En el área de facturación se utiliza un sistema desarrollado en lenguaje FOXPLUS, que es un lenguaje de tercera generación orientado al desarrollo de Archivos de Datos, haciendo hincapié que este tipo de lenguajes no manejan el concepto de bases de datos relacionales y además su desarrollo no siguió ningún esquema de diseño. Maneja una base de datos de clientes independiente a la utilizada en el corporativo de Estafeta que, como ya se mencionó, maneja su información con la utilería llamada SENTAI cuyo manejador de bases de datos es PROGRESS, ocasionando que existan incongruencias en los datos de los clientes. Una consecuencia más grave aún de esto, es el hecho de que debido a las diferencias en las bases donde se guarda la información de clientes, es frecuente que no se sepa a que cliente pertenece determinada guía, haciendo imposible el cobro del sobrepeso.

Asimismo, cada mes se tiene la necesidad de realizar una exportación de información referente a facturas del sistema que maneja facturación, para posteriormente importarlas al sistema que se tiene en el corporativo, debido a las diferencias en ambos sistemas. Lo anterior ocasiona, entre otras cosas, problemas de duplicidad de información, pérdida de tiempo, alta posibilidad de pérdida de información, mala utilización de recursos, tanto humanos como materiales, además de pérdidas mensuales de aproximadamente $400,000.00.

Oportunidades

Con la realización de un nuevo sistema con la utilería del corporativo (SENTAI) y que además utilice las mismas estructuras en sus bases de clientes y facturas que tiene corporativo, se podrá eliminar la inconsistencia de información. De igual forma, su integración a la base será de manera automática. Se pretende que este nuevo sistema sea instalado en los tres principales centros operativos de la empresa que son: México, Guadalajara y Monterrey, existiendo un flujo e intercambio de información entre éstos vía módem.

Esto traerá como consecuencia la agilización de procesos, permitiendo localizar el cliente de un mayor número de guías y la entrega oportuna de facturas para su cobro. Con esto el ingreso mensual por concepto de cobro de sobrepeso aumentará de una manera considerable. De esta forma el área de facturación contribuirá a que la empresa conserve el liderazgo en México dentro del ramo de mensajería y paquetería que obtuvo en 1995.

III.5.- Selección y definición de los subsistemas prioritarios

El proceso del cobro de sobrepeso tiene cuatro partes primordiales, de las cuales surgen todas las actividades. Estas partes son las siguientes:

1. Recopilación de guías.
2. Búsqueda de clientes de las guías.
3. Generación y revisión de facturas.
4. Impresión de facturas.

Estas partes requieren de los siguientes subsistemas para su realización:

1. Proceso de carga automática de guías.
2. Proceso de captura manual de guías.
3. Proceso de Interface.
   - Generación de archivo ASCII en facturación Ceylán.
   - Carga de archivo ASCII y búsqueda de clientes en facturas en sistemas aeropuerto.
   - Envío de archivos de guías encontradas y no encontradas a facturación Ceylán.
4. Captura de clientes a las guías no encontradas en aeropuerto.
5. Generación de reporte previo de facturas y validación de información.
6. Impresión de facturas.

III.6.- Descripción del sistema propuesto

Debido al crecimiento que Estafeta tuvo en cuanto a sus servicios y para tener un mayor control sobre el manejo de facturación de sus clientes, se plantea el siguiente sistema cuyo objetivo es ayudar a cubrir y solucionar todas sus necesidades. La base de éste será la instalación de un servidor más poderoso con un manejador de bases de datos relacionales, que permita dar solución a la problemática de la empresa, dando una mejor relación entre los principales centros de operación.

Se utilizará un lenguaje de cuarta generación que permita hacer de una manera más rápida y confiable el proceso de facturación. Se pretende utilizar la utilería (SENTAI) cuyo manejador sea PROGRESS. El proceso pretende utilizar la misma base de clientes que tiene el corporativo, al igual que se seguirá utilizando la estructura de las facturas que se han venido manejando, para que, al ser enviadas, su integración a la base sea automática. Así será posible el intercambio y el flujo de información entre los centros operativos.

El procedimiento del sistema propuesto es aplicable dentro del área de facturación de sobrepeso. Este constará de los siguientes pasos:

1. Un operador será el encargado de identificar los paquetes que generen sobrepeso por el tipo de servicio, éstos pasan por el sistema de básculas donde se confirma si el paquete está excedido en peso.

2. Al otorgarse las guías de un servicio cuyo pago no es inmediato (Programado Plus, Intercambio, Autorización, etc.), el responsable de la venta registrará los datos del cliente, así como el rango de guías entregadas. Posteriormente el operador del sistema en Ceylán capturará la información que permita la identificación del envío y de la guía:

   - Núm-cliente
   - Guía-inicial
   - Cantidad
   - Fecha de Captura

   esta información se guardará en alguna tabla, que se podría definir como la **tabla guías-cll**.

El sistema se encargará de generar un registro único con esta información, además de asignar la fecha en que el usuario la capturó.

3. Con un periodo quincenal, los centros operativos que cuenten con el sistema de pesado serán responsables de mandar vía módem su información a Sistemas Aeropuerto, para su posterior análisis y registro. Cada vez que ingrese la información de cada uno de ellos al sistema se generará un registro por guía. En caso de existir una guía que se pese en dos o más centros diferentes se tomará aquella que registre un mayor sobrepeso. Además se originará un registro en una tabla auxiliar, **tabla scan**, siempre y cuando el centro operativo en el que se peso la guía sea diferente. Estafeta pretende en un futuro realizar el pesado de las guías en un solo Centro Operativo, por lo que esta tabla ya no tendría incidencias. El formato del archivo que sistemas aeropuerto recibirá será el siguiente:

   - Archivo ASCII
   - Un sólo renglón por registro
   - Los campos separados delimitados por doble comilla
   - Los campos con sus comillas separados por un espacio
   - El orden de los campos deberá ser el siguiente:

     1. número de la guía
     2. origen
     3. destino
     4. sobrepeso
     5. volumen
     6. monto

4. El sistema será capaz de buscar los clientes de las guías las veces que sea necesario, tomando en cuenta todas aquellas guías en las cuales no se tenga nada en el campo de cliente. Cuando esto suceda se efectuará el siguiente proceso:

   Se ejecutará un módulo de generación de interface que dará como resultado la generación de un archivo ASCII conteniendo el listado de guías que no contengan nada en el campo cliente. Este módulo deberá pedir tres opciones para generarse, que son:

   - guía-inicial
   - guía-final
   - Lista del centro operativo

5. A continuación, se ejecutará el módulo de la búsqueda de la factura en que se vendió la guía, el cual tendrá como entrada el archivo anteriormente generado. Este módulo deberá tener como salida los siguientes reportes:

   - Reporte de guías a las cuales SI se encontró la factura correspondiente (Reporte **a**).
   - Reporte de guías a las cuales NO se encontró la factura correspondiente (Reporte **b**).
   - Reporte de guías que se encuentren en más de una factura. Este reporte tendrá que ser validado por el personal de facturación haciendo la detección

de la factura correcta, ya que sólo puede estar facturado una vez el envío (Reporte **c**).

La estructura del nombre de los reportes estará formado de la siguiente forma:

gf + número de mes + número de día + consecutivo + letra del reporte

Por ejemplo, si generan los reportes el día 12 de Junio por primera vez los nombres de los reportes serán:
gf6121a,
gf6121b,
y gf6121c, respectivamente.

Notas:
- El consecutivo de los reportes lo llevará el sistema.
- Los operadores que ingresen las facturas en la base de producción de Estafeta serán los responsables de la buena captura del número inicial y número final de las guías en las facturas. De lo contrario existirá el riesgo de no encontrar a los clientes de las guías.

6. El siguiente punto es buscar los clientes de aquellas guías que no se facturaron de manera inmediata, asignando el cliente correspondiente al campo "cliente".

7. El reporte "a" será procesado mediante un módulo que ligue los clientes de las guías encontradas. La función se encargará de actualizar el campo de "cliente" en el registro de una **tabla GUIA** y el número de factura que asignará al campo **guia-fact** de este registro.

8. El reporte "b" se trabajará manualmente, este reporte actualmente constituye el 50% (formato 2) de las guías buscadas. El usuario responsable tendrá que hacer la investigación necesaria para poder determinar la factura que ampara la guía, ya que no se encontró el registro. Una vez que se tenga este dato, será responsabilidad del usuario actualizar, en la base de sobrepeso, el cliente correspondiente a estas guías en la **tabla GUIA**, esto mediante un módulo que actualice el cliente, el cual preguntará por el número de la guía, desplegará **TODOS** los datos de dicha guía, y en esa misma pantalla preguntará por el número de cliente. Esta función se utilizará también para cambiar el cliente en cualquier guía, o inclusive para borrarlo y que esa guía se tome en cuenta en la siguiente corrida del proceso.

9. El reporte "c" desplegará las guías que, dentro de la base de facturación, están amparadas en más de una factura. Esto se puede dar debido a la mala captura o duplicidad en el número de la guía. El usuario responsable deberá determinar a qué factura corresponden estas guías. Este reporte se le dará al departamento de facturación para que, por medio de eliminación, determine las facturas mal capturadas y proceda a la corrección. Una vez que el usuario haya corregido la factura, deberá actualizar el campo de "cliente" en la **tabla GUIA**.

   Con los procesos anteriores, el sistema tendrá todas las guías con sus respectivos clientes, o sea que cada registro deberá tener el número de cliente correcto en el

campo "cliente" en la **tabla GUIA**. Con esta información se podrán generar las facturas.

10. Para la generación de las facturas se elaborará un reporte, el cual ordenará las guías que no se han facturado por los campos "cliente" y "num-ofi" (se tomará en cuenta la entidad y dirección de la oficina donde fueron comprados los servicios). Para esto, las opciones que pedirá el reporte son las siguientes:

    - Guia-ini — Número de guía inicial
    - Guia-fin — Número de guía final
    - Lista-cli — Lista de clientes a facturar
    - Fecha-ini — Fecha inicial de las guías
    - Fecha-fin — Fecha final de las guías
    - Monto-limite ($50) — Monto a partir del cual se generará factura por cliente

11. El sistema generará, en la **tabla FACTURA**, un registro por cada factura generada por la función, y emitirá un reporte de facturación por cliente y por oficina. El reporte deberá ser revisado por el usuario y, en caso de tener que modificar cualquier dato erróneo de la factura, lo hará mediante un módulo especial.

12. El sistema al momento de generar la factura, deberá asignar el consecutivo de facturación de sobrepeso, y asignará en el campo de "factura" del registro de la guía, el número de factura consecutivo que el sistema esté asignando en ese momento, esto es para poder tener la liga o el detalle entre las facturas de sobrepeso y las guías. En el caso de que el usuario detecte un error y borre la factura, el sistema se encargará de asignar un cero en el campo de "factura" de las guías que estaban ligadas o contempladas en la factura cancelada.

13. Una vez que el usuario ya verificó que las facturas estén correctas en los datos de clientes, guías e importes, se podrán imprimir mediante un módulo para impresión de facturas. Este módulo imprimirá exclusivamente aquellas facturas que en el campo "num-impres" (número de veces impresa) tenga el valor de 0 o, en otras palabras, aquellas facturas que no hayan sido impresas. Este módulo solicitará la siguiente información con el fin de establecer el rango de facturas a imprimir:
    - Numero-ini — Número inicial de factura
    - Numero-fin — Número final de factura
    - Lista-cli — Lista de clientes
    - Fecha-ini — Fecha inicial
    - Fecha-fin — Fecha final
    - Monto — >= 50.00

14. Por último se elaborará un proceso para el envío de las facturas generadas a sistemas aeropuerto para que puedan ser agregadas a la base de producción de Estafeta.

III.7.- Análisis costo / beneficio

El análisis de costo / beneficio delinea los costos por proyecto a desarrollar, esto es, algo que se puede medir tangiblemente, cuantificarlo en términos monetarios, pero siempre habrá beneficios del sistema que serán intangibles.

El análisis de costo / beneficio por criterio varía de acuerdo a las características del sistema a desarrollar, el tamaño del proyecto y realmente si cubre las exigencias pedidas. Como ejemplo de características intangibles de un sistema de computación tenemos: un mejor diseño, una mejor calidad, la satisfacción de los usuarios finales y poder dar mejores soluciones empresariales y administrativas basadas en datos reales proporcionados por el sistema.

En primera instancia lo que se busca del sistema es cuantificar mejor la información, su calidad de manejo, su oportunidad y organización de la misma.

El cálculo de los costos involucra el número de líneas del programa fuente, el tipo de esfuerzo, la renta del equipo, el mantenimiento de éste, los sueldos, etc.

En nuestro caso realizamos el siguiente estudio costo-beneficio tomando en cuenta Hrs. / hombre, costos y beneficios que generará la instalación del sistema.

A continuación se calcularán las Hrs. / Hombre requeridas para la realización del proyecto y así poder estimar los costos. Las cantidades que se muestran a continuación se calcularon tomando en cuenta cinco personas :

ACTIVIDAD	HRS / HOMBRE
Metodología	40
Análisis	350
Diseño	300
Programación	400
Arranque y pruebas	120
Mantenimiento	40
Total	**1250**

Gastos de Operación:

Suministro (Discos, hojas, cintas, luz, teléfono, etc.)	$ 2,000.00
Mantenimiento del Equipo	$ 4,000.00
Mantenimiento del Software	$ 6,500.00
Total (mes)	**$12,500.00**

Costos de arranque :

Análisis del sistema y especificaciones de lo que se necesita ($ 40.00 /hr)	$ 14,000.00
Diseño del sistema ($ 45.00 /hr)	$ 13,500.00
Programación, puesta en marcha y pruebas ($ 25.00 /hr)	$ 10,000.00
Mantenimiento, manuales de usuario y cursos ($ 25.00 /hr)	$ 1,000.00
Costo del equipo:	
Computadora 486 a 66 MHz, 16Mb RAM, DD 800Mb, Monitor SVGA	$ 10,000.00
Impresora Matriz de puntos de 15 pulgadas	$ 4,100.00
Módem 9,600 bauds	$ 4,000.00
Terminales (4)	$ 18,200.00
Tarjeta y software de comunicaciones	$ 6,200.00
Licencia de software (PROGRESS)	$ 9,000.00
Total	**$ 90,000.00**

Beneficios tangibles:

Ahorro de personal adicional	$ 4,000.00
Ahorro de papelería (por errores o ajustes)	$ 2,000.00
Total (mes)	**$ 6,000.00**

Beneficios intangibles:

Mejores relaciones con las demás áreas con las que interactúa.
Mayor eficacia en la información de planeación.
Satisfacción en el trabajo realizado.
Acceso a la información inmediata, es decir datos actualizados.
Recuperación de facturas no cobradas de aproximadamente $400,000.00 mensuales.

Los anteriores precios ya incluyen el IVA así como gastos adicionales que se ocasionen.

III.8.- Características principales de los manejadores de bases de datos

Dentro del estudio comparativo realizado se analizaron 6 manejadores de bases de datos, de los cuales DELPHI,SYBASE y DATABASE DB/2 tienen interface gráfica, por lo cual fueron descartados, ya que no se cuenta con la infraestructura necesaria en cuanto a equipo que requieren, lo cual implicaría una fuerte inversión para la empresa.

De acuerdo a las necesidades de la empresa y a las políticas de trabajo, como es el uso de aplicaciones de tipo carácter, las cuales observan un mejor performance en el manejo masivo de información, se optó por profundizar más en el estudio de tres manejadores, que son ORACLE, PROGRESS e INFORMIX, ya que los demás no trabajan con aplicaciones tipo carácter.

Haciendo referencia a la tabla 3.3 hacemos las siguientes observaciones:

- Requerimientos de hardware: De aquí se observa que ORACLE tiene requerimientos mayores y más costosos que PROGRESS e INFORMIX. Por otra parte vemos que INFORMIX requiere de Mainframes y sistemas abiertos.
- Clientes que soporta: En este rubro el que mayor número de clientes soporta es PROGRESS.
- Sistema operativo red: Nuevamente PROGRESS es el que maneja un mayor número de sistemas operativos en red.
- Interface gráfica del sistema operativo de red: Esta opción queda descartada debido a los requerimientos de la empresa antes mencionados. A pesar de ello PROGRESS lleva la vanguardia en este renglón.
- Sistema manejador de la base de datos y/o lenguajes: En este rubro vemos que ORACLE maneja más lenguajes de desarrollo.
- Herramientas para desarrollar la aplicación: INFORMIX es el que tiene más herramientas de desarrollo.
- Módulos de conectividad preestablecidos: El manejador que mayor número de módulos de conectividad maneja es PROGRESS.

De acuerdo a las observaciones obtenidas de la tabla 3.3 y a los costos de cada uno de ellos, tomamos la decisión de proponer a PROGRESS como manejador y lenguaje de desarrollo del sistema.

Aunado a esto nos encontramos con que la empresa ya cuenta con el lenguaje mencionado, por lo que no requeriría de una inversión adicional en este rubro, además de que con esto se unificaría y estandarizaría nuestro sistema con los del resto de la empresa, obteniendo grandes ventajas.

PRODUCTO DE INGENIERIA	ORACLE	INFORMIX	DELPHI	DATABASE DB2	SYBASE	PROGRESS
REQUERIMIENTOS EN HARDWARE	PC's con procesador 386 o mayor 16 Mb en RAM (32 Mb recomendados) de 25 a 50 Mb en disco Unidad CD-ROM -IBM RS/600 -SEQUENT DYNIX/ptx -Sun 4/SPARC /Soulborne Solaris -Apie Mac OS, A UX. -Bull DPX20 AIX v7 -Control Data Server 4000 (CDC) -DEC Ultrix/RISC, VAX MVS	Mainframes Sistemas abiertos (Virtualmente se puede instalar en cualquier plataforma y los requerimientos varían de acuerdo al equipo que se instale).	(50 Mb en disco) 486 o mayor 8 Mb en Ram (16 Mb recomendados) Unidad de CD-ROM	RS/6000 AS/400 Mainframes PC's Work Station HP Work Station Sun	De 72 Mb a 111 Mb en disco. Por lo menos 16 Mb en RAM (recomendable 32 Mb) Sybase corre en las plataformas más utilizadas del mercado, tales como: IBM, HP, Intel, Digital, Sequent, Data General, AT&T, Silicon Graphics, Tandem, Unysis, Pyramid, Siemens y Stratus.	De 18 Mb a 25 Mb en disco. Procesador 386 o mayor. Estaciones de trabajo basadas en la tecnolgía RISC o incluso equipos como la ALPHA 10,000 De 500 Kb a 4 Mb de memoria para el usuario. Los procesadores pueden ser de tecnología Intel, RISC, Motorola, AS/400, VAX.
CLIENTES QUE SOPORTA	MS-Windows MACINTOSH OS/2 MS-DOS DG/UX Open VMS Ultrix HP-UX Solaris DYNIX SCO UNIX	UNIX Windows Macintosh	Windows	MS-DOS Windows OS/2 AIX MVS VSE & VM HP-UX Solaris UNIX	UNIX VMS Solaris OS/2 IRIX DG/UX Windows MS-DOS	IBM AIX HP-UX NCROS Sun OS Sun Solaris DEC Alpha OSF-1 DEC- Open VMS Ultrix SCO-UNIX MS-DOS UNIX-V PTX AUX DG UX IRIX

Tabla 3.3 Manejadores de bases de datos (continúa).

PRODUCTO DE INGENIERIA	ORACLE	INFORMIX	DELPHI	DATABASE DB2	SYBASE	PROGRESS
SISTEMA OPERATIVO RED	Novell Windows NT SCO UNIX OS/2	Novell Windows NT	Novell Windows NT Banyan VINES DecNEt Pathworks IBM LAN Server, Microsoft LAN Manager Lantastic, UNIXWARE, AIX RS6000	OS/2-LANServer MVS AIX	UNIX VMS OS/2 Netware Windows NT	IBM AS/400 Windows NT Novell SCO-UNIX OS/2
INTERFACE GRAFICA DEL SISTEMA OPERATIVO DE RED	WINDOWS MOTIF MACINTOSH	WINDOWS Motif, Macintosh Carácter	Windows	OS/2	Windows Motif Macintosh OS/2	MS-Windows Motif X-Windows Cáracter UNIX
SISTEMA MANEJADOR DE LA BASE DE DATOS Y/O LENGUAJES	RDBMS (relacional) SQL PL/SQL C FORTRAN COBOL C++ Net TCP/IP PL/1	RDBMS (relacional) ODBC C COBOL	RDBMS (relacional) dBASE DDE DLLS C, C++ Pascal SQL ODBC	RDBMS (relacional) C FORTRAN COBOL SQL CLI ODBC	RDBMS (relacional) ANSI SQL Transact SQL	RDBMS (relacional) HLC ESQL/C SQL ODBC DDE C
HERRAMIENTAS PARA DESARROLLO DE APLICACION	CDE2 TOOLS ORACLE FORMS 4.5 ORACLE GRAPHICS 2.5 ORACLE REPORTS 2.5 CDE Soporta objetos multimedia	INFORMIX NewEra INFORMIX-NewEra Point Pro INFORMIX-4GL INFORMIX-4GL Forms INFORMIX-Menus INFORMIX-SQL INFORMIX-ESQL para C o COBOL	Two-Way-Tools Form Expert Crystal Reports for dBASE Borland Database Engine 2.0 ReportSmith 2.5	CAE (client aplication enabler) SDK (software developer's kit)	Las aplicaciones se pueden desarrollar con más de 125 herramientas de desarrollo distintas.	Ambiente de Desarrollo de Aplicaciones (ADE) herramientas de desarrollo graficas basadas en el 4GL Diccionario de Datos
MODULOS DE CONECTIVIDAD PREESTABLECIDOS	DB2 SQL Rdb Server de Microsoft	Base de Datos de IBM	dBASE Paradox Oracle Sybase MS/SQL Server Informix Interbase	Microsoft Access Base de Datos DRDA	Sybase tiene más de mil asociados en aplicaciones y herramientas de desarrollo, para una gran variedad de mercados	ORACLE, RMS Rdb-VMS C-ISAM, CT-ISAM DB2 Object Access, Object store Sybase

Tabla 3.3 Manejadores de bases de datos.

III.9.- Programa de actividades

La planeación de actividades se llevó únicamente en los días hábiles de trabajo de la empresa, debido a que es dentro de sus instalaciones y en colaboración con sus empleados que llevaremos a cabo el desarrollo del presente proyecto. Esta planeación la dividimos en cinco fases como se explica a continuación y se muestra en la figura 3.5.

FASE I
Debido a la complejidad que conllevan las dos primeras actividades que son análisis y diseño del sistema, se contempla dedicar 15 días para el análisis y 15 para el diseño, que son los días del 18 de marzo al 26 de abril del presente año.

FASE II
La generación de la base de datos se llevará a cabo en la semana del 29 de abril al 3 de mayo, continuando con el diseño y la generación de tablas que se realizará en una semana, del 6 al 10 de mayo. Para la carga de librerías de SENTAI a la base de datos se necesitarán dos días, 13 y 14 de mayo. El desarrollo de los módulos de configuración del sistema lo llevaremos entre los días 15 y 24 de mayo. Para completar la fase II se desarrollará el módulo de generación de facturas del 27 de mayo al 5 de junio.

FASE III
La fase III comenzará con el módulo de corrección y cancelación de facturas del 6 al 14 de junio. Continuaremos con la impresión de facturas del 17 al 25 de junio. El módulo de refacturaciones se realizará del 26 de junio al 5 de julio y concluiremos esta fase con las primeras pruebas del sistema del 8 al 15 de julio.

FASE IV
Una vez terminadas las pruebas procederemos con el desarrollo de las interfaces (esto es, el flujo de información entre Ceylán y Aeropuerto) del 16 al 26 de julio. Los módulos de depuración y respaldos se llevarán a cabo del 29 de julio al 2 de agosto. Para el desarrollo de reportes y listados, debido a la amplia variedad que el sistema requiere, dedicaremos 10 días hábiles para su elaboración, que estarán comprendidos del 5 al 16 de agosto. Por último, en esta fase se verá uno de los puntos primordiales del proyecto: el análisis del equipo requerido. Esto se llevará en conjunto con la gente que operará el sistema del 19 al 23 de agosto.

FASE V
En la última fase del proyecto se verá la adquisición del equipo del 26 al 28 de agosto. La instalación del equipo incluyendo software, módems y línea telefónica, se realizará en los días comprendidos del 29 de agosto al 4 de septiembre. Una vez teniendo listo el equipo se instalará la base de datos y se trasladará en lo posible la información del sistema anterior al sistema nuevo. El plan para esta actividad es del 5 al 18 de septiembre. La elaboración del manual de usuario será del 19 al 26 de septiembre y por último se tendrá el proceso de arranque del sistema del 27 de septiembre al 3 de octubre.

Actividades del Proyecto de Sobrepeso.

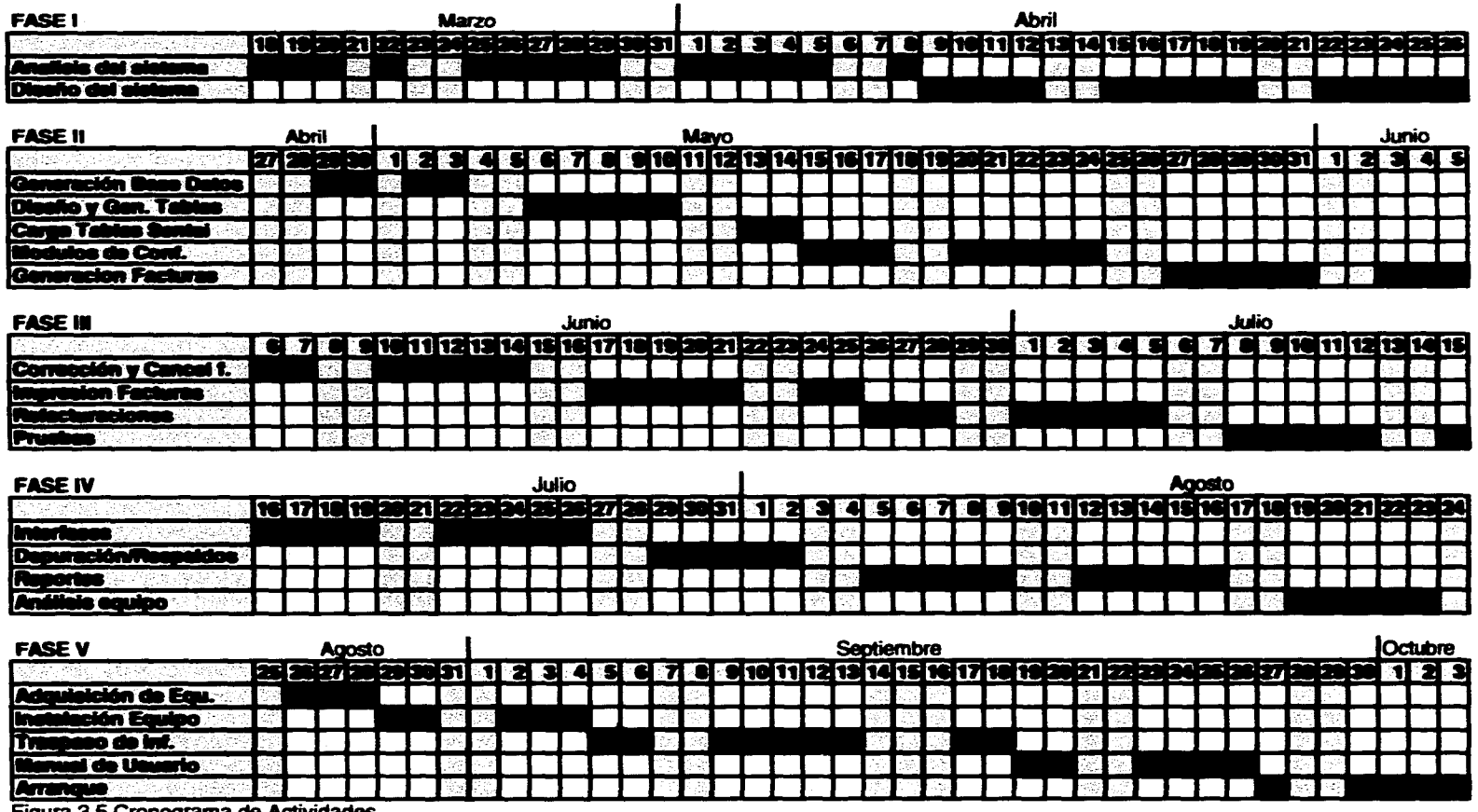
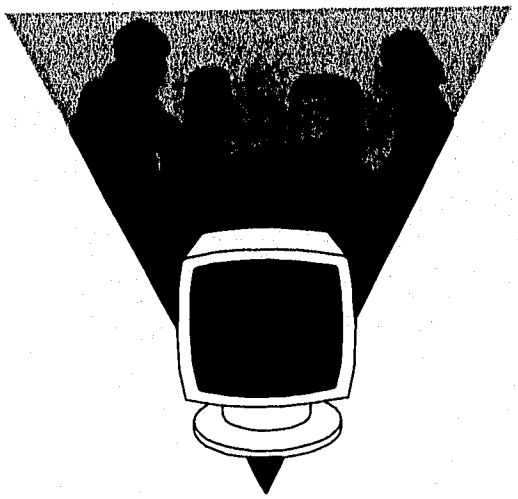


Figura 3.5 Cronograma de Actividades.



