Universidad Nacional Autónoma de México

Facultad de Estudios Superiores Acatlán

Caso práctico de la reingeniería de software. Punto de venta Infocaja.

Tesina

que para obtener el título de:

Licenciado en matemáticas aplicadas y computación

Presenta:

Oscar Mejía Rescalvo

Asesor: Lic. Martínez José Socorro

México D.F. Noviembre 2008

Índice.

Introducción.

Desde la década de 1970 se ha dirigido cada vez más la atención a la tecnología del desarrollo del software. Conforme los sistemas de cómputo se multiplican, se hacen más complejos y penetran con mayor profundidad en la sociedad moderna, de tal forma que se evidencia la necesidad de enfoques sistemáticos para el desarrollo de software; así como para su mantenimiento.

La ingeniería de software es el campo de estudio relacionado con esta nueva tecnología.

A lo largo de las últimas cuatro décadas, la ingeniería del software se ha hecho mayor, es decir, está reconocida como una verdadera disciplina, derivada de una investigación seria, un estudio minucioso y un debate multitudinario.

Los métodos, procedimientos y herramientas de la ingeniería de software han sido adoptados con éxito en una gran variedad de aplicaciones industriales. Tanto los gestores como los desarrolladores reconocen la necesidad de un enfoque más disciplinado del desarrollo del software.

Actualmente, el software ha superado al hardware como la clave del éxito de muchos sistemas basados en computadoras. Tanto si se utiliza la computadora para llevar un negocio, controlar un producto o capacitar un sistema; el software es el factor que marca la diferencia.

Lo que distingue a una compañía de su competidora es la suficiencia y oportunidad de la información dada por el software y bases de datos relacionadas. El diseño de un producto de software amigable a los humanos lo hace diferente al de los demás productos competidores que tengan unas funciones similares. El software es el que marca la diferencia.

La industria del software, hoy está en una situación en la que grandes y pequeñas empresas se están convirtiendo en fábricas de software que envejece, hay cientos de aplicaciones basadas en software que están en una situación crítica y necesitan ser renovados urgentemente tales como:

- Las aplicaciones de sistemas de información escritas hace veinte años, que han sufrido cuarenta generaciones de cambios y que ahora son virtualmente imposibles de mantener. Incluso la más pequeña modificación puede hacer que falle todo el sistema.

- Por su misma edad y estado de conservación ya no se entienden, nadie tiene un conocimiento detallado sobre la estructura interna de esos programas.

- Sistemas que parecen extraños y a veces tienen un comportamiento inexplicable; pero que no se pueden poner fuera de servicio porque no hay nada disponible para reemplazarlas (controlan plantas de potencia, tráfico aéreo y fábricas entre sus aplicaciones, etc.)

No será suficiente reparar lo que está mal y dar una imagen moderna a estas aplicaciones. Muchos componentes de la fábrica de software requieren una reingeniería o reestructuración importante o, de lo contrario, no serán competitivos durante los siguientes años.

Desafortunadamente, muchos directores de empresas parecen poco dispuestos a invertir recursos para emprender este esfuerzo de reestructuración: "las aplicaciones todavía funcionan ¿para qué mejorarlos?, es un gasto innecesario"; dicen.

Algunas empresas subcontratan o reducen al mínimo el personal dedicado a sus sistemas de información y, en otros casos, pactan con un tercero la administración de todo el desarrollo del nuevo software y el mantenimiento del hardware, con la idea de ahorrar dinero.

Algunas otras optan por la nueva tendencia hacia el software libre para tratar de conseguir el mismo objetivo, pero se enfrentan a un problema: no existen las aplicaciones necesitadas en este tipo de programas o no hay profesionales capaces de dar un servicio con estas nuevas herramientas.

Para el desarrollo del presente trabajo, se exponen tres capítulos que se describen a continuación:

- Capítulo I Ingeniería de Software y origen de Infocaja como empresa.

  El objetivo es mostrar la importancia de la Ingeniería de Software y ubicar dentro de esta rama de la informática a la Reingeniería de Software para dar mantenimiento a los sistemas computacionales ya existentes mediante el uso de técnicas que han sido desarrolladas.

  Así mismo se comentarán los orígenes de Infocaja como empresa y posteriormente del producto con el mismo nombre.

- Capítulo II Análisis del punto de venta Infocaja.

  Este capítulo tiene como objetivo mostrar las condiciones actuales bajo las que se encuentra el punto de venta a nivel base de datos, procedimientos e interfases, el software con el que fue desarrollado y los requerimientos de hardware que se necesitan para su uso.

- Capítulo III Cambios propuestos

Finalmente, en este capítulo se describen los cambios realizados a Infocaja en diversas partes (base de datos, reestructuración de código, cambios a reportes), las alternativas de software libre para desarrollar el nuevo punto de venta y los nuevos requerimientos de hardware que se necesitan para su ejecución.

Capítulo I

Ingeniería de Software y origen de Infocaja como empresa.

Ingeniería de Software y origen de Infocaja como empresa.

1.1 Introducción a la Ingeniería de Software

1.1.1 Evolución del software

1.1.2 Definición y objetivos de la Ingeniería de Software.

1.1.3 Metodología de la Ingeniería de Software.

1.1.4 Clasificación de herramientas C.A.S.E.

1.2 Reingeniería de Software como sub rama de la Ingeniería de software para la reestructuración de sistemas.

1.3 Ejemplo Infocaja: historia y antecedentes.

1.4 Necesidad de aplicar la Reingeniería de Software.

1.1 Introducción a la Ingeniería de Software

1.1.1 La evolución del software.

Durante los primeros años de la era de la computadora, la mayoría del software fue desarrollado por personas selectas. Los programas estaban escritos y utilizados por la misma persona y en caso de falla esta misma persona lo depuraba. Debido a este entorno el diseño era un proceso implícito, realizado en la mente de alguien y la documentación normalmente no existía debido a que no era necesaria.

A partir de la segunda evolución de los sistemas de computadora, (segunda mitad de la década de los sesenta hasta finales de los setenta) se introducen los sistemas multiusuario. Estos sistemas podían recoger, analizar y transformar datos de múltiples fuentes, controlando así los procesos y produciendo salidas en milisegundos en lugar de minutos. Los avances en los dispositivos de almacenamiento condujeron a la primera generación de sistemas de gestión de bases de datos.

Conforme crecía el número de sistemas informáticos, las aplicaciones comenzaron a crecer. Los programadores desarrollaban proyectos en los que se producían programas con cientos de líneas de código fuente. Todos esos programas, tenían que ser corregidos cuando se detectaban fallos, modificarlos cuando cambiaban los requisitos de los usuarios o adaptarlos a nuevos dispositivos de hardware que se hubieran adquirido.

Es en esta época cuando nace lo que se conoce como el mantenimiento del software, pero el esfuerzo requerido para este mantenimiento era en la mayoría de los casos tan elevado que se hacía imposible su mantenimiento.

La tercera evolución de los sistemas de computadora comenzó a mediados de los años setenta y continuó más allá de una década. El sistema distribuido con múltiples computadoras, cada una ejecutando funciones concurrentes y comunicándose con alguna otra, incrementó notablemente la complejidad de los sistemas informáticos. Las redes de área local y de área global, las comunicaciones digitales de alto ancho de banda y la creciente demanda de acceso inmediato a los datos, supusieron una fuerte presión sobre los desarrolladores del software.

En esta etapa aparecen la programación estructurada, metodologías de diseño y las herramientas C.A.S.E. (del inglés Computer-Aided Software Enginiering)[1], aunque como podemos imaginar eran muy rudimentarias.

La cuarta evolución de los sistemas informáticos (ochentas) se aleja de las computadoras individuales y de los programas de computadoras creados con las técnicas de la programación estructurada y da paso a nuevas y potentes máquinas personales controladas por sistemas operativos sofisticados, en redes globales y locales, acompañadas por aplicaciones de software creadas con una nueva técnica de desarrollo conocida como programación orientada a objetos.

Sin embargo, un conjunto de problemas relacionados con el software ha persistido a través de la evolución de los sistemas basados en computadora, y estos problemas continúan aumentando.

1. Los avances en la construcción de programas no son suficientes para que estos alcancen todo potencial que el hardware ofrece.

2. Nuestra habilidad de construir nuevos programas no puede ir al mismo ritmo de la demanda de nuevas aplicaciones, ni podemos construirlos lo suficientemente rápido como para cumplir las necesidades del mercado y de los negocios.

3. El uso extenso de computadoras ha hecho de la sociedad cada vez más dependiente de la operación fiable del software. Cuando este falla, pueden ocurrir daños económicos enormes y ocasionar sufrimiento humano.

4. Luchamos por construir aplicaciones que tengan fiabilidad y alta calidad.

5. Nuestra habilidad de soportar y mejorar los programas existentes se ve amenazada por diseños pobres y recursos inadecuados.

En respuesta a estos problemas, las prácticas de la Ingeniería del Software se están adoptando en toda la industria.

[1] Libro Software Engineering, autor Ian Sommerville, séptima edición, página 12.

1.1.2 Definición y objetivos de la Ingeniería de Software.

Como ya se mencionó, la programación de las computadoras no disponía de métodos sistemáticos para la realización de software.

Hoy en día existen muchas herramientas y metodologías para la creación de software de calidad, e informalmente comenzaremos diciendo que a este conjunto de herramientas y metodologías se les conoce como Ingeniería de Software.

La expresión Ingeniería de Software fué usada por primera vez por Fritz L. Bauer en 1968, en la conferencia de la OTAN de Garmisch, Alemania,

Según Fritz L. Bauer "La Ingeniería del Software es el establecimiento y uso de principios robustos de la ingeniería a fin de obtener software que sea fiable y que funcione eficientemente sobre máquinas reales".[2]

Pero la definición más aceptada dentro del campo de la Ingeniería de Software es la proporcionada por la I.E.E.E. (del inglés Institute of Electrical and Electronics Engineers)[3]:

"la Ingeniería de Software es la aplicación de un enfoque sistemático, disciplinado y cuantificable hacia el desarrollo, operación y mantenimiento del software".[4]

En cuanto al surgimiento de la Ingeniería de Software como ciencia no hay una fecha exacta, algunos autores como M. Jackson consideran que la Ingeniería de Software se inicia con Wheeler al inventar la subrutina en 1950.

El objetivo de la Ingeniería de Software es la construcción y desarrollo de proyectos aplicando métodos y técnicas para resolver los problemas, la informática aporta herramientas y procedimientos sobre los que se apoya la Ingeniería de Software.

La Ingeniería del Software es usada en áreas muy diversas de la informática y de las Ciencias de la Computación, como ejemplos podemos citar la construcción de compiladores, sistemas operativos o desarrollos de Intranet/Internet, abordando todas las fases del ciclo de vida del desarrollo de cualquier tipo de sistemas de información y aplicables a una infinidad de áreas tales como: negocios, investigación científica, medicina, producción, logística, banca, control de trafico, meteorología, el mundo del derecho, la red de redes Internet, redes Intranet y extranet, etc.

La Ingeniería de Software también se relaciona con muchos campos en diferentes formas:

[2] http://es.wikipedia.org/wiki/Industria_del_software
[3] Significado obtenido de la dirección www.ieee.org
[4] http://es.wikibooks.org/wiki/Evoluci%C3%B3n_del_software

Matemáticas:

Los programas tienen muchas propiedades matemáticas. Por ejemplo la corrección y la complejidad de muchos algoritmos son conceptos matemáticos que pueden ser rigurosamente probados. Edsger Dijkstra ha dicho que la Ingeniería de Software es una rama de las matemáticas.

Ciencia:

Los programas tienen muchas propiedades científicas que se pueden medir. Por ejemplo, el desempeño y la escalabilidad de programas se miden bajo diferentes cargas de trabajo. La efectividad de los procesadores más grandes, la velocidad de redes en la transmisión de datos y las nuevas tecnologías de base de datos tienen que ver con la ciencia. Se pueden deducir ecuaciones matemáticas de medidas para estas áreas.

Administración de Proyectos:

El software comercial (y mucho no comercial) requiere manejo de proyectos. Hay presupuestos y tiempos establecidos para finalizar un proyecto. Líderes de proyectos, recursos (espacio de oficina, computadoras) por adquirir, etc. Todo esto encaja apropiadamente con la visión de la administración de Proyectos.

Arte:

Los programas contienen muchos elementos artísticos. Las interfaces de usuario, la codificación, etc. Incluso la decisión para un nombre de una variable o una clase. Donald Knuth es famoso porque ha argumentado que la programación es un arte.

1.1.3 Metodología de la Ingeniería de Software.

No existe un único enfoque para solucionar los problemas relacionados con el software. Sin embargo, mediante la combinación de métodos para todas las fases del desarrollo, mejores herramientas para automatizar estos métodos, bloques de construcción más potentes para la implementación, mejores técnicas para la garantía de la calidad y una filosofía predominante para la coordinación, control y gestión, podemos conseguir una disciplina para el desarrollo de este llamada Ingeniería del Software (I.S.).

Se han dado muchas definiciones de sobre la I.S. y todas refuerzan la importancia de una disciplina de ingeniería para el desarrollo del software.

La I.S. surge de la Ingeniería de Sistemas y del Hardware, abarca un conjunto de tres elementos clave: métodos, herramientas y procedimientos. Estos elementos facilitan al gestor controlar el proceso del desarrollo y suministrar las bases para construir aplicaciones de alta calidad de una forma productiva.

Los métodos indican cómo construir técnicamente el software y consta de una amplia gama de tareas que incluyen la planeación y estimación de los proyectos, análisis de los requisitos del sistema, diseño de estructuras de datos, arquitectura de programas y procedimientos algorítmicos, codificación, prueba y mantenimiento.

Las herramientas suministran un soporte automático o semiautomático para los métodos. Existen herramientas para soportar cada uno de los métodos mencionados anteriormente. Cuando se integran las herramientas de forma que la información creada por una de ellas pueda ser usada por otra, se establece un sistema para el soporte del desarrollo del software llamado Ingeniería del Software asistida por computadora (C.A.S.E.). C.A.S.E. combina programas, hardware y base de datos sobre I.S. para crear entornos automatizados como CAD o AUTOCAD.

Los procedimientos de la I.S. son el pegamento que une los métodos y las herramientas y facilita un desarrollo racional y oportuno del software de computadora. Los procedimientos definen la secuencia en que se aplican los métodos, las entregas como documentos, informes, formas, entre otras, los controles que ayudan a asegurar la calidad y coordinar los cambios y directrices que apoyan a los líderes de proyecto a evaluar el progreso.

La I.S. esta compuesta por una serie de pasos que abarcan los métodos, las herramientas y los procedimientos antes mencionados, estos pasos se denominan frecuentemente paradigmas de la I.S. Tres son los paradigmas que se han tratado y debatido ampliamente: el ciclo de vida clásico, construcción de prototipos y el modelo espiral.

El ciclo de vida clásico.

este paradigma exige un enfoque sistemático y secuencial del desarrollo del software que comienza en el nivel del sistema y progresa a través del análisis, diseño, codificación, prueba y mantenimiento. Modelado a partir del ciclo convencional de una ingeniería, el paradigma del ciclo de vida abarca las siguientes actividades:

- Ingeniería y análisis del sistema. Debido a que el software es siempre parte de un sistema mayor, el trabajo comienza estableciendo los requisitos de todos los elementos del sistema y luego asignando algún subconjunto de estos requisitos al software. Este planteamiento del sistema es esencial cuando el software debe interrelacionarse con otros elementos, tales como hardware, personas y base de datos. La ingeniería y el análisis del sistema abarcan los requisitos globales a nivel del sistema con una pequeña cantidad de análisis y de diseño a un nivel superior.

- Análisis de los requisitos del software. El proceso de recopilación de los requisitos se centra e intensifica especialmente para el software, es decir, como para comprender la naturaleza de los programas que hay que construir, el analista debe comprender el ámbito de la información del software, así como la función, el rendimiento y las interfaces requeridos.

Los requisitos, tanto del sistema como del software, se documentan y se revisan con el cliente.

- *Diseño.* El proceso de diseño traduce los requisitos en una representación del software que pueda ser establecida de forma que obtenga la calidad requerida antes de que comience la codificación. Al igual que los requisitos, el diseño se documenta y forma parte de la configuración del software.

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

- *Prueba.* Una vez que se ha generado el código, comienza la prueba del programa. La prueba se centra en la lógica interna del software, asegurando que todas las sentencias se han probado y en las funciones externas, realizando pruebas que aseguren que la entrada definida produce los resultados que realmente se requieren.

- *Mantenimiento.* El software, indudablemente sufrirá cambios después de que se entregue al cliente. Los cambios ocurrirán debido a que se hayan encontrado errores o a que la aplicación deba adaptarse a cambios del entorno externo o debido a que el cliente requiera ampliaciones funcionales o del rendimiento. El mantenimiento del software aplica cada uno de los pasos precedentes del ciclo de vida a un programa existente en vez de a uno nuevo.

El ciclo de vida clásico es el paradigma más antiguo y más ampliamente usado en la I.S.

Las fases de este paradigma se muestran gráficamente en la figura 1.1.

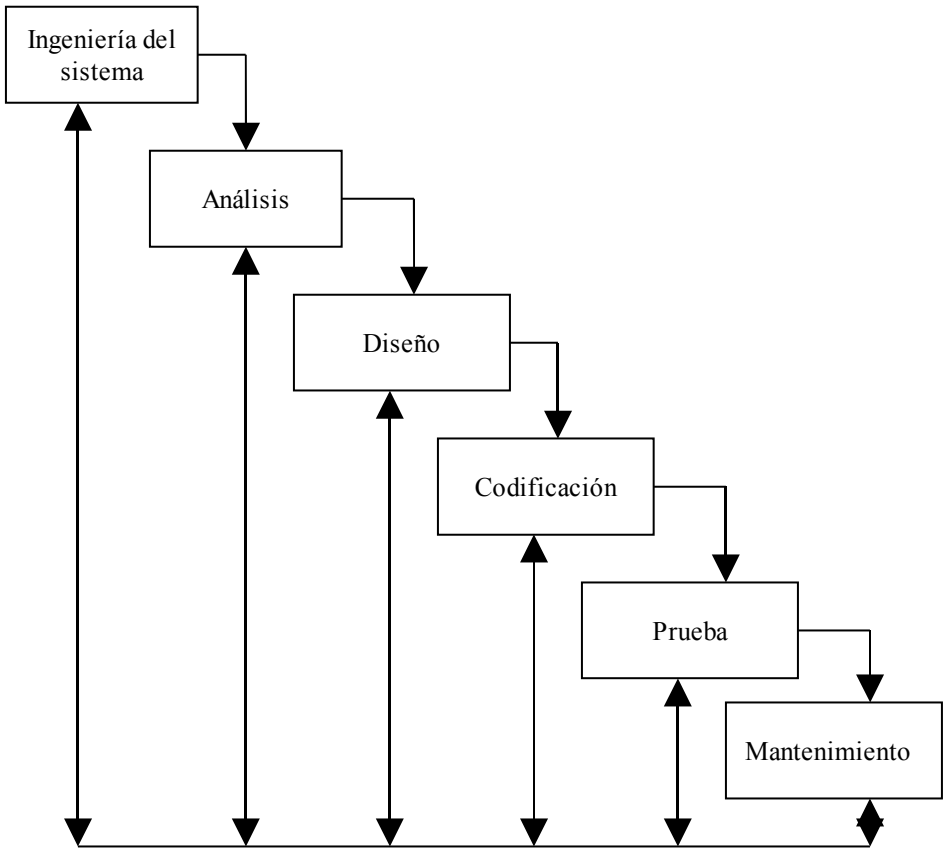


Figura 1.1. El ciclo de vida clásico.

Construcción de prototipos

Normalmente un cliente define un conjunto de objetivos generales para el software, pero no identifica los requisitos detallados de entrada, proceso o salida. En otros casos el programador puede no estar seguro de la eficiencia de un algoritmo, de la adaptabilidad de un sistema operativo etc. En estas y muchas otras situaciones, puede ser mejor método de I.S. la construcción de un prototipo.

La construcción de prototipos es un proceso que facilita la creación de un modelo del software a construir. El modelo tomará una de las tres formas siguientes:

1. Un prototipo en papel o un modelo basado en computadora que describa la interacción hombre-máquina, de forma que facilite al usuario la comprensión de cómo se producirá tal interacción.

2. Un prototipo que implemente algunos subconjuntos de la función requerida del programa deseado.

3. Un programa existente que ejecute parte o toda la función deseada, pero que tenga otras características que deban ser mejoradas en el nuevo trabajo de desarrollo.

La figura 1.2 muestra la secuencia de sucesos del paradigma de construcción de prototipos.

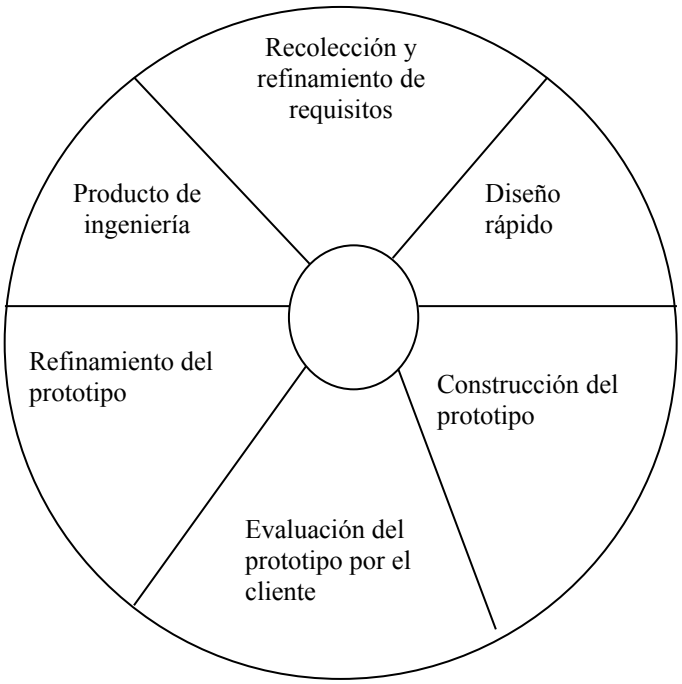


Figura 1.2 Creación de prototipos.

Como en todos los métodos de desarrollo, la construcción de prototipos comienza con la recolección de los requisitos.

El técnico y el cliente se reúnen y definen los objetivos globales para el software, identifican todos los requisitos conocidos y perfilan las áreas en donde será necesaria una mayor definición. Luego se produce un diseño rápido que se enfoca sobre la representación de los aspectos del software visibles al usuario, por ejemplo métodos de entrada y formatos de salida. El diseño rápido conduce a la construcción de un prototipo.

El prototipo es evaluado por el cliente/usuario y se utiliza para refinar los requisitos que se van a desarrollar.

Idealmente, el prototipo sirve como mecanismo para identificar los requisitos del software. Si se va a construir un prototipo que funcione, el realizador intenta hacer uso de fragmentos de programas existentes o aplica herramientas que faciliten la rápida generación de programas que funcionen.

Aunque pueden aparecer problemas como reglas de negocio mal programadas, se pasó por alto la validación de entrada de datos, el comportamiento del sistema ante situaciones no contempladas o que no suceden frecuentemente y no fueron contempladas, entre otros, la construcción de prototipos es un paradigma efectivo para la I.S.

Modelo en espiral.

El modelo en espiral ha sido desarrollado para cubrir las mejores características tanto del ciclo de vida clásico, como de la creación de prototipos, añadiendo al mismo tiempo un nuevo elemento: el análisis de riesgo, que falta en esos paradigmas. El modelo representado mediante la figura 1.3, define cuatro actividades principales representadas por los cuatro cuadrantes de un plano cartesiano.

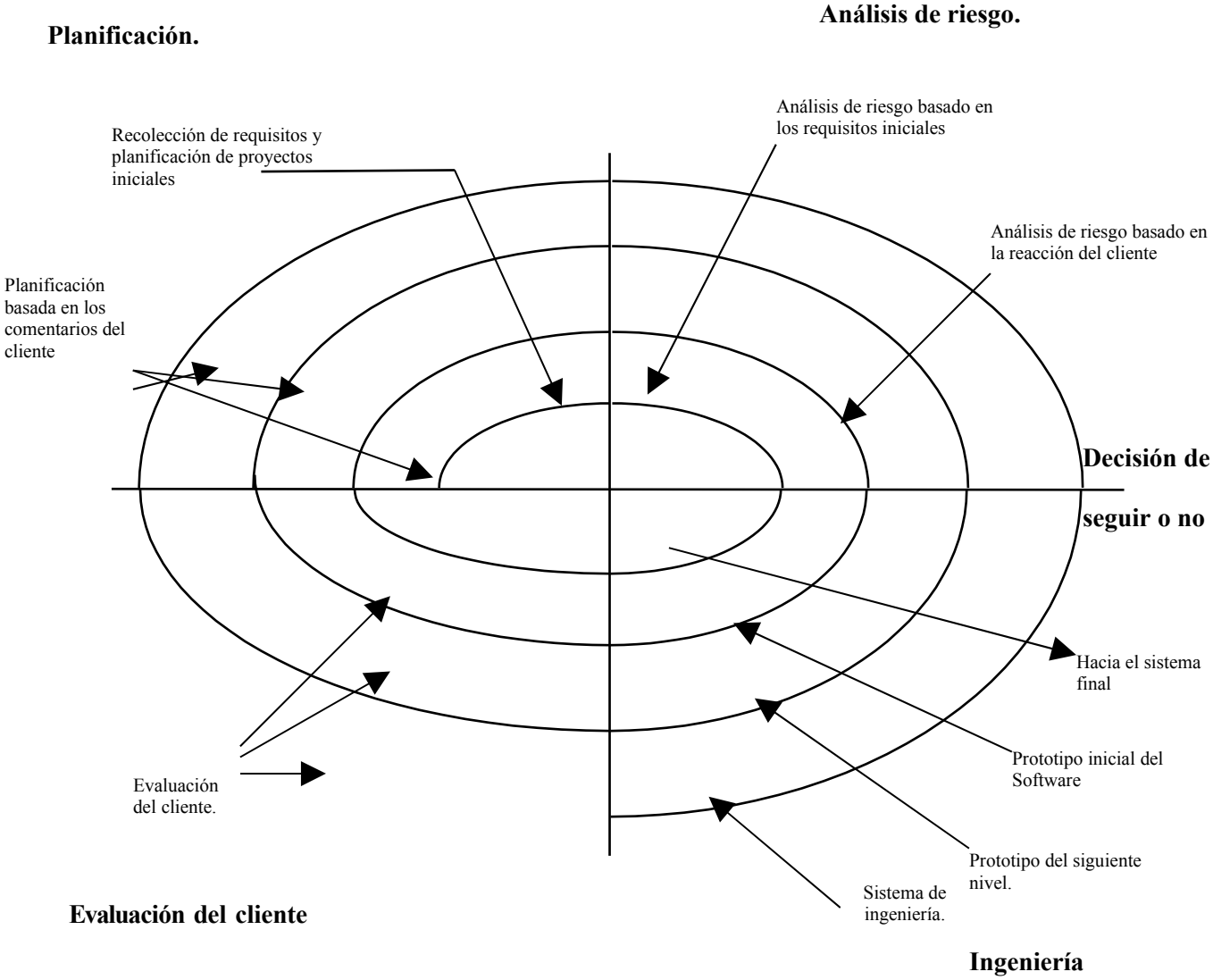


Figura 1.3 Modelo en espiral.

1. Planificación: determinación de objetivos, alternativas y restricciones
2. Análisis de riesgo: análisis de alternativas e identificación/resolución de riesgos
3. Ingeniería: desarrollo del producto de siguiente nivel.
4. Evaluación del cliente: valoración de los resultados de la ingeniería.

En cada iteración alrededor de la espiral (comenzando en el centro y siguiendo hacia el exterior), se construyen sucesivas versiones del software, cada vez más completas. Durante la primera vuelta alrededor de la espiral se definen los objetivos, las alternativas y las restricciones y se analizan e identifican los riesgos.

Sí el análisis de riesgo indica que hay una certidumbre en los requisitos, se puede usar la creación de prototipos en el cuadrante de ingeniería para dar asistencia tanto al encarado del desarrollo como al cliente. Se pueden usar simulaciones y otros modelos para definir más el problema y refinar los requisitos.

El cliente evalúa el trabajo de ingeniería (cuadrante de evaluación del cliente) y sugiere modificaciones. En base a los comentarios del cliente se produce la siguiente fase de planificación y de análisis de riesgo. En cada bucle alrededor de la espiral, la culminación del análisis de riesgo resulta en una decisión de seguir o no seguir. Si los riesgos son demasiado grandes, se puede dar por terminado el proyecto

Sin embargo, en la mayoría de los casos, se sigue avanzando alrededor del camino de la espiral y ese camino lleva a los desarrolladores hacia fuera, hacia un modelo más completo del sistema y al final, al propio sistema operacional.

Cada vuelta alrededor de la espiral requiere ingeniería (cuadrante inferior derecho), que se puede llevar a cabo mediante el enfoque del ciclo de vida clásico o de la creación de prototipos. Debe tenerse en cuenta que el número de actividades de desarrollo que ocurren en el cuadrante inferior derecho aumenta al alejarse del centro de la espiral.

El paradigma del modelo en espiral para la I.S. es actualmente el enfoque más realista para el desarrollo del software y de sistemas a gran escala. Utiliza un enfoque evolutivo permitiendo al desarrollador y al cliente entender y reaccionar a los riesgos en cada nivel evolutivo. Utiliza la creación de prototipos como un mecanismo de reducción del riesgo, pero, lo que es más importante, permite a quien lo desarrolla aplicar el enfoque de creación de prototipos en cualquier etapa de la evolución del producto.

1.1.4 Clasificación de las herramientas C.A.S.E.

La abreviación C.A.S.E. proviene de la expresión en inglés Computer-Aided Software Engineering (Ingeniería de Software asistida por computadora)[5].

C.A.S.E. es el nombre dado al software usado para ayudar a los líderes de proyectos o desarrolladores a automatizar sus actividades.

Estas herramientas abarcan un amplio rango de programas que son usados para soportar las actividades realizadas para desarrollar aplicaciones, entre estas se encuentran el análisis de los requerimientos, el modelo y pruebas para encontrar errores en los programas, etc.

Las herramientas C.A.S.E. se pueden clasificar por su función como instrumentos para el personal técnico o los directivos, por la arquitectura del entorno (hardware y software) que las soportan o incluso por su origen y coste. La principal clasificación se da por la funcionalidad como criterio principal.

Herramientas de planificación de sistemas de gestión.

Mediante el modelado de los requisitos de información estratégica de una organización, las herramientas de planificación de sistemas de gestión proporcionan un modelo del cual se pueden obtener sistemas de información específicos. El objetivo principal de las herramientas de esta categoría, es ayudar a comprender mejor cómo se mueve la información entre las distintas unidades organizativas.

Este tipo de herramientas proporciona una ayuda importante cuando se diseñan nuevas estrategias para los sistemas de información y cuando los métodos y sistemas actuales no satisfacen las necesidades de la organización.

Herramientas de gestión de proyectos.

Estas herramientas se centran en un elemento específico de la gestión del proyecto, en lugar de proporcionar un soporte global para la actividad de administración. Utilizando estas herramientas, el director del proyecto puede hacer estimaciones útiles de esfuerzo, coste y duración de un proyecto, definir una estructura de partición del trabajo y hacer una planificación realista del mismo y hacer el seguimiento de proyectos de forma continua.

Además el director puede utilizar estas herramientas para recoger datos que le permitan realizar una estimación de la productividad del desarrollo y de la calidad del producto.

[5] Libro Software Engineering, Ian Sommerville, séptima edición, página 12.

Herramientas de soporte.

La categoría de herramientas de soporte engloba las herramientas de aplicación y de sistemas que complementan el proceso de I.S. Las herramientas que caen en esta amplia gama recogen las actividades aplicables en todo el proceso de I.S. Estas incluyen instrumentos de documentación, de gestión de redes y software del sistema, control de calidad, otros más para la administración de bases de datos y configuración de aplicaciones.

Herramientas de análisis y diseño.

Las herramientas de análisis y diseño permiten al ingeniero de software crear un modelo del sistema que se va a construir. Dicho modelo contiene una representación del contenido y flujo de datos a través de una definición en un diccionario de requisitos, representaciones de los procesos y especificaciones de control entre otros detalles.

Toda esta representación ayuda al ingeniero de software evaluar la calidad del modelo y su consistencia, así como la localización y eliminación de errores antes de que se propaguen al código.

Herramientas de programación.

La categoría de herramientas de programación engloba los compiladores, los editores y los depuradores que se utilizan con los lenguajes de programación convencionales. También entran en esta categoría los entornos de programación orientados a objetos, los lenguajes de cuarta generación, los generadores de aplicaciones y los lenguajes de consulta a bases de datos.

Herramientas de mantenimiento.

Las herramientas de gestión de pruebas se utilizan para controlar y coordinar la prueba del software en cada una de sus etapas principales. Estas herramientas gestionan y coordinan las pruebas de regresión, realizan comparaciones para encontrar diferencias entre la salida esperada y la actual y realizan pruebas no interactivas de programas entre las interfaces.

También sirven como controladores genéricos de pruebas. Un controlador de pruebas lee uno o varios casos de un archivo, da formato a los datos para ajustarlos a las necesidades del software y después ejecuta el programa que va a ser verificado.

Estas herramientas a veces trabajan junto con otros instrumentos de seguimiento de requisitos para poder realizar un análisis de satisfacción de estos.

Las herramientas C.A.S.E. más populares en la actualidad son:

- Data Architect
- Erwin
- Racional Rose
- Together Control Center

1.2 Reingeniería de Software como sub rama de la Ingeniería de Software para la reestructuración de sistemas.

Cuando una aplicación ha servido a una compañía durante varios años, se vuelve inestable debido a las correcciones, adaptaciones y mejoras que se le realizan.

Esto lleva a que cada vez que se intenta efectuar un cambio, se produzcan efectos colaterales graves e inesperados y esto debido a que muchos sistemas siguen siendo desarrollados y mantenidos sin aplicar ninguna práctica de Ingeniería de Software (I.S.) por lo que hoy en día, muchas organizaciones se ven obligadas a seguir trabajando con sistemas de muy baja calidad y rendimiento, debido a que son vitales para el buen funcionamiento de la empresa.

La Reingeniería de Software (R.S.) es la actividad con la cual se intenta dar solución a estas organizaciones, su objetivo primordial es examinar los sistemas y aplicaciones de información con la intención de extraer de ellos conocimientos que permitan reestructurarlos o reconstruirlos de manera que el nuevo sistema sea confiable, mejore su rendimiento y su mantenimiento sea sencillo.

En muchos de los casos, la R.S. determina primero qué debe hacer un proceso para después decidir cómo debería hacerlo, olvidándose por completo de lo que es y se concentra en lo que debe ser.

Durante el proceso de análisis, se busca información acerca de la arquitectura, diseño y conexión de procedimientos. También se analiza el código fuente en busca de violaciones a las reglas de programación estructurada para reorganizar el código y de esta forma corregirlas.

Otro aspecto a analizar son los datos, principalmente se toma en cuenta el origen y se trata de identificar si son producidos por reglas de negocio o por otras causas; esto es importante porque si se llegan a reestructurar algunos de ellos, esto producirá un cambio inevitable del código.

Debido a que esta labor es estratégica, es conveniente conocer cuando conviene realizar la tarea de reingeniería para una aplicación y cuándo es más rentable sustituirla e implementar una nueva partiendo desde cero.

1.3 Historia y antecedentes de Infocaja.

Infocaja es una empresa que surge como una consultoría en el ramo restaurantero, que ingresa al mercado con una serie de aplicaciones que permiten llevar un control tanto de la operación de un restaurante como de la parte administrativa y estadística de sus productos y bodegas.

Sobre todo estas últimas aplicaciones son las que permiten a Infocaja posicionarse rápidamente en el mercado restaurantero como una empresa que ofrecía buenos productos para aquéllas personas que tuvieran la necesidad de controlar estos rubros de su negocio.

Las aplicaciones que vendía Infocaja estaban desarrolladas bajo el sistema operativo Msdos y programas como Clipper, Dbase, etc.

Con el paso del tiempo surgen problemas entre los socios de la empresa y terminan por separarse, dividiéndose también las diversas aplicaciones con la que contaban, una parte se adueña de los programas administrativos y la otra se queda con el punto de venta y con la gente del departamento de ventas.

Al tener de su lado al personal de ventas, se tiene una gran ventaja dado que estas personas saben perfectamente muchas de las sugerencias que los clientes les hacían con respecto de la aplicación, y se decide modificar el punto de venta original agregando varias de ésas sugerencias; la idea principal es crear el mismo punto de venta con programas que den una mejor presentación con nuevas características incorporadas.

El proyecto es asignado a un líder que no es del área de sistemas, sino un vendedor. Este líder por el contacto que tenía con los clientes, sabía que tipo de producto era el que se necesitaba y que necesidades tenía que cubrir ese nuevo punto de venta, pero no contaba con el conocimiento de las técnicas de diseño de bases y de programación, su único conocimiento en esta área se limitaba a saber que existía Visual Basic, que este era un lenguaje fácil de usar y con una presentación de pantallas similar a lo que usa el sistema operativo Windows.

Para cumplir con esta nueva meta, se crea un equipo de programadores sin experiencia alguna en el diseño de bases de datos y programación, esto con la finalidad de reducir los costos en el pago de sus salarios ya que contratar a gente con experiencia implicaba gastar más en su paga y la empresa no se encontraba en las mejores condiciones económicas debido a la división que se acababa de dar.

Así en tanto el líder de proyecto como su equipo tenían que aprender y estudiar sobre la marcha, se obtuvo un producto que cumplía con las necesidades que los clientes tenían pero con un diseño en la base de datos muy pobre y un código que no esta desarrollado con las reglas de programación estructurada u orientada a objetos.

El producto obtenido fue Infocaja visual, el cual satisface muchas de las necesidades de los dueños de restaurantes pero ahora se tiene el problema de volver a posicionar el producto en el mercado además de tener que competir con un equivalente de la otra parte de la empresa.

Lo antes mencionado junto con una debilidad en las finanzas de esta pequeña empresa, impiden que se pueda promocionar el producto con la fuerza necesaria para su venta, sin embargo se logra obtener algunos clientes nuevos con este producto, pero la empresa no ve un crecimiento en su economía por lo que el dueño de esta parte de la empresa dividida decide abandonar el mercado y cerrar el negocio.

1.4 Necesidad de aplicar la reingeniería de Software.

El programa ha sufrido cambios, debido a fallas y errores en la programación, pero principalmente debido a que el cliente requiere ampliaciones funcionales o de rendimiento.

Desde la concepción de la aplicación no se empleó ninguna metodología de las expuestas anteriormente y en su mantenimiento tampoco se planeó utilizar a la I.S. para resolver y mantener un control estricto en el desarrollo del sistema.

La falta de una metodología como las que ofrece la I.S. al desarrollo y mantenimiento, ha llevado a que el programa ahora ya sea difícil de corregir o ampliar ya que cualquier movimiento en el código fuente afecta al resto de los procedimientos, e incluso las mismas personas encargadas del mantenimiento ya desconocen la función de tal o cual módulo de procedimiento, tabla en la base de datos o columnas en estas. De aquí la necesidad de aplicar la reingeniería para tener un producto al cual se le pueda dar mantenimiento y a la vez agregar nuevas funciones al programa resultante.

La R.S. como ya se analizó, es una disciplina tecnológica preocupada de la producción sistemática y mantenimiento de los productos de software. Difiere de la programación tradicional en que se utilizan técnicas de ingeniería para especificar, diseñar instrumentar, validar y mantener los productos dentro del tiempo y presupuestos establecidos para el proyecto.