CAPITULO IV

DISEÑO

Una vez que hemos efectuado el análisis de nuestro sistema, procederemos con la fase de ***DISEÑO,*** para lo cual, con base en la metodología planteada en el capítulo dos, cubriremos los aspectos siguientes:

1. Objetivos y alcances del sistema

2. Descripción general del sistema propuesto
   - Principales componentes de software.
   - Interfaces de software, hardware, y humanas.
   - Limitaciones y consideraciones de diseño.

3. Descripción del diseño
   - Revisión del D.F.D.
   - Carta estructurada.
   - Estructura de archivos.

IV.1.- Objetivos y Alcances del sistema

Los objetivos del sistema de sobrepeso son los siguientes:

- Emisión de facturas de manera oportuna para su cobro.
- Estandarización de las bases de datos tanto de clientes como de facturas, para eliminar la inconsistencia de información y duplicidad de datos entre áreas.
- Aumentar los ingresos de la empresa por concepto de cobro de sobrepeso.
- Actualización de los datos de los clientes.

Para tal efecto, las metas propuestas son:

- Encontrar hasta el 95% de los clientes que generaron el sobrepeso en las guías detectadas en los centros operativos.
- Proporcionar a los operadores las herramientas necesarias para la búsqueda manual de las guías a las que no se localizó su respectivo cliente, esto es, reportes conteniendo el número de guía y la oficina que realizó la venta.
- Automatizar la carga de guías al sistema, eliminando la captura de las mismas.
- Llevar un registro acumulativo de los clientes para poder cobrar el sobrepeso registrado en las guías de meses anteriores, hasta que alcance el mínimo requerido para la emisión de su factura, de acuerdo a la política administrativa adoptada por la empresa.
- Flujo de información entre los diferentes centros operativos a nivel nacional.
- Mantenimiento oportuno en los datos de los clientes, evitando así las refacturaciones.
- Facilitar el intercambio de información referente a las facturas entre las sucursales de Aeropuerto y Ceylán.
- Proporcionar las herramientas necesarias para la emisión oportuna de las facturas.
- Brindar al cliente "estados de cuenta" de sus paquetes.

IV.2.- Descripción general del sistema propuesto

En este apartado efectuaremos un análisis de los puntos más importantes relacionados con el diseño de nuestro sistema.

Principales componentes de software

Para la selección del lenguaje a utilizar en el desarrollo del sistema, se siguieron los puntos que propone la metodología del capítulo dos, los cuales son:

- *Area de aplicación*
  Se trata de un ambiente administrativo contable en un 90% y otro 10% de análisis estadístico.

- *Complejidad del algoritmo*
  Podemos ver que la complejidad del algoritmo no es muy grande por ser un medio en el cual los procesos son relativamente sencillos.

- *Medio ambiente de ejecución del software*
  En este punto se menciona que se trabajará sobre una plataforma de UNIX, se debe tener en cuenta que el lenguaje elegido y sus utilerías deberán trabajar en este sistema operativo.

- *Funcionamiento general del software (performance)*
  Uno de los parámetros previamente definidos es trabajar con el equipo ya existente, por lo cual nos abocaremos a trabajar también con la base de datos PROGRESS, como se concluyó en el capítulo anterior.

- *Complejidad en las estructuras de datos*
  En el trabajo a desarrollar existen múltiples estructuras de datos, algunas de las cuales presentan una complejidad mayor que las otras, por lo que deben ser tomadas en cuenta en la selección del lenguaje, en nuestro caso se considera que la complejidad es media.

- *Experiencia y conocimiento del personal de desarrollo*
  En este punto es conveniente recordar que se puede elegir un lenguaje gráfico y más amigable con el usuario, sin embargo, si el personal de programación no tiene la suficiente experiencia en su manejo, y no se cuenta con los recursos para la compra de equipos que soporten este tipo de lenguaje, entonces es más conveniente utilizar uno que se conozca mejor y cumpla con las necesidades de estandarización de la empresa.

Interfaces de software-hardware y humanas

Dados los objetivos que se persiguen y como el trabajo es en una microcomputadora que funcionará como servidor con cuatro terminales, formando un sistema multiusuario, con lo cual el usuario interactuará con la computadora a través del teclado y el video, tanto del servidor como de las terminales.

Limitaciones y consideraciones de diseño

Este es el momento apropiado de plantear que limitaciones y consideraciones se harán en el diseño, por lo que, consultando el diagrama de flujos de datos, figura 4.1, podemos observar que:

Existen actividades manuales que no se automatizarán por medio del sistema, como son:

- Clasificación de los envíos por los servicios generadores de sobrepeso.
- Pesado de los paquetes para la detección del sobrepeso.
- Búsqueda de clientes generadores de sobrepeso de las guías que no fueron localizadas en la facturación de Aeropuerto. Esta podrá ser personalmente o vía telefónica, auxiliándose en el reporte generado por el sistema.
- Captura (mínima) de guías que por alguna causa no pasen por la báscula.
- Llenado y captura de formato de los servicios que entregan guías sin que exista una factura de por medio (Servicio Programado Plus, Autorización, Intercambio, etc.)
- Generación de disquetes para el flujo de información entre las áreas participantes.

IV.3.- Descripción del Diseño

Es el producto final de la etapa iniciada en el análisis, la cual nos lleva a obtener el diagrama de flujo de datos definitivo, a partir del cual obtendremos la carta estructurada y la respectiva estructura de archivos.

Revisión del D.F.D.

En el capítulo anterior (análisis) se hizo un estudio detallado de cómo está funcionando el sistema, así como los cambios que se podrían hacer o agregar; de todo esto se obtuvo un DFD desde el punto de vista del área de facturación de sobrepeso, figura 4.1. Este será nuestro punto de partida, del cual haremos una revisión más detallada para posteriormente llegar a la carta estructurada.

En la figura 4.1 se presenta el diagrama general de flujo de datos que tendrá el sistema de sobrepeso, el cual describiremos a continuación :

Una vez que se recolectan los paquetes y se llevan a las áreas de pesaje, que serán tres: México, Guadalajara y Monterrey, se separan aquellos paquetes susceptibles de generar sobrepeso, se colocan en las bandas de distribución y con un scanner se leen las guías. Esta información se almacenará en una base de datos que podría llamarse *sobrepeso.dbf* y sería manejada con FoxPlus, ésta se pasará al operador de Ceylán con cortes quincenales, con el fin de generar un archivo tipo ASCII que podría denominarse *gulamex*, así como *gulamty* y *gulagdl* respectivamente como se muestra en la figura 4.2. Con estos archivos se alimentará el sistema de sobrepeso. Se contempla utilizar una base denominada *guia* para almacenar la información de todas las guías del país, por lo que se hará un cruce de información de las guías contenidas en cada uno de los archivos con el fin de verificar si una guía se pesó en uno ó más centros y dejar así la que mayor peso haya registrado, además de actualizar en las guías que tengan cliente el costo del sobrepeso con una base denominada *precios*, que

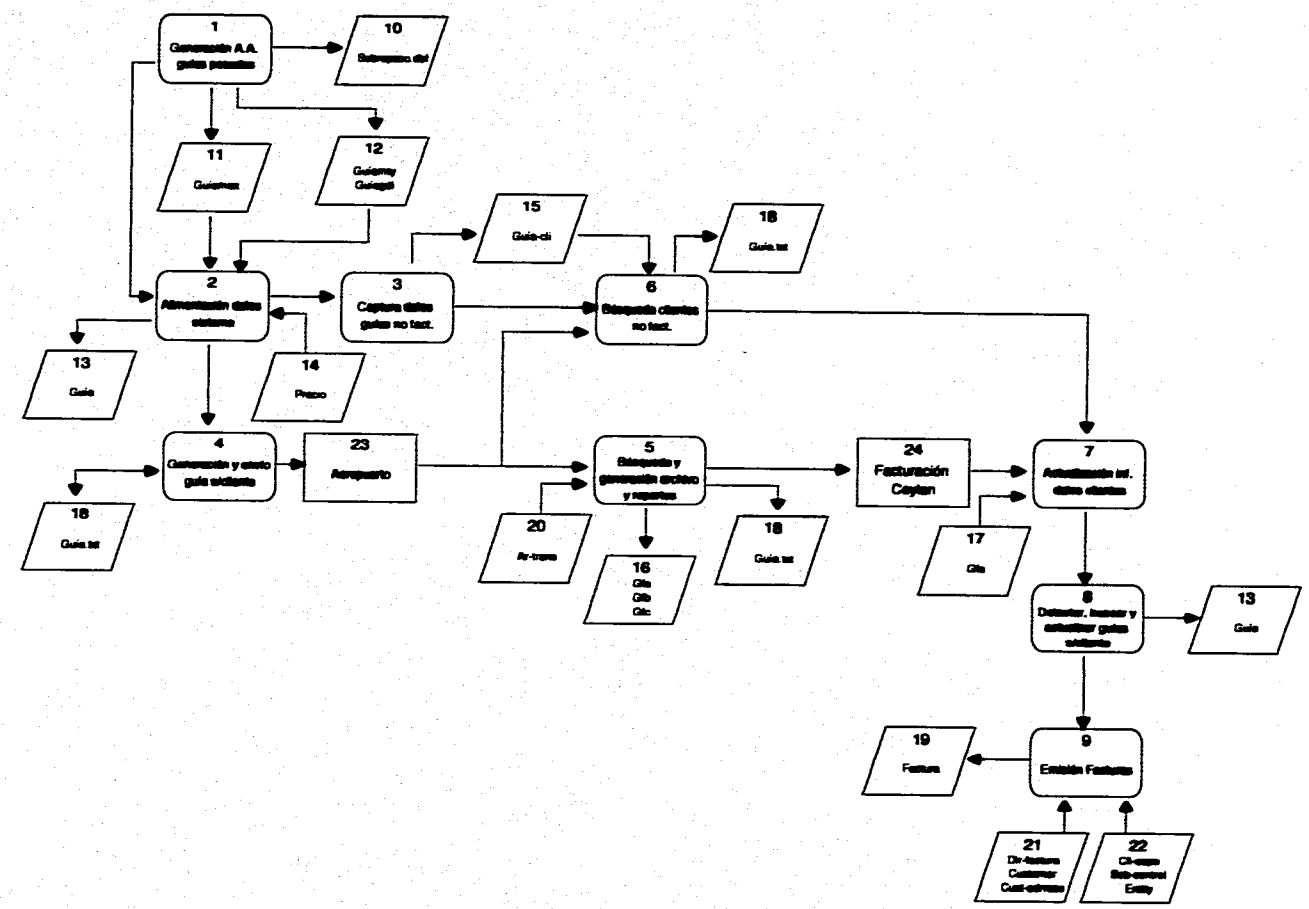


Figura 4.1 Diagrama general de flujo de datos.

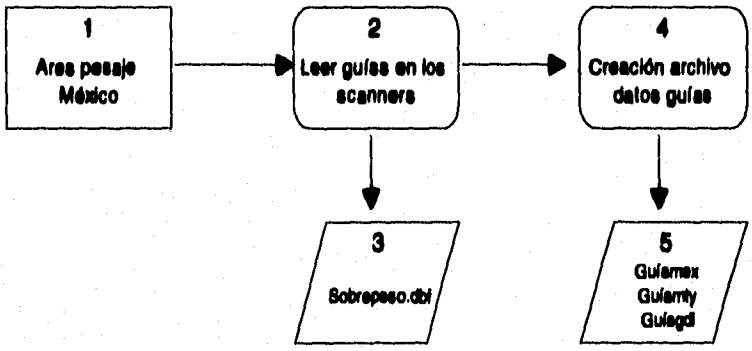


Figura 4.2 Descripción del proceso 1.

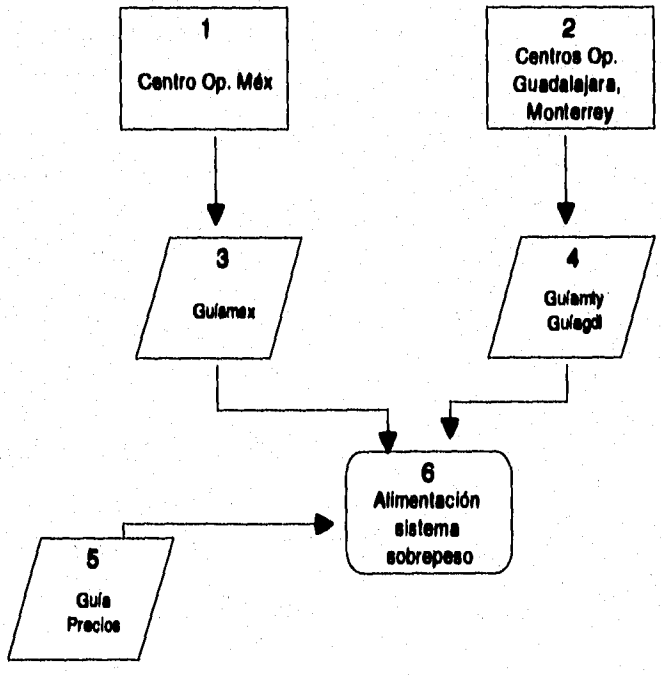


Figura 4.3 Descripción del proceso 2.

contendrá los precios por cada kilogramo de más en cada uno de los servicios que se ofrecen como se muestra en la figura 4.3.

Hay servicios en los que al adquirir las guías no se paga el servicio sino hasta su utilización, por lo cual es necesario tener un control sobre los clientes a los cuales se les vendieron esas guías, para esto habrá un proceso de captura de guías no facturadas, esta información podría almacenarse en una base denominada *guia_cli*. Enseguida se generará un archivo que contendrá las guías sin cliente denominado guia.txt como se muestra en la figura 4.4. En la figura 4.5 podemos apreciar que este archivo se enviará al área de sistemas en Aeropuerto para que las guías sean buscadas en una tabla de la base de datos de Estafeta denominada *ar-trans*, donde se almacena la información de todas las facturas generadas en la empresa y así determinar el cliente al cual fueron vendidas. Una vez buscadas en esta base se procederá a generar un archivo que podría denominarse *gfa*, que contendrá las guías a las que se les encontró su cliente, y dos reportes que podrían denominarse *gfb* y *gfc*; el primero será de las guías no encontradas y el segundo de las guías encontradas más de una vez, esto debido a que podría cometerse un error en la captura de guías; estos archivos son enviados a Ceylán.

En la figura 4.6 podemos apreciar que al mismo tiempo que se realiza la búsqueda en Aeropuerto, en Ceylán se realiza la búsqueda en la base de datos donde se almacenaron las guías no facturadas para poder detectar el mayor número de clientes. Una vez encontrados se actualiza el archivo denominado *guia*. Cuando tengan el archivo *gfa* se dedicarán a actualizar la información de los clientes en la base de *guia* como se aprecia en la figura 4.7. Con los reportes *gfb* y *gfc* podrán detectar las guías que aún no tienen cliente, y los localizarán ya sea vía telefónica o en el lugar donde les fueron vendidas las guías. Una vez localizados la mayoría de estos se procederá a actualizar la información en la base de *guia* como se muestra en la figura 4.8.

Por último se dará el proceso de facturación que se muestra en la figura 4.9. Para asegurar que las facturas tendrán todos los datos correctos se emitirá un borrador, para lo cual se podrían utilizar algunas bases como : *dir-fac* que contendrá la dirección de emisión de la factura, el lugar donde se elaboró; *customer* contendrá datos de los clientes como razón social, rfc, teléfono, etc.; *cust-adrress* contendrá la dirección de los clientes; *sob-control* donde se llevará el control de los números consecutivos del sistema; *entity* donde se tendrán las entidades federativas de las oficinas de venta, la oficina a la que corresponde la factura de la venta; *cli-espe* que contendrá información de los clientes especiales y *factura* que es donde se almacenará toda esta información.

Con el reporte se procederá a validar toda la información de las facturas, una vez validada se modificará la base de *factura* ó se cancelarán las que se considere necesario. Estando completamente seguros de los datos de las facturas se procederá a la impresión de las mismas

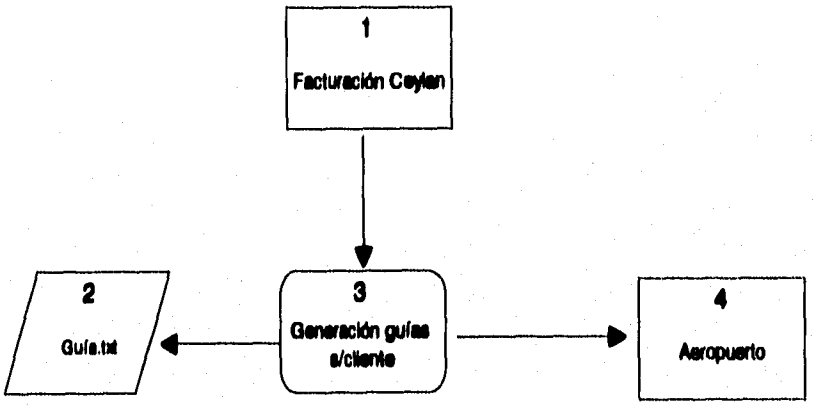


Figura 4.4 Descripción del proceso 4.

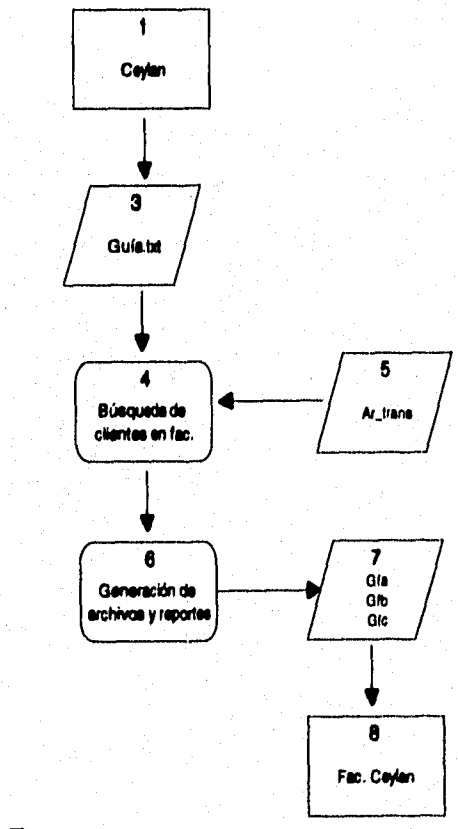


Figura 4.5 Descripción del proceso 5.

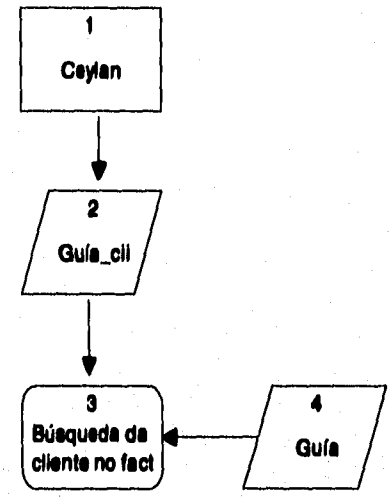


Figura 4.6 Descripción del proceso 6.

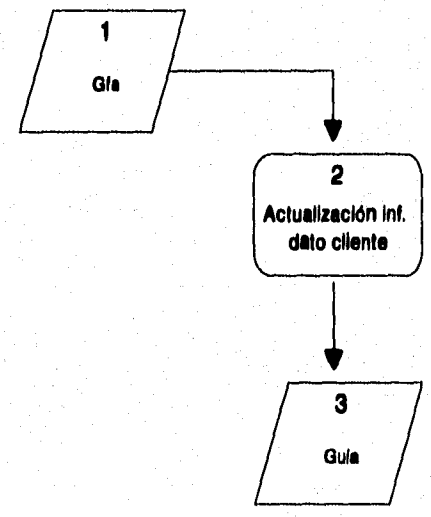


Figura 4.7 Descripción del proceso 7.

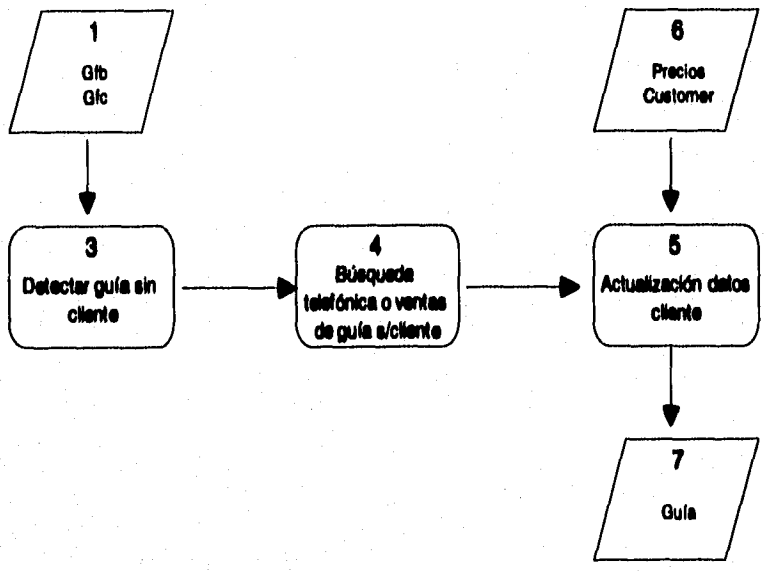


Figura 4.8 Descripción del proceso 8.

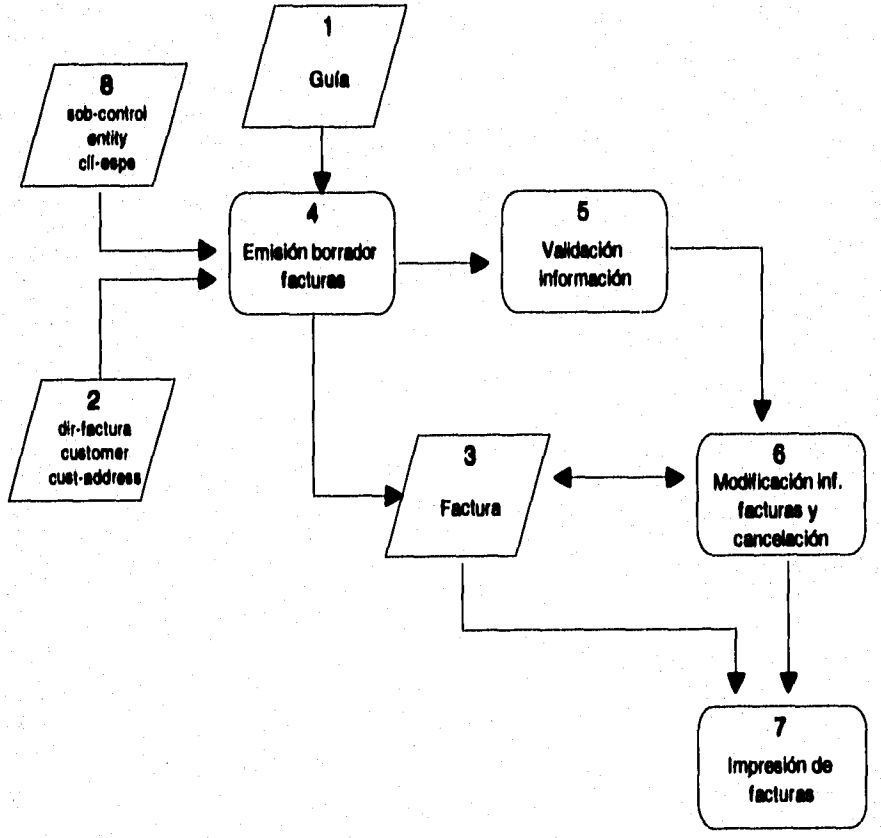


Figura 4.9 Descripción del proceso 9.

La tabla 4.1 muestra los procesos que se ocuparán, así como el número que los identifica y el encargado de llevarlos a cabo.

Proceso	Descripción	Encargado
1	Generación archivos ASCII de guías pesadas	Ceylán
2	Alimentación datos sistema	Ceylán
3	Captura de datos de guías no facturadas	Ceylán
4	Generación y envío de archivo de guías sin cliente	Ceylán
5	Búsqueda y generación archivo *gfa* y reportes *gfb* y *gfc*	Aeropuerto
6	Búsqueda de clientes no facturados	Ceylán
7	Actualización de información de datos de los clientes	Ceylán
8	Detectar, buscar y actualizar guías sin cliente	Ceylán
9	Emisión de facturas	Ceylán

Tabla 4.1 Descripción de procesos.

Carta estructurada

La carta estructurada es la representación hasta cierto punto física de los requerimientos del sistema, ya que éstos y el intercambio de datos, archivos y estructuras de datos, representan la culminación de la etapa del análisis, y propiamente es el paso intermedio entre el diseño del sistema y la programación de éste.

Existe diversos métodos para llevar a cabo la carta estructurada, partiendo del diagrama de flujo de datos, éstos se ajustan de acuerdo al tipo de proyecto desarrollado como son:

- *Estructura descendente*

  Son utilizados a través del proceso de análisis y diseño, su valor radica en el enfoque descendente, iniciando por los niveles generales para tener una mayor visión del sistema hasta llegar gradualmente a niveles de mayor detalle. Su uso se ha difundido ampliamente en la ingeniería de sistemas y diseño de software, en donde cada función que ha de desempeñar el sistema se identifica primero y posteriormente se desarrolla con mayor detalle.

- *H.I.P.O Hierarchial Input Process Output.*
  Diagramas jerárquicos entrada/proceso/salida

Los diagramas HIPO son gráficos que ayudan a saber qué función realizan los módulos, cómo lo hacen, qué entradas y salidas se tienen. Así es común encontrar de tres a cinco niveles de módulos, proporcionando facilidad para tener sistemas bien documentados.

- *Diagramas Warnier/Orr*

También llamados construcción lógica de programas/construcción lógica de sistemas. Ayuda al diseño de estructuras de un programa identificando la salida y los resultados del procesamiento y después retrocediendo para determinar los pasos y combinaciones de entrada que se necesitan para producirlos. Este método gráfico hace evidente los niveles del sistema y el movimiento de datos experimentado.

- *Métodos de transacciones y de transformación*

Una vez obtenido el diagrama de flujo de datos y teniendo definidos los archivos y las estructuras de datos, tenemos dos caminos a elegir, para llegar a la carta estructurada, uno de los cuales es el diseño por transacciones, útil para proyectos técnicos, y establece que un sistema (programa) queda definido en su entrada de software por las estructuras de datos de entradas y salidas. Sus características son:

- Requiere de estructuras de datos bien definidas.
- No se preocupa por el flujo de datos interno.
- Revisión de cada una de las transacciones.
- Generar una estructura de software para cada transacción.
- Optimización de la carta estructurada en general.

El método por transformación parte del hecho de que los datos al entrar y salir de los procesos sufren transformaciones. A partir de este criterio se genera la carta estructurada, considerando además la ayuda de otras metodologías como la de Jackson, que incluye el uso de estructuras de software predefinidas como secuencia, iteración y selección propias de los lenguajes estructurados.

De acuerdo a cada salida se derivan estructuras de proceso con la metodología de Jackson y se aplican estructuras combinacionales para derivar los subsistemas y procesos finales. Estas estructuras combinacionales parten de las estructuras fundamentales y se mezclan para formar nuevas estructuras.

También existen técnicas que permiten tener a los componentes de la carta estructurada en forma independiente, es decir utilizando módulos, como a continuación se explica.

- *Modularidad*

Con este tipo de técnica los contenidos de los módulos se diseñan de tal manera que desarrollen una función específica y se entiendan con más facilidad. Existen grupos de tipos de módulos que realizan tareas específicas como:

- *Módulo según su función*

- Control ó interface lógica: Selecciona la parte del proceso requerido.
- Proceso: Realiza el trabajo de una función determinada.
- Rutinas auxiliares: Son módulos comunes, como inicializa, borra, etc.
- De interface archivos: De entrada/salida a periféricos.
- Errores y excepciones: Realiza manejos de información no requerida.

- *Módulos según su ejecución*

  - Secuencial: Es ejecutable completamente sin interrupciones del software aplicativo.
  - Incremental: Es ejecutado y puede tener una interrupción de software y reiniciarse posteriormente en el BREAK-POINT.
  - Paralelo: Dos ó más módulos pueden ejecutarse simultáneamente.

El método utilizado para obtener la carta estructurada es el H.I.P.O. ya que describe el sistema en su generalidad y un conjunto de diagramas funcionales. En donde cada diagrama muestra la entrada, la salida, los pasos del procesamiento y los flujos de datos. Además de ser muy efectivo para documentar el sistema.

La simbología utilizada en la carta estructurada se muestra en la figura 4.10, así como una breve descripción:

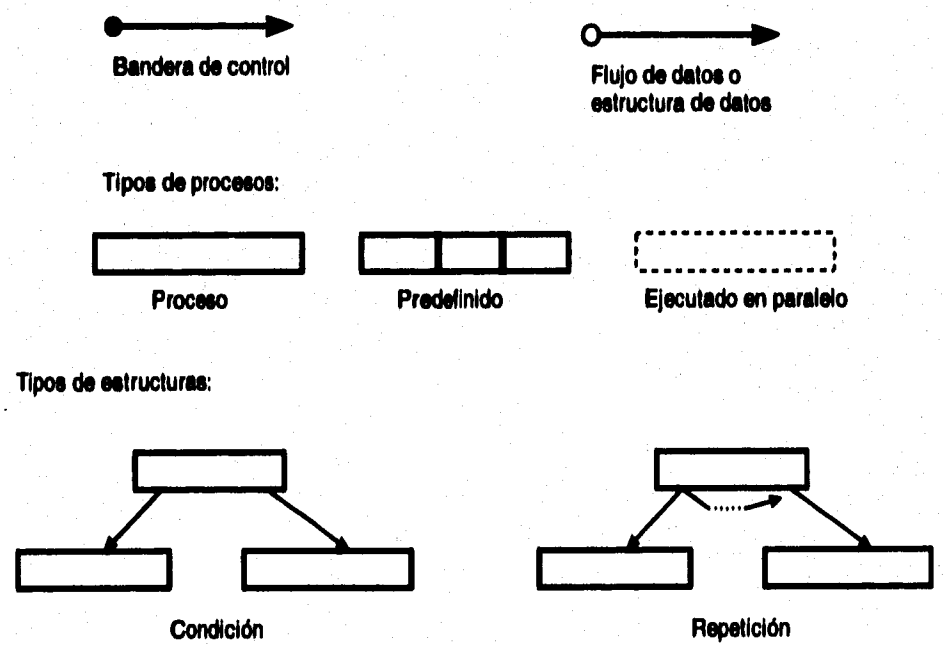


Figura 4.10 Simbología utilizada en la carta estructurada.

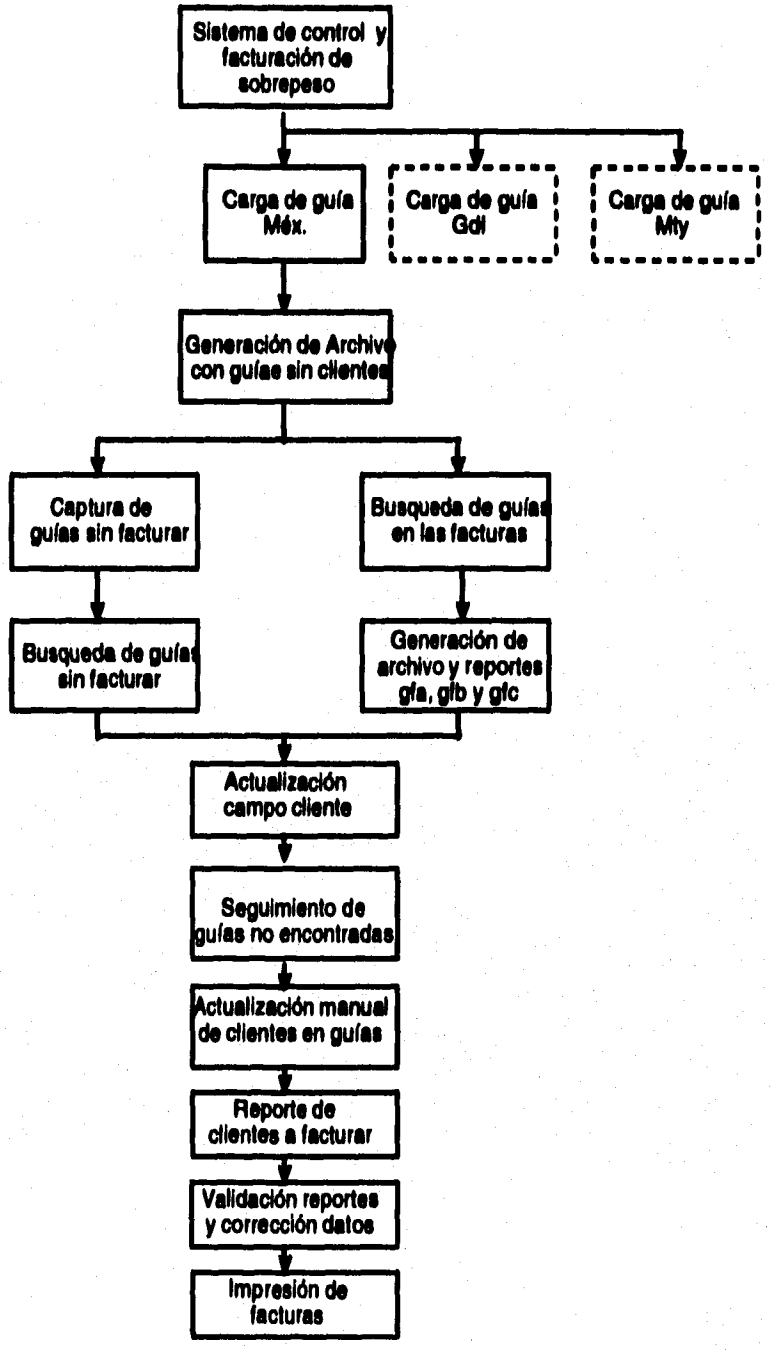


Figura 4.11 Carta estructurada.

En la figura 4.11 se presenta la carta estructurada del sistema de facturación de sobrepeso, en la cual se puede apreciar más claramente el flujo de información y el flujo entre los procesos que intervienen en el sistema. Dado que los procesos se explicaron a detalle en la revisión del diagrama de flujo de datos, a continuación sólo se mencionan tal como los muestra la carta estructurada.

Proceso 1. Generación de archivos ASCII de las guías pesadas.
Proceso 2. Carga de los archivos ASCII de los diferentes centros operativos al sistema.
Proceso 3. Captura de datos de guías no facturadas.
Proceso 4. Generación y envío de archivo ASCII de guías sin cliente a Sistemas Aeropuerto.
Proceso 5. Búsqueda de guías en las facturas, generación de archivo ASCII con guías encontradas y reportes de guías no encontradas y encontradas en más de una factura.
Proceso 6. Búsqueda de clientes en guías no facturadas.
Proceso 7. Actualización del campo cliente en las guías encontradas.
Proceso 8. Localización y actualización del cliente en las guías no encontradas.
Proceso 9. Emisión e impresión de facturas.

Estructura de datos

La estructura de datos es una representación de la relación lógica existente entre los elementos individuales de datos. Debido a que la estructura de la información afectará al diseño procedimental final, la estructura de datos es tan importante como la estructura del programa en la representación de la arquitectura del software.

La estructura de datos dicta la organización, los métodos de acceso, el grado de asociatividad y las alternativas de procesamiento para la información.

Reporte de Leyendas

En la columna Banderas aparecerán las siguientes siglas que significan :

c Campo de caso sensible
m Campo obligatorio
I Campo participante en la indexación
v Campo componente de vista

TABLA dir-factura

Tabla que almacena la dirección de emisión de la factura.

Nombre de campo	Tipo	Banderas	Longitud	Descripción
numero	entero	I	1	Número
direccion	carácter		40	Dirección
servicios	carácter		40	Servicios

INDEXACIONES DE LA TABLA

Nombre de Indexación(indicador primario)*	*Unico*	*Nombre del campo*

dir-factura*	si	numero

TABLA cli-espe

Tabla donde se almacenan los datos del cliente especial.

Nombre de campo	Tipo	Longitud	Descripción del campo
Cust-ID	carácter	8	Identificador del cliente
servicio	carácter	3	Tipo de servicio
M1	carácter	40	Observaciones
M2	carácter	40	Observaciones
M3	carácter	40	Observaciones
M4	carácter	40	Observaciones
Revision	carácter	5	Número de revisión
Pago	carácter	5	Identifica el pago
Rfc	carácter	20	R.F.C. del cliente
Gramos	carácter	6	Cantidad en gramos
Desxter	decimal	11	Destino terrestre
Desxaere	decimal	9	Destino aéreo
Plaza	entero	3	Plaza
Oficina	carácter	3	No. de oficina
kilo-terres	decimal	12	Cantidad de kilos terrestres
kilo-aereo	decimal	11	Cantidad de kilos aéreos
precio-aer	decimal	10	Cantidad por vía aérea
precio-terr	decimal	10	Cantidad por vía terrestre

INDEXACIONES DE LA TABLA

Nombre de la indexación (* indicativo primario) Nombre Unico de campo

cli-espe*	Cust-ID
servicio	

CRITERIO PARA EL CAMPO DE VALIDACIÓN

Cust-ID:

Criterio: cust-id ne ""

TABLA guia-control

Tabla que controla la numeración de las facturas.

Nombre de campo	Tipo	Banderas	Longitud	Descripción
Trans-no	entero	m	10	No-Trans
num-impre	decimal		10	núm-impre

INDEXACIONES DE LA TABLA

Nombre de indexación(indicador primario)*	*Unico*	*Nombre del campo*
default*	no	

TABLA cust-address

Tabla donde se almacena la dirección del cliente.

Nombre de campo	Tipo	Banderas	Longitud	Descripción
Machine-ID	carácter	im	4	Máquina
Cust-ID	carácter	im	8	Cliente
Address-no	entero	im	4	No-Dirección
Name	carácter	i	60	Nombre
Address	carácter		40	Dirección
City	carácter		20	Ciudad
Prov	carácter		4	Estado
Postal-code	carácter		10	Código Postal
Country-ID	carácter		4	País
Telephone	carácter		16	Teléfono
Memo	carácter		20	Memo
Tax-group-ID	carácter		4	Grupo-Impto
Cust-PO	carácter		20	OC-Clte
Contract-ID	carácter		20	Contrato
Carrier-ID	carácter		4	Transportista
Sales-list	carácter		60	Lista-Vendedor
Ship-instr	carácter		60	Instr-Embrq
Route	entero		4	Ruta
Rev-day	carácter		14	Día-Rev
Pay-day	carácter		14	Día-Pago

INDEXACIONES DE LA TABLA

Nombre de indexación(indicador primario)*	*Unico*	*Nombre del campo*
cust-address*	si	Machine-ID
		Cust-ID
		Address-no
name	no	Machine-ID
		Cust-ID
		Name

CRITERIO PARA EL CAMPO DE VALIDACIÓN

Cust-ID:

Criterio: Poder localizar(customer de cust-address)

Country-ID:

Criterio: Poder localizar(country de cust-address)

Tax-group-ID:

Criterio: tax-group-id eq "" de poder localizar(tax-group donde tax-group.tax -group-id eq cust-address.tax-group-id y tax-group.module-id eq "a/r")

Carrier-ID:

Criterio: Poder localizar(carrier de cust-address)

Route:

Criterio: Poder localizar(ar-route de cust-address)

Rev-day:

Criterio: CAN-DO("LU,MA,MI,JU,VI",Rev-day)

TABLA entity

Tabla de Entidades Federativas de las oficinas de Estafeta.

Nombre de campo	Tipo	Banderas	longitud	Descripción
Entity-ID	carácter	im	12	Entidad
Name	carácter	i	40	Nombre
Address	carácter		40	Dirección
GST-reg-no	carácter		20	R.F.C.
Control-ent	carácter	i	12	Ent-Control
Active	lógico		si/no	Activa

INDEXACIONES DE LA TABLA

Nombre de indexación(indicador primario)*	*Unico*	*Nombre del campo*
control-ent	no	Control-ent Entity-ID
entity*	si	Entity-ID
entity-name	no	Name

CRITERIO PARA EL CAMPO DE VALIDACIÓN

Entity-ID:

Criterio: entity-id ne ""

Control-ent:

Criterio: Poder localizar(gl-control de entity)

TABLA customer

Tabla donde se almacenan datos de clientes.

Nombre de campo	Tipo	Banderas	Longitud	Descripción
Machine-ID	carácter	im	4	Máquina
Cust-ID	carácter	im	8	Cliente
Name	carácter		60	Nombre
Legal-name	carácter		60	Razón Social
Search-name	carácter	i	16	Nom-Búsqueda
Address	carácter		40	Dirección
City	carácter	i	20	Ciudad
Prov	carácter		4	Estado
Postal-code	carácter	i	10	CP
Country-ID	carácter		4	País
Telephone	carácter	i	16	Teléfono
FAX	carácter		16	Fax
Report-code	carácter		12	Cod-Reporte
Status-code	carácter		1	Cod-Estatus
Term-ID	carácter		4	Término
Tax-group-ID	carácter		4	Grupo-Impto
Carrier-ID	carácter		4	Transp
Terr-ID	carácter		4	Territorio
Stmt-freq	carácter	m	1	Frec-Edo
Credit-limit	entero		10	Límite-Cred
Grace-days	entero		4	Días-Gracia
Cred-rating	carácter		4	% crédito
Interest-ID	carácter		4	Intereses
Stmt-type	carácter		1	Tipo-Edo
Ship-caracterge	carácter		2	Cargo-Embrq
Charge-cust	carácter	i	8	Cargo-Clte
Mail-cust	carácter		8	Corr-Clte
Entity-list	carácter		60	Lista-Entidad
Priority	entero		2	Prioridad
Comm-ID	carácter		4	Com
Create-date	fecha		99/99/99	Fecha Creación
Currency-ID	carácter		4	Divisa
AR-gl	carácter		8	CG-CxC
Tax-reg-no	carácter		20	NoReg-Impto
Non-maintained	lógico		si/no	No-Mantenible
SIC-code	carácter		8	CIS
Level-ID	carácter		4	Nivel
UDF-1	carácter		20	CDU-1
UDF-2	carácter	i	20	CDU-2
Phrase-ID	carácter		4	Frase
Cash	lógico		si/no	Ctdo
On-hold	lógico		si/no	En-Ret
Review-code	carácter		8	Código Revisión
Sales-list	carácter		60	Lista Vendedor

Lang-ID	carácter	2	Idioma
Perform-code	carácter	2	Código desempeño
Route	entero	4	Ruta
Rev-day	carácter	14	Día Revisión
Pay-day	carácter	14	Días de Pago

INDEXACIONES DE LA TABLA

Nombre de indexación(indicador primario)*	*Unico*	*Nombre del campo*
charge-cust	no	Machine-ID Charge-cust
city	no	Machine-ID City
customer*	si	Machine-ID Cust-ID
df2	no	UDF-2
GST-reg-no	no	Machine-ID GST-reg-no
postal-code	no	Machine-ID Postal-code
search-name	no	Machine-ID Search-name
telephone	no	Machine-ID Telephone

CRITERIO PARA EL CAMPO DE VALIDACIÓN

Cust-ID:
Criterio: cust-id ne ""
Country-ID:
Criterio: Poder localizar(country de customer)
Status-code:
Criterio: lookup(status-code,"a,I") > 0
Term-ID:
Criterio: Poder localizar(credit-term de customer)
Tax-group-ID:
Criterio: Poder localizar(tax-group donde tax-group.tax-group-id eq customer.
tax-group-id y tax-group.module-id eq "a/r")
Carrier-ID:
Criterio: Poder localizar(carrier de customer)
Terr-ID:
Criterio: Poder localizar(territory de customer)
Stmt-freq:
Criterio: lookup(stmt-freq,"m,s,w,o") > 0
Interest-ID:
Criterio: Poder localizar(interest de customer)

Stmt-type:
Criterio: lookup(stmt-type,"o,b,u") > 0
Ship-charge:
Criterio: lookup(ship-charge,"co,pr,pc") > 0
Comm-ID:
Criterio: comm-id eq "" de poder localizar(commission de customer)
Currency-ID:
Criterio: Poder localizar(currency de customer)
AR-gl:
Criterio: Poder localizar(gl-account donde gl-account.gl-code eq ar-gl)
Phrase-ID:
Criterio: phrase-id eq "" de poder localizar(phrase de customer)
Lang-ID:
Criterio: Poder localizar(wb_language de customer)
Route:
Criterio: Poder localizar(ar-route de customer)

TABLA factura

Tabla que guarda la información con las facturas que amparan las diferentes guías que se cobran en el sistema de sobrepeso.

Nombre de campo	Tipo	Banderas	Longitud	Descripción
Numero	entero	I	9	Número
Num-cliente	carácter	i	8	Núm-Cliente
Nombre-cli	carácter		60	Nombre-Cli
Direccion-cli	carácter		40	Dirección-Cli
rfc-cli	carácter		15	RFC-cli
fecha	fecha		99/99/99	Fecha Factura
monto	decimal		14	Monto
entidad	carácter	i	12	Entidad
num-impres	entero		3	Núm-impres
User-ID	carácter		8	Usuario
fecha-creacion	fecha		99/99/99	Fecha-Creación
cancelada	lógico		si/no	Cancelada
num-impre	decimal		9	núm-impre
num-oficina	entero		4	Núm-Oficina
num-dir	entero		1	Núm-Dir
servicio	carácter		3	Servicio
number-serv	entero		7	Núm-Servicios
Num-Interno	entero	I	9	No-Interno
Address-no	entero	m	9	Dirección No.

INDEXACIONES DE LA TABLA

Nombre de Indexación(Indicador primario)*	*Unico*	*Nombre del campo*
cli-fact	no	Num-cliente entidad

Nombre de Indexación(Indicador primario)*	*Unico*	*Nombre del campo*
*factura**	si	Numero Num-Interno

TABLA oficina

Tabla que contiene las oficinas que expiden guías.

Nombre de campo	Tipo	Banderas	Longitud	Descripción
num-oficina	entero	I	4	Núm Oficina
ofi-id	carácter	I	3	Sigla
nombre	carácter		50	Nombre
represent	carácter		20	Representante
Entity-ID	carácter	m	12	Entidad
Terr-Aer	carácter	I	1	Terrestre/Aéreo

INDEXACIONES DE LA TABLA

Nombre de Indexación(Indicador primario)*	*Unico*	*Nombre del campo*
oficina*	si	num-oficina
sigla	si	ofi-id Terr-Aer

CRITERIO PARA EL CAMPO DE VALIDACIÓN

Entity-ID:
Criterio: Poder localizar (entity donde entity.entity-id = entity-id)

TABLA servicio

Tabla que guarda los servicios que se ofrecen y el peso que amparan.

Nombre de campo	Tipo	Banderas	Longitud	Descripción
service-type	carácter	m	4	Tipo-Serv
description	carácter		40	Descripción
peso-ampara	decimal		10	Peso Amparado

INDEXACIONES DE LA TABLA

Nombre de Indexación(Indicador primario)*	*Unico*	*Nombre del campo*
default*	no	

TABLA guia

Tabla maestra con la información de guías con sobrepeso.

Nombre de campo	Tipo	Banderas	Longitud	Descripción
Num-Guia	decimal	i	12	Número Guía
origen	carácter		6	Origen
destino	carácter		6	Destino
Fec-Guia	fecha		99/99/99	Fecha Guía
sobrepeso	decimal		10	Sobrepeso
servicio	carácter	i	3	Servicio
medidas	decimal		10	Medidas
cop-scan	carácter		6	Ctro.Op.
monto	decimal		10	Monto
cliente	carácter	i	8	Cliente
Factura	entero	i	10	Factura
Guia-fact	decimal		10	Guia-Fact
Observ	carácter		3	Observ
num-oficina	entero	i	4	Núm-Oficina

INDEXACIONES DE LA TABLA

Nombre de indexación(indicador primario)*	*Unico*	*Nombre del campo*
cliente	si	Num-Guia
		cliente
factura	no	Factura
guia*	si	Num-Guia
num-ofi	si	servicio
		num-oficina
		cliente
		Num-Guia

TABLA scan

Tabla con información de las guías y el centro operativo donde fueron scaneadas.

Nombre de campo	Tipo	Banderas	Longitud	Descripción
Num-Guia	carácter	i	15	Núm-Guía
sobrepeso	decimal		10	Sobrepeso
medidas	decimal		10	Medidas
cop-scan	carácter	i	6	Ctro.Op.
destino	carácter		6	Destino

INDEXACIONES DE LA TABLA

Nombre de indexación(* indicador primario)	Unico	Nombre del campo
scan*	si	Num-Guia
		cop-scan

TABLA Guias-Cli

Tabla que almacena las guías con su respectivo número de cliente.

Nombre de campo	Tipo	Banderas	Longitud	Descripción
Guia-Ini	decimal	i	12	Guía-Ini
Cantidad	entero		6	Cantidad
cliente	carácter	i	8	Cliente
User-ID	carácter		8	Usuario
fecha	fecha		99/99/99	Fecha Guía
Update-date	fecha		99/99/99	Fecha-Act
num-oficina	entero		4	Núm-Oficina

INDEXACIONES DE LA TABLA

Nombre de Indexación(Indicador primario)*	*Unico*	*Nombre del campo*
cliente	si	Guia-Ini
		cliente
guias-cli*	si	Guia-Ini

CRITERIO PARA EL CAMPO DE VALIDACIÓN

cliente:

Criterio: Poder encontrar(customer donde customer.machine-id = '' y customer.cust-id = guias-cli.cliente)

TABLA precios

Tabla que contiene los servicios que generan sobrepeso y sus precios.

Nombre de campo	Tipo	Banderas	Longitud	Descripción
num-servi	entero	i	12	Num-Servi
nombre	carácter		40	Nombre
kilo-ampara	entero		12	Kilo-Ampara
precio	decimal		12	Precio
Observaciones	carácter		40	Observaciones
servi-id	carácter		5	Servi-ID
kilo-ampara-aer	entero		9	Kilo-Aer
precio-aer	decimal		12	Precio-Aer.

INDEXACIONES DE LA TABLA

Nombre de Indexación(Indicador primario)*	*Unico*	*Nombre del campo*
precios*	si	núm-servi

TABLA guia-concilia

Tabla que almacena las guías borradas de la tabla guia.

Nombre de campo	Tipo	Banderas	Longitud	Descripción
Num-Guia	decimal	i	12	Número Guía
origen	entero		6	Origen
destino	carácter		6	Destino
Fec-Guia	fecha		99/99/99	Fecha Guía
sobrepeso	decimal		9	Sobrepeso
servicio	carácter		3	Servicio
medidas	decimal		9	Medidas
cop-scan	carácter		6	Ctro.Op.
monto	decimal		14	Monto
cliente	carácter		8	Cliente
Factura	entero		12	Factura
Guia-fact	decimal		12	Guía-Fact
Observ	carácter		3	Observaciones
num-oficina	entero		4	Núm-Oficina

INDEXACIONES DE LA TABLA

Nombre de indexación(indicador primario)*	*Unico*	*Nombre del campo*
guia-concilia*	si	Núm-Guia

TABLA sob-control

Tabla que controla los consecutivos de la factura, el número interno y fiscal del sistema de sobrepeso.

Nombre de campo	Tipo	Banderas	Longitud	Descripción
Control-ent	carácter	mi	12	Ent-Control
Fiscal-yr	entero		4	Año Fiscal
Fiscal-prd	entero		2	Prd-Fiscal
num-impre	decimal		10	Núm impresión
Trans-no	entero	m	10	No-Trans

INDEXACIONES DE LA TABLA

Nombre de indexación(indicador primario)*	*Unico*	*Nombre del campo*
sob-control*	si	Control-ent

CRITERIO PARA EL CAMPO DE VALIDACIÓN

Control-ent:

Criterio: Poder localizar(entity donde entity.entity-id eq control-ent)

TABLA user-default

Tabla que lleva el control de los usuarios que realizan transacciones en el sistema.

Nombre de campo	Tipo	Banderas	Longitud	Descripción
User-ID	carácter	ml	8	Usuario
Terminal-ID	carácter	im	12	Terminal
Module-ID	carácter	mi	4	Módulo
Entity-ID	carácter		12	Entidad
Fiscal-yr	entero		4	Año Fiscal
Fiscal-prd	entero		2	Prd-Fiscal
Default-date	fecha		99/99/99	Fech-Def

INDEXACIONES DE LA TABLA

Nombre de indexación(indicador primario)*	*Unico*	*Nombre del campo*
user-default*	si	User-ID Module-ID Terminal-ID

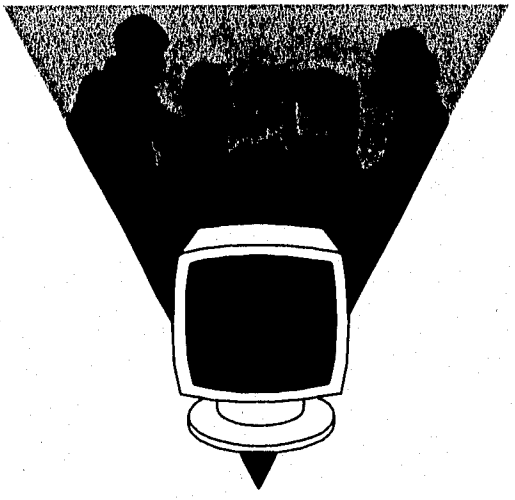
CRITERIO PARA EL CAMPO DE VALIDACIÓN

User-ID:
Criterio: Poder localizar(user-status de user-default)

Module-ID:
Criterio: Poder localizar(module de user-default)

Entity-ID:
Criterio: Poder localizar(entity de user-default)

Una vez obtenida la carta estructurada y la estructura de los archivos de las bases de datos, procederemos al desarrollo del sistema utilizando las técnicas de diseño de bases de datos relacionales y las de programación estructurada de PROGRESS, además utilizaremos como herramienta de desarrollo de menúes la utilería de SENTAI, como lo veremos en el siguiente capítulo.



CAPITULO V

DESARROLLO

V.1.- Conceptos generales

En este punto daremos una pequeña introducción a lo que son las bases de datos, el porque se utilizan así como los tipos de enfoques que existen.

Base de datos

Una base de datos es un conjunto de datos relacionados. El almacenamiento de datos para una base se logra empleando uno o más archivos.

Sistema de bases de datos

Un sistema de bases de datos es, en esencia, un sistema de mantenimiento de registros basado en computadoras, es decir, un sistema cuyo propósito general es registrar y mantener información. Un sistema de bases de datos incluye cuatro componentes principales: *datos*, *hardware*, *software* y *usuarios*.

Un sistema de bases de datos proporciona a la empresa un control centralizado de sus datos de operación. Algunas de las ventajas de tener un control centralizado de los datos son las siguientes :

- *Puede reducirse la redundancia.* En sistemas que no usan bases de datos, cada aplicación tiene sus propios archivos privados. Esto a menudo origina enorme redundancia en los datos almacenados (duplicidad de datos en diferentes archivos), así como desperdicio resultante del espacio de almacenamiento.
- *Puede evitarse la inconsistencia.* Una base de datos es inconsistente cuando existe redundancia y alguno de los registros existentes no se modifica después de una actualización, creación o supresión.
- *Los datos pueden compartirse.* Las aplicaciones existentes pueden compartir los datos de la base de datos, además de que es factible desarrollar nuevas aplicaciones que operen con los mismos datos almacenados. Las necesidades de datos de las nuevas aplicaciones pueden atenderse sin tener que crear nuevos archivos almacenados.
- *Pueden hacerse cumplir las normas establecidas.* Con un control central de la base de datos se pueden cumplir todas las normas aplicables a la representación de datos, éstas pueden comprender la totalidad o parte de lo siguiente: normas de la compañía, de instalación, departamentales, industriales, nacionales o internacionales.
- *Pueden aplicarse restricciones de seguridad.* Al tener los datos centralizados se puede asegurar que el único medio para accesar la base de datos sea a través de los canales establecidos y, por tanto, definir controles de autorización para que se apliquen cada vez que se intente el acceso a datos sensibles.
- *Puede conservarse la integridad.* El problema de la integridad es garantizar que los datos de la base sean exactos. La inconsistencia entre dos entradas que representan al mismo "hecho" es un ejemplo de falta de integridad.