Además la I.S. abarca un conjunto de tres elementos claves: métodos, herramientas y procedimientos.

Los métodos indican como construir técnicamente el software, estos abarcan tareas como planificación y estimación de proyectos, análisis de los requisitos del sistema y del software, diseño de estructuras de datos, arquitectura de programas y procedimientos algorítmicos, codificación, prueba y mantenimiento.

Las herramientas suministran un soporte automático o semiautomático para los métodos. Hoy existen herramientas para soportar cada uno de los métodos mencionados anteriormente. Dentro de las múltiples herramientas que existen se encuentran las denominadas herramientas C.A.S.E. las cuales combinan software, hardware y bases de datos, para crear un entorno de I.S.

Los procedimientos son el pegamento que une a los métodos y las herramientas y facilita un desarrollo racional y oportuno del software de computadora. Los procedimientos definen la secuencia en la que se aplican los métodos, las entradas (documentos, informes, formas, etc.) que se requieren, los controles que ayudan a asegurar la calidad y coordinar los cambios que ayudan a los desarrolladores del software a evaluar el progreso.

Como se puede observar la Reingeniería de Software nos ofrece técnicas y herramientas adecuadas para las necesidades de depuración y mantenimiento de esta aplicación.

Capítulo II

Análisis del punto de venta.

Análisis del punto de venta.

2.1 Programas usados por el sistema.

2.2 Decisiones que motivaron a usar este software.

2.3 Requerimientos del sistema.

2.4 Análisis de la base de datos de la aplicación.

2.4.1 Control de clientes y usuarios.

2.4.2 Control mesas.

2.4.3 Captura y armado de platillos.

2.4.4 Control de comandas.

2.4.5 Control de pagos.

2.4.6 Impresión de documentos.

2.5 Análisis de pantallas de la aplicación.

2.5.1 Pantallas representativas del aspecto de la aplicación.

2.1 Programas usados por el sistema.

El punto de venta Infocaja esa basado en los siguientes programas:

- Base de datos: Microsoft Access 87 no posterior.
- Lenguaje de programación: Microsoft Visual Basic versión 6.0.
- Sistema operativo: Microsoft Windows cualquier versión.

Como se advierte, el punto de venta tiene una clara tendencia por software comercial de la empresa Microsoft conocido como programas de código cerrado o protegido.

2.2 Decisiones que motivaron a usar este software.

Las razones por las que se inclinó a crear esta aplicación con el software antes mencionado son las siguientes:

Como se mencionó anteriormente, la falta de experiencia pero sobre todo de conocimiento por parte del líder de proyecto y del equipo de programadores motivo a inclinarse por programas que fueran fáciles de usar como Visual Basic y Access 97. Es bien sabido que este manejador de base de datos es sencillo de usar en comparación con otros existentes en el mercado como Oracle, Progress, Microsoft Sql Server, Mysql, etc. Además de exigir muy pocos recursos en la computadora, Access es una excelente opción para aplicaciones que no soporten una gran cantidad de información.

El lenguaje de programación Visual Basic versión 6, también es muy sencillo de usar y ofrece una muy buena apariencia en las pantallas que presentan la información al usuario; además de ofrecer una conexión muy sencilla a la base de datos Access.

Pero el principal motivo para usar estas herramientas es el hecho de que la mayoría de las personas ya están familiarizadas con el entorno del sistema operativo Windows, así que una aplicación que ofrece un ambiente similar al de este sistema operativo resulta más amigable, fácil de usar y aprender.

2.3 Requerimientos del sistema.

Este punto de venta necesita los siguientes requerimientos para su correcto funcionamiento. Computadora Pentium III o superior (incluido su similar amd) con sistema operativo Windows; 96 megabytes de memoria RAM que garantizan el correcto funcionamiento del programa y 1 gigabyte de espacio en disco duro para almacenar tanto la aplicación como los respaldos que se realizan a diario.

En cuanto a monitor, el programa ofrece la posibilidad de usar un monitor de toque (touch screen) o un monitor normal. Este primero reemplaza el teclado y le da una gran apariencia y elegancia al punto de venta.

Entorno de red a 100 mbps. si es que se piensa utilizar este programa con varias terminales distribuidas en el negocio.

Es importante aclarar que Infocaja no puede ser usado en computadoras distintas a las compatibles con IBM, es decir, no existe una versión de Infocaja para computadoras Apple así como sistemas operativos diferentes a Windows (Linux, Unix, etc.).

El sistema trabaja bajo un ambiente de red simulado; esto es; que en la mayoría de las instalaciones realizadas a clientes se instalaba en programa en una computadora y se compartía la carpeta donde se encontraba el archivo ejecutable, mientras que en el resto de las computadoras se creaba un acceso directo hacia el ejecutable y con este mecanismo se simulaba un ambiente de red debido a que las diferentes terminales podían ejecutar la aplicación y esta radicaba solo en una computadora

Como se puede concluir, Infocaja no es una aplicación que cumpla con las características de una verdadera aplicación distribuida para redes. A pesar de que las necesidades de la mayoría de los clientes eran un buen motivo para realizar esfuerzos de agregar la característica de aplicación distribuida, nunca se dedicó el tiempo para realizarlo y se decidió mantener el programa simulando esta característica.

2.4 Análisis de la base de datos de la aplicación.

2.4.1 Control de clientes y usuarios.

Control de clientes.

Muchos de los restaurantes tienen la necesidad de llevar un control de los clientes que frecuentan el lugar, esto con el objetivo de expedir documentos fiscales tales como notas o facturas o si son clientes asiduos del lugar darles un trato más personalizado y ofrecerles algún descuento especial o alguna bebida de cortesía cuando visitan nuevamente el lugar, etc.

También el negocio necesita llevar un control sobre su personal. Información como datos personales, direcciones y números telefónicos donde pueden ser localizados son parte del control que se necesita tanto en clientes como en empleados.

Análisis interno.

Para llevar a cabo la tarea del control de clientes y el personal que labora para un restaurante, el sistema usa dos grupos de clasificación llamados Usuario y Cliente así como dos tablas para guardar la información ingresada, en el siguiente recuadro se muestra el nombre de estas tablas y su función.

Nombre de la tabla	Uso
Colonias	Guarda el nombre de la colonia, ciudad y estado donde se encuentra, así como la fecha, hora e identificador del usuario que capturó dicha información.
Cliente	Guarda la información personal de un cliente, así como su dirección y teléfono así como la fecha, hora e identificador del usuario que capturó dicha información.

La estructura y relación entre estas dos tablas se muestra en la figura 2.1.

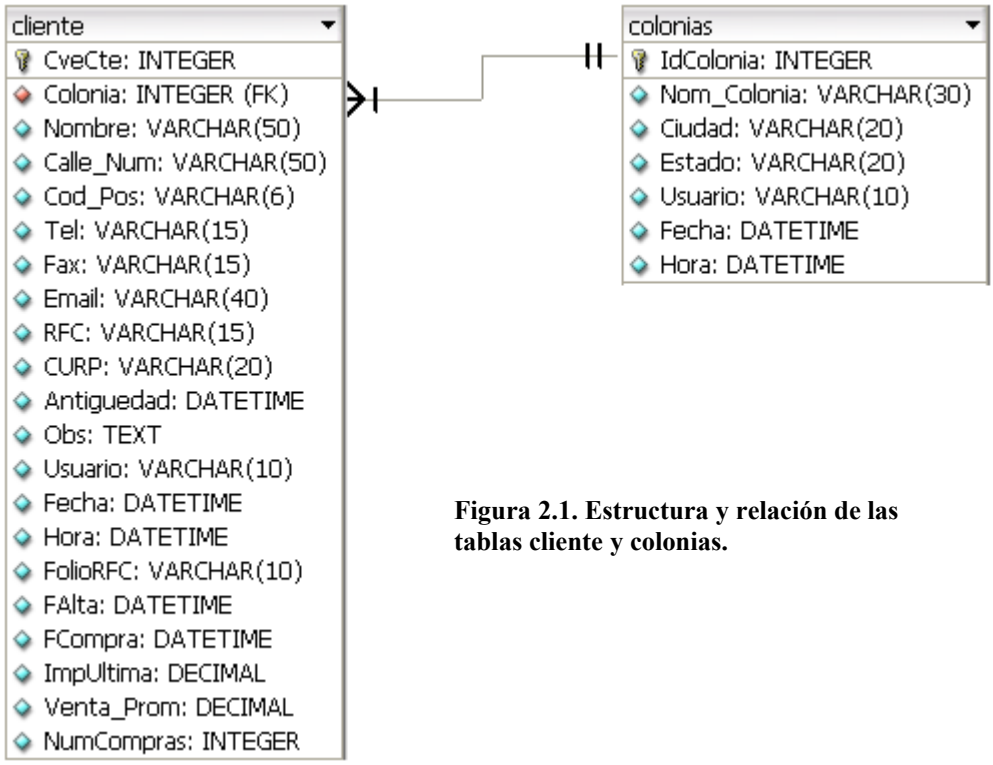


Figura 2.1. Estructura y relación de las tablas cliente y colonias.

Como se observa, la estructura de la tabla *Colonias* esta formada por una clave primaria *IdColonia* y campos como *Nom_Colonia, Ciudad y Estado* que contienen el nombre de la colonia y el nombre de la ciudad y del estado donde se ubica dicha colonia; los campos *Usuario, Fecha, Hora* sirven para saber quien agregó dicho registro en la tabla, que día y a que hora aunque estos últimos nos son utilizados por el sistema ya que dicho control no fue programado, además con esta estructura no se permite la reutilización de datos como el nombre de la ciudad o del estado ya que estos datos tienen que ser repetidos por cada colonia que tenga en común esta información.

Por otra parte, la estructura de la tabla *Cliente* esta formada por una clave primaria *CveCte*, una clave externa *Colonia* y campos como *FAlta, FCompra, Obs, FolioRfc* e *ImpUltima* que no son utilizados por el sistema. El resto de los campos tienen como finalidad guardar información personal de cada cliente como *RFC, CURP, Nombre,* etc. Cabe resaltar que también se tienen los campos *Usuario, Fecha, Hora* cuyo objetivo era el mismo descrito en el párrafo anterior y no son utilizados por los mismos motivos ya mencionados; aunado a esto, esta estructura restringe a solo dos números telefónicos por cada cliente (campos *Tel y Fax*) por lo que no se puede tener información extra como por ejemplo número de celular y un número de oficina .

Control de usuarios.

Análisis interno.

El sistema puede llevar el control del acceso de los usuarios que sean registrados para usarlo; parte del control consiste en permitir a un usuario agregar un platillo, eliminarlo, cobrar la cuenta, levantar órdenes, etc. Para ello los usuarios de la aplicación deben se darse de alta y posteriormente asignárseles los derechos adecuados.

Internamente el programa utiliza las tablas descritas en el siguiente recuadro.

Nombre de la tabla	Uso
Puesto	Guarda los distintos puestos que están definidos dentro del negocio, así como la fecha, hora e identificador del usuario que capturó dicha información.
Usuario	Guarda la información relacionada con un usuario, así como la fecha, hora e identificador del usuario que capturó dicha información y el puesto con el que se relaciona dicho usuario.
Derecho	Guarda los derechos aplicados a un usuario en particular.

La estructura y relación entre estas dos tablas se muestra en la figura 2.2.

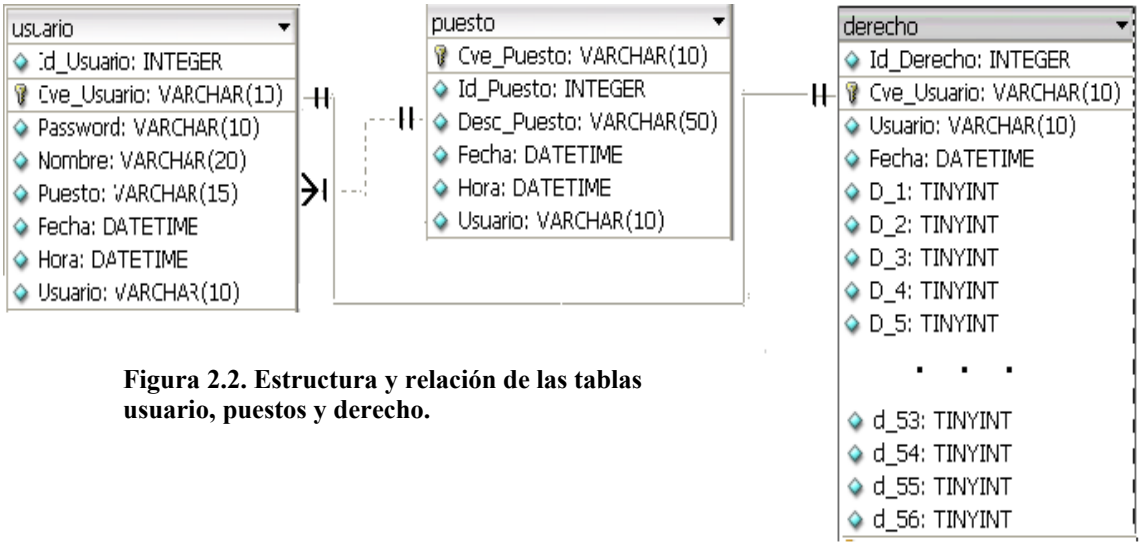


Figura 2.2. Estructura y relación de las tablas usuario, puestos y derecho.

De la estructura mostrada en la figura 2.2 se puede observar que en la tabla *Puesto* existen dos campos cuyo nombre da la idea de que juegan el papel de clave primaria*: Cve_Puesto* e *Id_Puesto*, siendo la clave principal el campo *Cve_Puesto*. Nuevamente se aprecia la aparición de los campos *Usuario, Fecha, Hora* que buscaban dar la funcionalidad ya descrita y que a partir de este momento ya no se mencionará más a pesar de que estos campos aparecerán en el resto de las tablas del sistema.

Por consecuencia el único campo importante además de la clave primaria es *Desc_Puesto* que contiene el nombre o descripción del puesto.

La estructura de la tabla *Usuario* también contiene el error de tener dos campos cuyo nombre da la idea de que cada uno de ellos es la clave primaria del sistema; los campos mencionados son *Id_Usuario* y *Cve_Usuario* siendo la clave principal *Cve_Usuario.*

El objetivo de tener esta tabla es guardar el nombre y clave de un empleado que serán utilizados por la aplicación para permitir o negar el acceso a dicho empleado. Los campos *Cve_Usuario* y *Password* almacenan dicha información, por lo que se puede concluir que el campo *Id_Usuario* debería ser la clave principal de la tabla y no el campo *Cve_Usuario* como aparece resaltado en el diagrama entidad relación. La tabla también contiene el nombre del usuario y el puesto que ocupa dentro del negocio.

Cabe resaltar la forma en que fue establecida la relación entre las tablas *Usuario* y *Puesto*.

Como se mencionó anteriormente, la clave primaria de la tabla *Puesto* es el campo *Cve_Puesto*, por lo que la relación entre estas tablas debería llevarse a cabo por medio de esa columna, pero la relación entre dichas tablas se realiza por medio de las columnas *Desc_Puesto* de la tabla *puesto* y *Puesto* de la tabla *usuario*, por lo que se concluye que a pesar de que existe la relación entre estas tablas, esta mal diseñada tomando en cuenta las reglas de la normalización.

En relación con los derechos, no existe una tabla que contenga el significado de cada derecho existente, dicha información esta mezclada con el código del sistema, para dar una idea se muestra un fragmento de código:

```
Comando_SQL$ = "Select * " _
        & "From   derecho " _
        & "Where  cve_usuario = '" & Text2.Text & "'"
    Set rst_uno = Db.OpenRecordset(Comando_SQL$, dbOpenDynaset)
    With rst_uno
```

```
If !d_4 Then    'abrir mesa
Frmderechos.Label1(4).Caption = "si"
Else
Frmderechos.Label1(4).Caption ="no"
End If

If !D_5 Then    'cambio de mesa
Frmderechos.Label1(5).Caption = "si"
Else
Frmderechos.Label1(5).Caption ="no"
End If

If !d_6 Then    'cancelar mesa
Frmderechos.Label1(6).Caption = "si"
Else
Frmderechos.Label1(6).Caption ="no"
End If

If !d_7 Then    'alta de comanda
Frmderechos.Label1(7).Caption = "si"
Else
Frmderechos.Label1(7).Caption ="no"
End If

If !d_8 Then    'cancelar comanda
Frmderechos.Label1(8).Caption = "si"
Else
Frmderechos.Label1(8).Caption ="no"
End If
```

Para hacer más sencilla la comprensión de este fragmento, se han resaltado los comentarios donde se explica o da una pista de cual derecho se esta controlando con las líneas de código; así, del bloque resaltado en gris, se deduce que ese bloque de código sirve para visualizar o no en la ventana de asignación de derechos al permiso con el nombre *alta de comanda*.

Como se mencionó, esta es la única fuente de información que se tiene en relación con los derechos y como se puede observar la palabra d_7 es el nombre de una columna de la tabla *Derecho*.

Al analizar la estructura de la tabla *derecho*, se recalca la columna *Id_Derecho* la cual por su nombre da la impresión de ser la clave primaria de la tabla pero en realidad, la clave primaria de esta tabla es el campo *Cve_Usuario* como se puede advertir de la figura 2.2.

Esta ambigüedad se origina por la necesidad de restringir a cada usuario con solamente un registro (los permisos que tiene asignado); de ahí que se haya hecho del campo *Cve_Usuario* la clave primaria de esta tabla para de esta manera evitar que un usuario tenga datos duplicados.

Otro punto importante a recalcar en esta estructura, es que solo permite el uso de 56 derechos, si se llegase a necesitar un permiso extra se tendría que modificar la estructura de la tabla y el código para que se tenga control sobre ese nuevo permiso.

2.4.2 Control mesas.

El sistema puede llevar el control de todas las mesas con las que cuenta el local para atender a los clientes. Para llevar a cabo esta actividad se cuenta con una sola tabla llamada *Mesa*, la relación de esta tabla con las demás que componen la base de datos se muestra en la figura 2.3.

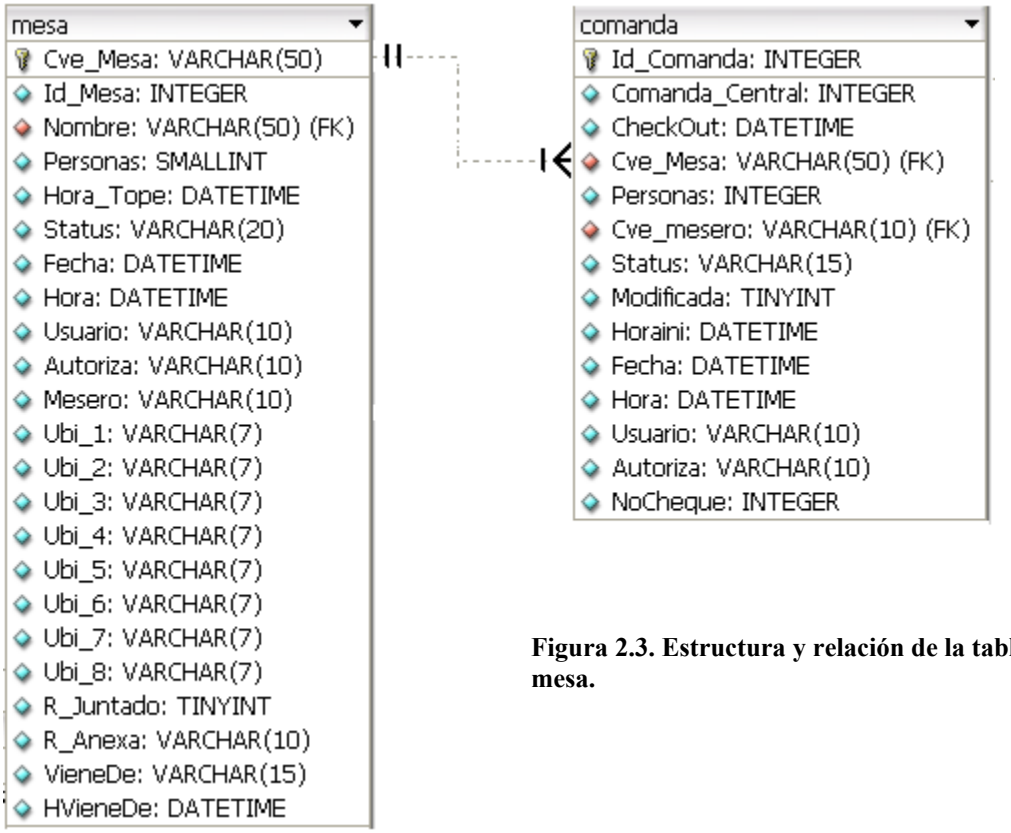


Figura 2.3. Estructura y relación de la tabla mesa.