Tipos de acceso

En un principio, los datos eran meros parámetros dados al programa para su ejecución y obtención de unos resultados. A menudo estaban incluidos en el propio código fuente de los programas, por lo que, si cambiaban los datos había que modificar el programa y recompilarlo.

Posteriormente, en lo que podría llamarse la época de los ficheros, los datos salen de los programas y se agrupan en registros que se almacenan en dispositivos generalmente magnéticos (cintas o discos) para su tratamiento. El acceso a estos registros podía ser :

- *Secuencial.* La lectura de registros se realiza de forma consecutiva desde el inicio del fichero, por lo que para localizar un registro determinado de un fichero hay que leer todos los registros que le preceden. Es el acceso adecuado para un almacenamiento en un soporte sin posibilidad de acceso directo (como la cinta magnética) y para ficheros usados para tratamientos masivos de registros. Los atributos de los datos se clasifican por categorías, de manera que los registros individuales contienen todos los valores de atributos de los datos en el mismo orden y posiblemente en la misma posición.
- *Secuencial indexado.* Permiten un acceso directo mediante índices y también un acceso secuencial y ordenado a través del campo clave. Un índice consiste en un conjunto de anotaciones, una para cada registro dato, que contienen el valor de un atributo llave para ese registro y un apuntador que permite el acceso inmediato a ese registro.
- *Indexado.* El acceso a los registros se logra sólo mediante uno o más índices. Este se utiliza cuando no existe necesidad de que el manejo secuencial proporcione un acceso serial eficiente. Un índice estará asociado con un atributo llave y pueden existir índices para todos los atributos de los que se espere un argumento de búsqueda.
- *Directo.* En este caso el dispositivo sobre el que se almacenan los registros deberá tener la posibilidad de acceso o posicionamiento directo en una determinada posición o dirección dentro del fichero. Para la localización de un registro es necesario un campo o clave que identifique al registro.
- *Multianillo.* Está orientado al procesamiento eficiente de subconjuntos de registros. Un subconjunto se define como un grupo de registros que contienen algún valor común de atributo. Este enfoque se utiliza mucho con sistemas de base de datos.

Tipos de enfoques

Para determinar qué estructuras de datos y operadores asociados debe soportar el sistema es conveniente clasificar a los sistemas de bases de datos de acuerdo con el enfoque que adoptan. Los tres enfoques mejor conocidos son :

- el enfoque relacional
- el enfoque jerárquico y
- el enfoque de red

Enfoque relacional

El modelo relacional de datos es definido en 1970 por el Dr. F. Codd por medio de una serie de reglas cuyo objetivo es lograr la independencia de la representación lógica de los datos de su almacenamiento físico.

Esta independencia física / lógica se refiere a tres aspectos:

- *Independencia de la ordenación*, es decir, que el resultado obtenido en un acceso no dependa de cómo estén ordenados los datos físicamente.
- *Independencia de la indexación*, separando los índices de los datos haciendo que la creación y mantenimiento sean manejados por el sistema.
- *Independencia de los caminos de acceso*, haciendo que la navegación a través de los datos no tenga que estar previamente establecida consiguiendo así unas formas de acceso más flexibles.

A través de sus reglas, Codd pretende lo siguiente:

- Independencia física / lógica
- Eliminación de redundancia
- Flexibilidad
- Uniformidad
- Sencillez
- Sólido fundamento teórico

El modelo relacional se distingue por la reducción de los datos a estructuras planas o *tablas* con filas y columnas. A estas tablas se les denomina *relaciones* y equivalen cada una de ellas a un *fichero*. Un *dominio* es un depósito de valores del cual se sacan los que aparecen en una columna específica. Un *atributo* representa el uso de un dominio dentro de una relación. Una *llave primaria* es un grupo de columnas o columna que identifican de manera única cada renglón de la tabla, todas las tablas deben tener una llave primaria.

Una característica crucial de la estructura de datos relacional es que las asociaciones entre renglones se representan únicamente por valores de datos en columnas sacadas de un dominio común.

Hay una correspondencia entre los términos utilizados al hablar de ficheros, tablas y relaciones. Así, si un fichero corresponde a una tabla, cada registro del fichero se corresponde con una fila y cada campo dentro del registro con una columna.

Enfoque jerárquico

En el modelo jerárquico los datos están estructurados en forma arborescente y las relaciones entre los diferentes tipos de registros se resuelven mediante punteros o enlaces entre ellos. Se establece una jerarquía de modo que las relaciones entre un registro y otro relacionado con él (relación padre-hijo) tienen como condición que un registro "hijo" no puede existir si no existe el registro "padre" asociado a él.
El tipo de registro en el tope del árbol se conoce en general como "raíz". En general, la raíz puede tener cualquier número de dependientes; cada uno de éstos puede tener

cualquier número de dependientes de nivel inferior, y así sucesivamente, hasta cualquier número de niveles.

Enfoque de red

Si en la estructura jerárquica se permiten relaciones entre "hermanos", es decir, entre registros de un mismo padre lógico, tenemos una base de datos en red.

Debe considerarse a la base de datos jerárquica como un tipo de base de datos en red con ciertas restricciones. Por tanto, es aplicable a este tipo de bases de datos todo lo dicho para las bases de datos jerárquicas en cuanto a tipos de enlaces, accesos, etc.

Una red es una estructura más general que una jerarquía porque una ocurrencia de registro específica puede tener cualquier número de superiores inmediatos (así como cualquier número de dependientes inmediatos), es decir, no se está limitado a un máximo de uno, como ocurre con una jerarquía. De esta manera, un enfoque de red permite modelar una correspondencia de muchos a muchos de manera más directa que el enfoque jerárquico.

Normalización

Un problema del diseño de bases de datos es el decidir qué relaciones se necesitan y qué atributos deben tener dado un conjunto de datos que se va a representar en una base de datos. Para esto podemos auxiliarnos de la teoría de normalización que a continuación explicamos.

Formas normales

La teoría de la normalización está basada en el concepto de *formas normales*. Se dice que una relación está en una forma normal particular si satisface cierto conjunto específico de restricciones. A continuación presentamos cada una de las formas normales.

Primera forma normal

Dada una tabla T, con una llave primaria P y un atributo A, se dice que la tabla está en primera forma normal si y sólo si el valor del atributo A en cualquier renglón depende del valor de la llave primaria P en ese renglón.

Segunda forma normal

Dada una tabla T en primera forma normal, con una llave primaria P en múltiples columnas con componentes $P_1,P_2,...,P_n$ y un atributo A, se dice que la tabla está en segunda forma normal si y sólo si el valor del atributo A en cualquier renglón depende de los valores $P_1,P_2,...,P_n$ para ese renglón. Las tablas con llave primaria en una sola columna siempre están en segunda forma normal.

Tercera forma normal

Dada una tabla T en segunda forma normal, con una llave primaria P y dos atributos A_1 y A_2, se dice que la tabla está en tercera forma normal si y sólo si el valor del atributo A_1 en cualquier renglón no depende del valor del atributo A_2 y viceversa

V.2.- Normas de Programación

La programación es un arte que depende de la habilidad individual, la atención al detalle y el conocimiento de cómo utilizar los instrumentos disponibles de la mejor manera. El programador debe comprender el sistema de computación, saber algo de teoría de programación y adquirir práctica. La programación es una actividad práctica que sólo se puede aprender mediante la experiencia. No obstante, se puede utilizar la experiencia de otros para aprender principios generales que ayuden al programador a evitar algunos errores.

Para el desarrollo del sistema de sobrepeso conjuntamos precisamente la experiencia adquirida en el plano laboral por cada uno de nosotros, para tomar la decisión de aplicar la metodología conocida como programación estructurada. Este término se usa de manera amplia y un tanto vaga en los análisis de programación. Parece que no hay una definición generalmente aceptada del término; en algunos casos significa programación sin el empleo de **goto**, en otros casos significa adoptar una metodología de diseño descendente y aun en otros implica limitar las construcciones de control de la programación a ciclos **while** y proposiciones **if**.

V.3.- Descripción de la utilería SENTAI

SENTAI es un ambiente de desarrollo basado en el lenguaje de cuarta generación Progress. Este ambiente es diseñado para permitir a los desarrolladores crear funciones de aplicación sin tener que rehacer porciones de código repetidamente.

Menuing, seguridad, *report frontend* y consistencia son sólo algunos beneficios que obtenemos con SENTAI. El único código requerido por cada aplicación es incorporado dentro de la estructura de la plantilla completa de SENTAI.

SENTAI utiliza una filosofía orientada a objetos permitiendo reutilizar bloques de código. Cada utilería de SENTAI es un archivo parametrizado en Progress, que tiene numerosas opciones disponibles para el programador.

SENTAI cubre las necesidades del programador, dando la opción de usar o no la presentación de menús, manejo de pantallas y control de programa a través de teclas predefinidas por el programador (*keystroke).*

SENTAI no es un generador de código, los generadores de código tienen una limitación inherente de no permitir la edición subsecuente de código generado sin remover la capacidad de generación de actualización de código.

SENTAI permite el uso de plantillas de diseño. La ventaja del uso de plantillas es que los cambios sean hechos para una biblioteca central de código y se aplique a todos llamando a esas plantillas. Esto permite al usuario remarcar las plantillas y tener esos

realces disponibles para todo el código que llama la plantilla sin regeneración de código.

SENTAI usa estas plantillas generalizadas en conjunción con registros de la base de datos que llaman diferentes programas en el contexto. Esto permite que una plantilla simple sea usada en muchos contextos y que el contexto sea determinado por las banderas en la que llama al registro. Por ejemplo, la plantilla Setup maneja todos los procesos de: agregar, cambios y borrar. El registro que llama al programa indica el modo en que fue seleccionado por el usuario para que la plantilla funcione adecuadamente.

Los programas que el usuario puede correr están definidos como funciones en SENTAI. Cada función puede ser de muchos tipos. Los tipos posibles son: *screen*, *transaction*, *subscreen*, *report*, *form* y *batch*. El usuario puede correr directamente funciones de *screen*, *transaction*, *report* y *batch*. *Subscreen* y *forms* están sólo disponibles como programas cuando son llamados dentro de otro programa. Todas las funciones definidas arriba están almacenadas con su programa ejecutable (archivo de sistema operativo que contiene el código fuente o código objeto) en el archivo wb_function.

El camino para que los programas estén disponibles durante el uso de otros programas es a través del uso del registro wb_help. Las funciones que incluyen *subscreen* pueden ser agregadas al archivo wb_help y estarán disponibles a los usuarios como esta especificado en el registro de ayuda. Los registros de ayuda pueden ser tanto globales (sistema amplio) como específicos, ya sea a un archivo, campo o programa de modo definido. Esto permite que las *subcreens* sean enganchadas a un nivel de campo por alteración de un registro de la base de datos.

SENTAI consiste de una serie de funciones que el programador usa para manejar el desempeño del proyecto terminado al usuario y una biblioteca de plantillas que ofrece un conjunto de características que pueden ser aplicadas fácilmente a cualquier base de datos de Progress. Los programas de plantilla manejan tareas de programación comunes tales como: búsqueda, mantenimiento de archivos, marca de archivo, petición de programas, programas de acceso a transacción y formato de reportes.

Tipo de funciones definidas dentro de SENTAI

Group. Un *group* es una colección de funciones (*screen*, *report*, *transaction*, etc) que están presentados juntos sobre un menú para el usuario. Cada grupo puede contener 15 funciones y la seguridad puede ser un nivel de grupo para una expresión can-do de departamentos o usuarios. Todos los grupos están almacenados en el archivo wb_function con un tipo "g".

Screen. Un *screen* es una función que es diseñada para aceptar datos de entrada de un usuario. Algunos ejemplos de funciones *screen* incluyen *setup* o petición de programas. Todas las *screen* son almacenadas como un registro wb_function con un tipo "s".

Subscreen. Son programas que no pueden ser ejecutados directamente por el usuario. Estos programas son llamados por otros procedimientos ya que necesitan alguna forma de entrada (por ejemplo customer.change necesita que el registro customer sea seleccionado por el usuario). Los *subscreen* son conectados con sus funciones

relacionadas a través del uso de registros de wb_help. Los *subscreen* son almacenados como registros en wb_function con un tipo "u".

Report. Son programas que usará la opción estandar Report de SENTAI para la elaboración de reportes. Las opciones son usadas para ofrecer criterios de selección al usuario de una manera consistente. Los *reports* son registros de wb_function con un tipo "r".

Form. Una *form* es un tipo de reporte que no ofrece opciones a los usuarios y como *subscreen* puede ser ejecutada dentro de una función. Un posible uso de las *forms* sería la demanda de impresión de documentos en un programa dado. Las *forms* están en los registros de wb_function con un tipo "f".

Transaction. Son idénticos a los *screens*, la característica adicional es que reduce el número de funciones posibles seleccionadas durante la búsqueda de una transacción. Las *transactions* están en los registros de wb_function con un tipo "t".

Batch. Un *batch* es una serie de reportes que pueden ser ejecutados bajo un simple identificador. Cada *batch* tiene una serie de funciones *batch* asociadas con el mismo *batch*. Estas funciones *batch*, junto con las opciones *batch*, pueden correr cuando el *batch* se ejecuta. Los *batches* están en registros de wb_function con un tipo "b".

Seguridad del sistema

SENTAI maneja tablas organizacionales por departamentos. Algunas de las tablas más importantes son:

- Tabla USER
- Tabla USER-STATUS
- Tabla DEPARTAMENT
- Tabla MODULE

SENTAI opera a través de funciones, cada una de las cuales tiene un módulo asignado, es decir, cada función pertenece a un módulo. Por ejemplo, cuando se da de alta a un usuario, le especificamos a SENTAI a qué departamento pertenece ese usuario, figura 4.1. A su vez cada departamento tiene definidos los módulos a los que puede tener acceso, así por ejemplo, el departamento de Contabilidad sólo puede accesar al módulo de Contabilidad (GL).

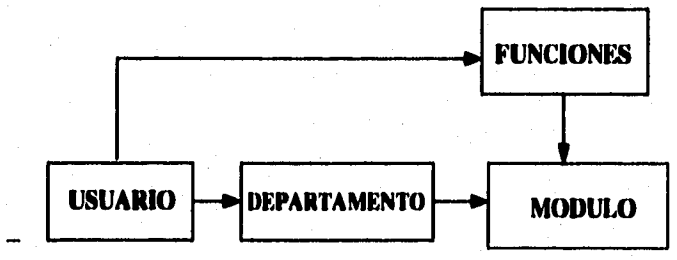


Figura 4.1 Ejemplo de funciones de SENTAI.

Para el mismo ejemplo, el Departamento de Sistemas es el único que puede tener acceso a todos los módulos.

Para el caso en estudio, el área de sobrepeso de Ceylán solo tiene acceso al módulo de sobrepeso, lo cual conlleva a un manejo de seguridad de la información que es llevado a cabo por SENTAI.

SENTAI maneja dos tablas principales que son :

- wb_functions, que contiene todas las operaciones que puede desarrollar SENTAI a través de funciones.
- wb_messages, que proporciona los mensajes cuando existe algún error.

Expresiones CAN-DO

La metodología tradicional "desde/hasta" para seleccionar datos se encuentra limitado por su diseño. SENTAI ofrece una técnica para seleccionar datos por medio del uso de expresiones CAN-DO de Progress. Las expresiones CAN-DO pueden consolidar datos de una variedad de grupos donde la codificación es no-secuencial y el agrupamiento no está determinado.

Una expresión CAN-DO es un método que utiliza una forma libre para especificar los datos que serán seleccionados para un reporte dado. La utilización de una serie de caracteres comodines y de exclusión, provee una posibilidad casi infinita de combinaciones. Una serie de reglas simples gobiernan la creación de una expresión CAN-DO válida. Un resumen de caracteres involucrados y su significado se muestran a continuación.

Carácter	Propósito
*	Reemplaza caracteres múltiples
.	Reemplaza un sólo carácter
!	Excluye una combinación de caracteres
,	Separa elementos

Por ejemplo:

!MEMRECBIC,* selecciona todo menos la entidad MEMRECBIC.
!*BIC,* selecciona todo excepto los productos "BIC".
!*REC*,* selecciona todo con excepción del centro de costo "REC"

V.4.- Descripción de PROGRESS

PROGRESS es un sistema diseñado para crear aplicaciones. Una aplicación es una solución computarizada para resolver algunos problemas de una empresa.

Normalmente, estas aplicaciones pueden desarrollarse simplemente para resolver problemas en la ocupación que se lleva a cabo, y usualmente se desarrollan en computadoras utilizando diferentes sistemas de archivos o base de datos. PROGRESS

desarrolla estas aplicaciones fácilmente. Todas las aplicaciones están basadas en tres partes:

Un espacio para almacenar la información
Un modo para traer la información
La información misma

PROGRESS cuenta con dos de estas tres partes. Esto es, PROGRESS almacena la información y la búsqueda fácilmente para traer los datos.

PROGRESS esta compuesto de cinco componentes:

- Base de datos relacionales, donde una base relacional es un lugar para almacenar información.
- Diccionario de datos, donde se describe la información que esta almacenada en la base de datos.
- El editor de procedimientos es una herramienta para escribir, complementadose con el uso del lenguaje de aplicación de PROGRESS para escribir los procedimientos donde se desarrollará la aplicación.
- PROGRESS tiene la herramienta de Formateador de Pantallas y reportes con el cual se pueden crear los reportes o las pantallas en forma sencilla.

PROGRESS organiza la información, o los datos, de una forma lógica, que es similar a trabajar con un archivo de papeles.

PROGRESS organiza la información de manera sencilla y fácil de localizar. Se crea ***database*** el cual esta compuesto por diversos ***files***. Donde un ***file*** es simplemente la acumulación de datos por tipo de cosas. PROGRESS acumula la información en una computadora en forma más organizada. A la vez un archivo esta compuesto por ***records*** y al final un ***record*** es formado por ***fields***. Un ***field*** guarda un valor para el ***record***.

PROGRESS utiliza en sus ***fields*** a los ***indexes*** para encontrar los ***record*** con mayor eficiencia. Puede usarse en ***field*** simples o una combinación de ***fields*** para crear un ***indexe***. Para decidir que archivo se usara para los ***indexes***, hay que escoger la forma de como se accesan los datos.

Además, PROGRESS puede usar ***indexes*** para ordenar ***records*** más eficientemente, permitiendo que se puedan generar reportes según se requieran.

V.5.- Entradas y Salidas del sistema

El sistema de sobrepeso proporciona diferentes pantallas con las cuales interactúan los usuarios. A continuación en la figura 4.2 se muestra la carta estructurada de entradas y salidas, en ella se verán las diferentes pantallas, que se irán explicando así como la manera de manejarlas a lo largo del capítulo.

Entradas del sistema

Se cuenta con un conjunto de teclas que realizan funciones determinadas, las cuales nos auxilian a navegar dentro del Sistema. Estas teclas las mencionamos a continuación:

Tecla	Función
F1	Aceptación
F2	Invocación de ayuda
F4	Tecla de Salida
F8	Tecla de Detalle del registro

Tabla 4.1 Teclas de Funciones.

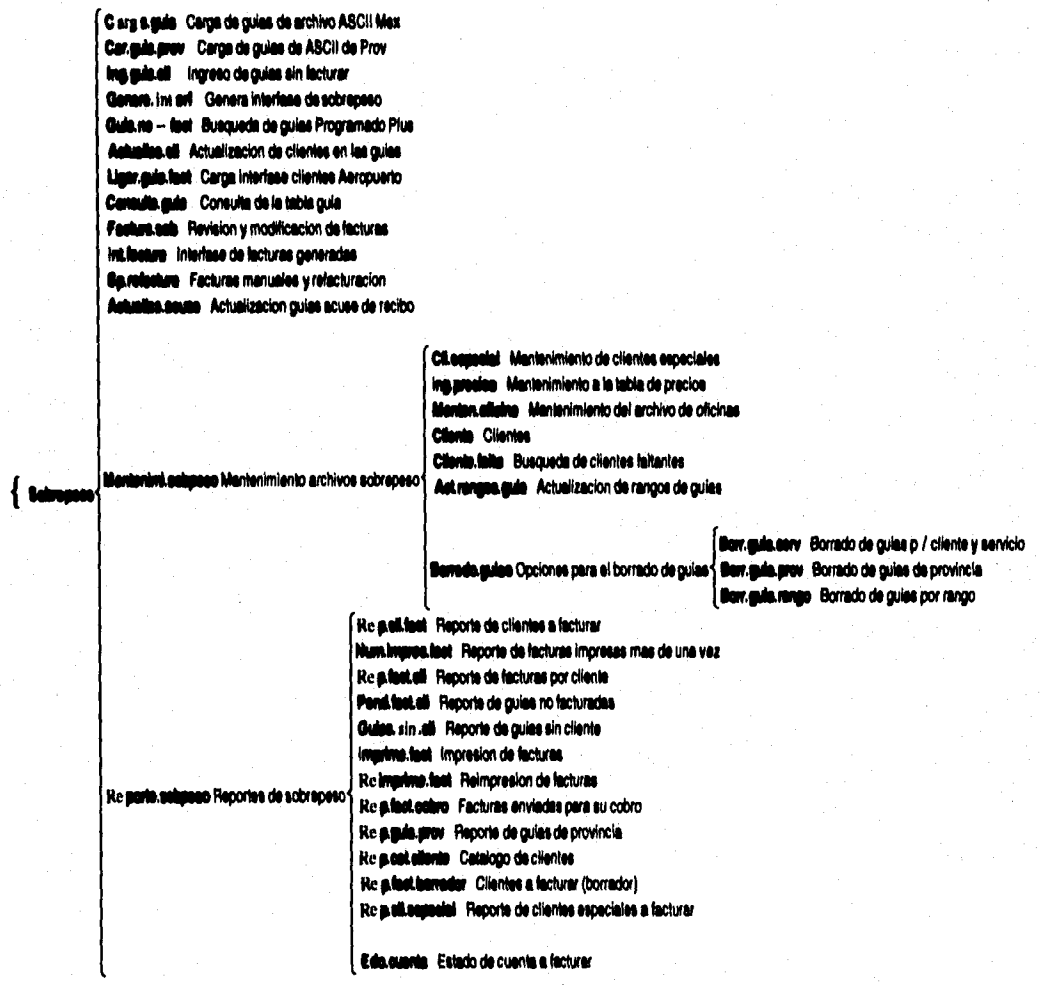


Figura 4.2 Carta estructurada de entradas/salidas.

Pantalla principal del sistema de sobrepeso

En la figura 4.3 se muestra la primer pantalla del sistema de sobrepeso que nos muestra los módulos que conforman el sistema. Debido a que el sistema funcionará de manera independiente de SENTAI, por el momento el único módulo que se nos presenta es justamente el de sobrepeso. Para accesar esta opción es necesario oprimir la tecla *Enter;* para las opciones salir y terminar se navega con las flechas hacia abajo hasta llegar a la opción deseada y se oprime la tecla *Enter.*

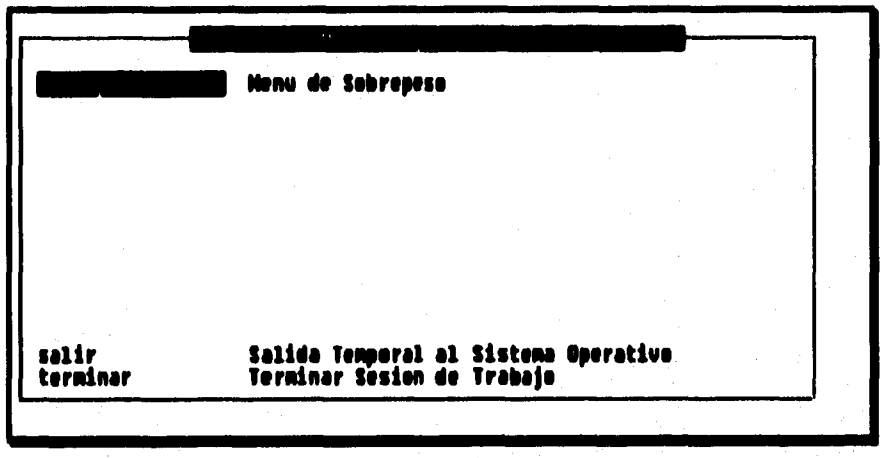


Figura 4.3 Pantalla principal.

Menú principal del sistema de sobrepeso

El menú principal del sistema de sobrepeso se muestra a continuación en la figura 4.4. Este menú se conforma por el momento por 12 opciones y dos submenús. Es importante hacer mención que el menú puede ir creciendo en sus opciones conforme las necesidades lo requieran.

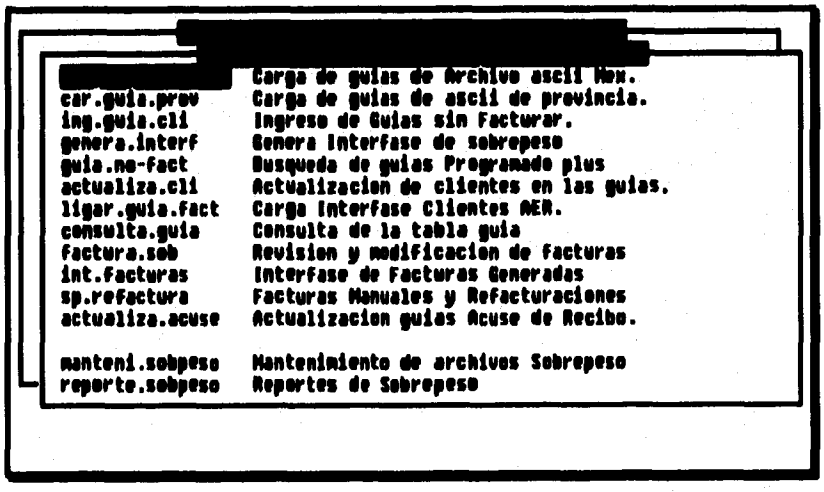


Figura 4.4 Menú de sobrepeso.

Como se puede observar, el menú está conformado por las opciones de carga, ingreso, generación, búsqueda, actualización, etc. El submenú está integrado por las opciones de mantenimiento de archivos y reportes de sobrepeso. Como en el menú principal, las opciones pueden accesarse posicionandose en ellas y oprimiendo la tecla *Enter*.

Pantalla para carga de guías leídas en los centros operativos

Las opciones 1 y 2 de nuestro menú principal nos muestran las funciones **carga.guia** y **car.guia.prov**. Estas funciones llevan a cabo la carga de los archivos que serán mandados a facturación Ceylán por los centros operativos que llevan a cabo el pesado de los paquetes. La figura 4.5 nos muestra la pantalla perteneciente a dichas funciones.

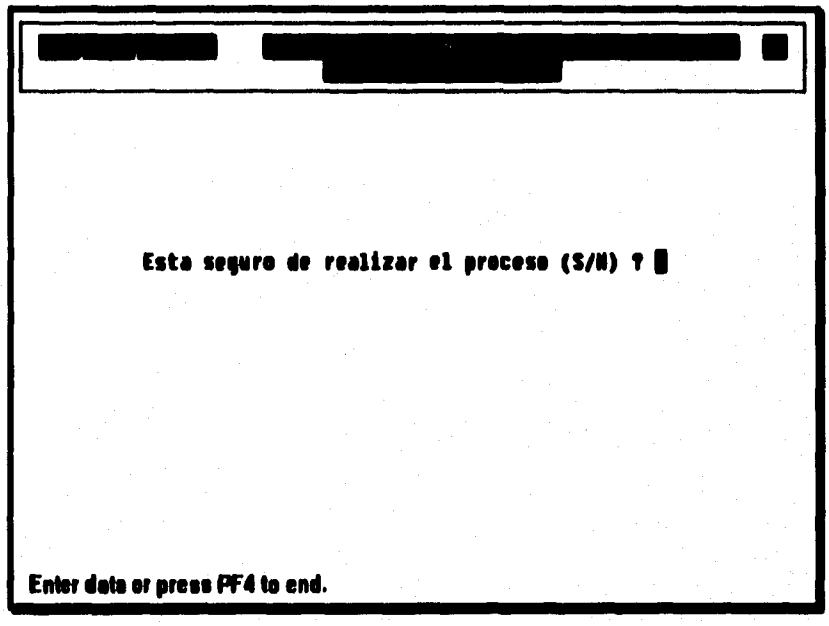


Figura 4.5 Pantalla de carga.guia.