Del diagrama anterior, se aprecia que la única relación que mantiene la tabla *Mesa* es con la tabla comanda; esta relación se analizará mas adelante en la parte del análisis de control de comandas.

Se puede observar que la tabla *Mesa* contiene dos campos *Cve_Mesa* y *Id_Mesa* siendo que la clave primaria es el campo *Cve_Mesa* dado que también es la clave externa en la tabla *Comanda* y es la que permite el enlace entre estas dos tablas por lo que la columna *Id_Mesa* es un campo innecesario. Cabe resaltar nuevamente el hecho de que el nombre de estos dos campos sugiere que ambos pueden ser la clave primaria de la tabla.

A continuación aparece el campo *Nombre*, el cual es utilizado para guardar el nombre de algún cliente que haya realizado una reservación.

El campo *Personas* se utiliza para indicar el número de personas que se esperan cuando existe una reservación o para indicar cuantas personas están sentadas en esa mesa en un momento dado.

El campo *Hora_Tope* sirve para saber hasta que hora es posible esperar a un cliente cuando este haya realizado una reservación y así poder decidir si es mejor cancelar la reservación y asignar la mesa a otros comensales que estén en espera de ser atendidos.

El campo Status indica en que fase del ciclo de atención al cliente se encuentra una mesa y permite al sistema tomar la decisión de cómo reflejar este hecho en pantalla para los usuarios como meseros o hostes (personas encargadas de asignar mesa a los clientes).

El ciclo de atención al cliente es el siguiente:

Fase	Significado	Representación del sistema para el usuario.
Vacía	La mesa esta lista para ser asignada a algún cliente.	La mesa se representa en pantalla con un color blanco.
Abierta	Los comensales están ocupando sus lugares en la mesa recién asignada a ellos.	La mesa se representa en pantalla con un color verde claro.
Ocupada o Modificada	Los comensales están degustando los platillos solicitados.	La mesa se representa en pantalla con un color naranja.
Cerrada	Los comensales han solicitado la cuenta del servicio.	La mesa se representa en pantalla con un color azul claro.
Pagada	Los comensales están pagando su cuenta y están a punto de abandonar la mesa.	La mesa se representa en pantalla con un color amarillo.
Reservada	La mesa no puede ser asignada a clientes porque fue reservada previamente.	La mesa se representa en pantalla con un color rojo.

Como ejemplo práctico se puede observar la figura 2.4 donde se muestra la pantalla utilizada por el sistema para indicar al usuario en que status se encuentra una mesa.

Figura 2.4. Pantalla de la aplicación mostrando mesas en diferentes status o etapas.

De la figura se puede observar que la mesa 12 (número más grande dentro de cada círculo) se encuentra en la fase *abierta (se puede asignar a un comensal)* la mesa 20 se encuentra en la fase *ocupada o modificada*, la mesa 21 se encuentra en la fase *cerrada (el o los comensales han solicitado la cuenta del consumo)*, la mesa 22 se encuentra en la fase *pagada*, la mesa 23 esta *reservada* y que el resto de las mesas están en la fase *vacías*.

De esta manera la aplicación ayuda a los encargados de asignar mesa a los clientes para tomar la decisión de ofrecerle mesa al cliente en un corto plazo o invitarlo a regresar un poco más tarde por falta de mesas que vayan a desocuparse pronto.

2.4.3 Captura y armado de platillos.

Los restaurantes presentan los platillos a los clientes en una lista llamada carta. La carta contiene los nombres de los platillos o bebidas, los ingredientes que los forman y el precio de cada uno de ellos.

Por ejemplo en la carta de un restaurante podemos encontrar el siguiente platillo y la siguiente bebida:

Platillo o bebida	Ingredientes	Precio
Ravioles de ricota y espinaca	Con tres quesos, salsa de chile poblano y con anchoas	$65
Malteada	De chocolate, vainilla o fresa con hielo.	$24

De esta forma los clientes eligen los platillos que desean consumir y si desean pueden omitir algún ingrediente si no es de su agrado.

Control interno.

La primera actividad que tiene que realizar el operador del sistema para llevar el control de la carta, es capturar todos los ingredientes (modificadores) que se utilicen al preparar los diferentes platillos o bebidas que ofrece el negocio; esto debido a que más adelante se capturarán los platillos y si no se tienen los ingredientes previamente capturados no se podrán relacionar porque el programa no mostraría ingredientes en la pantalla correspondiente.

Cabe señalar que el sistema llama modificadores a todos los ingredientes de un platillo, pero se hará mención a ingredientes para evitar confusiones con el término modificador.

Análisis interno del sistema.

Para ofrecer toda la funcionalidad mostrada anteriormente, el programa utiliza una serie de tablas para guardar la información ingresada, en el siguiente recuadro se muestra el nombre de estas tablas.

Nombre de la tabla	Uso
Tgrupo	Grupos para clasificar a los tipos de grupo (A)limentos, (B)ebidas y (O)tros, sus precios e impuestos
Grupo	Subgrupos dentro de los grupos para clasificar a los platillos, precios e impuestos aplicables a estos grupos
Platillo	Nombre de los platillos que sirve el restaurante, sus precios e impuestos
Modificador	Nombre de los ingredientes o modificaciones hechas a un platillo, su precio extra e impuestos aplicables.
Enlace_modificador	Relación entre platillos y los modificadores

La estructura y relación entre estas dos tablas se muestra en la figura 2.5.

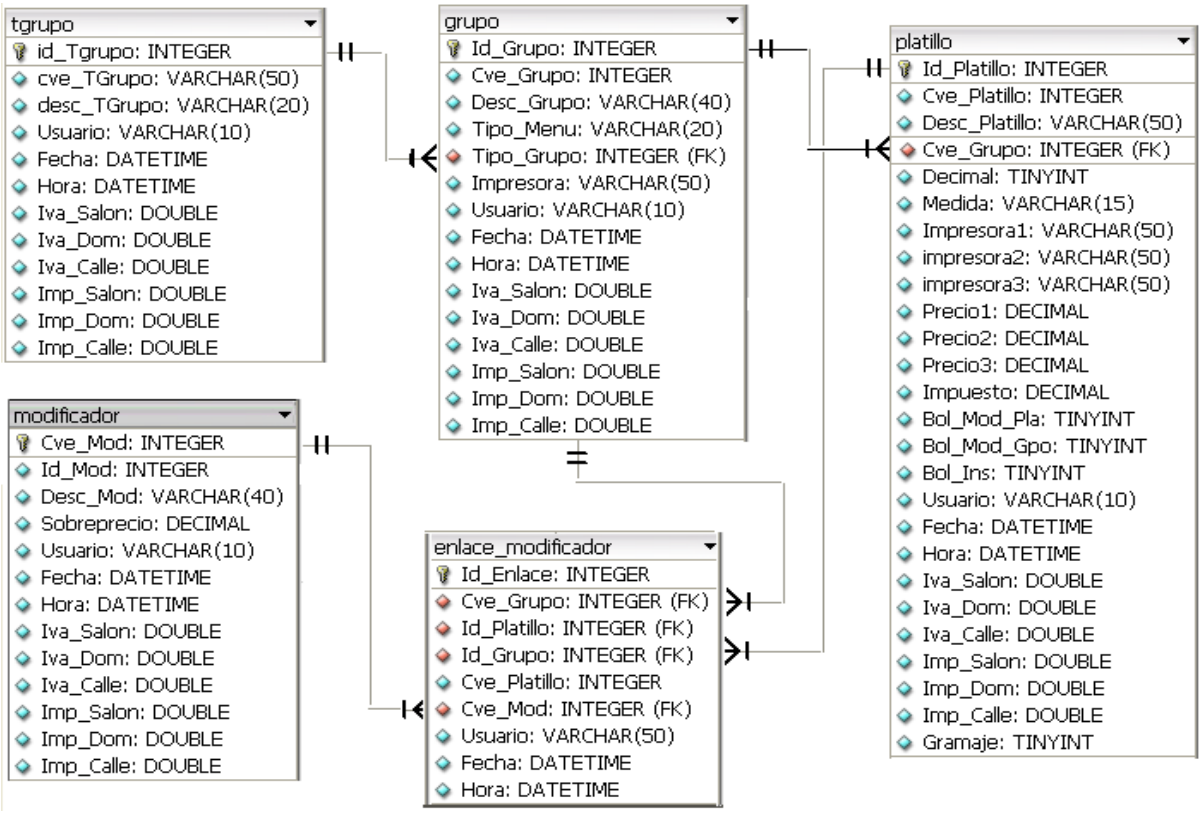


Figura 2.5. Estructura y relación de las tablas tgrupo, grupo, platillo, modificador y enlace_modificador.

La primera tabla leída y modificada por el sistema es la tabla *Modificador*, que como se mencionó anteriormente guarda la información relacionada con los ingredientes que se utilizan en la preparación de los platillos del negocio.

Esta tabla contiene dos campos con valores únicos llamados *Id_Mod* y *Cve_Mod* cuyos nombres dan la idea de ser la clave primaria de la tabla, siendo *Cve_Mod* la que juega este papel dentro de la estructura.

Enseguida aparece el campo *Desc_Mod* que contiene el nombre de un ingrediente y la columna *Sobreprecio* que almacenará el precio extra por el uso de este ingrediente.

Posteriormente se encuentran tres columnas para tres tipos de *IVA* e *impuestos*, estas columnas tenían como objetivo dar la oportunidad de tener tres valores diferentes del *IVA* y tres valores para otro tipo de impuestos. De esta manera si en alguna ley se exigía cobrar un impuesto adicional por el ingrediente se tenía la oportunidad de reflejar ese cambio en alguna de estas columnas. Normalmente estos IVAS e impuestos no se utilizan ya que como se puede observar en la figura anterior los valores de estas columnas son 0.

La siguiente tabla en ser usada es la tabla *Grupo*. Esta tabla es la encargada de almacenar la información relacionada con los subgrupos.

Del diagrama entidad relación de la figura 2.5, se observan dos columnas *Id_Grupo* y *Cve_Grupo*, las cuales tienen el mismo valor en sus columnas siendo *Id_Grupo* la clave primaria de la tabla.

A continuación el campo *Desc_Grupo* guarda el nombre, la columna *Tipo_Menu* indica si este grupo se puede servir en primer tiempo, segundo tiempo o tercer tiempo, su valor no es usado regularmente al tomar una orden. La columna *Tipo_Grupo* indica si este subgrupo esta clasificado dentro de los alimentos (A), bebidas (B) o dentro del grupo principal otros(O); la información de estos grupos se encuentra dentro de la tabla *Tgrupo*.

La columna *Impresora* tiene el nombre o identificador de la impresora donde se va a enviar la información para la preparación de un platillo contenido en este grupo, por ejemplo, todos los platillos que se clasifiquen dentro del grupo Aves (Id_Grupo = 6) se imprimirán en la impresora cocina que se supone deberá estar ubicada en el mismo lugar que indica su nombre.

Los campos *Iva_Salon, Iva_Dom, Iva_Calle, Imp_Salon, Imp_Dom* e *Imp_Calle* buscan dar la funcionalidad de manejar tres ivas distintos y tres impuestos extras en caso de ser requeridos.

Cabe señalar que esta última idea de manejar tres ivas e impuestos distintos para un registro ya se había manejado en la tabla *Ingredientes*.

Dentro del ciclo de armado de platillos, se emplean las tablas *Platillo y Enlace_Modificador*.

La tabla *Platillo* contiene dos campos idénticos *Id_Platillo* y *Cve_Platillo* los cuales tienen la misma información siendo la clave primaria de la tabla la columna *Id_Platillo*.

La columna *Desc_Platillo* contiene el nombre del platillo, la columna *Cve_Grupo* aloja el identificador del subgrupo dentro del cual esta clasificado el platillo en cuestión.

Los campos *Decimal, Medida* y *Gramaje* fueron añadidos a la tabla con la finalidad de que el sistema pudiera calcular el precio de un platillo que se vendiera en proporciones como la barbacoa, que se puede vender por *Gramaje* o cantidad variada y su precio depende de la cantidad pesada, esta característica no se agrego al sistema por lo que los campos no son usados.

Las columnas *Impresora1, Impresora2* e *Impresora3*, tienen el nombre de tres impresoras distintas; estos campos buscaban incrementar la capacidad del sistema en el sentido de poder redireccionar o imprimir en distintos lugares un platillo para su preparación.

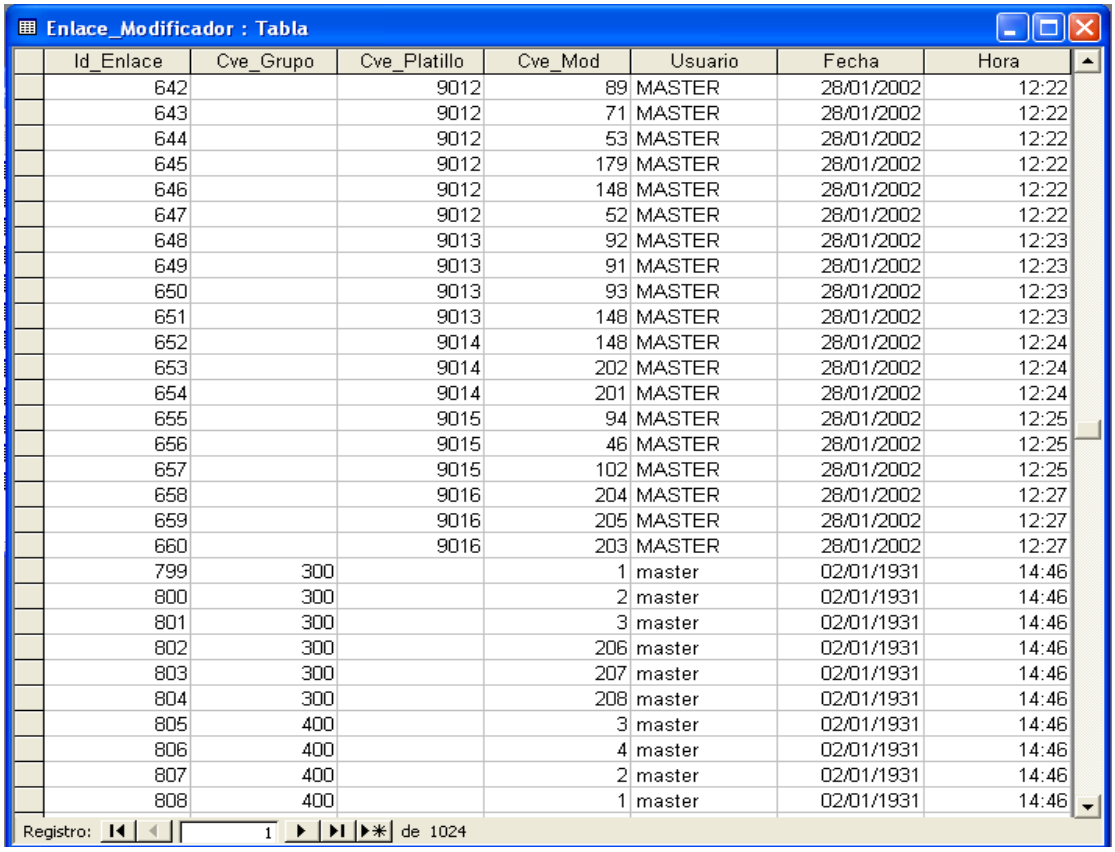
También es notoria la presencia de tres campos llamados *Precio1, Precio2* y *Precio3*, la finalidad de estas columnas era la de permitir tener más de un precio asignado a un platillo; la idea era la misma que con las impresoras.

Con la finalidad de aumentar la versatilidad del programa en el manejo de varios impuestos para un solo platillo, se agregan tres columnas para el IVA y tres más para impuestos de otro tipo, como se ha venido haciendo con tablas anteriores, así podemos encontrar nuevamente las columnas Iva*_Salon, Iva_Dom, Iva_Calle, Imp_Salon, Imp_Dom, Imp_Calle* más una columna nueva *Impuesto*.

Las columnas *Bol_Mod_Pla y Bol_Mod_Gpo* son de tipo falso o verdadero, estas columnas indican si algunos de los ingredientes que formaban parte de este, estaban asignados al subgrupo donde se encontraba clasificado o eran exclusivos del platillo, normalmente estos campos no eran de gran utilidad para el sistema.

El campo *Bol_Ins* también es del mismo tipo de las columnas del párrafo anterior y servía únicamente para retrasar el envío de un platillo a una impresora para su preparación, normalmente este campo tampoco era muy utilizado por el sistema.

La última tabla en usarse en esta fase es la tabla *Enlace_Modificador*, parte de la información de esta tabla se muestra en la figura 2.6.



Enlace_Modificador : Tabla

Id_Enlace	Cve_Grupo	Cve_Platillo	Cve_Mod	Usuario	Fecha	Hora
642		9012	89	MASTER	28/01/2002	12:22
643		9012	71	MASTER	28/01/2002	12:22
644		9012	53	MASTER	28/01/2002	12:22
645		9012	179	MASTER	28/01/2002	12:22
646		9012	148	MASTER	28/01/2002	12:22
647		9012	52	MASTER	28/01/2002	12:22
648		9013	92	MASTER	28/01/2002	12:23
649		9013	91	MASTER	28/01/2002	12:23
650		9013	93	MASTER	28/01/2002	12:23
651		9013	148	MASTER	28/01/2002	12:23
652		9014	148	MASTER	28/01/2002	12:24
653		9014	202	MASTER	28/01/2002	12:24
654		9014	201	MASTER	28/01/2002	12:24
655		9015	94	MASTER	28/01/2002	12:25
656		9015	46	MASTER	28/01/2002	12:25
657		9015	102	MASTER	28/01/2002	12:25
658		9016	204	MASTER	28/01/2002	12:27
659		9016	205	MASTER	28/01/2002	12:27
660		9016	203	MASTER	28/01/2002	12:27
799	300		1	master	02/01/1931	14:46
800	300		2	master	02/01/1931	14:46
801	300		3	master	02/01/1931	14:46
802	300		206	master	02/01/1931	14:46
803	300		207	master	02/01/1931	14:46
804	300		208	master	02/01/1931	14:46
805	400		3	master	02/01/1931	14:46
806	400		4	master	02/01/1931	14:46
807	400		2	master	02/01/1931	14:46
808	400		1	master	02/01/1931	14:46

Registro: 1 de 1024

Figura 2.6. Tabla Enlace_Modificador.

Esta tabla representa la relación entre un platillo y sus ingredientes o de un subgrupo e ingredientes comunes a todos los platillos agrupados en dicho subgrupo. Como se advierte, para realizar esta función, existen los campos *Cve_Platillo, Cve_Platillo* y *Cve_Grupo* entre otros.

La idea de esta tabla es que en el campo *Cve_Mod* (identificador del ingrediente) siempre tendrá un valor, en cambio las columnas *Cve_Platillo* y *Cve_Grupo* son excluyentes, es decir, cuando la columna *Cve_Platillo* tiene valor el campo *Cve_Grupo* no tiene valor y viceversa.

En la figura 2.6 se observa esta combinación, por ejemplo, los registros con el campo *Enlace_Id* con valores del 642 al 647, son la combinación del platillo 9012 y los ingredientes con identificadores 89, 71, 53, 179, 148 y 52.