La función **carga.guia** lleva a cabo la carga de guías leídas en Centro Operativo Ceylán y el procedimiento a seguir es el siguiente:

1. Recibir Archivo ASCII de la información recabada en Centro Operativo Ceylán. Depositar el archivo en la computadora localizada en facturación Ceylán que funge como servidor.

2. Recibir Archivos ASCII de la información recabada en los centros operativos de Guadalajara y Monterrey. Depositar dichos archivos en el servidor de facturación Ceylán. El depósito de estos archivos deberá ser realizado directamente por el encargado de sistemas para el caso de GDL y MTY y por el operador del sistema de sobrepeso para el caso de CEYLAN. Este depósito deberá llevarse a cabo **vía Módem**.

3. Una vez que estemos seguros de tener nuestros archivos en el servidor, correr primero la función **carga.guia**, diciéndole que queremos realizar el proceso y luego tecleando el nombre del archivo que contiene la información.

4. Correr la función **car.guia.prov** para cargar la información de provincia. Es muy importante realizar los pasos 3 y 4 en riguroso orden para poder asegurar que queden en el sistema las guías leídas con el mayor sobrepeso, ya que esta función realiza una sustitución de las guías repetidas.

Pantalla de ingreso de guías sin facturar

La opción tres del menú principal ejecuta la función **Ing.guia.cli**. Esta función realiza el ingreso de todas aquellas guías que se otorgan sin una factura que la cubra, como es el caso del servicio Programado Plus, Intercambios, etc. Por ejemplo, en la figura 4.6. presentamos la opción del servicio Programado Plus. En ella se desglosa el número de la guía inicial, la cantidad de guías que se dieron y la clave de a que cliente se otorgaron. En caso de que necesitemos consultar alguna otra información bastará con posicionarse en la guía deseada y oprimir la tecla **F8**. Inmediatamente aparecerá en pantalla toda la información referente a esta guía.

Dentro de la misma pantalla, en el recuadro inferior, se presentan las diferentes acciones que podemos realizar, como son agregar, borrar y modificar una guía. Esta pantalla aparecerá siempre que oprimamos la tecla **F2**.

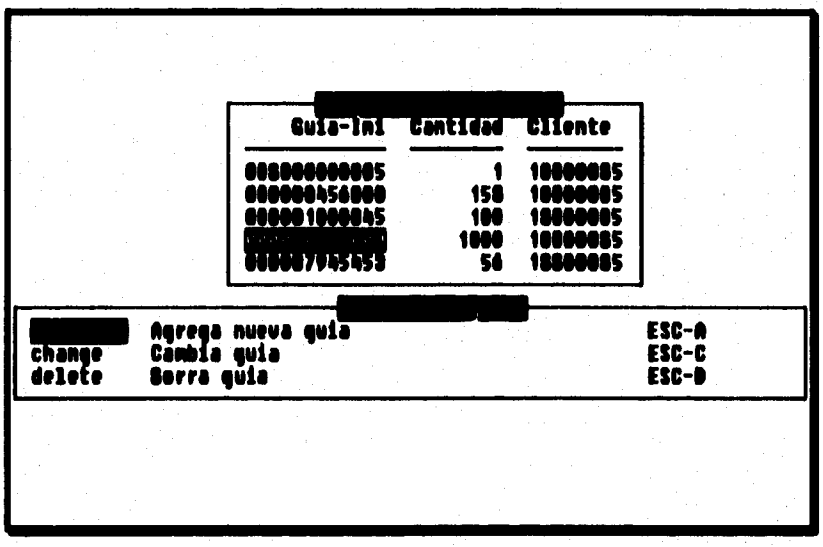


Figura 4.6 Pantalla de ingreso de guías del servicio Programado Plus.

En la figura 4.7 se presenta la pantalla que captura los campos de la guía. Como se puede ver aquí aparecen todos los campos que componen nuestra guía para su manejo por parte del usuario.

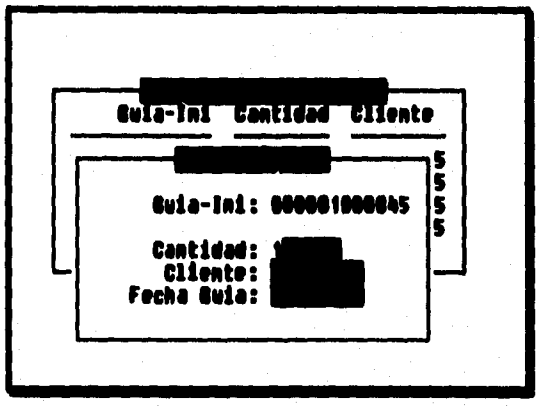


Figura 4.7 Pantalla de ingreso de guías del servicio Programado Plus.

Generación de interfase de sobrepeso

La generación de la *interfase* de sobrepeso es una de las funciones más importantes del sistema, ya que con ella generamos el archivo de *interfase* que será mandado a Aeropuerto. Este archivo contiene la información de las guías de las cuales desconocemos su cliente. El procedimiento para generar este archivo es el siguiente:

1. Seleccionar la opción 4 del menú principal **genera.interf.**
2. En la figura 4.8 se muestra en la pantalla superior los destinos que vamos a poner para nuestro proceso. Se recomienda que estos destinos sean siempre los mismos, por lo que bastará con que oprimamos la tecla **F1.**
3. La pantalla inferior se refiere a las opciones que vamos a manejar. Si queremos mandar todas las guías pendientes de todos los centros operativos basta con oprimir F1. Si queremos sólo un rango de algún centro en específico haremos lo siguiente:

   Supongamos que queremos de la guía 100 a la 5000 del Centro Operativo Monterrey, las opciones deberán quedar de la siguiente manera:

   guia-inicial: 000000000100
   guia-final: 000000005000
   Centro: MTY.

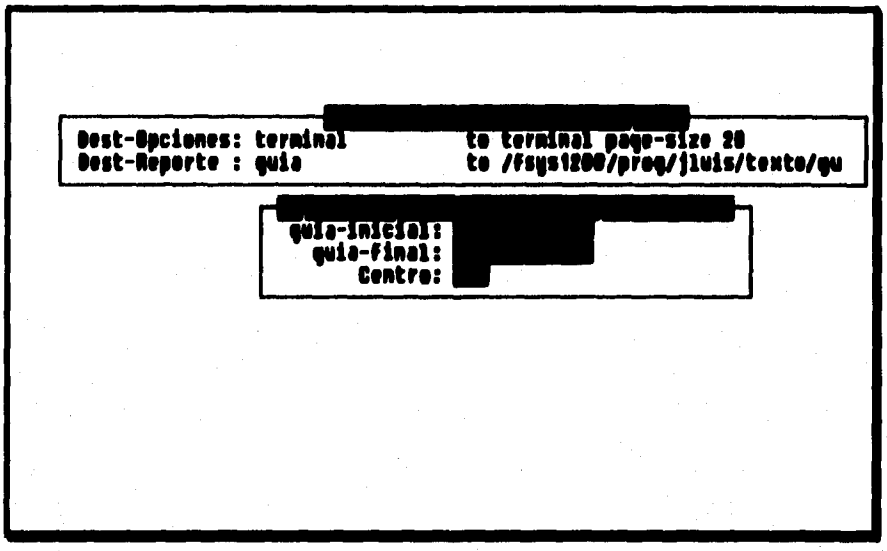


Figura 4.8 Pantalla Genera interfase de sobrepeso.

Búsqueda de guías Programado Plus

Esta búsqueda se lleva a cabo mediante la función **guia.no-fact** y es muy importante, ya que con ella buscamos aquellas guías que se dieron sin una factura correspondiente y que sin embargo tenemos registradas en el sistema. Esta **siempre** deberá ser corrida antes de mandar el archivo de interfase a Aeropuerto, para así evitar búsquedas infructuosas, ya que no debemos de perder de vista que en el sistema SENTAI no tenemos ninguna factura que registre dichas guías.

En la figura 4.9 se muestra la pantalla de dicha función, la cual nos pregunta el archivo que contiene las guías a las cuales se va a buscar el cliente correspondiente. Este archivo es el mismo que generamos con la función **genera.interf** y se llama **guía**.

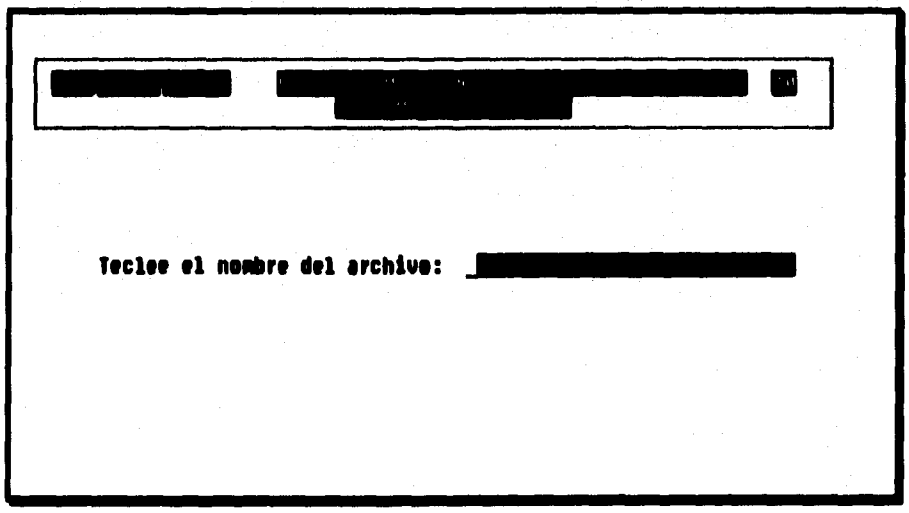


Figura 4.9 Pantalla de búsqueda de clientes.

Actualización de clientes en las guías

La opción **actualiza.cli** del menú principal nos permite ingresar el cliente correspondiente en todas aquellas guías en las cuales no se les encontró en el sistema de facturación de SENTAI, o que en su defecto se les asignó un cliente erróneo.

Su manejo es muy sencillo:

- En la pantalla superior de la figura 4.10 se ingresa el número de guía deseado.
- En la pantalla inferior se despliega la información referente a dicha guía además de ingresarse el cliente correspondiente.

Esto significa que para esta actualización sólo necesitamos capturar dos campos: numero de guía y cliente.

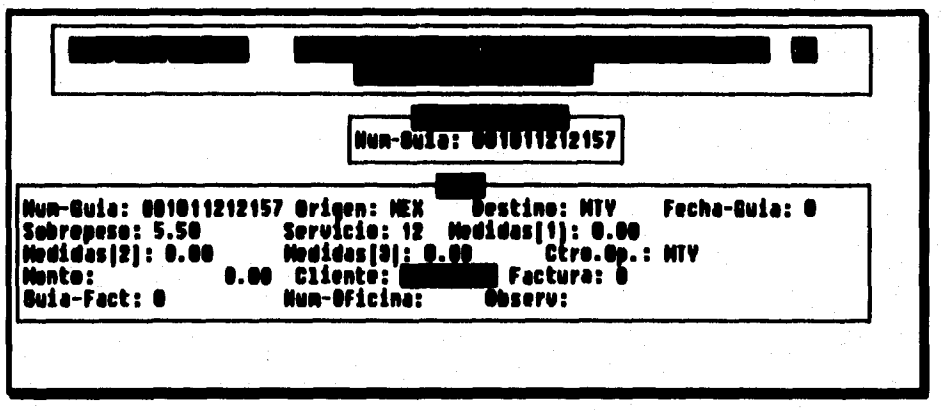


Figura 4.10 Pantalla de actualización de clientes.

La Carga Interfase clientes Aeropuerto

función para actualizar las guías con los clientes que si se encontraron en el sistema de facturación SENTAI, se llama **ligar.guia.fact.**, esto se muestra en la figura 4.11 y el proceso a seguir es el siguiente:

1. Coordinar con Aeropuerto para recibir vía módem el archivo ASCII de las guías a las que se encontró el cliente.
2. Asegurarse de que el archivo esté en el servidor de facturación Ceylán. El nombre del archivo estará compuesto de la siguiente forma:

   gf + número de mes + día mes + número consecutivo + a

   Por ejemplo: si el archivo se generó el 19 de octubre el archivo será gf10191a

3. Ejecutar la función **ligar.guia.fact** e ingresar el nombre del archivo mandado por Aeropuerto.

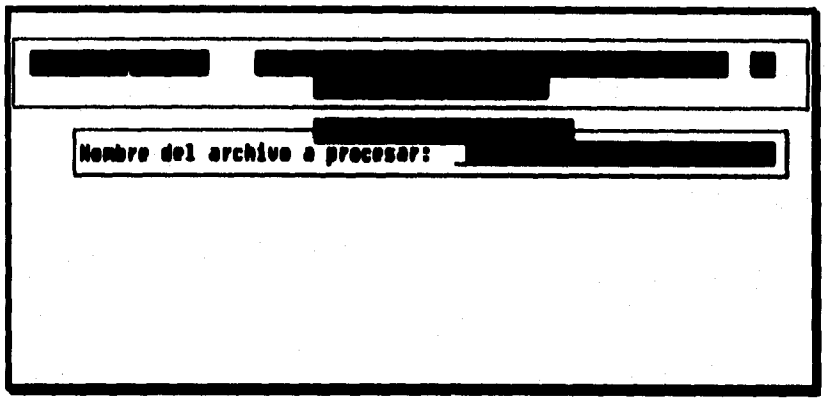


Figura 4.11 Pantalla de actualización cliente.

Consulta de la tabla guía

El modo de operar para la función **consulta.guia** es exactamente el mismo que el de la función **ing.guia.cli**. La pantalla correspondiente a esta opción se presenta en la figura 4.12. Primeramente nos posicionamos en la guía deseada, con F8 desplegamos toda la información referente a esa guía y con F2 accesamos a las funciones de generación, consulta y borrado.

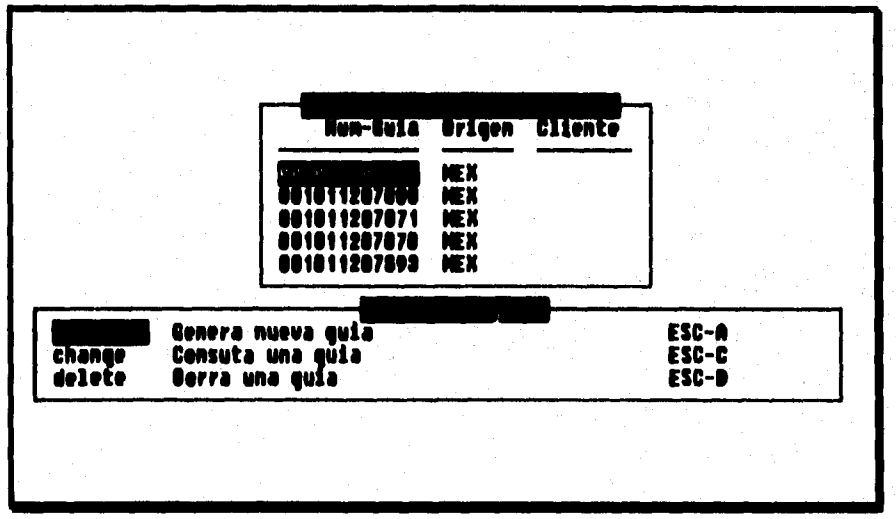


Figura 4.12 Pantalla de consulta de la tabla guia.

Revisión y modificación de facturas

La metodología para el manejo de esta función es la misma que para la consulta de guías y se muestra en la figura 4.13.

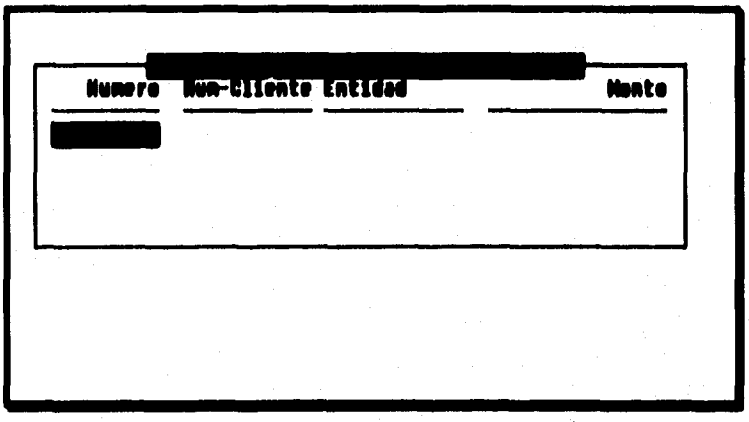


Figura 4.13 Pantalla de revisión y modificación de facturas.

Interfase de facturas generadas

Con esta opción se copian en un diskette las facturas generadas para ser enviadas a la sucursal de Aeropuerto con el fin de anexarlas a la base de datos de Estafeta, como se muestra en la figura 4.14.

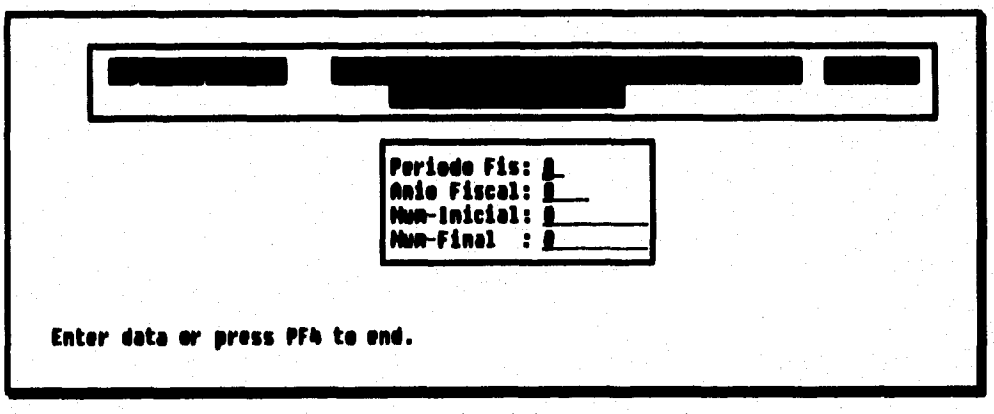


Figura 4.14 Pantalla de interfase de facturas generadas.

Facturas manuales y refacturaciones

Esta opción nos permite elaborar de manera manual las facturas así como la refacturación de las mismas, esta pantalla se muestra en la figura 4.15.

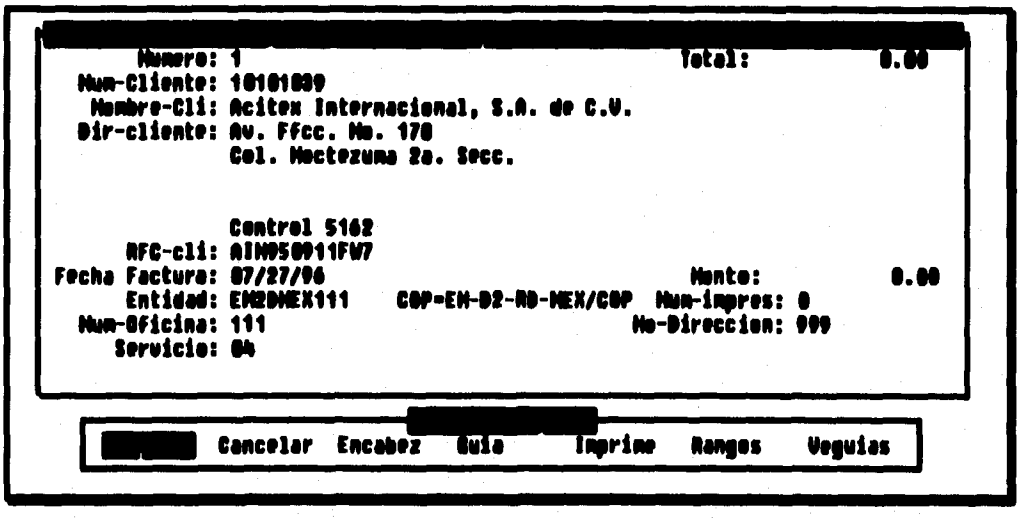


Figura 4.15 Pantalla de facturas manuales y refacturaciones.

Actualización de guías acuse de recibo

Actualmente existe un servicio llamado acuse de recibo, el cual engloba a todos los servicios que anteriormente se mencionaron, como es Facilito, Económico, etc. Debido a esto el precio de este servicio es variable, por lo que surgió la necesidad de crear una función, la cual calculará automáticamente el precio a un rango de guías específico, dependiendo del tipo de servicio que se trate, como se muestra en la figura 4.16.

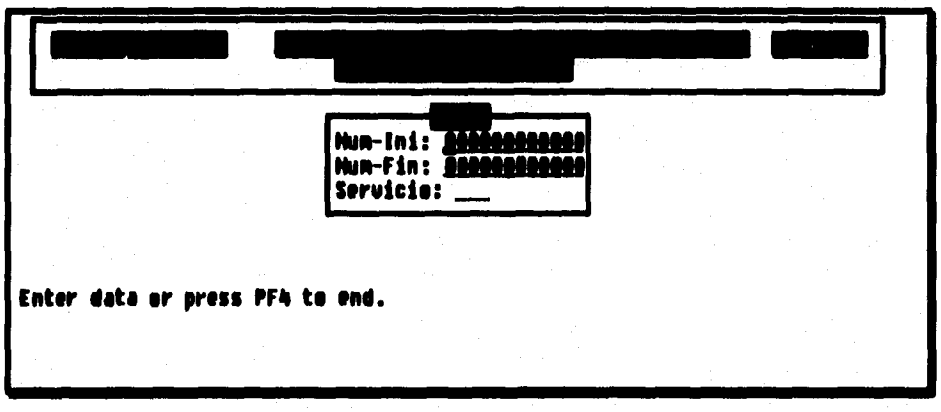


Figura 4.16 Actualización de guías acuse de recibo.

Submenú Mantenimiento de archivos sobrepeso

Como se muestra en la figura 4.17, este submenú nos muestra las opciones para realizar las actualizaciones de las diferentes tablas con las que trabaja el sistema, como son: clientes especiales, precios, archivo de oficinas, clientes, etc.

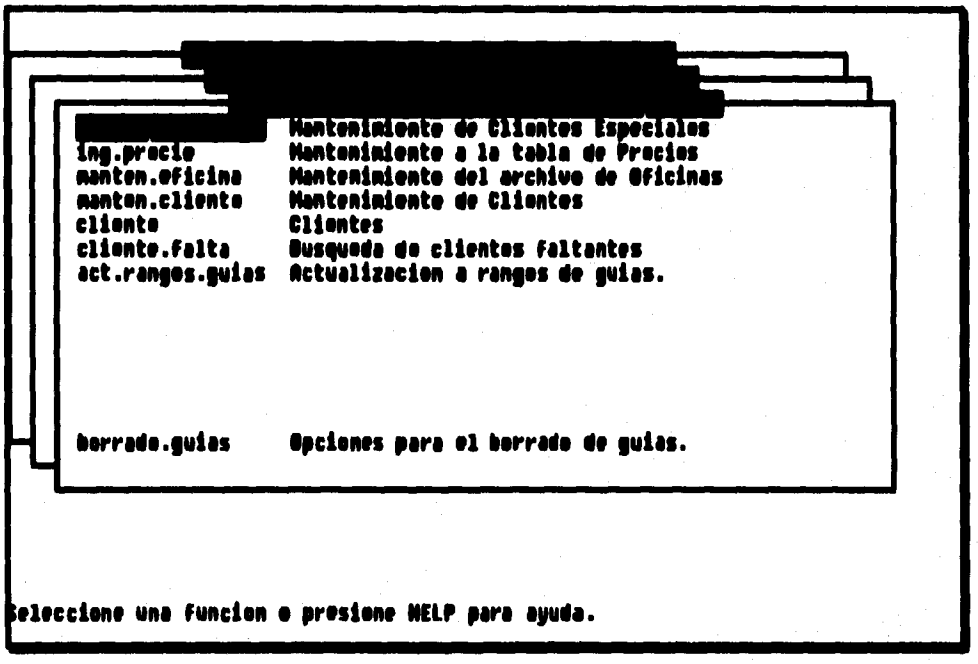


Figura 4.17 Submenú de mantenimiento de archivos de sobrepeso.

Así mismo, en este submenú tenemos el submenú de borrado el cual se muestra en la figura 4.18, este submenú esta compuesto por tres opciones que nos permiten borrar guías por cliente y servicio, de provincia y por rango.

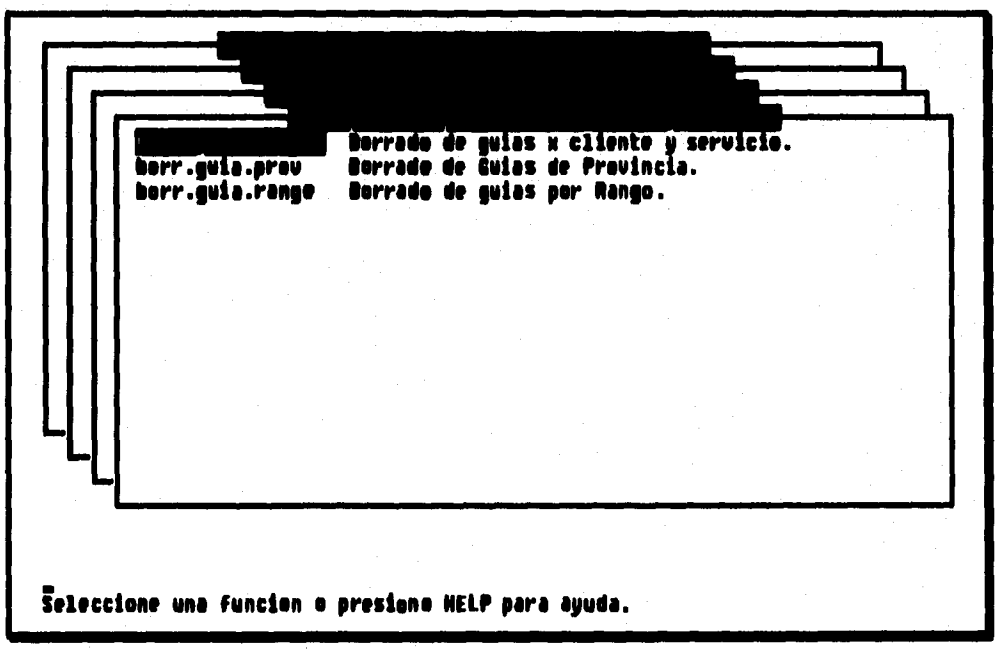


Figura 4.18 Submenú de borrado de guías.

Salidas del sistema

Los usuarios pueden recibir la información del sistema ya sea por medio de pantalla o por medio de reportes. A continuación describimos los reportes más importantes del sistema y la manera de manipularlos.

Menú de reportes de sobrepeso

En la figura 4.19 se muestran los diferentes reportes que podremos manejar en el sistema de sobrepeso.

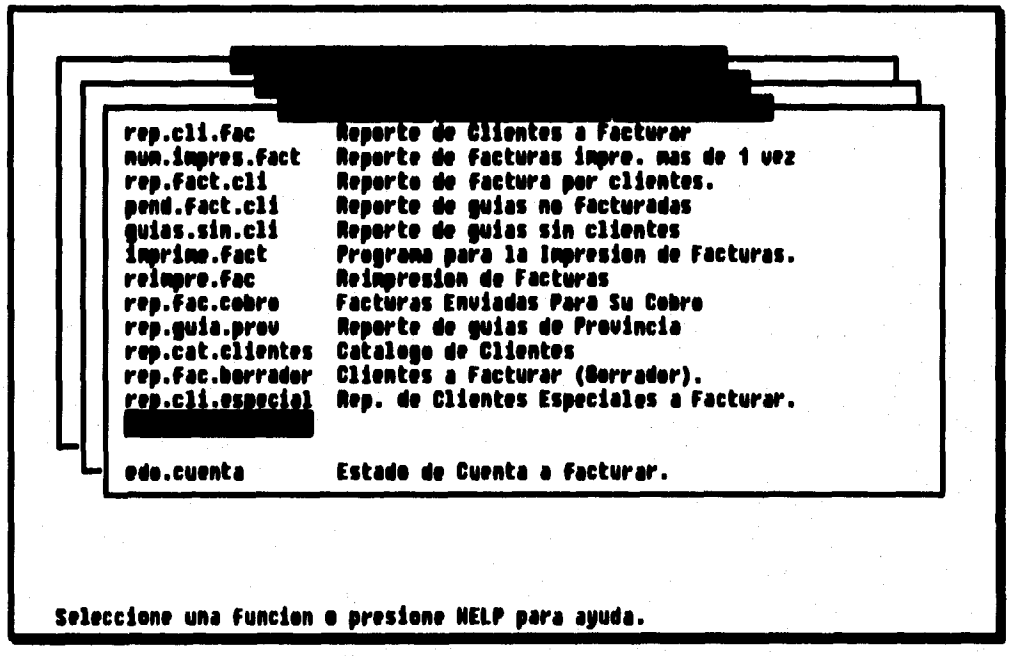


Figura 4.19 Pantalla de menú de reportes de sobrepeso.