En cambio los registros con el campo *Id_Enlace* con valores del 799 al 804, son la combinación del subgrupo 300 y los ingredientes con identificadores 1, 2, 3,206, 207 y 208.

De esta forma el programa es capaz de representar la carta de los platillos y bebidas de un restaurante y llevar un control de lo que un comensal va a pagar por su consumo, aunque es obvio que no se aplicaron las reglas de normalización de base de datos en su diseño.

2.4.4 Control de comandas.

En la terminología del mundo de los restaurantes, se conoce como *comanda*, al hecho de que un mesero apunte en una libreta o bloc los platillos y bebidas que los clientes desean se les preparen. Se puede decir que la comanda es el equivalente a que un mesero *tome la orden* a cada cliente, que es la forma más conocida entre la gente común.

Esta parte del análisis se centra en la forma en que el sistema efectúa el control de las comandas.

Análisis interno del sistema.

Para guardar la información relacionada a una comanda, la aplicación hace uso de dos tablas, cuyos nombres y función se describe en el siguiente recuadro.

Nombre de la tabla	Uso
Comanda	Guarda el número de comanda, su status, el identificador de la mesa a la que pertenece y el identificador del mesero que la capturó.
Detalle	Guarda el nombre de cada platillo de la comanda así como los ingredientes con los que fueron solicitados. También guarda el precio de cada platillo, como el grupo al que pertenece.

La figura 2.7 muestra la relación y estructura de estas tablas.

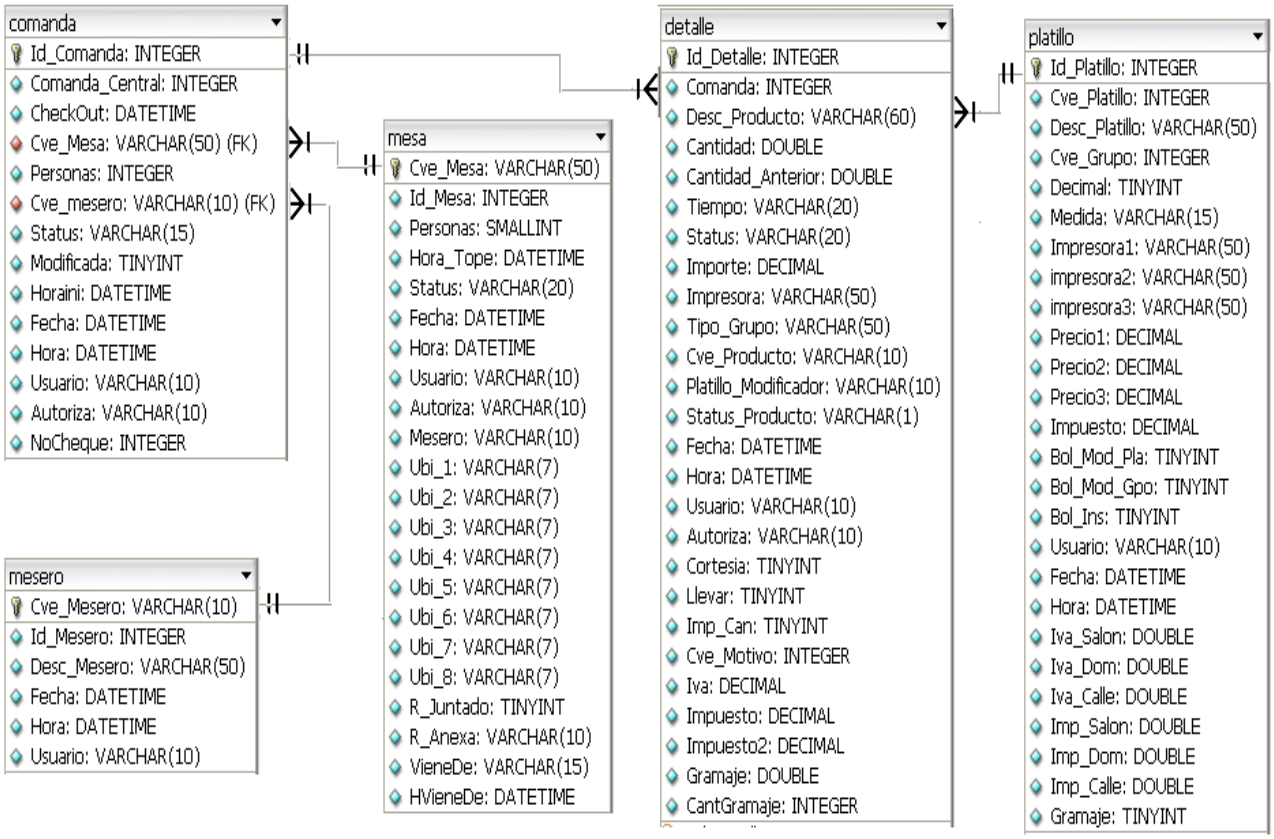


Figura 2.7. Estructura y relación entre las tablas comanda y detalle.

La primera tabla en ser usada es la tabla *Comanda*, entre los campos que forman esta tabla destacan el *Id_Comanda* siendo la clave primaria de esta tabla, *Cve_Mesa* es el identificador de la mesa que esta siendo atendida, *Personas* contiene el número de personas sentadas en la mesa, *Cve_mesero* es el identificador del mesero que atiende la mesa, *Status* es utilizado para identificar la comanda activa de una mesa, esto debido a que una mesa puede tener asociadas muchas comandas durante un día, pero solo una o varias de ellas son las que representan lo que han pedidos los clientes en este momento, *NoCheque* indica en que cheque se encuentra el precio pagado por los platillos de las comandas, la suma total de los precios así como el IVA y otros impuestos que se cobraron por el servicio y *Horaini* que registra la hora en que fue capturada la comanda.

El resto de los campos no son importantes para el control de las comandas, de esta manera tenemos campos como *Comanda_Centra* y *CheckOut* cuyo objetivo era indicar si las comandas habían sido capturadas en el local o por llamada telefónica para un servicio a domicilio (*Comanda_Central*) e indicar también a que hora fueron entregados los alimentos. Estos campos no se utilizan ya que esta funcionalidad no fue programada.

El campo *Modificada*, fue agregado con el objetivo de indicar si una comanda era modificada, esto es, si se le agregaba un nuevo platillo o si se cancelaba un platillo de la comanda, el campo no se utiliza ya que este control no es requerido por los propietarios de restaurantes.

El campo *Autoriza* no es utilizado por el sistema.

Para determinar los platillos que fueron solicitados en las comandas, se hace una relación entre la tabla *Comanda* y la tabla *Detalle*.

Dentro de la estructura de la tabla *Detalle*, hay muchos campos que no son requeridos, ejemplo de ellos son:

- *CantGramaje* que no se utiliza, dado que la intención de este campo era ofrecer la funcionalidad de poder vender un platillo con diferentes medidas por ejemplo, vender barbacoa por kilo o gramos o pan por pieza o por caja, etc. Esta funcionalidad no fue programada.

- *Gramaje*, este campo guardaría el precio del producto dependiendo de la medida que tuviera el campo *CantGramaje*. No se utiliza por el motivo descrito en el campo anterior

- *Autoriza*, este campo guardara el identificador del usuario que tuviera privilegios para modificar un registro y que autorizara el cambio en los datos del registros, no se utiliza como se puede observar de las figuras 2.16

- *Llevar*, la intención de este campo era solo informativa, para indicar que los platillos habían sido pedidos para llevar y no para consumirse en el restaurante, no es utilizado por el sistema.

- *Imp_cam*, campo añadido solo para guardar algún tipo de información que no se hubiera tenido contemplado, por lo que no es utilizado por el sistema normalmente salvo en muy pocas excepciones.

- *Impuesto2* campo agregado para guardar el precio que se cobra de un impuesto que no se tuviera contemplado, normalmente no se utiliza.

El resto de los campos de esta tabla si son necesarios para guardar información relevante:

- *Id_Detalle* campo que es la clave única de la tabla, este campo a pesar de ser utilizado, no es indispensable para jugar el papel de identificador único, debido a que este rol lo podrían jugar los campos *Comanda* y *Cve_Producto* ya que la combinación de ellos permite la identificación única de un producto en una comanda específica.

- *Comanda*, campo que permite la relación con la tabla *Comanda*.

- *Desc_Producto* contiene el nombre del platillo o bebida solicitado, se hace una copia del campo *Desc_Platillo* de la tabla *Platillo* debido a que este nombre puede ser susceptible de cambio en un futuro y si se da este caso, se puede saber que nombre tenía este platillo al momento de ser vendido.

- *Cantidad* y *Cantidad_Anterior* el campo *Cantidad* es la cantidad de platillos que se van a cobrar al momento de pagar la cuenta, esta cantidad se puede ver modificada al solicitarse la cancelación de uno de ellos, por ejemplo, un comensal al inicio de su orden probablemente solicito tres helados, después se arrepiente y termina pidiendo la cancelación de uno de ellos por lo que solo se le cobrarían dos helados. El campo *Cantidad_Anterior* contiene la cantidad de platillos pedidos originalmente.

- *Tiempo*, indica en que tiempo se debe servir el platillo, es decir, si primero se sirve la entrada (primer tiempo), después la sopa (segundo tiempo) y al último el guisado (tercer tiempo).

- *Status*, indica si un platillo esta cancelado y no se debe tomar en cuenta su información a la hora de calcular el total a pagar en un ticket.

- *Importe*, precio del platillo o bebida.

- *Impresora*, indica a que impresora se envía la solicitud de preparación del platillo o bebida. Estas impresoras están ubicadas en lugares estratégicos como la cocina, el bar, etc.

- *Tipo_Grupo*, indica en que grupo se encuentra clasificado el platillo. A pesar de usarse este campo no es necesario que aparezca en esta tabla, debido a que este dato se puede obtener al ubicar el registro equivalente en la tabla *Platillos* mediante la clave del platillo.

- *Cve_Producto*, el identificador único del platillo asignado dentro de la tabla *Platillos*.

- *Platillo_Modificador*, identificador único del ingrediente asignado dentro de la tabla *Modificador*.

- *Status_Producto*, su valor sirve para diferenciar entre un platillo y un modificador. Se utilizan las letras P para indicar que la información es de un platillo o M, si contiene datos de un modificador. Este campo no es necesario, debido a que la combinación de los campos *Cve_Producto* y *Platillo_Modificador* son suficientes para conocer dicha diferencia. Debido al hecho de que el registro contiene un valor en el campo *Cve_Producto* pero no en el campo *Platillo_Modificador* y viceversa cuando se trata de un registro de un ingrediente la columna *Platillo_Modificador* tiene valor y *Cve_Producto* no tiene.

- *Cortesía*, este campo que indica si el platillo o bebida es una cortesía del restaurante para con un cliente, por lo que no se consideraría su precio a la hora de calcular el total a pagar por parte del cliente.

- *Cve_Motivo*, contiene el identificador de algún registro de la tabla *Motivos*, estos registros son los motivos por los cuales se podría cancelar la preparación de un platillo o bebida.

- *Iva*, el total de i.v.a. a pagar por el platillo.

- *Impuesto*, el total de algún otro impuesto que tenga el platillo y que deba pagar el cliente.

La estructura de la tabla *Platillo* ya fue analizada previamente.

2.4.5 Control de pagos.

El pago del servicio consiste en que los comensales una vez que han consumido sus platillos y deciden retirarse, solicitan al mesero la cuenta con el total que van a pagar. En ese momento el mesero imprime un ticket (cheque para la aplicación) y se lo entrega a los clientes para que procedan a liquidar su deuda.

Esta parte del análisis se centra en la forma en que el sistema efectúa el control de dichos pagos.

Análisis interno del sistema.

Para guardar y recuperar la información relacionada al pago de una comanda, la aplicación hace uso de cinco tablas, cuyos nombres y función se describe en el siguiente recuadro.

Nombre de la tabla	Uso
Comanda	Además del uso descrito anteriormente, contiene información sobre el número de cheque que relaciona a varias comandas con un solo pago.
Cheque	Asigna un número único al total pagado por los platillos de una comanda o varias comandas.
F_Pago	Contiene las diversas formas de pago en que un cliente puede saldar su deuda.
Cobro	Contiene las diversas formas de pago utilizadas para pagar un cheque.
Factura	Contiene el monto facturado, el identificador del cliente al que se expidió la factura así como el número de cheque que fue facturado.

La figura 2.8 muestra la estructura y relación entre las tablas.

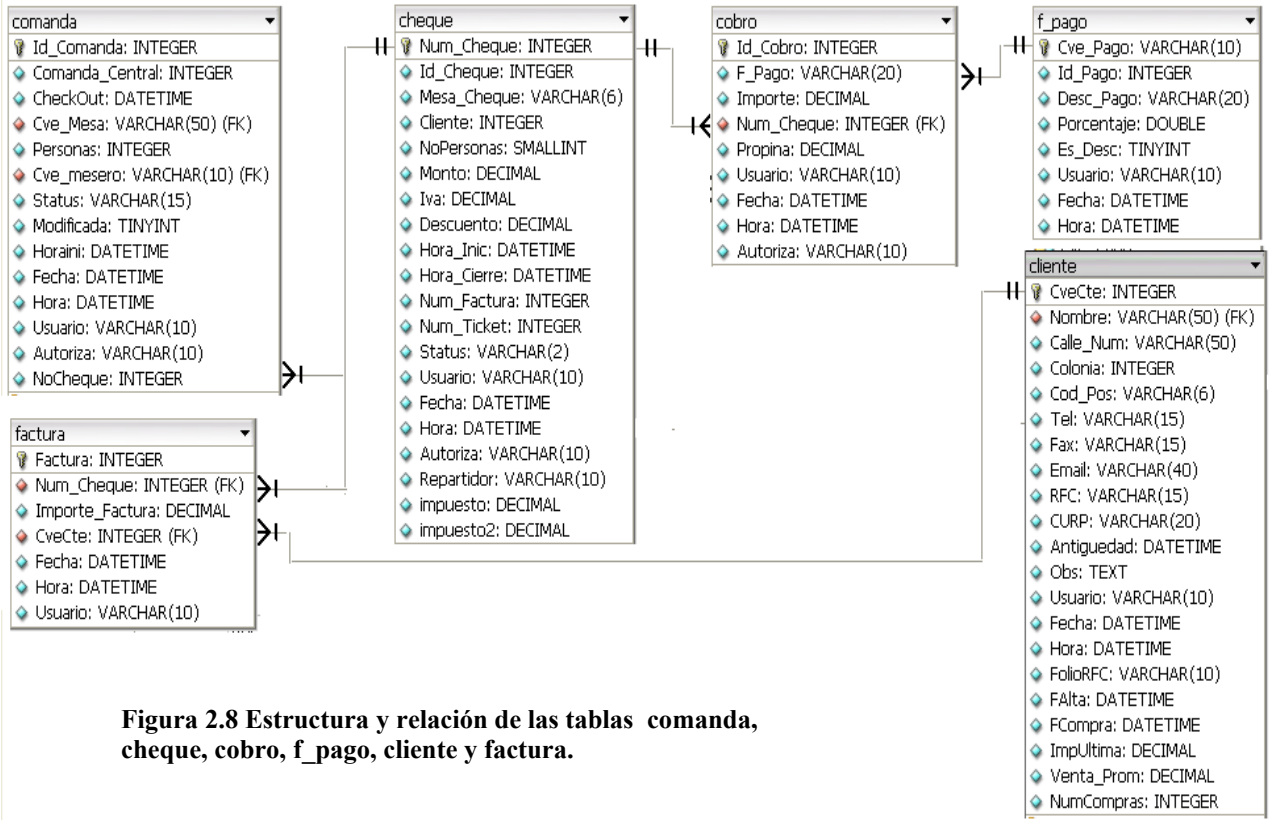


Figura 2.8 Estructura y relación de las tablas comanda, cheque, cobro, f_pago, cliente y factura.

La tabla *Comanda* ya fue analizada previamente en la sección control de comandas, por lo que no se volverá a analizar nuevamente.

La primera tabla en ser usada es *F_Pago*, la cual contiene información sobre las diversas formas con las que un cliente puede pagar por el servicio. Es importante que esté lista antes de efectuar pagos ya que si esta vacía no habría manera de elegir como se esta liquidando una cuenta.

Entre los campos que forman esta tabla destacan los siguientes:

- *Cve_Pago* sus valores son el identificador de cada forma de pago que ingrese a esta tabla.

- *Desc_Pago* contiene el nombre o la descripción de la forma de pago de que se trata.

- *Porcentaje* es el porcentaje que el establecimiento puede cobrar por el uso de alguna forma de pago. Existen algunas tarjetas de crédito que por su uso, se cobra un porcentaje adicional al cliente.

- *Es_Desc* indica si un registro es en realidad una forma de pago o un descuento a aplicarse a la cuenta en un cliente.

- *Usuario* es la clave del usuario del sistema que capturó la información del registro en cuestión.

Cabe resaltar que en general la tabla se encuentra bien diseñada a excepción de las columnas *Id_Pago* y *Cve_Pago* cuyos valores son únicos y una de estas columnas (*Cve_Pago*) podría ser la clave principal de esta tabla, por lo que *Id_Pago* no debería existir.

La siguiente tabla utilizada durante el proceso es *Cheque*, esta contiene el total a pagar por los platillos de las comandas. Esta información es ingresada por el sistema al momento de imprimir un ticket para que el cliente sepa cuanto es el monto que debe pagar.

Los campos más destacados en esta tabla son:

- *Num_Cheque* cuyos valores son únicos y forman la clave única de la tabla.
- *No_Personas* es la cantidad de personas que ocuparon la mesa.
- *Monto* es el total de la cuenta a pagar.
- *IVA* el monto del impuesto a pagar del monto.
- *Descuento* en caso de haber descuentos esta columna tendrá la cantidad que se descontó.
- *Impuesto* e *Impuesto2* almacenarán la cantidad que se agrega al precio del producto. Estos son impuestos fuera de lo común o impuestos que sean autorizados y permiten al sistema tener la flexibilidad de registrarlos.
- *Status* indica si el cheque esta cancelado debido a que fue creado con fines informativos, si hubo algún error en su captura o si el cliente pagó el monto señalado en él.

El resto de los campos no son necesarios para seguir ofreciendo la funcionalidad de esta tabla; así por ejemplo; el campo *Id_Cheque* es un campo numérico cuyos valores son únicos y el cual no es necesario debido a que ya se tiene una columna con las mismas características de el campo *Num_Cheque* y que juega el papel de clave primaria.

El campo *Mesa_Cheque* buscaba ofrecer una relación entre la tabla *Mesa* y *Cheque*. Este campo no es necesario debido a que esa relación se puede obtener mediante la liga entre las tablas *Cheque – Comanda – Mesa*.

El campo *Cliente* no se utiliza debido a que no es importante saber a que cliente pertenece un cheque (ticket).

Los campos *Hora_Inic* y *Hora_Cierre* indicaban la hora en que se imprimía el ticket y a que hora era pagado, esta información es irrelevante debido a que como se sabe un ticket solo es un papel informativo para que el cliente este enterado a cuento asciende su deuda con el restaurante por el servicio y no sirve como comprobante fiscal.

Num_Factura buscaba ofrecer la capacidad de relación entre un cheque y una factura. Este campo esta mal planeado en esta tabla debido a que un cheque puede tener varias facturas sobre su monto, es decir, la relación *Cheque – Facturas* se da mejor al relacionar a una o varias facturas con un ticket. Por este hecho es que no es utilizado por el sistema.

Num_Ticket buscaba tener la misma función que la columna *Num_Cheque*. Es irrelevante porque como se mencionó anteriormente al describir la columna *Id_Cheque* toda la funcionalidad de clave primaria ya la ofrece *Num_Cheque*.

Los campos *Autoriza* y *Repartidor* no se utilizan en el sistema., mientras que los campos *Impuesto* e *Impuesto2* trataban de dar flexibilidad al programa en el manejo de impuestos que no están contemplados o que pudieran aparecer en un futuro. La desventaja es que se esta restringido a solo dos impuestos. Una mejor forma de agregar esta flexibilidad sería crear una tabla especial con su respectiva relación para permitir el manejo no solo de dos impuestos extras si no el manejo de un número infinito de impuestos.

Una vez pagado un cheque, el sistema actualiza el contenido de la tabla *Cobro*. Esta tabla es necesaria debido a que un cliente puede saldar su deuda haciendo uso de diferentes medios que le permitan disponer de dinero. Así, un cliente podría liquidar la cuenta haciendo uso de dinero en efectivo, un cheque y su tarjeta de crédito y pagando con cada uno de estos medios una cantidad diferente; el cual es precisamente el objetivo de esta tabla, registrar esas formas de pago y su monto.

En general esta tabla esta bien diseñada a excepción de los campos *Usuario*, *Fecha*, *Hora* y *Autoriza*, ya analizados previamente.

El resto de los campos si son necesarios en esta tabla, así *Id_Cobro* y *F_Pago* ayudan a identificar el medio o la forma de pago que utilizó el cliente. *F_Pago* aparece en esta columna debido a que una forma de pago puede cambiar su nombre con el tiempo. De esta manera aunque se llegase a dar el caso antes mencionado se tendría la capacidad de saber cual fue el medio original que utilizaba la misma clave.

El campo importe contiene la cantidad del total a pagar que se utilizo con este medio.

Num_Cheque indica el cheque que contiene el total a liquidar con las diferentes formas de pago y propina es la cantidad de propina que se dejó con una forma de pago específica.

Hasta este momento se considera que la cuenta de un cliente ha sido pagada y la impresión de una factura muchas de las ocasiones es opcional.

Para imprimir una factura se utiliza la tabla con el mismo nombre.

En general la tabla también esta bien formada con campos que son necesarios tanto para las relaciones con otras tablas como para la información de la propia tabla; ejemplo de esto son las columnas *Num_Cheque* y *CveCte*. Estas columnas permiten saber que cheque fue facturado y a que cliente se le entregó la factura.

La columna *Factura* es la clave primaria de la tabla e *Importe_Factura* es el importe por el que se expidió la factura.

2.4.6 Impresión de documentos.

El control de impresión de documentos se lleva a cabo mediante tablas. Estas contienen el número de línea y de columna donde se debe iniciar la impresión de algún dato dentro del área de un papel.

Para guardar y recuperar la información relacionada al pago de una comanda, la aplicación hace uso de tres tablas, cuyos nombres y función se describe en el siguiente recuadro.

Nombre de la tabla	Uso
Fac_ult	Valores de las ubicaciones donde se imprimirá un campo de la factura.
Cheq_omi	Valores de las ubicaciones donde se imprimirá un campo del cheque.
Cheq_ult	Valores de las ubicaciones donde se imprimirá un campo del cheque.
Fact_omi	Valores de las ubicaciones donde se imprimirá un campo de la factura.

La figura 2.9 muestra la estructura y relación entres las tablas.

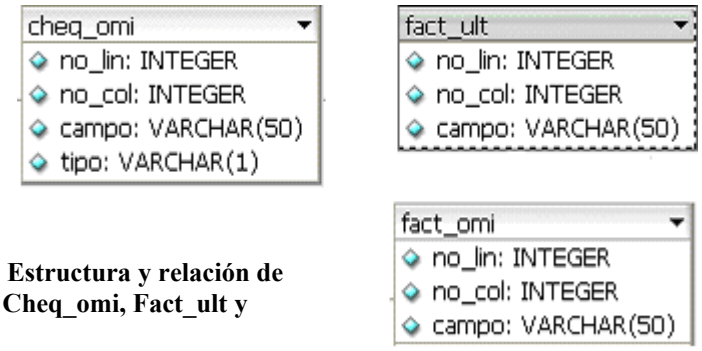


Figura 2.9 Estructura y relación de las tablas Cheq_omi, Fact_ult y Fact_omi.

Diversos documentos como la impresión de un cheque o la factura requieren este tipo de control. La tabla *Cheq_omi* contiene el formato y ubicaciones para la impresión de un cheque mientras que la tabla *fac_ult* guarda la misma información para la impresión de una factura.

Como se advierte, estas tablas tienen la misma estructura a excepción del campo *tipo* en la tabla *Cheq_omi*, por lo que el contenido de estas tablas bien pudiera alojarse en una sola bien diseñada haciendo pequeñas modificaciones tomando en cuenta las reglas de la normalización para poder identificar que campos pertenecen a uno u otro documento.

Para ejemplificar el contenido de estas tablas, la figura 2.10 muestra el contenido de *Fact_omi*.

Figura 2.10 Información contenida en la tabla *Fact_omi*.

fact_omi : Tabla

	no_lin	no_col	campo
▶	1	1	dd-mm-aa
	2	1	dd
	3	1	mm
	4	1	mes
	5	1	año(aaaa)
	6	1	año(aa)
	7	1	año(a)
	8	1	nombre
	9	1	direccion
	10	1	direc1
	11	1	direc2
	12	1	rfc
	13	1	folio
	14	1	consumo en letra
	15	1	consumo
	16	1	total sin iva
	17	1	iva
	18	1	total
	19	1	propina
	20	1	total+propina
	21	1	total en letra
*	0	0	

Registro: 1 de 21

Como se aprecia, las columnas *no_lin* y *no_col* contienen el renglón y columna en donde se debe imprimir cierta información. Tomando como ejemplo el primer renglón de la tabla se observa que la fecha con el formato día mes año, se imprime en el renglón uno columna uno y así sucesivamente con el resto de los campos.

Estas tablas no tienen relación con alguna otra y se limitan solamente a guardar la información que se mencionó previamente.

2.5 Análisis de pantallas de la aplicación.

En este apartado se analizan algunas pantallas de la aplicación que son representativas del aspecto de esta y algunas otras que contienen deficiencias en su diseño.

2.5.1 Pantallas representativas del aspecto de la aplicación.

A continuación se presentan algunas pantallas de la aplicación que servirán para tener una idea de la presentación que tiene el sistema Infocaja. Las figuras 2.11 y 2.12 muestran dos pantallas que son comunes en toda la aplicación.

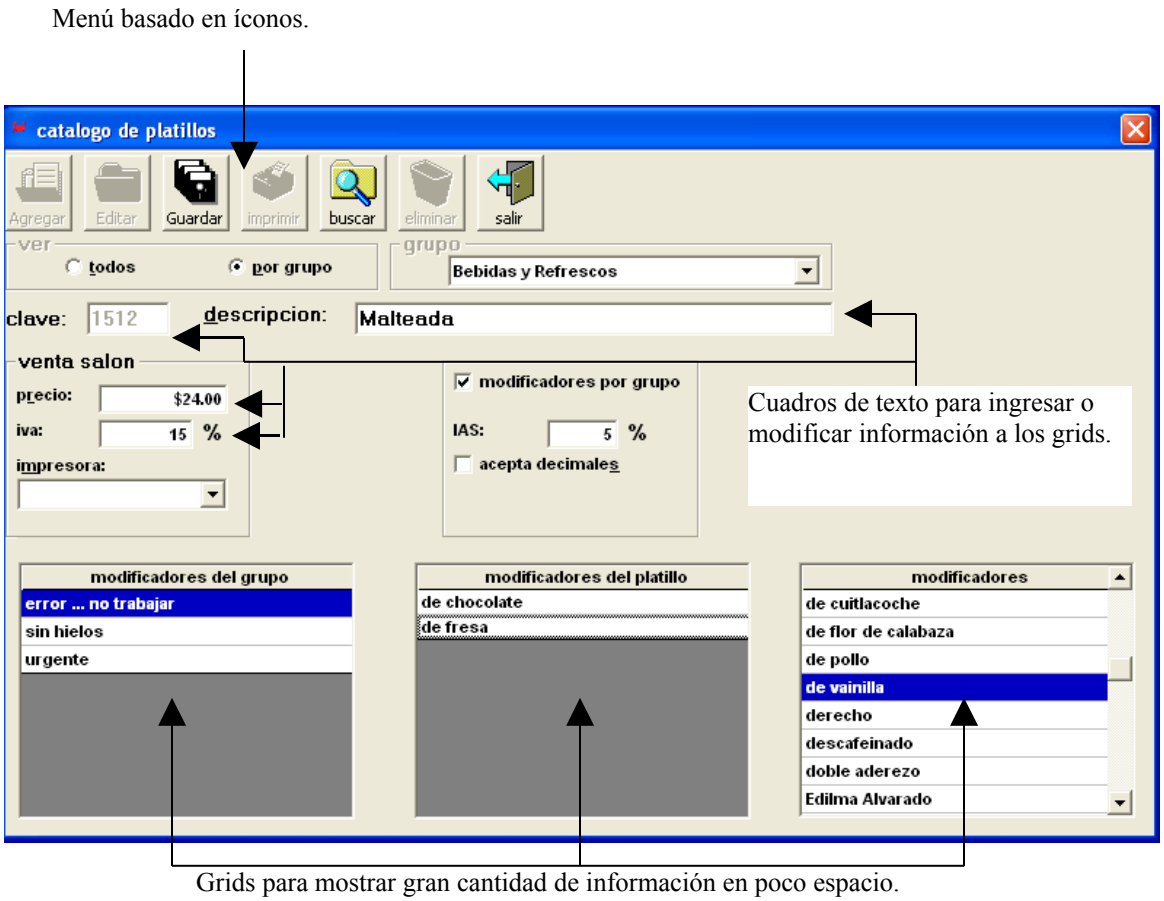


Figura 2.11 Primer pantalla representativa de la aplicación.

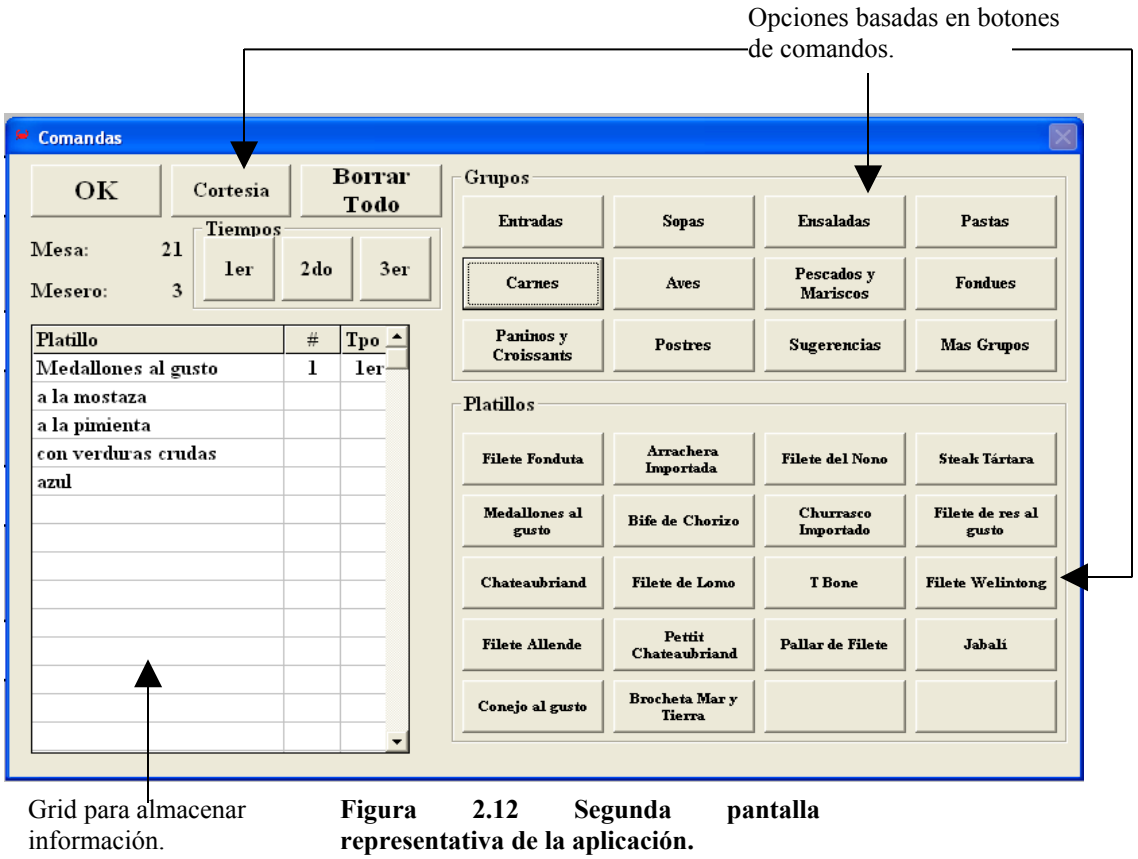


Figura 2.12 Segunda pantalla representativa de la aplicación.

Como se puede advertir de las figuras 2.11 y 2.12 las pantallas de la aplicación por lo general estarán formadas por un menú basado en íconos, cuadros de texto que permiten modificar o ingresar información a la aplicación y grids para mostrar gran cantidad de información en poco espacio.

Otro tipo de ventana (figura 2.12) utiliza botones para presentar la información y grids.

En cuanto al diseño efectivo de pantallas de captura se puede observar que algunas de estas ventanas no siguen el movimiento lógico de la vista que es de izquierda a derecha y de abajo hacia arriba. [6]

[6] Libro: Análisis y Diseño de Sistemas, autores Kendall y Kendall, capítulos 14 y 15 "Diseño de una entrada eficaz".

Así, tenemos que el flujo de la vista del usuario en la ventana 2.21 es de izquierda a derecha, arriba hacia abajo y de derecha a izquierda como se muestra en el siguiente diagrama de flujo:

1. Seleccionar un botón del *grupo* tiempos
2. Buscar dentro de la opción *grupos*, el nombre del *grupo de alimentos* dentro del cual está clasificado el *platillo* que se esta buscando.
3. Buscar el nombre del *platillo* solicitado dentro de la *opción platillos.*
4. Asegurarse de que tanto el nombre del platillo y la cantidad solicitada son correctos en el grid con título *platillo*.

Este flujo se enumera dentro de la figura 2.13

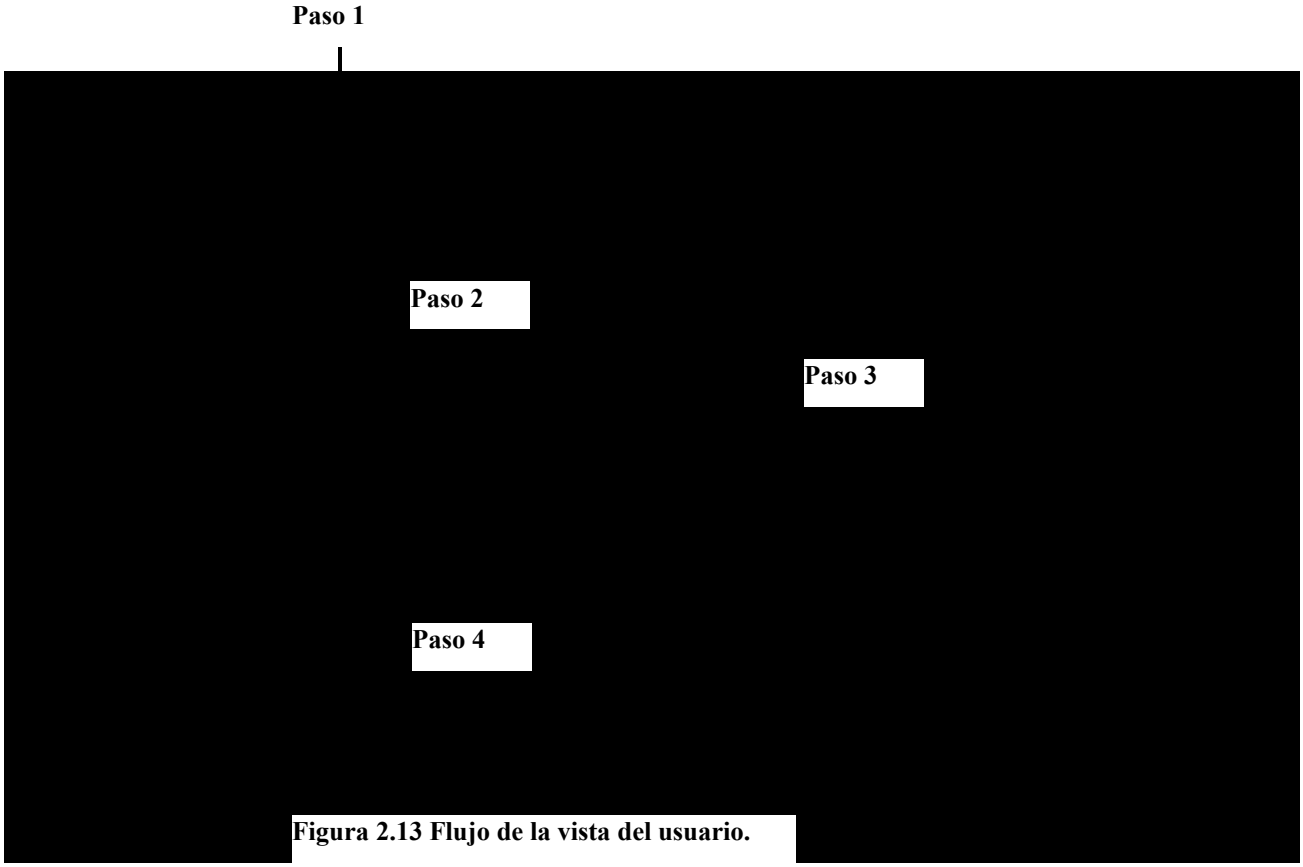


Figura 2.13 Flujo de la vista del usuario.

Capítulo III

Cambios propuestos.

Cambios propuestos.

3.1 Elección de programas sustitutos.

3.2 Cambios a la base de datos.

3.3 Cambios a pantallas del sistema.

3.4 Cambios al código de la aplicación.

3.5 Cambios a reportes.

3.6 Nuevos requerimientos para su uso.

3.1 Elección de programas sustitutos.

El software libre se ha convertido en una gran opción y la mayoría de sus programas más populares como el sistema operativo Linux, la base de datos Mysql y/o el lenguaje de programación Java, etc. han llegado a una madurez que permiten utilizarlos con la confianza y seguridad de que no van a fallar tan fácilmente, aunado a la creciente información que podemos encontrar acerca de estos programas en Internet, publicaciones y libros, lo que nos permite conocerlos más a fondo y resolver cualquier problema que tengamos con ellos.

El uso de estos programas representa un gran ahorro para todo tipo de empresas ya que disminuyen los gastos que se generan al pagar por licencias (permisos) para usar programas con derechos de autor de empresas como Microsoft, Oracle, Sun, etc.

Para el desarrollo de este prototipo, se eligieron los siguientes programas para sustituir a los programas con los que fué creada la aplicación originalmente.

Base de datos: Preferentemente usaremos Mysql versión 5.0, distribución para el sistema operativo Linux (se puede utilizar la versión para el sistema operativo Windows).

La elección de esta base de datos sobre otras como Postgress, Mssql server (nombre abreviado de la base de datos Microsoft sql server), Informix, Sybase, etc. se debe a que Mysql es una de las bases de datos más rápidas que se pueden encontrar, esto debido a que no es una gran base de datos como Oracle o Postgress pero tampoco es tan chica como un Acces o Foxpro; el hecho de ser una base de datos de mediano tamaño le permite ofrecer características de las grandes bases y la rapidez de una base de tamaño pequeño.

Otras ventajas son la seguridad en el control de los datos y su fácil administración.

Lenguaje de programación: Visual Basic versión 6

El lenguaje de programación para la aplicación de la R.I.S. al sistema será Visual Basic versión 6, debido a su gran facilidad y rapidez para desarrollar la interfaz y la programación de la misma; aunado a que las aplicaciones desarrolladas en este lenguaje son muy amigables para los usuarios debido a su gran parecido al sistema operativo Windows.