Reporte de clientes a facturar

Este es uno de los reportes más importantes del sistema, ya que con este reporte hacemos la validación de todas las guías que vamos a facturar. El reporte arroja los cortes por factura, por cliente y por oficina. Las opciones que manejamos en este reporte se muestran en la figura 4.20 y las detallamos a continuación:

- Dest-Opciones y Dest-Reporte: El destino físico hacia el cual vamos a mandar nuestras opciones y nuestro reporte. Este puede ser nuestra pantalla, una impresora o un archivo.
- guia-ini: El número inicial del rango de guías que queremos facturar.
- guia-fin: El número final del rango de guías que queremos facturar.
- fecha-ini: La fecha inicial que queremos que abarquen las guías.
- fecha-fin: La fecha final que queremos que abarquen las guías.
- Lista-cli: La lista de clientes a los cuales se va a facturar. Esta es una expresión CAN-DO.

- Monto-limite: El monto mínimo por el que vamos a facturar. Este monto es muy importante, ya que requiere un análisis de cuanto es lo menos que vale la pena que facturemos, para que no se presente el caso de que implique más gastos el facturar que lo que pretendemos cobrar con dicha factura.

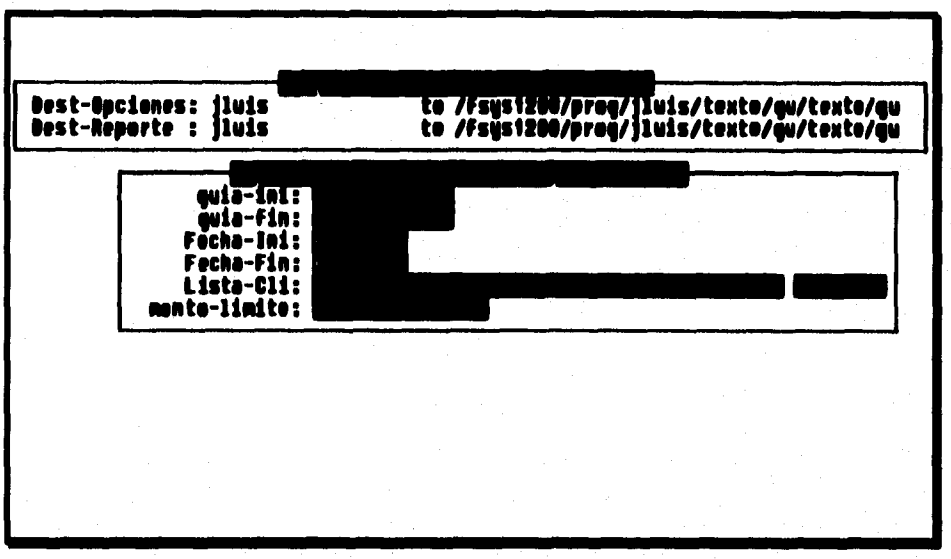


Figura 4.20 Pantalla de opciones para reporte.

El reporte tiene la facilidad de que si queremos facturar todas las guías sólo tenemos que oprimir la tecla *Enter* en las opciones que nos piden un número de guía para aceptar el *default* que tiene el sistema. Lo mismo sucede con la lista de clientes y el monto límite.

Reporte de facturas impresas más de una vez

Este es un reporte que nos va a permitir controlar las impresiones para cada una de las facturas. La información que nos brinda es la factura con todo su detalle y el número de impresiones. En la figura 4.21 se puede observar la pantalla de entrada a este reporte.

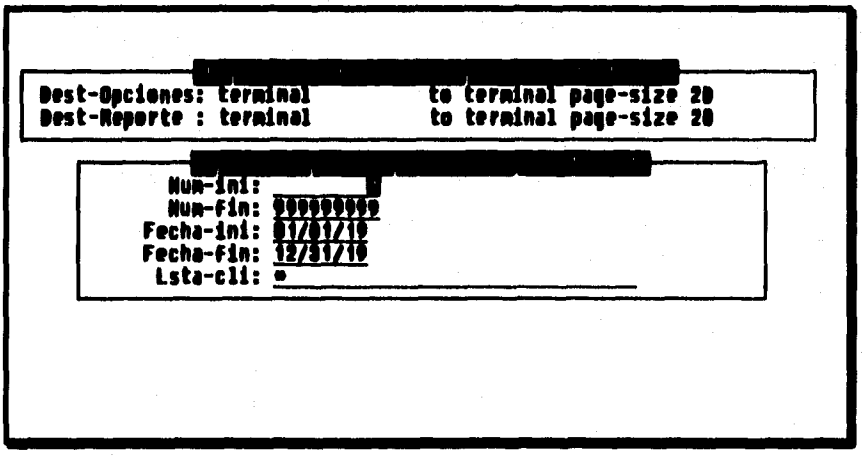


Figura 4.21 Pantalla de reporte de facturas impresas más de una vez.

Las opciones de este reporte son las siguientes:

- Num-Ini: El número inicial de factura que deseamos revisar.
- Num-fin: El número final de factura a analizar.
- Fecha-ini: La fecha a partir de la cual queremos incluir las facturas.
- Fecha-fin: La fecha hasta la cual queremos incluir las facturas.
- Lista-cli: Lista de clientes a analizar. Esta es una expresión CAN-DO.

Reporte de factura por clientes

Con este reporte podemos emitir todas aquellas facturas que por alguna razón no se han impreso y que el sistema sigue registrándolas como vivas. La pantalla de este reporte se muestra en la figura 4.22. y las opciones que manejamos son las siguientes:

- Lista-cli: Lista los clientes que existen con facturas pendientes. Esta es una expresión CAN-DO.
- Fecha-ini: La fecha a partir de la cual queremos incluir las facturas.
- Fecha-fin: La fecha hasta la cual queremos incluir las facturas.

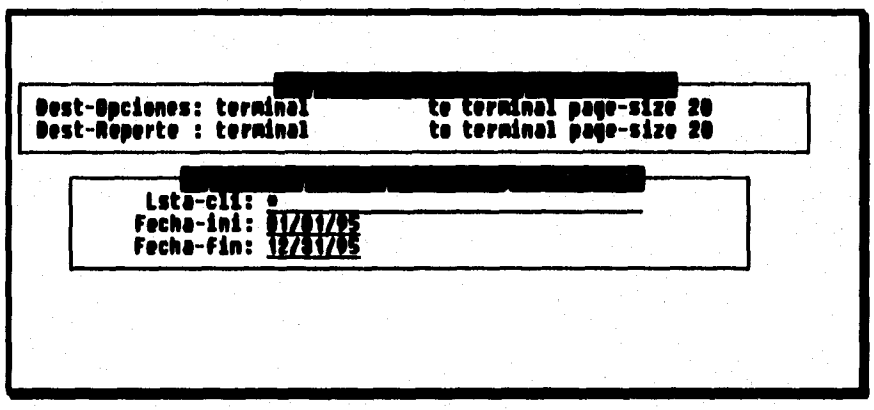


Figura 4.22 Pantalla de clientes por facturar.

Reporte de guías no facturadas

Este reporte engloba todas aquellas guías que aún no han sido facturadas. Nos da toda la información de la guía.

Sus opciones como lo muestra la figura 4.23 son las siguientes:

- Lista-cli: Lista de clientes a analizar. Esta es una expresión CAN-DO.
- Fecha-ini: La fecha a partir de la cual queremos incluir las facturas.
- Fecha-fin: La fecha hasta la cual queremos incluir las facturas.

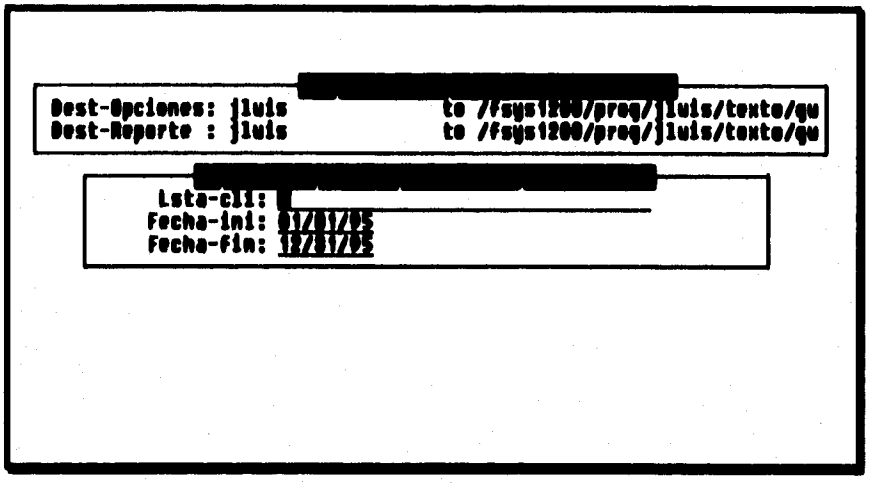


Figura 4.23 Pantalla de reporte de guías no facturadas.

Reporte de guías sin clientes

La figura 4.24 nos muestra el reporte de guías sin clientes, el cual nos detalla las guías existentes en el sistema que no registran un cliente. Sus opciones son las siguientes:

- Guia-ini: El número inicial de guía que deseamos analizar.
- Guia-fin: El número final de guía que deseamos analizar.

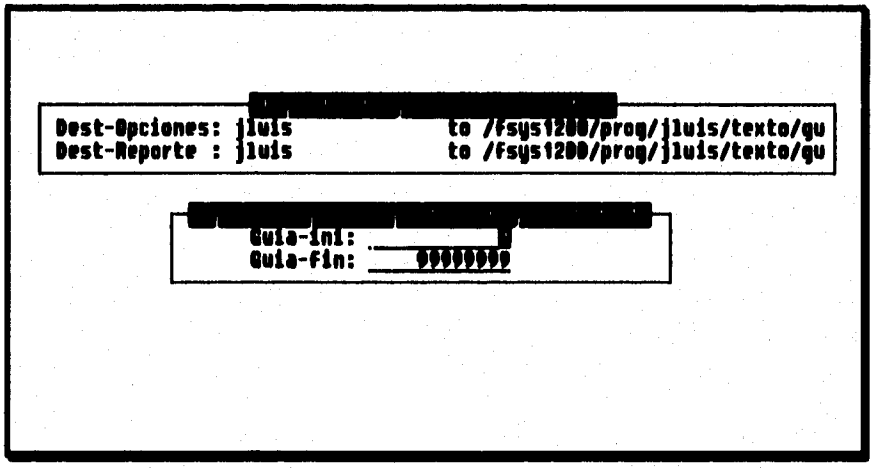


Figura 4.24 Pantalla de reporte de guías sin cliente.

Impresión de facturas

Una vez que tenemos generadas nuestras facturas en el sistema y que hemos validado que están correctas, el siguiente paso es imprimirlas. Para esto vamos a manejar las siguientes opciones :

- numero-ini: El número inicial de factura que deseamos imprimir.
- numero-fin: El número final de factura que deseamos imprimir.

- Lista-cli: Lista de clientes a analizar. Esta es una expresión CAN-DO.
- Fecha-ini: La fecha a partir de la cual queremos incluir las facturas.
- Fecha-fin: La fecha hasta la cual queremos incluir las facturas.

En este caso el destino de las opciones y el destino del reporte son de gran importancia. Como lo muestra la figura 4.25, es necesario que el destino de las opciones sea **siempre** terminal, para que las opciones no sean impresas en nuestras facturas. El destino del reporte será la impresora que utilizaremos para dicha impresión o en su defecto un archivo que posteriormente podrá ser impreso.

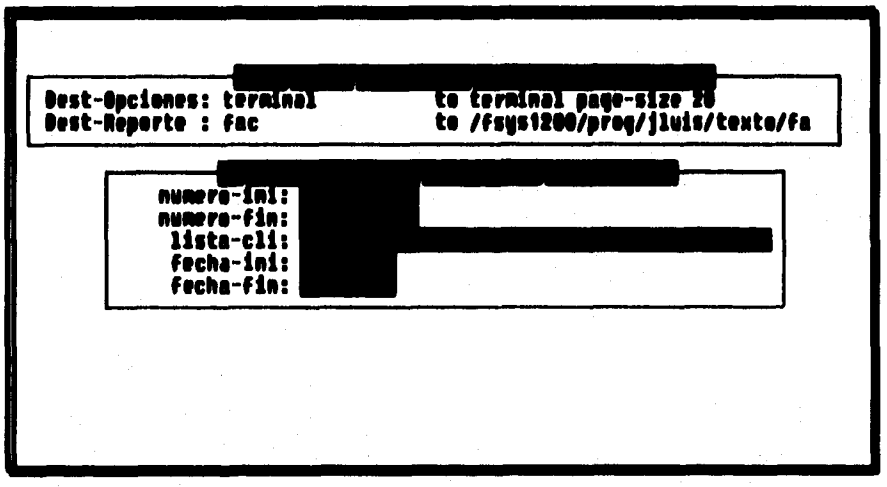


Figura 4.25 Pantalla de opciones para impresión de facturas.

Reimpresión de facturas

El programa para la reimpresión de una factura se tiene como una opción para el caso de que se nos eche a perder la impresión original. Nosotros podemos volver a imprimir una factura cuantas veces queramos. De igual forma que el programa de impresión de facturas, el destino de las opciones deberá ser terminal y el destino del reporte la impresora indicada, como se ven en la figura 4.26.

Además de las opciones antes mencionadas también se cuenta con otras opciones tales como facturas enviadas para su cobro, reporte de guías de provincia, catálogos de clientes, clientes a facturas (borrador) y reporte de clientes especiales a facturar, las cuales tienen la misma metodología de manejo que las opciones anteriores.

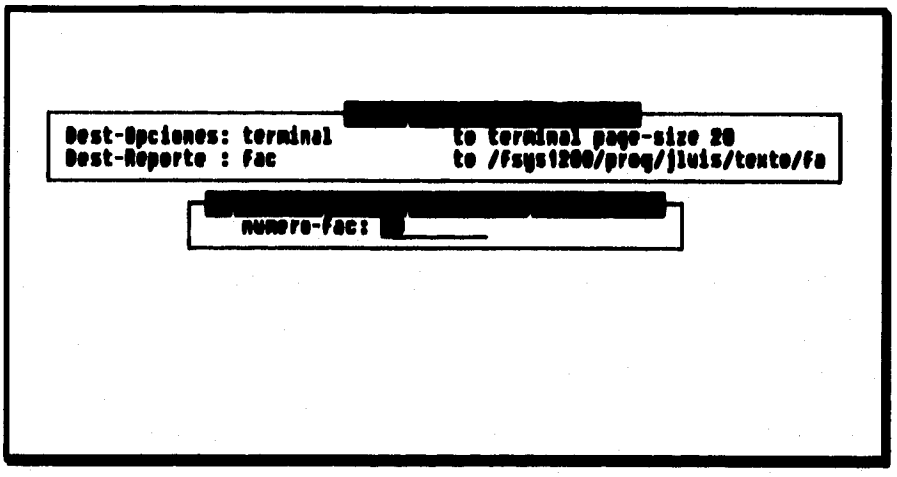


Figura 4.26 Pantalla de reimpresión de facturas.

Estado de cuenta a facturar

En esta opción se pretende dar información al cliente sobre las guías que han utilizado y el sobrepeso que han generado, como se muestra en la figura 4.27.

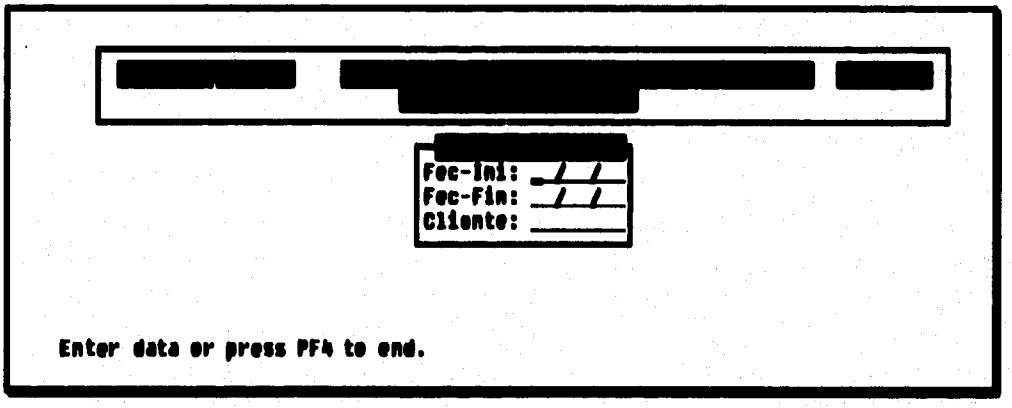
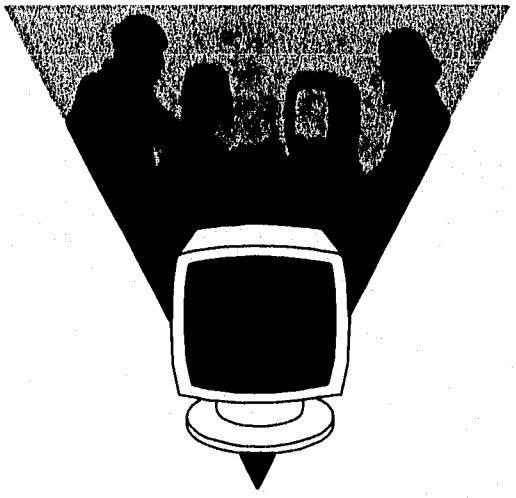


Figura 4.27 Estado de cuenta a facturar.



CAPITULO VI

PRUEBAS, INSTALACION Y MANTENIMIENTO

En el capítulo II de Metodología se describió el desarrollo de pruebas para un sistema. Se resalta la importancia que tiene esta fase para todo sistema, pues la prueba del software es un elemento crítico, que le da garantía de calidad al software y que da la revisión final de las especificaciones, del propio diseño y de la codificación.
La prueba de los programas es la parte del proceso de confirmación que suele realizarse durante las aplicaciones y también en una forma distinta cuando éste se ha terminado. La prueba consiste en ejercitar el programa utilizando datos similares a los datos reales que habrán de ser ejecutados; observar resultados y depurar los errores o insuficiencias a partir de las anomalías de los resultados.

VI.1.- Plan de pruebas

El plan de pruebas es el documento que describe las pruebas propuestas. El proceso de prueba, por lo tanto, constará de un documento para cada una de las siguientes etapas, en las cuales se caracterizarán las entradas y salidas del módulo a probar, se considerará la dependencia funcional entre clases y se seleccionarán casos de prueba, de tal forma que se verifique si el módulo de prueba satisface o no la transformación funcional para el cual fue desarrollado.

El método que se aplicará será el de prueba descendente (ya antes visto en el capítulo II de nuestra tesis), el cuál abarca las siguientes fases.

1. Prueba de unidad o por módulos (prueba de la caja blanca).
2. Prueba de integración (prueba de la caja negra).
3. Prueba de aceptación.

De unidad o por módulos

En la prueba de módulos o prueba de la caja blanca se evalúan las funciones y procedimientos que componen un módulo, para probar su operación correcta.

Se utilizará la codificación del programa y su estructura para crear casos de prueba. Este proceso de prueba de la caja blanca, está enfocado a la aplicación de pruebas en cada módulo. Para el sistema se crearon varias matrices de prueba de los principales módulos que lo integran, los cuales son: alimentación de datos al sistema, captura de guías no facturadas, generación y envío de guías sin cliente, búsqueda de clientes no facturados, actualización de información de los datos de clientes y emisión de facturas. La descripción de los nueve procesos a los que se aplicó un plan de pruebas se da a continuación.

Alimentación de datos al sistema. En este módulo se incorporó la información recabada en Ceylán corriendo la función *carga.guia*. Aquí se dio el nombre del archivo que tiene la información de las guías, de la misma forma se cargó la información que viene de provincia con la función llamada *carga.guia*. Al realizar esta prueba el sistema registró la información de las guías que tienen mayor sobrepeso, los resultados de esta prueba se muestran en la tabla 6.1.

Objetivo de la prueba	Resultados esperados			Diseño de casos de prueba	Resultados reales
	Rechazar	Exhibir mensaje de error	Aceptar la información de las guías que generan sobrepeso		
Determinar si el módulo carga información recabada en los centros operativos			X	Introducir los archivos con la información de los centros operativos	Quedo registrada la información de los centros operativos
Verificar si en el sistema quedan residentes las guías que generan sobrepeso			X	Entrada de las guías capturadas en los centros operativos	Quedaron registradas las guías que generan sobrepeso

Tabla 6.1 Prueba de alimentación de datos.

Captura de guías no facturadas. Como se muestra en la tabla 6.2, en este módulo se ejecuta la función *ing.guia.cli* que realiza el ingreso de las guías que se otorgaron sin una factura que las cubra. En la pantalla nos desglosa el número de la guía inicial, la cantidad de las guías que se dieron y la clave del cliente al que se otorgaron. Así también se tiene la opción de dar de alta una guía, borrarla o modificarla. La prueba consistió en elegir una guía para pedir la información, obteniendo el número de la guía inicial, la cantidad de guías y la clave del cliente, logrando manipular y modificar esta información, borrarla o dar una nueva alta, asignándole una fecha de ingreso.

Objetivo de la prueba	Resultados esperados			Diseño de casos de prueba	Resultados reales
	Rechazar	Exhibir mensaje de error	Aceptar la información de las guías		
Ingreso de guías a las que no se localizó el cliente que respalde el cobro			X	Der de alta guías no capturadas, modificaciones a guías ya dadas y borrado de algunas	El sistema acepta las altas, bajas y modificaciones
Entrada de datos con mayor extensión a los determinados	X	X		Solicitar la consulta de una guía residente en el sistema	Se logro obtener la información correspondiente, indicando a que cliente pertenece, cuantas guías le incluyen

Tabla 6.2 Pruebas a guías.

Generación y envío de guías sin cliente. Este módulo es parte principal del proceso ya que se generará el archivo de interfase que será mandado a Aeropuerto, el cual contiene la información de las guías a las cuales se desconoce el cliente. Este módulo utiliza la función *genera.interf*, en la cual se da un destino de donde dejar la información. Se aplicó la prueba de mandar la información de una guía dando al módulo el número de la guía inicial, guía final y el centro de operación. Se tiene la opción de

mandar sólo parte de ella o todo un bloque de guías a las cuales no se les encontró un cliente. Los resultados de esta prueba se muestran en la tabla 6.3.

	Resultados esperados				
Objetivo de la prueba	Rechazar	Exhibir mensaje de error	Interfase realizada por el sistema de información de las guías	Diseño de casos de prueba	Resultados reales
Realizar el proceso que genere la interfase de la información de las guías a las que se desconoce el cliente			X	Generación de información para hacer posible la interfase	Se logro hacer la interfase de información que manejará Aeropuerto para hacer la búsqueda del cliente a quien corresponde la guía

Tabla 6.3 Prueba de interfase de guías.

Búsqueda de guías sin facturar. Este módulo utiliza la función *guia.no-fact*, la cual se encarga de realizar la búsqueda de clientes en las guías que fueron dadas sin ninguna factura. Esta función deberá correrse antes de hacer la interfase para Aeropuerto. El nombre del archivo que se ingresa es el que se generó con las funciones *genera.interf* que se llama *guia*. La matriz de resultados se muestra en la tabla 6.4.

	Resultados esperados				
Objetivo de la prueba	Rechazar	Exhibir mensaje de error	Búsqueda realizada por el sistema	Diseño de casos de prueba	Resultados reales
Hacer una búsqueda de guías no facturadas			X	Encontrar los clientes que corresponden a las guías que se encuentran en el sistema	La búsqueda fue realizada y se encontraron los respectivos clientes

Tabla 6.4 Prueba de búsqueda de guías sin facturar.

Actualización del campo cliente en las guías encontradas. Esta parte del sistema utiliza la función ligar.guia.fact, la cual ejecuta el programa *actcli.p* que realiza el ingreso de los clientes a los que corresponden las guías que se encontraron en el sistema de facturación. Así mismo, el sistema debe actualizar el campo del número de oficina que realizó la venta. Los resultados de esta prueba se muestran a continuación en la tabla 6.5.

Objetivo de la prueba	Resultados esperados			Diseño de caso de prueba	Resultados reales
	Rechazar	Exhibir mensaje de error	Actualización realizada por el sistema		
Lograr la actualización de clientes de las guías localizadas			X	Verificar que el sistema haya actualizado determinadas guías las cuales no tenían cliente	La actualización fue realizada

Tabla 6.5 Prueba de ingreso de clientes a las guías encontradas.

Ingreso manual de clientes en guías no encontradas. En esta parte el operador debe actualizar manualmente todas aquellas guías a las cuales no se localizó el cliente al que pertenecen, la matriz de pruebas se muestra en la tabla 6.6.

Objetivo de la prueba	Resultados esperados			Diseño de caso de prueba	Resultados reales
	Rechazar	Exhibir mensaje de error	Actualización realizada por el sistema		
Lograr actualización de clientes de las guías localizadas			X	Solicitar al sistema la actualización de determinadas guías, las cuales no tienen cliente	La actualización fue realizada
Ingreso en la guía de clientes no existentes	X	X		Se ingreso un número no registrado de cliente	El sistema marca error en el ingreso del cliente

Tabla 6.6 Pruebas de ingreso de clientes a las guías.

En cuanto a la parte de reportes, el sistema trabaja con varios módulos, el cual cuenta con una pantalla de menú. A continuación se hace una descripción del tipo de prueba que se presentó:

Reporte previo para validación de facturas. Este módulo debe generar las facturas, así como el reporte en el que se realizarán las validaciones correspondientes, como son: datos de los clientes, montos de las facturas, oficina que genera la factura, etc. La función a ejecutar se llama *rep.cli.fac* y ejecuta el programa *repcligui.p.* Las pruebas realizadas se muestran en la tabla 6.7.

Objetivo de la prueba	Resultados esperados			Diseño de caso de prueba	Resultados reales
	Rechazar	Exhibir mensaje de error	Aceptación		
Generación de reporte junto con los registros de las facturas			X	Consulta de registros de las facturas	Registros generados correctamente
Cancelación de facturas erróneas			X	Se canceló una factura con datos erróneos	El sistema cancela la factura.

Tabla 6.7 Validación de facturas.

Emisión de facturas de clientes. El módulo trabaja con la función *imprime.fact*, la cual a su vez ejecuta el programa *impfact.p.* Este reporte hace cortes por factura, por cliente y por oficina. Este módulo trabaja con opciones de destino físico el cual manda las

opciones y reportes a pantalla, a un archivo o a la impresora. Se pide el rango de guías que se quieren facturar, junto con la fecha inicial y final que abarquen las guías, lista de clientes que se van a facturar y el monto límite por el cual se va a facturar. Los resultados de esta prueba se pueden ver en la tabla 6.8.

Objetivo de la prueba	Resultados esperados			Diseño de casos de prueba	Resultados reales
	Rechazar	Exhibir mensaje de error	Aceptación de generación de reportes por el sistema		
Verificar que las facturas no hayan sido impresas anteriormente	X			Generer reporte de facturas que han sido impresas enteriormente	El sistema no genero nada ya que valida que las facturas no hayan sido impresas anteriormente
Dar reporte de facturas que no se han impreso			X	Pedir el reporte de facturas que no se han impreso	Impresión del reporte de facturas generadas
Introducción de datos no válidos al sistema	X	X		Se dieron como entrada al sistema datos erróneos	El sistema marco como incorrectos estos datos y no realizo la operación

Tabla 6.8 Prueba de impresión de facturas.

De integración

La prueba de integración o prueba de la caja negra, consiste en la integración de todos los módulos que componen el sistema para realizar una prueba piloto. Los procesos que se llevan a cabo para dicho exámen se desglosan en la tabla 6.9 junto con sus resultados.

Se diseñó el plan de pruebas de acuerdo a la especificación del requerimiento del sistema. Los casos de prueba se generan revisando las especificaciones externas y los requerimientos de los usuarios, concentrándose en las situaciones posibles que generen error.

Para esta parte se diseñaron pruebas que involucran a todos los módulos en conjunto. Se insertaron datos reales parecidos a los que se van a manejar, haciendo una simulación completa de datos válidos y no válidos.