Sistema operativo: se puede utilizar alguno de estos sistemas operativos: Linux en cualquiera de sus distribuciones (preferentemente Red Hat o Fedora) o el sistema operativo Windows en cualquiera de sus versiones.

Se ha elegido el sistema operativo Linux para instalar y ser soporte del servidor que contendrá la base de datos, debido a que Linux es el sistema operativo para el cual fué diseñado originalmente Mysql y en donde podemos obtener su mejor rendimiento. Existe la alternativa de usar una versión de esta base de datos para el sistema operativo Windows, por

lo que estaremos en libertad de elegir entre estos dos sistemas operativos para la computadora que funcionará como servidor.

Conector de la base de datos: Myodbc para Windows (en caso de elegir este sistema operativo.).

Este programa permitirá la conexión del prototipo creado con la computadora servidor y con la base de datos que almacena la información, independientemente del sistema operativo que se elija para esta.

Otras herramientas: Crystal Reports versión X.

Para la impresión de documentos y reportes se utilizará la herramienta llamada Crystal Reports que es muy popular en el mercado. Esta herramienta fué desarrollada específicamente para realizar este tipo de tareas mejorando la presencia de los reportes así como permitir la exportación de datos a otras aplicaciones como Excel o Access, entre otros.

En el siguiente cuadro se presenta los programas en los que se encuentra hecho el punto de venta y el sustituto a utilizar en la creación del nuevo punto de venta.

	Software protegido		Nuevo software
Sistema operativo	Windows cualquier versión	A	Linux cualquier distribución o cualquier versión de Windows para la base de datos
Lenguaje de programación	Visual Basic 6		Java
Base de datos	Access 2000	A	Mysql 5
Otras programas	_____	A	Conector Myodbc Crystal Reports

3.2 Cambios a la base de datos.

Control de clientes, usuarios y sus derechos.

Una vez analizada la base de datos original, ahora se proponen algunos cambios para eliminar defectos y agregarle funcionalidad.

Para el control de clientes y usuarios del sistema, se modificaron las tablas *Cliente, Mesero* y *Usuario*, debido a que tienen una estructura muy parecida y se unificaron en una sola tabla llamada *Usuarios*, de esta manera evitamos el uso de las tres tablas mencionadas anteriormente quedando toda la información que tienen en común en una sola.

Ahora, cada usuario será identificado con un puesto, los cuales se encontrarán dentro de la tabla *Puestos*. Se consideró un puesto especial llamado cliente que ayuda a identificar aquellos registros que están clasificados como clientes del negocio.

En resumen, la tabla puestos ahora solo tendrá una clave primaria llamada *Idpuesto* para identificar a un puesto de forma única y otro campo llamado *Nombre* el cual guardará el nombre o descripción del puesto. El resto de los campos originales no son importantes para que esta tabla cumpla con su función por lo que son eliminados de la nueva estructura; de esta manera se siguen cubriendo las necesidades de control de usuarios y se elimina la repetición de datos entre las tablas *Usuario* y *Mesero* que se hacia anteriormente. Por último las columnas de *Hora* y *Usuario* que aparecían en todas las tablas de la base original son eliminadas de todas las tablas en la nueva aplicación

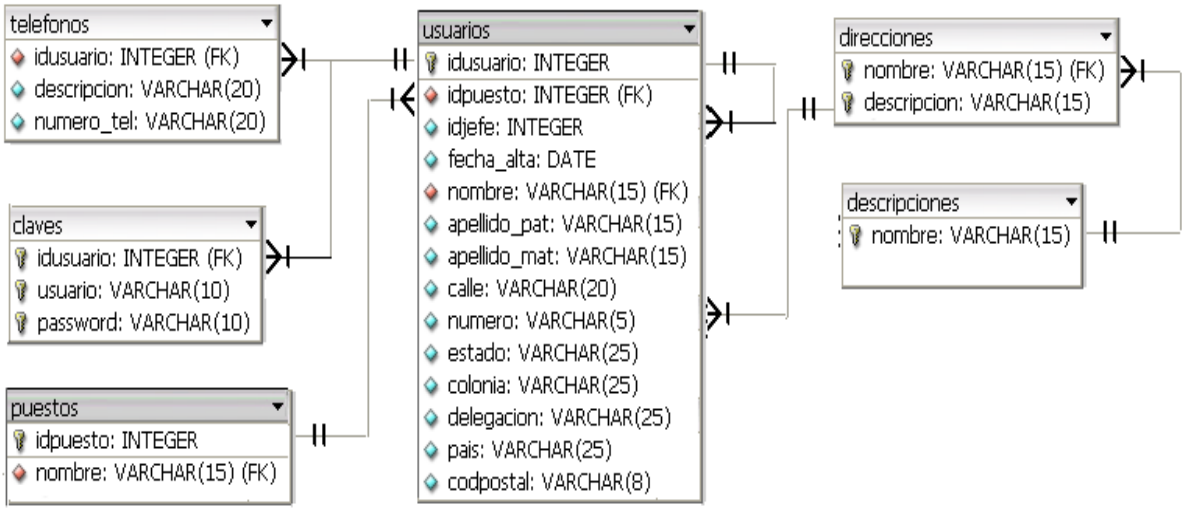


Figura 3.1. Relación de las nuevas tablas para el control de usuarios y clientes.

Como se advierte de la figura 3.1, se crea la tabla *Telefonos* con el fin de tener la capacidad de registrar más de un medio de contactar a un cliente o trabajador y no estar limitados a tres como se tenía en la tabla *Cliente* original, de esta manera se esta en condiciones de almacenar teléfonos, emails, faxes o cualquier tipo de medio de localización que utilice una persona.

La tabla *Claves* se agrega para llevar el registro y control de las claves de los usuarios que se ingresan a la aplicación. Esto es necesario porque al tener a clientes y usuarios dentro de una misma tabla, no todos los registros usarán una clave y contraseña de acceso, por lo cual es mejor tenerlas en esta tabla por separado.

Las tablas *Descripciones* y *Direcciones* se incorporan para mejorar el control de las direcciones en un domicilio. La tabla *Descripciones* contiene las formas en que se van a clasificar los datos, es decir, se puede decidir si un dato es una calle, colonia, etc. Así por ejemplo, la tabla *Direcciones* contendrá una descripción y algún nombre para dicha descripción, por ejemplo el par (calle, av. Revolución), indica que existe una calle llamada av. Revolución, en otro ejemplo el par (estado, Jalisco), indica que existe un estado llamado Jalisco. Con esta nueva forma de agrupar la información de una dirección evitamos que se tenga que escribir una dirección por parte de un usuario y que se comentan errores al momento de la captura ya que solamente se tendrá que seleccionar la calle, la delegación, etc. que ya estarán previamente capturados.

Con respecto a los derechos de un usuario de la aplicación, ahora se controlarán por medio de los permisos que la misma base proporciona para la creación de tablas, inserción, actualización y borrado de registros en las mismas, conexiones, etc.

Captura y armado de platillos.

La información de las tablas *Tgrupo* y *Grupo* originales se guardará en una llamada *Grupos* con menos campos de los que tenían las tablas originales (de 12 campos en promedio a solo 3) y permitiendo por medio de la columna *Pertenece* tener más de un grupo principal de clasificación para subgrupos.

También se crea una nueva tabla llamada *Impuestos* la cual tendrá los impuestos que se aplican a un grupo de platillos específico eliminando con esto los campos *Imp_salon*, *Imp_calle*, *Imp_dom*, *Iva_salon*, *Iva_dom* e *Iva_calle* que se tenían en las tablas *Tgrupo* y *Grupo*, aumentando con esto la capacidad de la base para agregar más de tres impuestos a cada grupo de platillos en caso de ser requerido.

Con las tablas *Platillo* y *Modificadores* se disminuye la cantidad de campos (de 13 campos en promedio a solo 4) debido a que campos como *Precio2, Precio3, Impresora2, Impresora3*, *Bol_mod_pla, Bol_mob_grupo, Bol_ins, Ivas_xx* e *Imp_xx* no son importantes para el control de los registros dentro de estas tablas.

La tabla *Modificadores* cambia su nombre por *Ingredientes* que es más descriptivo en cuanto a su función dentro de la aplicación y la tabla *Platillo* cambia su nombre por *Platillos*.

En cuanto a la relación entre las tablas Grupos y Platillos, esta se llevará a cabo de manera directa añadiendo un campo a la tabla *Platillos* llamado *Idgrupo* el cual indicará a que grupo pertenece un platillo, de esta manera se evita el uso de la tabla *Enlace_modificador* que ocasionaba confusión en la relación original.

La tabla que sufre una mayor modificación en su estructura es *Enlace_modificador*, debido a que se eliminan campos como *Id_enlace, Cve_grupo, Usuario, Fecha* y *Hora*. El campo *Id_enlace* se descarta porque al ser una tabla que ahora solo relaciona dos tablas no es necesaria una clave primaria en ella. El campo *Cve_grupo* se excluye debido a que esta tabla ya no tendrá relaciones entre *Grupos – Platillos* y *Platillos – Modificadores* ahora solo tendrá relaciones entre *Platillos* y *Modificadores*.

Pero el cambio mas importante que sufre esta tabla es el cambio de nombre por uno más descriptivo en relación con el papel que juega dentro de la nueva relación, su nuevo nombre será *Plaing* tratando de mostrar que esta tabla es una relación entre platillos e ingredientes.

Como en la mayoría de las tablas, esta nueva estructura también sufre una reducción en el número de campos que la forman, pasa de 7 campos a solo 2.

La relación entre las tablas se muestra en el diagrama entidad relación de la figura 3.2.

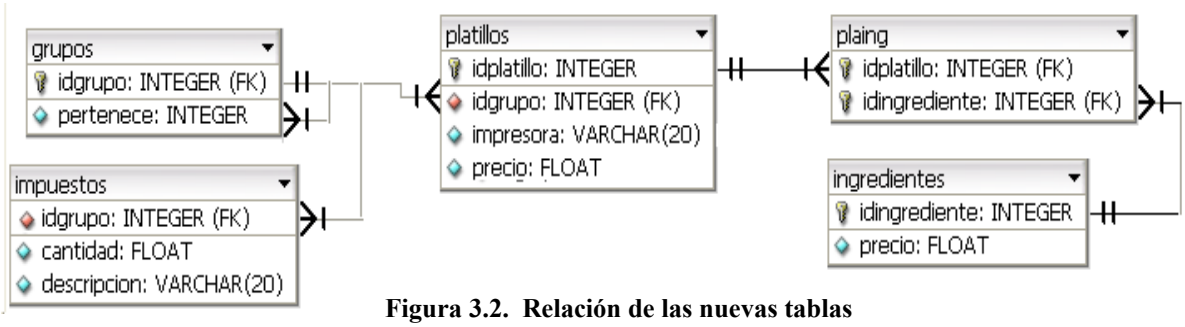


Figura 3.2. Relación de las nuevas tablas para el armado de platillos.

Manejo y control de comandas.

En esta parte de la aplicación se hacen ligeras modificaciones a las tablas originalmente involucradas rescatando la mayor parte de ellas.

En la tabla *Mesa* se elimina el campo *Cve_mesa* que es uno de los dos campos que tienen valor único; los campos *Ubi_1* al *Ubi_8* se excluyen debido a que estos campos eran usados para tratar de ubicar la imagen de una mesa en varias posiciones, esta característica no es usada y con solo dos campos *Ren* y *Col* se puede ofrecer esa funcionalidad. Los campos *R_juntado, R_anexa, VieneDe* y *HvieneDe* se eliminan debido a que son campos que se utilizan para tratar de juntar una mesa con otra, saber de donde viene la mesa que se junta y a que hora sucedió el juntado de mesas. Esta característica no esta lista y no se va a programar dentro del nuevo sistema por no ser una característica que los clientes soliciten.

En la tabla *Comanda* se hace una reestructuración importante, en la estructura original se tiene un campo como clave primaria llamado *Id_comanda* que es un consecutivo; ahora este campo se llamará *Numcomanda* el cuál será el número de comanda que corresponda a la captura y unido al campo *Fecha* formarán la nueva clave primaria para esta tabla.

También desaparecen campos como *Comanda_central* que era otro campo consecutivo igual a *Id_comanda* y *CheckOut* por ser un campo que no tiene función dentro de la tabla.

El campo *Personas* original contenía el total de personas que ocuparon una mesa, ahora este campo cambiara su nombre por *Persona* y ahora su función será la de indicar a que persona pertenece la comanda a la que esta relacionado, esto para poder agregar la funcionalidad de cuentas separadas y calcular lo que tiene que pagar la persona independientemente de otras que le acompañen.

Los campos *Modificada* y *Horaini*, se eliminan de la estructura de la tabla debido a que contenían valores tipo hora para indicar en que momento eran modificadas las comandas, a que hora se capturaba la comanda y quien modificaba y autorizaba dichas modificaciones.

En la nueva versión estas características no son necesarias. Por último esta tabla ahora se llamará *Comandas*.

La tabla *Detalle* era utilizada por el sistema para guardar una descripción de todos los platillos en una comanda junto con los ingredientes con los que fué pedido un platillo; para la nueva aplicación esta tabla cambia su función, ahora solo tendrá una descripción de los ingredientes con que fué pedido el platillo de una comanda específica, para lograr esto se eliminan campos como *Idplatillo* por no ser necesario, *Cantidad* y *Cantidad_anterior* ya que estas cantidades de un platillo se encuentran en la tabla *Platillos*. Los campos *Tiempo*, *Status*, *Status_producto* e *Impresora* son excluidos debido a que también aparecen en la tabla *Platillos*. El campo *Platillo_modificador* desaparece ya que no tiene utilidad en la nueva relación de la tabla *Detalle*, ya que solo indicaba si el registro era platillo o ingrediente ahora ya sabemos que esta tabla solo contiene ingredientes.

El resto de los campos como *Cortesía, Llevar, Imp_can, Cve_motivo, Iva, Impuesto1, Impuesto2, Gramaje* y *CantGramaje* se eliminan de la estructura al no ser necesarios en la nueva relación, esto debido a que solo son identificadores que ayudan para saber como se vendía un platillo (si era cortesía no se cobra) o que impuestos se deberían aplicar por el platillo.

Otro cambio importante es el manejo de los status en las tablas hasta el momento analizadas, estos pasan de ser una sola letra a palabras, por ejemplo, el status P en la tabla *Detalle* significaba pagado, o M significaba que el platillo había sido modificado, etc. El manejo de una palabra en lugar de letras facilita la comprensión acerca del status o acción realizada con una comanda y sus detalles.

Cabe mencionar que a esta parte de la aplicación se añade una tabla nueva llamada Salones y a la tabla Mesas se le añade un campo nuevo que no tenía contemplado originalmente que es el campo *Idsalon*. Con este nuevo campo se podrá relacionar a un conjunto de mesas con un salón específico y de esta manera la aplicación llevará el control de las mesas que existen en un salón.

Esto se hace para agregarle funcionalidad a la aplicación debido a que hay restaurantes que manejan varios salones y en cada uno llevan una venta independiente de cada salón, así que con esta tabla la aplicación ya puede llevar el control por separado de las ventas hechas en un salón en particular.

La relación entre las tablas se muestra en el diagrama entidad relación de la figura 3.3.

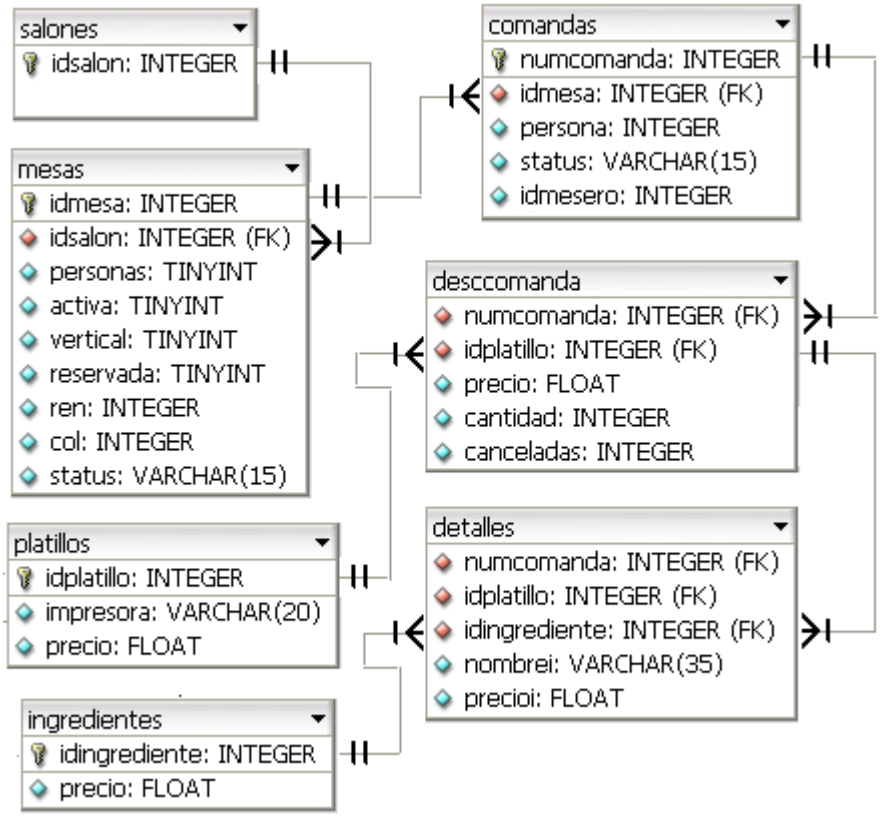


Figura 3.3. Relación de las nuevas tablas para el control de comandas.

Control en el pago del servicio.

Para la parte del pago de servicios también se hacen algunos cambios. En la tabla *Cheque* se elimina uno de los dos campos con valor único que se tenían originalmente y se deja solamente *Numcheque* como clave primaria; también se eliminan campos que solo se usaban esporádicamente como *Repartidor, Autoriza, Impuesto2, Usuario y Hora.*

Con respecto a la tabla *F_pago* (formas de pago) se procedió de la misma forma eliminando el campo *Id_pago, Es_desc, Usuario, Fecha y Hora*.

La tabla *Cobros* también sufrió una disminución en su estructura al eliminar *Id_cobro* por ser uno de los dos campos con valores únicos.

A la tabla *Factura* se le cambió el nombre a *Facturas* y solo se prescinde del campo *Usuario* debido a que solo se usaba esporádicamente, el resto de los campos permanecen sin cambio.

Como se observa el resultado final en la mayoría de las tablas es una notable reducción en su tamaño así como la supresión en la repetición de datos que se daba entre varias de ellas.

La nueva relación entre estas tablas se muestra en la figura 3.4.

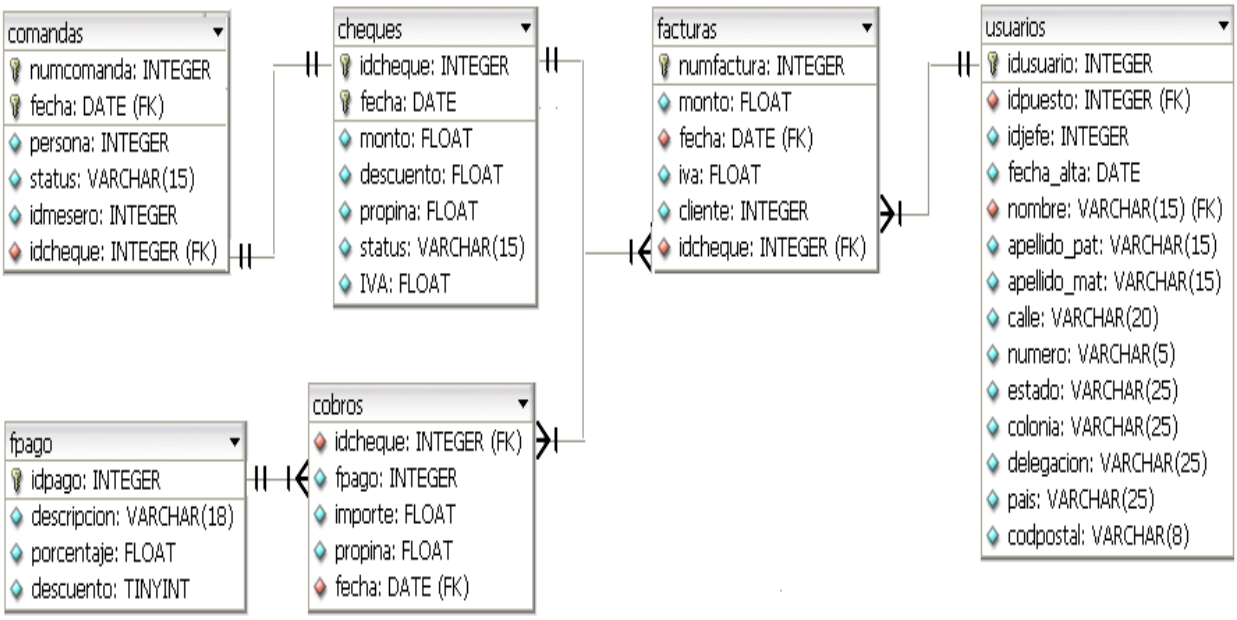


Figura 3.4 Relación de las nuevas tablas para el control de comandas.