Objetivo de la prueba	Resultados esperados			Diseño de casos de prueba	Resultados reales
	Rechazar	Exhibir mensaje de error	Hacerlo automáticamente la integración al sistema		
Cargar en el sistema la información de los centros operativos Guadalajara, Monterrey y México, que tengan el mayor sobrepeso			X	Introducción de datos para ser procesados en el sistema	Procesamiento de la información
Determinar si verifica la validez de los números de guías			X	Introducir números de guías inválidos	Rechazado y se exhibe mensaje de error
Verificar la exactitud de los cálculos del monto del sobrepeso			X	Ingresar una guía que tenga un sobrepeso	El cobro se calculó y dio buen resultado
Ingresar guías que se otorgan sin una factura			X	Búsqueda del cliente al cual no se le ha hecho la factura	Selección del cliente con su respectiva guía
Al hacer el ingreso de guías, meter otros caracteres a los números.	X	X		Introducir datos inválidos al sistema	El sistema rechaza los datos marcando error
Hacer la actualización del sistema referente a las guías encontradas			X	Se solicita hacer alteraciones a las guías contenidas en el sistema	Las modificaciones se actualizan en el sistema sin ningún error
Al pedir la información que se de el número de cliente			X	Solicitud de información de un cliente	Despliegue de la información de las guías que tiene para su cobro
Impresión de facturas			X	Selección de determinadas guías para la creación de una factura	Impresión de facturas

Tabla 6.9 Prueba de integridad.

De aceptación

Esta prueba consistió en introducir al sistema con los datos reales del ambiente de operación y la capacidad para satisfacer los requerimientos de los usuarios. Por lo tanto la realizaron los usuarios con el soporte técnico de los programadores. Estas pruebas fueron muy parecidas a las realizadas por nosotros, sin embargo, fueron de vital importancia, ya que surgieron dudas, solicitudes y recomendaciones que permitieron mejorar algunos aspectos del sistema.

Calendario de pruebas

El calendario de pruebas al sistema de sobrepeso se diseñó para un periodo de seis días, figura 6.10.

1er día	2° día	3er día	4° día	5° día	6° día
prueba del equipo	prueba de módulos	prueba de integración	prueba total del sistema	prueba de aceptación	

Tabla 6.10 Calendario de Pruebas.

El primer día asignado se hicieron pruebas del equipo para ver si soporta la configuración necesaria para la instalación del sistema.

El segundo día se probaron las principales funciones que tiene el sistema. Probando individualmente cada uno de ellos y encontrando posibles errores que se dieran antes de su instalación.

El tercer día se hizo una prueba, ya integrando todas las partes del sistema, analizando y probando cada una de las entradas y salidas. Se comprobó que se cumplió con los requerimientos planteados en el análisis.

El cuarto día se hizo una prueba total del sistema, únicamente la realizaron las partes que desarrollaron el sistema, es decir programadores y diseñadores.

El quinto y sexto día las pruebas las realizaron los usuarios en conjunto con diseñadores. Estos últimos fueron los encargados de supervisar y tomar nota de las condiciones que no eran satisfactorias, las cuales fueron corregidas posteriormente.

VI.2.- Plan de Instalación

La instalación se realizó en dos fases: instalación del equipo e instalación del software.

En la instalación del equipo se realizaron las siguientes actividades:

- Instalación del servidor
- Instalación de terminales
- Cableado de terminales a servidor
- Ampliación de memoria al servidor
- Instalación unidad de disco de 5 1/4
- Instalación del módem

En la instalación de software se realizaron los siguientes puntos:

- Instalación de UNIX
- Instalación de PROGRESS
- Instalación de SENTAI y del sistema de sobrepeso
- Instalación de software de comunicaciones BLAST

Estas actividades se realizaron en un periodo de dos semanas como se calendarizó en el capítulo tres.

Métodos de Conversión

Siempre que un nuevo sistema es liberado, ya sea en forma conjunta con el sistema ya existente o por sí sólo, debe preverse un periodo de conversión, lo que significa un proceso de cambio de un sistema antiguo al nuevo sistema de trabajo. Para ello existen

varios métodos entre los que podemos contar el de sistemas paralelos, el de cambio directo, el de enfoque piloto y el método por etapas.

La elección de uno u otro método depende de las circunstancias existentes.

Sistemas Paralelos

Es el método más seguro de conversión de un sistema viejo a uno nuevo, su secreto radica en que ambos sistemas operan en forma paralela, lo que significa que los usuarios continúan operando el viejo sistema de la manera acostumbrada, pero empiezan a usar el nuevo sistema. De ahí su seguridad, ya que garantiza que en el caso de que aparezcan problemas durante el empleo del nuevo sistema, como errores en el procesamiento o incapacidad para manejar determinadas situaciones, los usuarios pueden valerse del viejo sistema sin pérdidas de tiempo o servicios.

Su desventaja radica en que se duplica el trabajo y si llegase a existir falta de facilidad en el uso del nuevo sistema, los usuarios preferirán seguir usando el sistema antiguo, con el consecuente fracaso de implantación del nuevo sistema.

Cambio directo

En este caso el sistema nuevo sustituye al viejo sistema en forma repentina, durante periodos de tiempo de fines de semana o durante días. Y así, el sistema viejo se emplea hasta el día de conversión planeado y entonces se reemplaza por el sistema nuevo. Aquí no existen actividades paralelas. El método por cambio directo logrará el éxito cuando los usuarios no confíen en los métodos anteriores. Psicológicamente obliga a que todos los usuarios logren que funcione el sistema nuevo.

Aunque el no contar con un sistema de apoyo puede llegar a ser una grave desventaja, ya que de surgir problemas con el nuevo sistema, el trabajo que normalmente se lleva a cabo se detendría, ocasionando una larga cadena de problemas; provocando, si se llegase a dar, gran cantidad de trabajo extra, esfuerzos personales frustrados y costos muy elevados.

Por lo tanto el cambio directo requiere una planificación cuidadosa por adelantado. Donde las sesiones de capacitación se deben programar y mantener. Y cualquier preparación en el lugar debe ser completa antes de llevar a cabo la conversión.

Enfoque piloto

Cuando los sistemas nuevos implican técnicas nuevas o cambios drásticos en el desempeño de la empresa se prefiere el enfoque piloto. Cuando se usa este método, un prototipo funcional del sistema se pone en marcha en una parte de la empresa; por ejemplo en una sola área o departamento de trabajo. Así los usuarios de dicha área deben saber que están llevando a cabo una prueba piloto del nuevo sistema y que, por lo tanto, se pueden practicar cambios para mejorarlo. Así cuando el nuevo sistema está terminado por completo, se procede a instalarlo en todas las áreas que abarque, ya sea de una sola vez por cambio directo, o en forma gradual por etapas. Esto significa una desventaja, ya que proporciona además de un enfoque piloto, una sólida oportunidad para realizar pruebas, pero teniendo en cuenta que la puesta en marcha debe realizarse con toda seguridad basados en la solidez del sistema para dar una buena

impresión y alejar toda clase de problemas y desconfianzas que aún pudiera tener el nuevo sistema.

Método por etapas

Se recurre a este método cuando no se tiene la posibilidad de instalar el nuevo sistema en todas las áreas de trabajo a la vez. Así pues la conversión de archivos, la capacitación de personal o llegada de equipo pueden forzar a la puesta en marcha gradual realizada en periodos de semanas o meses. Desde el punto de vista programadores y analistas esta forma de implantación del nuevo sistema presenta algunas desventajas, tales como presión de algunos usuarios entusiastas con la funcionalidad del nuevo sistema, en áreas de trabajo diferentes a las suyas, llegando a niveles tales que cuando se presente la conversión total, si el sistema presenta algunas fallas, incluso provocadas por sus propios errores, los mismos usuarios reaccionarán de manera negativa, haciendo incluso comentarios sobre las dificultades mas mínimas del sistema.

Cuando en los sistemas se utilicen los métodos de conversión por etapas, deben trabajar adecuadamente en la primera conversión y en las que siguen.

Método de conversión elegido

Para nuestro caso, debido a la cantidad de información que se maneja referente al sobrepeso y a la importancia de todos sus procesos para la emisión de la factura, fue necesario asegurarnos que en caso de que existiera alguna falla en el sistema nuevo, tuviéramos un respaldo que le permitiera al área de facturación cumplir con su labor. Por esta razón el método que se eligió fue el de **sistemas paralelos**, el cual, a pesar de la carga de trabajo que significó, cumplió satisfactoriamente con su objetivo: aumento en los montos facturados.

Capacitación

Para el caso en cuestión, la capacitación se llevó a cabo directamente por los programadores a los usuarios finales del sistema, apoyados en el uso de manuales de usuario y de instalación del sistema, bajo un control de fechas previamente establecido. Dicho control se ejerció durante un periodo de una semana para su etapa inicial.

Debido a que el sistema en cuestión no es muy complejo, la capacitación fue básicamente una ambientación a las nuevas herramientas de trabajo, ya que los procesos, procedimientos y estándares en el nuevo sistema se respetaron al máximo.

Otro punto importante es la previa capacitación respecto al uso del nuevo equipo, ya que para las personas involucradas en el manejo del nuevo sistema, esto representa también un cambio, aunque no muy drástico puesto que ya tenían alguna experiencia en el manejo de equipo de cómputo.

Carga de archivos y datos

Ello involucra la identificación de todos los datos que se requieren para la construcción de archivos nuevos durente la conversión.

La solución de este punto fue previamente estudiada, con el uso de una lista de la estructura de los archivos utilizados y un diccionario de datos con la descripción de las características más importantes tanto de la información contenida en el diagrama de flujo de datos como en la carta estructurada.

VI.3.- Mantenimiento

El mantenimiento es una actividad constante que conserva al sistema en los niveles máximos de eficiencia y eficacia. Antes de conocer las causas de un mantenimiento es importante que éste abarque la actualización del diseño, de los archivos, la corrección de errores y la implantación de mejoras, dado que existe una necesidad de programación constante y una revisión al sistema conforme los requisitos cambiantes del mismo.

Plan de mantenimiento

Como se mencionó en el capítulo 2, existen diferentes tipos de mantenimiento, los cuales puede ser aplicables, dependiendo de las necesidades del sistema. Estos son:

Mantenimiento correctivo, enfocado a la reparación, involucra la corrección de errores en una parte del software. Este puede presentarse cuando los usuarios detecten errores en los procesos como pueden ser cálculos incorrectos o un mal manejo de la información.

Mantenimiento adaptativo, incluye la alteración de un programa para adecuarlo lo más posible a los requerimientos de los usuarios. Este mantenimiento es muy utilizado en la etapa de arranque del sistema, ya que al ser utilizado por el usuario final es cuando surgen todas las adaptaciones y mejoras que necesitan los procesos.

Mantenimiento perfectivo, no altera ni la especificación ni la adherencia del software a él, pero mejora el desempeño al hacer que el software consuma menos recursos o bien, que ejecute su función con mayor rapidez y eficacia. Este tipo de mantenimiento solamente es llevado a cabo por los programadores y depende de la experiencia que estos tengan. Un programador inexperto puede realizar mejoras al sistema conforme conozca el lenguaje de programación. Aquí se lleva a cabo la depuración de la base de datos.

Mantenimiento preventivo, implica hacer cambios al software, que por si mismos, no mejoran ni la corrección ni el desempeño, pero provocan que las actividades futuras de mantenimiento sean más fáciles de llevar a cabo. En este mantenimiento se realizan los respaldos a la base de datos, los cuales llevan una periodicidad determinada por la persona responsable del mantenimiento del sistema.

Dentro del plan de mantenimiento que se propone se toman en cuenta los siguientes puntos:

En el mantenimiento perfectivo, se lleva a cabo una depuración de los registros que no interesan o que ya son obsoletos para la empresa, pues sólo ocupan espacio innecesariamente además de provocar que el acceso a la información consuma más tiempo. Antes de explicar el proceso de depuración, es necesario mencionar que el sistema de sobrepeso consta de 3 archivos diferentes que son: *sobre.db*, es el que

contiene la base de datos, *sobre.bi* (before image) guarda información de las transacciones: altas, bajas, borrados de una sesión para evitar pérdidas de información en caso de alguna caída del sistema; y *sobre.lg* que es el archivo que contiene el historial de entradas y salidas de la base de datos, así como una breve explicación de los problemas que se presentan en la misma.

Existe otro archivo, que a diferencia de los anteriores, es un archivo temporal y funciona como candado si la base se encuentra en monousuario, es decir, evita la entrada a más personas una vez que entró la primera. Este archivo es llamado *sobre.lk*.

Una vez mencionados los diferentes archivos de los que consta el sistema de sobrepeso, pasaremos a explicar el proceso de depuración en los siguientes pasos:

1. Hacer por lo menos tres respaldos de la base de datos.
2. Hacer el *dump* (vaciado) de los archivos de la base de datos. Este será generado por la función *dump* de SENTAI. Estando en alguna de las pantallas del sistema de sobrepeso, con la tecla F2 SENTAI nos manda una ventana en la cual hay que indicar que función queremos ejecutar, en este caso la función a ejecutar es *dump*, como se mueatra en la figura 6.1.

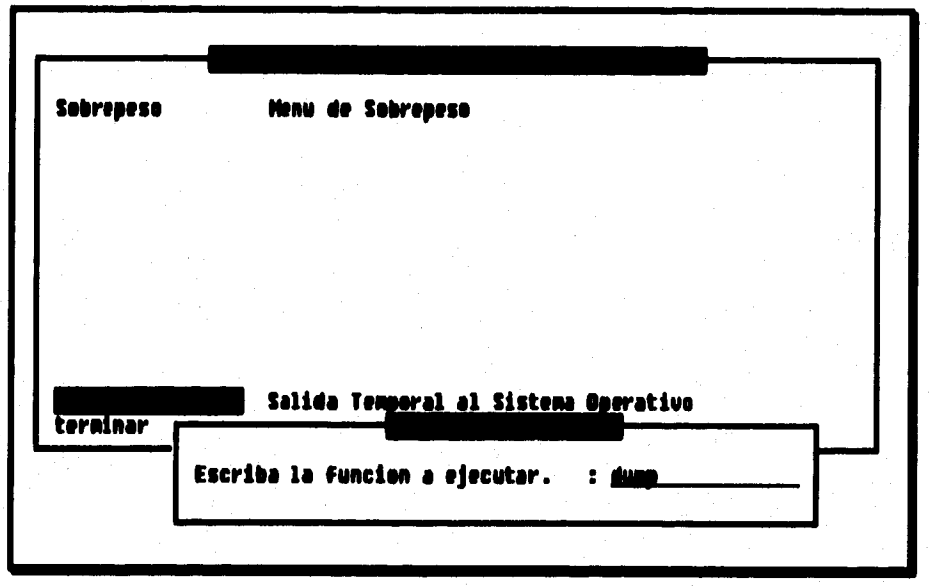


Figura 6.1 Ventana para ejecutar funciones de SENTAI.

Dicha función nos llevará a otra ventana con 4 opciones como se muestra en la figura 6.2.

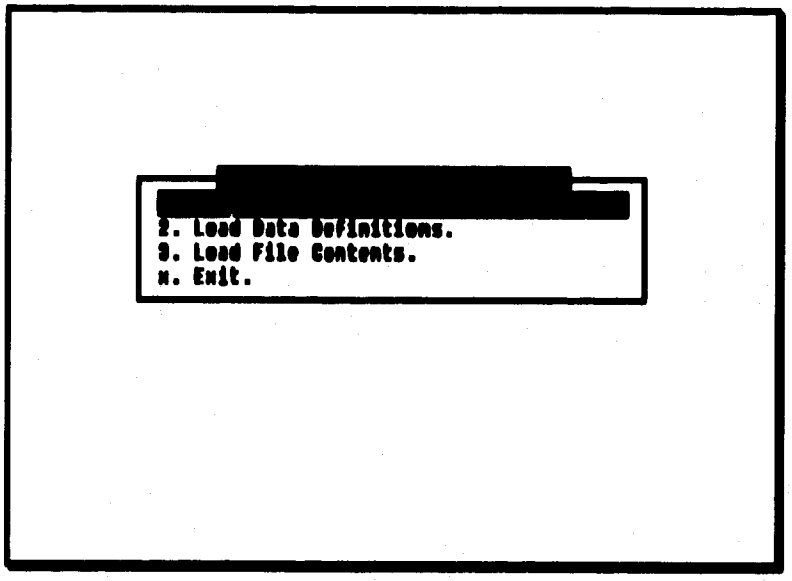


Figura 6.2 Opciones de la función *dump*..

Como se debe realizar el *dump* tanto de las definiciones de las tablas como de los datos contenidos en esas tablas, seleccionamos del menú de la figura 6.2 la opción 1 y escogemos la letra "b" (*both*) para realizar dicha tarea.

Para los archivos que se les va a dar un *dump* se usa la expresión CAN DO ya explicada anteriormente, en la opción de *File List* se teclea * para seleccionar todas las tablas del sistema, y en la opción *Output File* tecleamos *datos*, que será el nombre de los archivos que genera la función. Estos archivos son: *datos.d*, que contiene los datos de las tablas, y *datos.df* que corresponde a la definición de las tablas.

3. Posteriormente se debe generar una nueva base de datos vacía. Esto lo realizamos con el siguiente comando: *Prodb vacía empty*, donde *vacía* es la base de datos que se va a crear y *empty* es la base de datos que tiene PROGRESS por default, de la cual se va a copiar sólo la estructura a la base de datos *vacía*.

4. Entra a la base de datos con el comando: *Pro vacía*. Una vez dentro del editor de PROGRESS se tecleará lo siguiente:

   propath="<directorio donde reside SENTAI>/wbuser," + propath

   Después se carga y ejecuta el programa *wbdict/data.p*. Posteriormente se ejecutan las opciones 2 y 3 del menú de *dump* como se muestra en la figura 6.2. La opción 2 es para la carga de definiciones y la 3 para la carga de datos que se van a manejar.

5. Se debe hacer un borrado de la base de datos. Esto se lleva a cabo mediante el borrado de los archivos que maneja el sistema de sobrepeso: *sobre.db*, *sobre.bi* y *sobre.lg*. Una vez creados los archivos de la nueva base de datos vacía, que son:

vacia.db, *vacia.bi* y *vacia.lg*, deben ser renombrados por los archivos de sobrepeso borrados anteriormente.

6. Posteriormente se debe hacer una regeneración de índices con el comando:

   $/<direcorio donde reside PROGRESS>/dcl/proutil <nombre db> -C idxbuild -TB 31 TM31 -S 31 -T <directorio temporal>

   donde el directorio temporal es donde PROGRESS hace las escrituras a disco al momento de generar los índices.

7. Por último tenemos la recompilación de programas. Esto se lleva a cabo con la ayuda de la función *compile* propia de SENTAI. Al oprimir la tecla F2 y teclear *compile*, nos lleva a otra ventana que se muestra en la figura 6.3, en la cual se teclean los siguientes parámetros:

   En *Files Name* se teclea /*.p que toma todos los archivos con la extensión .p perteneciente a los programas fuente. En error File se le da el nombre de un archivo que va a contener la información cuando un programa no haya compilado. En *Background* se teclea YES si deseamos ejecutar dos tareas al mismo tiempo, para nuestro caso se deberá teclear NO para poder observar que el proceso se lleve a cabo de manera correcta. En Genera objeto se teclea YES, el cual genera el código objeto de los programas. En la línea de Programa indica cuantos programas ha compilado del total de programas. Y en *compiling* indica el nombre del programa que esta compilando en ese momento.

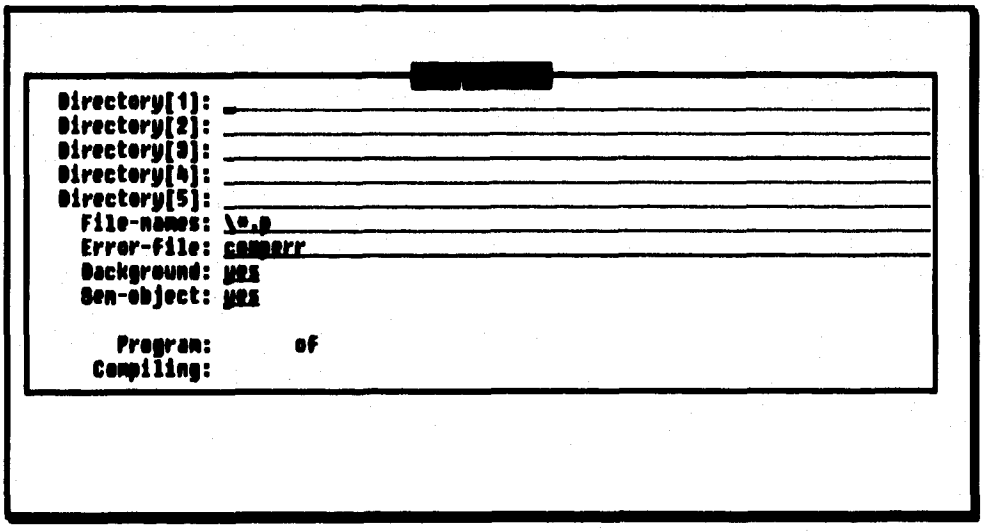


Figura 6.3 Ventana de recompilación de programas.

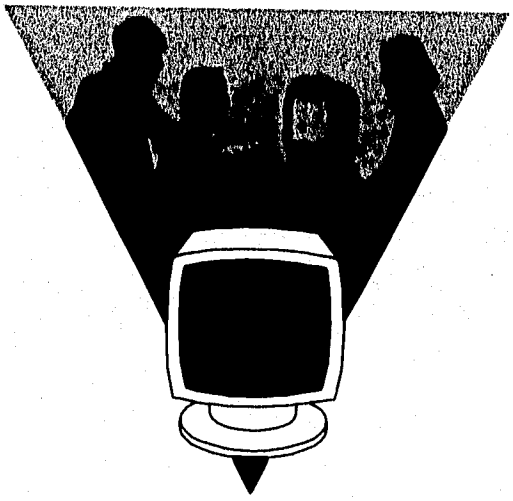
Dentro del mantenimiento preventivo se deben realizar respaldos de la información por lo menos cada quince días, de tal forma que tengan algún parámetro de seguridad, pues la información en Estafeta, como en cualquier empresa, es lo más importante.

Calendario de Mantenimiento

Como se mencionó, el mantenimiento debe llevarse de manera periódica dependiendo del manejo de la información llevado por la empresa, para nuestros sistema se propone el siguiente:

El mantenimiento perfectivo se llevará a cabo por lo menos una vez al año, mientras que, el mantenimiento preventivo deberá realizarse de manera diaria utilizando cintas de reuso. Además se tendrá un respaldo que podrá ser semanal o quincenal que será archivado, esto dependerá del volumen de información que se maneje en las diferentes semanas.

El mantenimiento del sistema tiene que ser desarrollado a la par del mantenimiento del manual de usuario, de tal forma que cualquier cambio en el sistema debe ser reflejado en dicho manual para que esté siempre actualizado.



RESULTADOS Y CONCLUSIONES

Las problemáticas que se analizaron en un principio, como eran: inconsistencia de datos, redundancia de información, complejidad en los procesos, utilización de sistemas no homogéneos, software y hardware obsoleto; daban como consecuencia pérdidas monetarias muy cuantiosas, por montos cercanos a los $300,000.00 pesos mensuales.

El desarrollo del nuevo sistema de sobrepeso eliminó todas estas problemáticas, y se necesitó de una inversión inicial de aproximadamente $90,000.00 pesos, la cual fue recuperada desde el primer mes de su funcionamiento, con un incremento en la facturación del sobrepeso de $300,000.00 a $450,000.00 pesos representando un 50 % de aumento.

Durante el proceso de arranque se tuvieron que realizar numerosas adaptaciones y correcciones al sistema que los operadores solicitaban conforme avanzaban en la realización del proceso. Como ejemplos podríamos citar los siguientes:

- El módulo de actualización manual de clientes en las guías tuvo que ser capaz de capturar una guía completa en caso de que así lo requiriera el operador.

- La impresión de facturas tuvo que desglosar un total por concepto de envíos terrestres y un total por concepto de envíos aéreos, ya que el cliente solicita se le facture de ese modo.

- Diversas modificaciones a los diferentes reportes que intervienen en el sistema, así como la creación de reportes que no se habían contemplado.

- Surgió la necesidad de un módulo de facturación manual y refacturaciones, el cual abarcó módulos propios de captura de guías e impresión de facturas.

Todas estas modificaciones pudieron haberse realizado en un período de prueba, en el cual se hiciera un proceso piloto en conjunto con los operadores y se realizaran las modificaciones, sin embargo, el factor tiempo y la carga de trabajo hizo imposible hacer las cosas de este modo.

El sistema cumplió con las siguientes metas:

- Actualmente se localiza entre el 90% y el 95% de los clientes generadores de sobrepeso. Este porcentaje se obtuvo de comparar el número total de guías capturadas con el número de guías a las cuales se localizó el cliente, estas últimas se conservan en la base con el fin de que en envíos posteriores se pueda localizar a sus respectivos clientes.

- Se proporciona a los operadores las herramientas necesarias para localizaciones manuales de clientes generadores de sobrepeso.

- Automatización en la alimentación de información de guías al sistema.

- Registro acumulativo para los clientes del sobrepeso que registran sus envíos, el cual es almacenado por el sistema hasta que alcanza el mínimo establecido para su facturación.

- Actualización oportuna en los datos de los clientes, minimizando las refacturaciones por datos incorrectos.

- Integración automática de la información de las facturas de Ceylán a Aeropuerto.

- Facturación oportuna y estado informativo de sus envíos al cliente.

El sistema presenta las siguientes desventajas:

- No registra las guías directamente de los *scanners*, hay procesos manuales intermedios, como intercambio de información en discos, que pueden acarrear errores sobre todo humanos.

- Utilización de módems para envío de información, sería más seguro utilizar una línea privada, algún tipo de red ó fibra óptica.

Dados los resultados presentados anteriormente, concluimos que:

- Todo el proceso de facturación fue actualizado y mejorado, desde la forma en que se realizaba anteriormente hasta la utilización de herramientas más viables para la generación de este proceso, como lo son los lenguajes de cuarta generación y el manejo de bases de datos relacionales, lo cual hace que sean más fáciles las futuras modificaciones.

- Se utilizaron bases de datos relacionales debido a que permiten obtener independencia lógica y física, eliminación de redundancia, flexibilidad, uniformidad y sencillez.

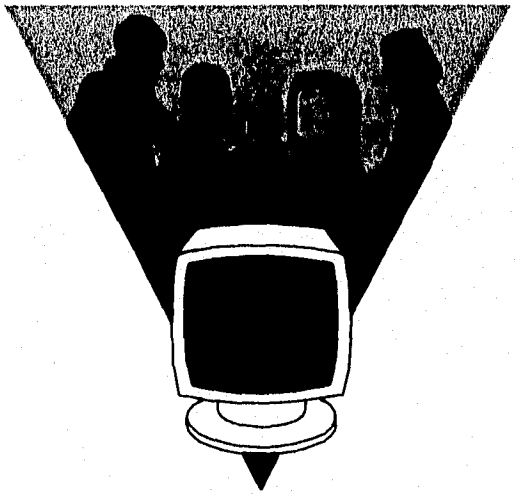
- El sistema de sobrepeso utiliza el manejador de bases de datos Progress debido a que se llegó a la conclusión (después de un estudio realizado) de que era el más adecuado para la elaboración del mismo.

Como se mencionó en capítulos anteriores, en el desarrollo del sistema se utilizó paquetería ya existente en la empresa, lo cual implicó menor inversión.

El sistema tiene las siguientes perspectivas a futuro:

- Instalación en los diferentes centros operativos de la república y flujo de información entre éstos.

- Que el sistema pueda ser integrado como un módulo independiente para la integración de nuevos sistemas.



BIBLIOGRAFIA

Advanced file structures. SENTAI versión 3.4 Workbook versión 4.2
SENTAI Software Corporation,
Canadá, 1993.

Análisis II
Colegio Nacional da Educación Profesional Técnica y Alfaomega Grupo Editor,
México, 995

Análisis III
Colegio Nacional de Educación profesional Técnica y Alfaomega Grupo Editor,
México, 1995

Beizer, B *Software testing techniques.*
Editorial Nostrand Reinhold, Segunda edición,
Alemania, 1990.

Date C.J *Introducción a los sistemas de bases de datos.*
Editorial Addison-Wesley Iberoamericana,
México, 1986

Fairley, Richard. *Ingeniería de software.*
Editorial Mc Graw Hill,
México, 1983.

Grudnitski, Burch *Diseño de sistemas de información,*
Editorial Noriega,
México, 1996.

Lucas Junior, Henry O. *Analysis Design and Implemention Information Systems*
Editorial Mc Graw Hill,
Singapur, 1979,

Deutsch, M. *Verificación y validación en Ingeniería de software.*
Editorial Prentice Hall,
Estadoe Unidos, 1979.

Pressman, Roger S. *Ingeniería del software Un enfoque práctico.*
Editorial Mc. Graw Hill, Tercera edición
España, 1993.

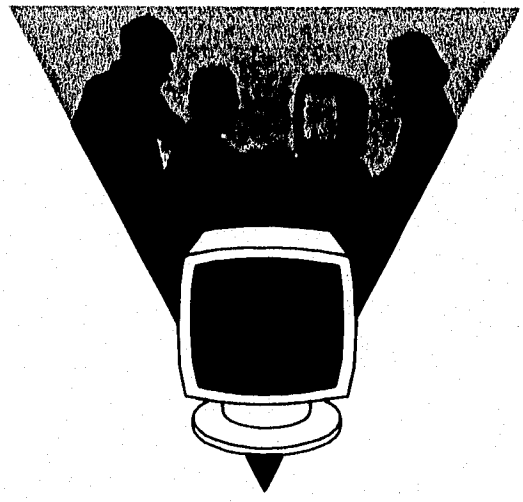
Progress Language Tutorial.
Progress Software Corporation,
Estados Unidos, 1991.

Senn, James A. *Análisis y diseño de sistemas de información*
Editorial Mc Graw Hill,
Segunda edición, México, 19988.

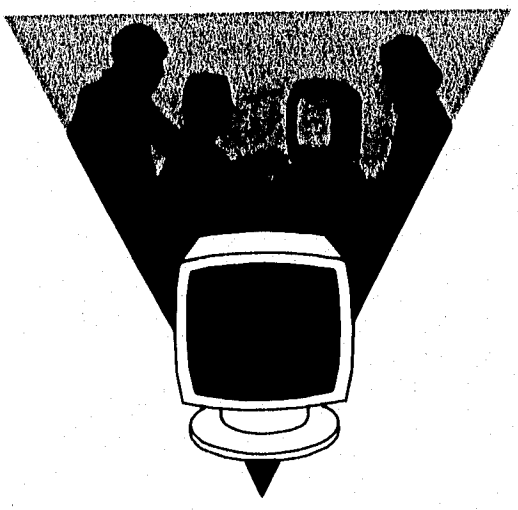
Somerville, Lan, *Ingeniería de software,*
Editorial Adisson-Wesley iberoamericana, Segunda adición,
México, 1988,

Wiederhold, Gio. *Diseño de bases de datos.*
Editora Mc Graw Hill, Segunda edición
México 1993.

Workbook Development Toolkit. SENTAI versión 3.4 Workbook versión 4.2.
SENTAI Software Corporation,
Canada,1993.



INDICE



APENDICE 1

Programas fuente

```
/*************************************************************************
* Programa: act-cli.p                                                    *
* Descripcion : Programa que carga clientes en las guias.                *
*************************************************************************/
def var archi as char format 'x(30)' no-undo.
def var num like guia.num-guia no-undo.
def var cli like guia.cliente no-undo.
def var fact like guia.guia-fact no-undo.
def var ofi like guia.num-oficina no-undo.
def var a as date.
def var b as char format 'x(8)'.
def var cia as char format 'x(40)'.
def var prg as char format 'x(15)'.
def var fecha as char format 'x(8)'.
def var descri as char format 'x(20)'.

form
prg space(4) cia space(2) fecha skip
space(25) descri
with frame w centered no-labels overlay.
pause 0 before-hide.

hide all.
cia = 'ESTAFETA MEXICANA, S.A. DE C.V.'.
prg = 'act-cli.p'.
descri = 'SISTEMA DE SOBREPESO'.
a = today.
b = string(a).
fecha = substring(b,4,2) + '/' + substring(b,1,2) + '/' +
                substring(b,7,2).
display cia prg fecha descri
        with attr-space frame w.
color display messages cia prg fecha descri
        with frame w.

repeat:
  update 'Nombre del archivo a procesar: ' archi
  with frame a centered title 'Actualizacion Clientes' no-labels.

  if search(archi) <> ? then do:
  hide frame a.
  message 'Procesando...'.
  input from value(archi).

  repeat :
    import num cli fact ofi.
    find guia where guia.num-guia = num no-error.
    if available guia then do:
        assign guia.cliente   = cli
            guia.guia-fact = fact
            guia.num-oficina = integer(ofi).
        find cli-espe where cli-espe.cust-id = guia.cliente and
                            cli-espe.servicio = guia.servicio
        no-lock no-error.
        if available cli-espe then do:
          find first oficina where (ofi-id = guia.origen or
                                    ofi-id = guia.destino) and
                                    terr-aer = 'A' no-lock no-error.
```

```
            if available oficina then do:
              find precios where precios.num-servi = integer(guia.servi)
              no-lock no-error.
              if available precios then
                    guia.sobrepeso = guia.sobrepeso + precios.kilo-ampara-aer.
              guia.sobrepeso = guia.sobrepeso - cli-espe.kilo-aereo.
              guia.monto = guia.sobrepeso * cli-espe.precio-aer.
              guia.monto = guia.monto - (guia.monto * cli-espe.desxaere * 0.01).
            end.
            else do:
              find precios where precios.num-servi = integer(guia.servi)
              no-lock no-error.
              if available precios then
                    guia.sobrepeso = guia.sobrepeso + precios.kilo-ampara.
              guia.sobrepeso = guia.sobrepeso - cli-espe.kilo-terres.
              guia.monto = guia.sobrepeso * cli-espe.precio-terr.
              guia.monto = guia.monto - (guia.monto * cli-espe.desxter * 0.01).
            end.
          end. /* end del if avai cli-espe */
          assign guia.num-oficina = int(substring(string(guia.num-guia),1,3)).
          display guia.num-oficina guia.monto.
          find customer where customer.cust-id = guia.cliente and
                              customer.machine-id = '' no-lock no-error.
          if not available customer then do:
            message 'Cliente no existe' .
            guia.cliente = ''.
            display guia.cliente.
          end.
      end.
    end.
    input close.
    end. /* ENd del if search */
    else message 'El archivo no existe.'.
end. /* END repeat principal */

/*************************************************************************
* Programa: actguia.p                                                    *
* Descripcion : Programa actualiza clientes en las guias.                *
*************************************************************************/
def var a as date.
def var b as char format 'x(8)'.
def var cia as char format 'x(40)'.
def var prg as char format 'x(15)'.
def var fecha as char format 'x(8)'.
def var descri as char format 'x(20)'.
def var noguia like guia.num-guia no-undo.
def var resp as logical format "s/n" initial "s" no-undo.

form
prg space(4) cia space(2) fecha skip
space(25) descri
with frame w centered no-labels overlay.
form
    guia.Num-Guia colon 15
    guia.origen colon 15
    guia.destino colon 15
    guia.Fec-Guia colon 15
    guia.sobrepeso colon 15
    guia.servicio colon 15
    guia.medidas[1] colon 15
    guia.medidas[2] colon 15
    guia.medidas[3] colon 15
    guia.cop-scan colon 15
```

```
    guia.monto colon 15
    guia.cliente colon 15
    guia.Factura colon 15
    guia.Guia-fact colon 15
    guia.num-oficina colon 15
with frame a title " Guia " centered
1 down row 5 side-labels attr-space overlay.
pause 0 before-hide.

hide all.
cia = 'ESTAFETA MEXICANA, S.A. DE C.V.'.
prg = 'actguia.p'.
descri = 'SISTEMA DE SOBREPESO'.
a = today.
b = string(a).
fecha = substring(b,4,2) + '/' + substring(b,1,2) + '/' +
                        substring(b,7,2).
display cia prg fecha descri
        with attr-space frame w.
color display messages cia prg fecha descri
        with frame w.

repeat:
update noguia with centered title "Actualizacion" side-labels 1 down frame r.
find guia where guia.num-guia = noguia no-error.
if available guia then do:
  display guia except observ with frame a.
  update guia.origen guia.destino guia.sobrepeso
         guia.servicio guia.cliente guia.factura guia.num-oficina
  with frame a.

  {peso/aux1.i}

  pause.
  clear frame a.
end. /* END del available guia */
else do:
  bell.
  update 'Guia no Existe. Crearla (S/N)? ' resp
  with frame b no-labels row 19 centered.
  hide frame b.
  if resp then do:
    create guia.
    assign
    guia.num-guia  = noguia
    guia.factura   = 0
    guia.guia-fact = 0
    guia.num-oficina = integer(substring(string(noguia,"999999999999"),1,3)).
    display guia.num-guia
            guia.factura
            guia.guia-fact
            guia.num-oficina
    with frame a.
    update guia except num-guia factura guia-fact observ num-oficina monto
    with frame a.

    {peso/aux1.i}

    pause.
    clear frame a.
  end. /* END del if resp */
end.
end.
```

```
/*****************************************************************************
* Programa: carguia.p                                                        *
* Descripcion: programa qua carga la informacion de las guias leidas en ceylan *
*          para despues cruzarla con la de gdl y mty.                        *
*****************************************************************************/
def workfile wmex
   field num as decimal format "999999999999"
   field ori as char format "x(8)"
   field des as char format "x(8)"
   field fecha as date format "99/99/99"
   field peso as decimal format ">>>,>>9.99"
   field servi as char format "x(3)".
def stream sali.
def var a as date.
def var b as char format 'x(8)'.
def var cia as char format 'x(40)'.
def var prg as char format 'x(15)'.
def var fecha as char format 'x(8)'.
def var descri as char format 'x(20)'.
def var resp as logical format 'S/N' initial 'S' no-undo.
def var tot like guia.monto no-undo.
def var pesot like guia.sobrepeso no-undo.
def var archi as char format "x(30)".
def var ofi like guia.num-oficina no-undo.

form
prg space(4) cia space(2) fecha skip
space(25) descri
with frame w centered no-labels overlay.

hide all.
cia = 'ESTAFETA MEXICANA, S.A. DE C.V.'.
prg = 'carguia.p'.
descri = 'SISTEMA DE SOBREPESO'.
a = today.
b = string(a).
fecha = substring(b,4,2) + '/' + substring(b,1,2) + '/' +
                    substring(b,7,2).
display cia prg fecha descri
        with attr-space frame w.
color display messages cia prg fecha descri
        with frame w.

update 'Esta seguro de realizar el proceso (S/N) ?' resp
with frame s no-labels centered no-box row 10.

if resp then do:
hide frame s.

update 'Teclee el archivo a procesar' archi
with centered frame d no-labels.
output stream sali to peso/repetidas.txt peged.

pause 0 before-hide.
if search(archi) = ? then
  message 'Archivo no existe'.
else do:
input from value(archi).
message 'Cargando Guias .....'.
repeat:
  create wmex.
  import wmex.
```

```
  find guia where guia.num-guia = wmex.num no-error.
  if not available guia and (wmex.servi <> '01' and
                             wmex.servi <> '03' and
                             wmex.servi <> '07' and
                             wmex.servi <> '10') then do:
    create guia.
    find first oficina where (ofi-id = wmex.ori or
                              ofi-id = wmex.des) and
                              terr-aer = 'A' no-lock no-error.
    if available oficina then do:
        find precios where precios.num-servi = integer(wmex.servi)
        no-lock no-error.
        if available precios then do:
          tot = (wmex.peso - precios.kilo-ampara-aer) * precios.precio-aer.
          pestot = wmex.peso - precios.kilo-ampara-aer.
        end. /* END del if available precios */
    end. /* END del if available oficina */
    else do:
         find precios where precios.num-servi = integer(wmex.servi)
         no-lock no-error.
         if available precios then do:
           tot = (wmex.peso - precios.kilo-ampara) * precios.precio.
           pestot = wmex.peso - precios.kilo-ampara.
         end.
         else tot = 0.
    end. /* END del else */
    ofi = integer(substring(string(wmex.num,"999999999999"),1,3)).
    assign
        guia.Num-Guia    = wmex.Num
        guia.Fec-Guia    = wmex.Fecha
        guia.origen      = wmex.ori
        guia.destino     = wmex.des
        guia.sobrepeso   = pestot
        guia.servicio    = wmex.servi
        guia.cliente     = ''
        guia.factura     = 0
        guia.guia-fact   = 0
        guia.cop-scan    = 'MEX'
        guia.monto       = tot
        guia.num-oficina = ofi.
  end.
  else do:
    export stream sali wmex.
  end.
  delete wmex.
end. /* END del repeat */
input close.
output stream sali close.
end.
end. /* END del if resp  */
else leave.

/*****************************************************************************
* Programa: carscan.p                                                        *
* Descripcion: programa que genera un solo ascii para el reporte de guias de *
*              sobrepeso conjuntando la informacion de ceylan, guadalajara y *
*              monterrey.                                                    *
*****************************************************************************/

def workfile wmex
    field num as char format "x(15)"
    field cop as char format "x(8)"
    field des as char format "x(8)"
```

```
   field peso as decimal format ">>>,>>9.99".

message 'Procesando .....'.
input from 'peso/repetidas.txt'.
repeat:
create wmex.
import wmex.num wmex.cop wmex.des ^ wmex.peso ^.
find scan where scan.num-guia = wmex.num and
                scan.cop-scan = wmex.cop no-error.
disp wmex.
if not available scan then do:
create scan.
assign
  scan.Num-Guia   = wmex.Num
  scan.cop-scan   = wmex.cop
  scan.destino    = wmex.des
  scan.sobrepeso  = wmex.peso.
end.
else do:
  message ' scan ya existe'.
  display wmex.num.
  end.
delete wmex.
end.
/*************************************************************************
* Programa: cruzados.p                                                   *
* Descripcion: programa que carga las guias de sobrepeso conjuntando la infor- *
*          macion de ceylan, guadalajara y monterrey. Arroja las guias cru-*
*          zadas o repetidas en algun centro operativo sustituyendo o dejan*
*          las de mayor sobrepeso.                                       *
*************************************************************************/
def workfile wgdl
   field num as decimal format "999999999999"
   field ori as char format "x(6)"
   field des as char format "x(6)"
   field fecha as date format "99/99/99"
   field peso as decimal format ">>>,>>9.99"
   field servi as char format "x(3)".
def workfile wmty
   field num as decimal format "999999999999"
   field ori as char format "x(6)"
   field des as char format "x(6)"
   field fecha as date format "99/99/99"
   field peso as decimal format ">>>,>>9.99"
   field servi as char format "x(3)".
def var a as date.
def var b as char format 'x(6)'.
def var cia as char format 'x(40)'.
def var prg as char format 'x(15)'.
def var fecha as char format 'x(8)'.
def var descri as char format 'x(20)'.
def var resp as logical format 'S/N' initial 'S' no-undo.
def var archi as char format "x(40)" no-undo.
def var archi1 as char format "x(40)" no-undo.
def var sobpeso like guia.sobrepeso no-undo.
def var tot like guia.monto no-undo.
def var ofi like guia.num-oficina no-undo.
def stream gdl.
def stream mty.
def stream malo.
def stream bueno.
def stream sali.
```

```
form
prg space(4) cia space(2) fecha skip
space(25) descri
with frame w centered no-labels overlay.

hide all.
cia = 'ESTAFETA MEXICANA, S.A. DE C.V.'.
prg = 'cruzados.p'.
descri = 'SISTEMA DE SOBREPESO'.
a = today.
b = string(a).
fecha = substring(b,4,2) + '/' + substring(b,1,2) + '/' +
                    substring(b,7,2).
display cia prg fecha descri
        with attr-space frame w.
color display messages cia prg fecha descri
        with frame w.

update 'Esta seguro de realizar el proceso (S/N) ?' resp
with frame e no-labels centered no-box row 10.

if resp then do:
hide frame e.
unix silent rm peso/repetidas.txt.
output stream malo to 'peso/pesomal.txt'.  /* arch. de guias duplicadas malas */
output stream bueno to 'peso/pesook.txt'.  /* arch de guias que fueron sustitu*/
output stream sali to 'peso/repetidas.txt' append. /* archivo de carga para
                                                                    scan */
update archi label "Archivo GDL"
     archi1 label "Archivo MTY"
with centered frame a side-labels.
hide frame a.

message 'Procesando Guias.....'.

input stream gdl from value(archi).
repeat: /* Barrido archivo gdl */
  create wgdl.
  import stream gdl wgdl.
  if wgdl.servi <> '01' and wgdl.servi <> '03' and
     wgdl.servi <> '07' and wgdl.servi <> '10' then do:
     find guia where guia.num-guia = wgdl.num no-error.
     if available guia then do:
          find precios where precios.num-servi = integer(wgdl.servi)
          no-lock no-error.
          if available precios then
            assign
              sobpeso = wgdl.peso - precios.kilo-ampara
              tot     = (wgdl.peso - precios.kilo-ampare) * precios.precio.
          if sobpeso < guia.sobrepeso then do:
            display stream malo wgdl.
            export stream sali wgdl.
          end.
          else do:
                    display stream bueno wgdl.
                    export stream sali wgdl.
                    update guia.origen    = wgdl.ori
                           guia.destino   = wgdl.des
                           guia.fec-guia  = wgdl.fecha
                           guia.sobrepeso = sobpeso
                           guia.monto     = tot
                           guia.servicio  = wgdl.servi.
     end.
```

```
  end. /* END del if available guia */
  else do:
        create guia.
        find precios where precios.num-servi = integer(wgdi.servi)
        no-lock no-error.
        if available precios then do:
          tot = (wgdi.peso - precios.kilo-ampara) * precios.precio.
          sobpeso = wgdi.peso - precios.kilo-ampare.
        end. /* END del if available precios */
        else assign tot    = 0
                    sobpeso = 0.
        ofi = integer(substring(string(guia.num-guia,"999999999999"),1,3)).
        assign
          guia.Num-Guia   = wgdi.Num
          guia.Fec-Guia   = wgdi.Fecha
          guia.origen     = wgdi.ori
          guia.destino    = wgdi.des
          guia.sobrepeso  = sobpeso
          guia.servicio   = wgdi.servi
          guia.cliente    = ''
          guia.factura    = 0
          guia.cop-scan   = 'GDL'
          guia.guia-fact  = 0
          guia.monto      = tot
          guia.num-oficina = ofi.
    end. /* END del else */
  end. /* END del if servicio <> 01 and 03 */
delete wgdi.
end. /* END barrido archivo gdi */
input stream mty from value(archi1).
repeat: /* Barrido archivo mty */
  create wmty.
  import stream mty wmty.
  if wmty.servi <> '01' and wmty.servi <> '03' and
    wmty.servi <> '07' and wmty.servi <> '10' then do:
    find guia where guia.num-guia = wmty.num no-error.
    if available guia then do:
        find precios where precios.num-servi = integer(wmty.servi)
        no-lock no-error.
        if available precios then
          assign
            sobpeso = wmty.peso - precios.kilo-ampara
            tot     = (wmty.peso - precios.kilo-ampara) * precios.precio.
        if sobpeso < guia.sobrepeso then do:
          display stream malo wmty.
          export stream sali wmty.
        end.
        else do:
                display stream bueno wmty.
                export stream sali wmty.
                update guia.origen    = wmty.ori
                       guia.destino   = wmty.des
                       guia.fec-guia  = wmty.fecha
                       guia.sobrepeso = sobpeso
                       guia.monto     = tot
                       guia.servicio  = wmty.servi.
        end.
    end. /* END del if available guia */
    else do:
        create guia.
        find precios where precios.num-servi = integer(wmty.servi)
        no-lock no-error.
        if available precios then do:
```

```
            tot = (wmty.peso - precios.kilo-ampara) * precios.precio.
            sobpeso = wmty.peso - precios.kilo-ampara.
          end. /* END del if available precios */
          else assign tot = 0
                      sobpeso = 0.
          ofi = integer(substring(string(guia.num-guia,"999999999999"),1,3)).
          assign
            guia.Num-Guia   = wmty.Num
            guia.Fec-Guia   = wmty.Fecha
            guia.origen     = wmty.ori
            guia.destino    = wmty.des
            guia.sobrepeso  = sobpeso
            guia.servicio   = wmty.servi
            guia.cliente    = ''
            guia.factura    = 0
            guia.cop-scan   = 'MTY'
            guia.guia-fact  = 0
            guia.monto      = tot
            guia.num-oficina = ofi.
      end.
    end. /* END del if servicio <> 01 and 03 */
  delete wmty.
  end. /* END del barrido del archivo mty. */
  end. /* end del resp */
  else leave.

run peso/carscan.p.
/*****************************************************************************
*            ESTAFETA MEXICANA S.A DE C.V.                                   *
*              Sistemas Areopuerto.                                          *
*                                                                            *
*  PROGRAMA   : peso/factcancel.p                                            *
*  MODULO     : SOB                                                          *
*  DESCRIPCION : Ayuda para la cancelacionde facturas y                      *
*            actualizacion de guias de esa factura.                          *
*                                                                            *
*****************************************************************************/

def var resp as char format "x(1)" initial 'L' no-undo.
def var archi as char format "x(15)" no-undo.
def stream sali.

{wb/vars.i}
find last wb_xfer where wb_xfer.fr-file eq "factura"
                    and wb_xfer.fr-level eq help-level - 1 no-lock no-error.
if available wb_xfer then
do transaction :
   find  factura where recid(factura) eq wb_xfer.fr-recid no-error.
   if factura.cancelada = false then
   do:
         /* busqueda de las guias de esta factura */
         hide all.
         update "[B]orrar o [L]iberar las guias de esta factura? " resp
         with frame a no-labels centered row 7.
         resp = caps(resp).
         if resp = 'B' then do:
           archi = 'texto/borr' + string(month(today)) + string(day(today))
                   + substring(string(time,"HH:MM:SS"),7,2) + '.d'.
           output stream sali to value(archi).
           for each guia where guia.factura = factura.num-interno:
             export stream sali guia.
             delete guia.
           end.
```

```
            output stream sali close.
          end. /* END del if resp = 'B' */
          if resp = 'L' then
            for each guia where guia.factura = factura.num-interno:
               assign guia.factura = 0.
            end.
          hide frame a.
          /* assignacion de statue */
          assign factura.cancelada = true
              factura.monto    = 0.

    end.
    else
          message "La factura ya esta cancelada!...".
    delete wb_xfer.
end.

/*****************************************************************************
/*  Sentai documentation
Programme name: peso/Factura.p
Purpose: Lookup routine for Factura
*****************************************************************************/

{util/lookup.i
&file = Factura
&start-index = 1
&index-1 = factura &hilight-1 = Numero &non-char-1 = *
&index-2 = factura &hilight-2 = Num-interno &non-char-2 = *
&index-3 = cli-fact &hilight-3 = entidad
&small-fields = "Factura.Num-cliente factura.monto"
&big-fields = "Factura.Num-cliente Factura.Nombre-cli Factura.Direccion-cli Factura.rfc-cli Factura.fecha
Factura.monto    Factura.entidad    Factura.num-impresa    Factura.User-ID    Factura.fecha-creacion
Factura.cancelada factura.num-interno factura.addrees-no"
&frame-size = 5
&leave-values = "lookup,copy"
&forms = "~{ peso/factura.f &name = lu-big &title = function-desc &row = 1~}"}

/*  Sentai documentation

    $Header: $
Programme name: peso/facturas.p
Purpose: Setup routine for Factura

*/

{util/setup.i
&file = Factura
&key-1 = Num-interno
&field-01 = Num-cliente &add-01 = * &change-01 = *
&field-02 = Nombre-cli &add-02 = * &change-02 = *
&field-03 = Direccion-cli[1] &add-03 = * &change-03 = *
&field-04 = Direccion-cli[2] &add-04 = * &change-04 = *
&field-05 = Direccion-cli[3] &add-05 = * &change-05 = *
&field-06 = Direccion-cli[4] &add-06 = * &change-06 = *
&field-07 = Direccion-cli[5] &add-07 = * &change-07 = *
&field-08 = rfc-cli &add-08 = * &change-08 = *
&field-09 = fecha &add-09 = * &change-09 = *
&field-10 = monto &add-10 = * &change-10 = *
&field-11 = entidad &add-11 = * &change-11 = *
&field-12 = num-impres &add-12 = * &change-12 = *
&field-13 = User-ID &add-13 = * &change-13 = *
&field-14 = fecha-creacion &add-14 = * &change-14 = *
&field-15 = cancelada &xadd-15 = * &xchange-15 = *
```

```
&field-16 = address-no &add-16 = " &change-16 = "
&add-value = "add" &change-value = "change" &delete-value = "delete"
&look-values = "lookup"
&change-tests = "if factura.address-no <> 999 then do:
                  find customer where customer.cust-id = factura.num-cliente and
                                      customer.machine-id = '' no-lock no-error.
                  if not available customer then do:
                    message 'Cliente no existe' .
                    factura.num-cliente = ''.
                    display factura.num-cliente.
                  end.
                  else do:
                    find cust-address of customer where
                         cust-address.address-no = factura.address-no
                    no-lock no-error.
                    if available cust-address then do:
                      assign factura.nombre-cli       = cust-address.name
                             factura.direccion-cli[1] = cust-address.address[1]
                             factura.direccion-cli[2] = cust-address.address[2]
                             factura.direccion-cli[3] = cust-address.address[3]
                             factura.direccion-cli[4] = cust-address.address[4]
                             factura.direccion-cli[5] = cust-address.address[5].
                      display factura.nombre-cli factura.direccion-cli.
                    end.
                    else do:
                      message 'Numero de direccion no existe !'.
                      factura.address-no = 999.
                      display factura.address-no.
                    end.
                  end. /* END del else */
                end. /* END del if address-no <> 999 */
                "

&forma =
"~{peso/factura.f &name = setup-frame &title = function-desc &row = 4~}"
&change-disallow = "if factura.cancelada = true then
                    do:
                    message ""No se puede cambiar una factura cancelada!"".
                          undo change-loop, leave change-loop.
                    end.
                    "}

Programme name: peso/facturas.p
Purpose: Setup routine for Factura

*/

{util/setup.i
&file = Factura
&key-1 = Num-Interno
&field-01 = Num-cliente &add-01 = " &change-01 = "
&field-02 = Nombre-cli &add-02 = " &change-02 = "
&field-03 = Direccion-cli[1] &add-03 = " &change-03 = "
&field-04 = Direccion-cli[2] &add-04 = " &change-04 = "
&field-05 = Direccion-cli[3] &add-05 = " &change-05 = "
&field-06 = Direccion-cli[4] &add-06 = " &change-06 = "
&field-07 = Direccion-cli[5] &add-07 = " &change-07 = "
&field-08 = rfc-cli &add-08 = " &change-08 = "
&field-09 = fecha &add-09 = " &change-09 = "
&field-10 = monto &add-10 = " &change-10 = "
&field-11 = entidad &add-11 = " &change-11 = "
&field-12 = num-impres &add-12 = " &change-12 = "
&field-13 = User-ID &add-13 = " &change-13 = "
&field-14 = fecha-creacion &add-14 = " &change-14 = "
```

```
&field-15 = cancelada &xadd-15 = " &xchange-15 = "
&field-16 = address-no &add-16 = " &change-16 = "
&add-value = "add" &change-value = "change" &delete-value = "delete"
&look-values = "lookup"
&change-tests = "if factura.address-no <> 999 then do:
                   find customer where customer.cust-id = factura.num-cliente and
                                       customer.machine-id = " no-lock no-error.
                   if not available customer then do:
                     message 'Cliente no existe' .
                     factura.num-cliente = ".
                     display factura.num-cliente.
                   end.
                   else do:
                     find cust-address of customer where
                           cust-address.address-no = factura.address-no
                     no-lock no-error.
                     if available cust-address then do:
                       assign factura.nombre-cli      = cust-address.name
                              factura.direccion-cli[1] = cust-address.address[1]
                              factura.direccion-cli[2] = cust-address.address[2]
                              factura.direccion-cli[3] = cust-address.address[3]
                              factura.direccion-cli[4] = cust-address.address[4]
                             factura.direccion-cli[5] = cust-address.address[5].
                       display factura.nombre-cli factura.direccion-cli.
                     end.
                     else do:
                       message 'Numero de direccion no existe !'.
                       factura.address-no = 999.
                       display factura.address-no.
                     end.
                   end. /* END del else */
                 end. /* END del if address-no <> 999 */
                 "
&forms =
"~{peso/factura.f &name = setup-frame &title = function-desc &row = 4~}"
&change-disallow = "if factura.cancelada = true then
                   do:
                   message ""No se puede cambiar una factura cancelada!"".
                         undo change-loop, leave change-loop.
                   end.
                   "
}
```