Al final de todos estos cambios, se tiene una base de datos cumpliendo con la tres formas normales de la normalización.

3.3 Cambios a las pantallas.

Pantallas.

En general, la mayoría de las pantallas tienen buena presentación, por lo que solo se les hace una pequeña modificación en la manera de presentar la información y en la forma de capturarla.

Para ejemplificar esto se muestra la pantalla de inicio del sistema original en la figura 3.5, el inicio consistía en presentar una pantalla de bienvenida, para continuar con el ingreso a la aplicación se debe seguir las instrucciones que muestra esta pantalla.

Figura 3.5. Pantalla de inicio del sistema.

Después de presionar la tecla enter o un botón del mouse sobre la pantalla, aparece la pantalla de ingreso de la figura 3.6.

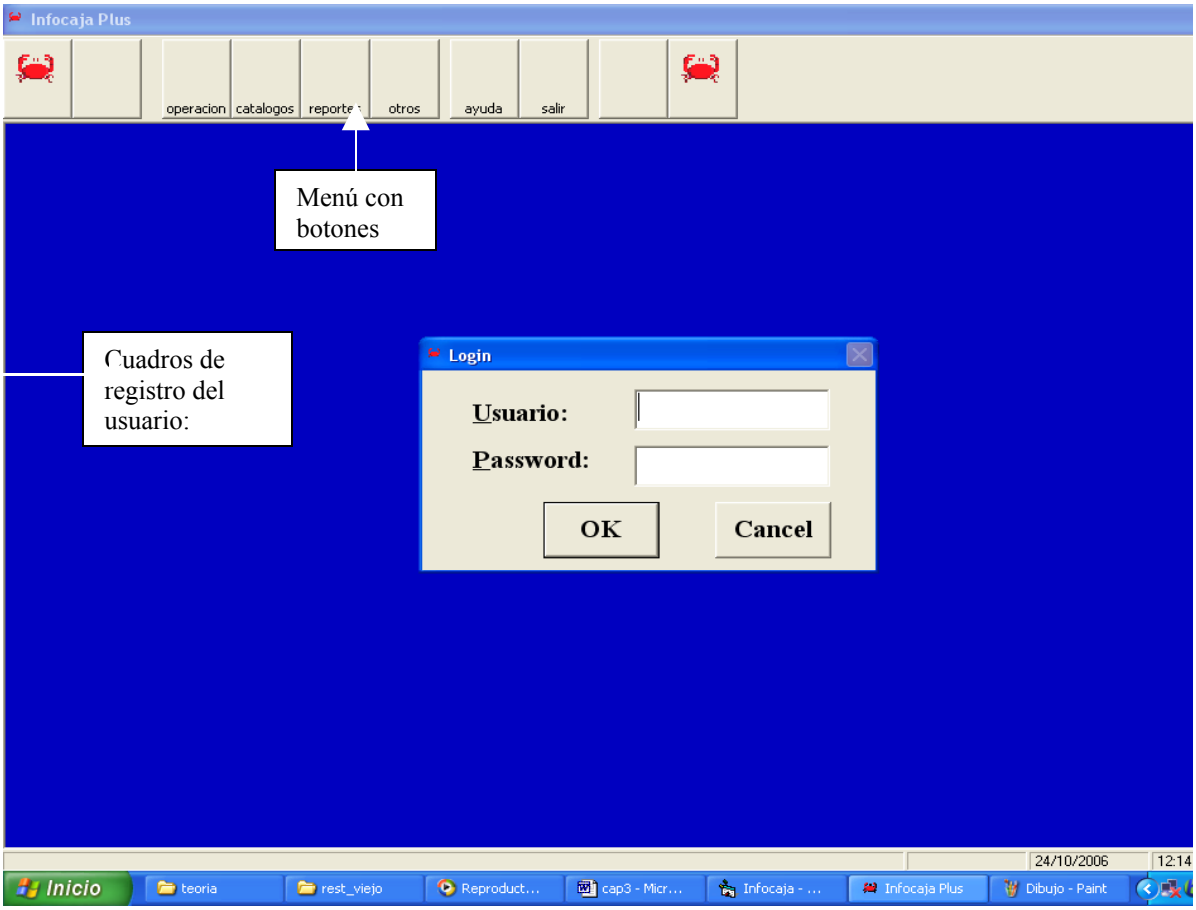


Figura 3.6 Pantalla de ingreso de la aplicación original.

Ahora el ingreso a la aplicación es más simple, mostrando solamente una pantalla al inicio quedando de la manera que se muestra en la figura 3.7

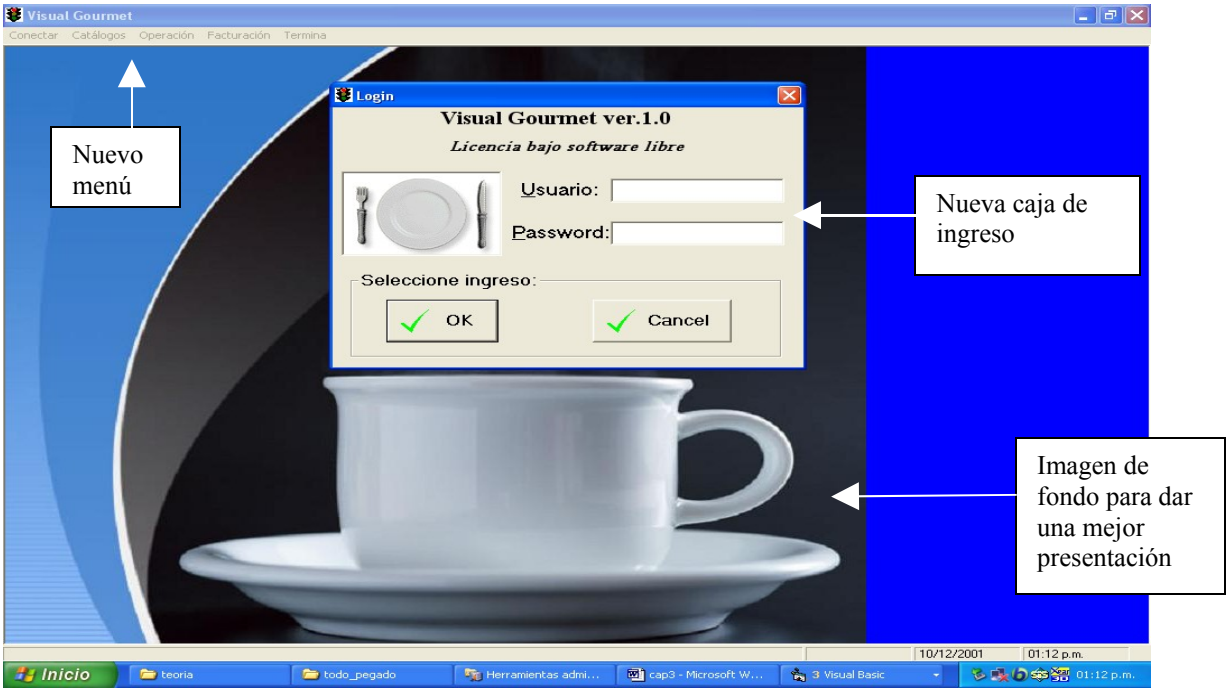


Figura 3.7 Nueva pantalla de ingreso a la aplicación.

En esta nueva pantalla también se modifica el menú principal de la aplicación original que estaba basado en botones por un menú desplegable, esto debido a que el menú basado en botones obliga a los clientes a tener un mouse para poder usar la aplicación; al cambiar a un menú del tipo antes mencionado ya no es obligatorio el Mouse y la aplicación puede usarse con un teclado.

También se agregó una figura al fondo de la pantalla principal, esto para darle una mejor presencia.

Otra pantalla rediseñada en su totalidad fué la que permite la captura de las comandas, esta pantalla se muestra a continuación en la figura 3.8

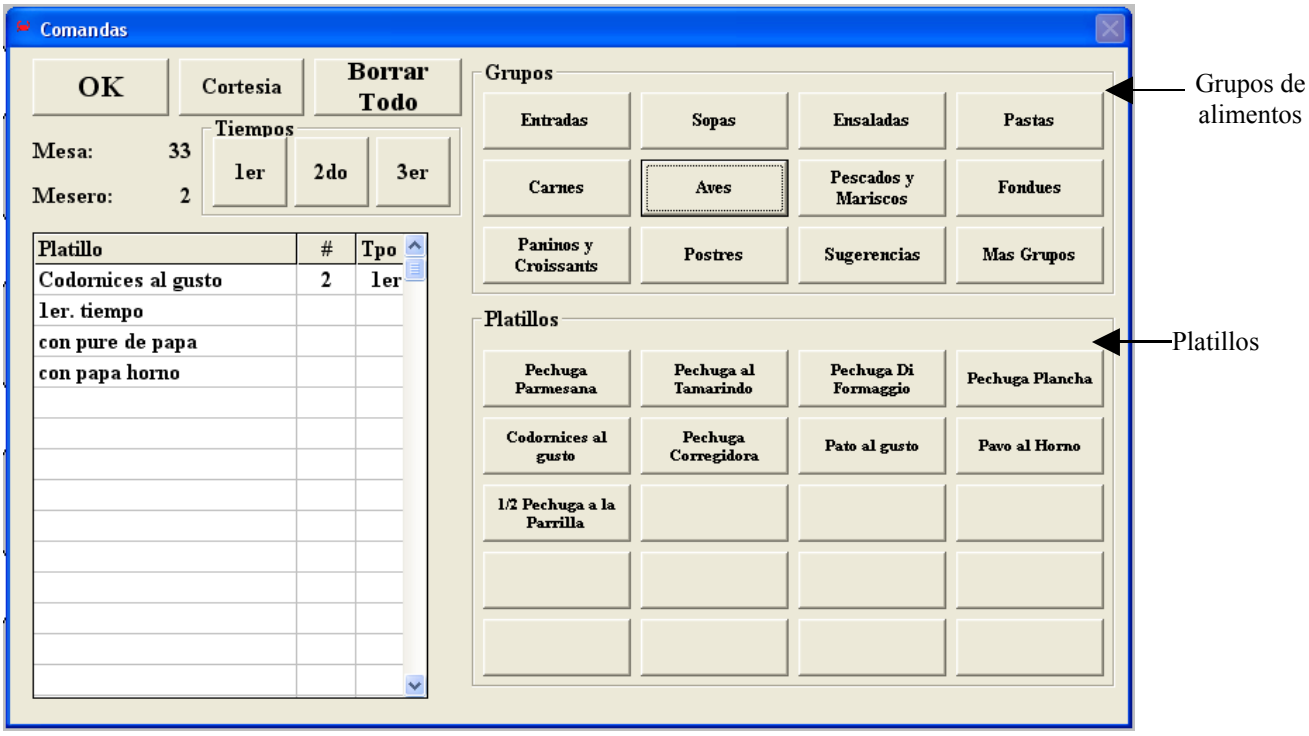


Figura 3.8 Pantalla para captura de comandas.

Como se advierte, esta pantalla muestra los grupos de alimentos y los platillos de cada grupo por medio de botones, esto a nivel código es complicado de manejar por que se tiene que borrar el título de cada botón y mostrar los siguientes o anteriores títulos en los botones.

La cantidad de información que muestra esta pantalla es demasiada y no respeta la secuencia en el diseño de pantallas que es de izquierda a derecha y de arriba hacia abajo.[7]

De la pantalla anterior se observa como la vista tiene que ir a la parte izquierda de la pantalla para localizar el grupo de platillos después ir hacia abajo para seleccionar el platillo y al final dirigimos la mira hacia la derecha para ver si hemos capturado bien el platillo solicitado.

[7] Libro: Análisis y Diseño de Sistemas, autores Kendall y Kendall, capítulos 14 y 15 “Diseño de una entrada eficaz”.

Otro defecto en esta pantalla es que oculta información al usuario; una vez seleccionado el platillo la pantalla cambia de aspecto al mostrado en la figura 3.9:

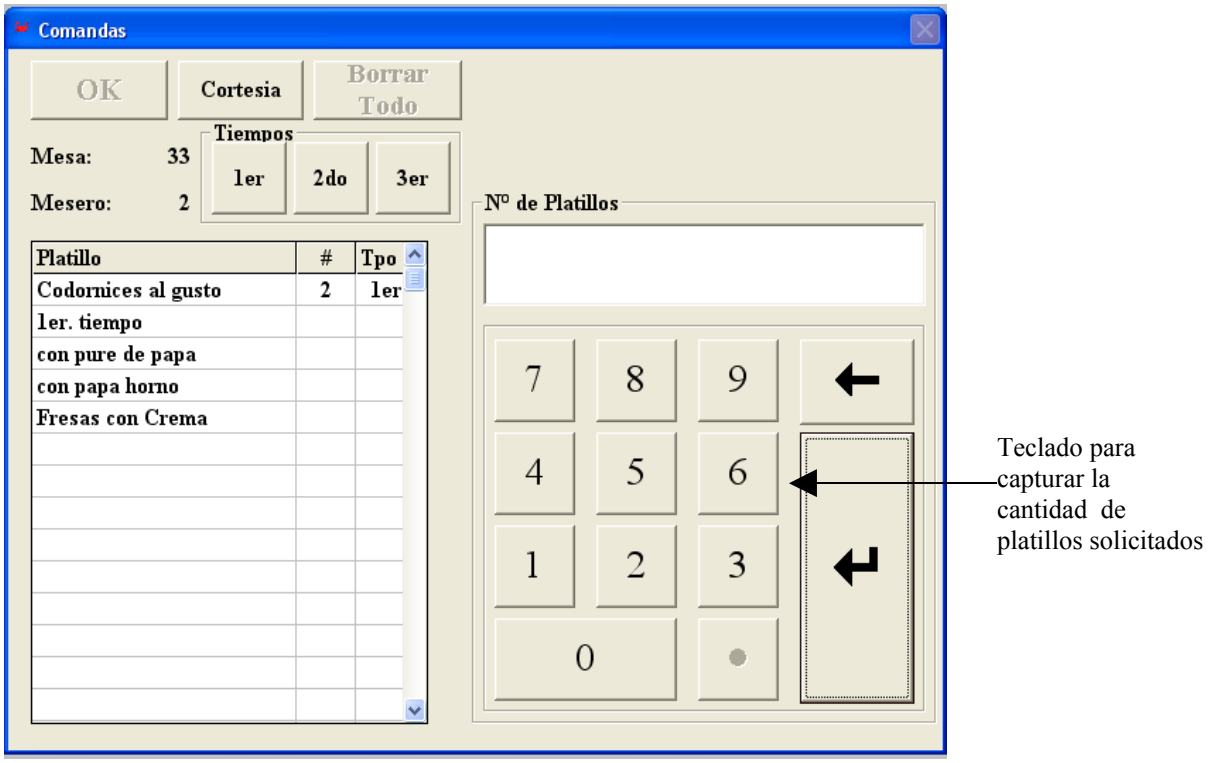


Figura 3.9 Nuevo aspecto de la pantalla de captura de comandas.

Nuevamente al capturar la cantidad de platillos requeridos, vuelve a cambiar de apariencia con otro tipo de información como se observa en la figura 3.10.

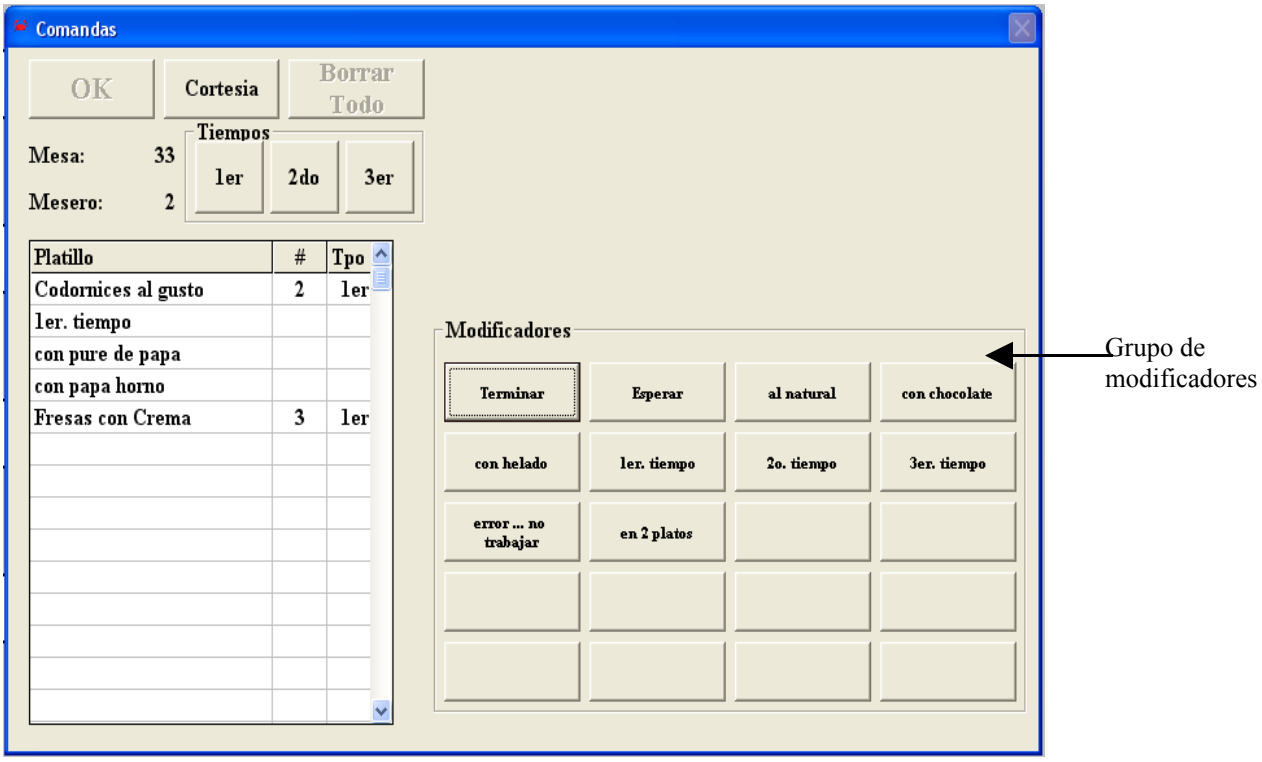


Figura 3.10 Otro aspecto de la pantalla de captura de comandas.

Desde el punto de vista de programación este tipo de pantallas es difícil de controlar así como su mantenimiento en posteriores cambios y como se ha mencionando es demasiada información oculta para el usuario lo que complica su operación.

En esta pantalla se hicieron modificaciones para evitar los cambios en su apariencia.

El nuevo aspecto de la pantalla de captura de comandas es el que se muestra a continuación en la figura 3.11:

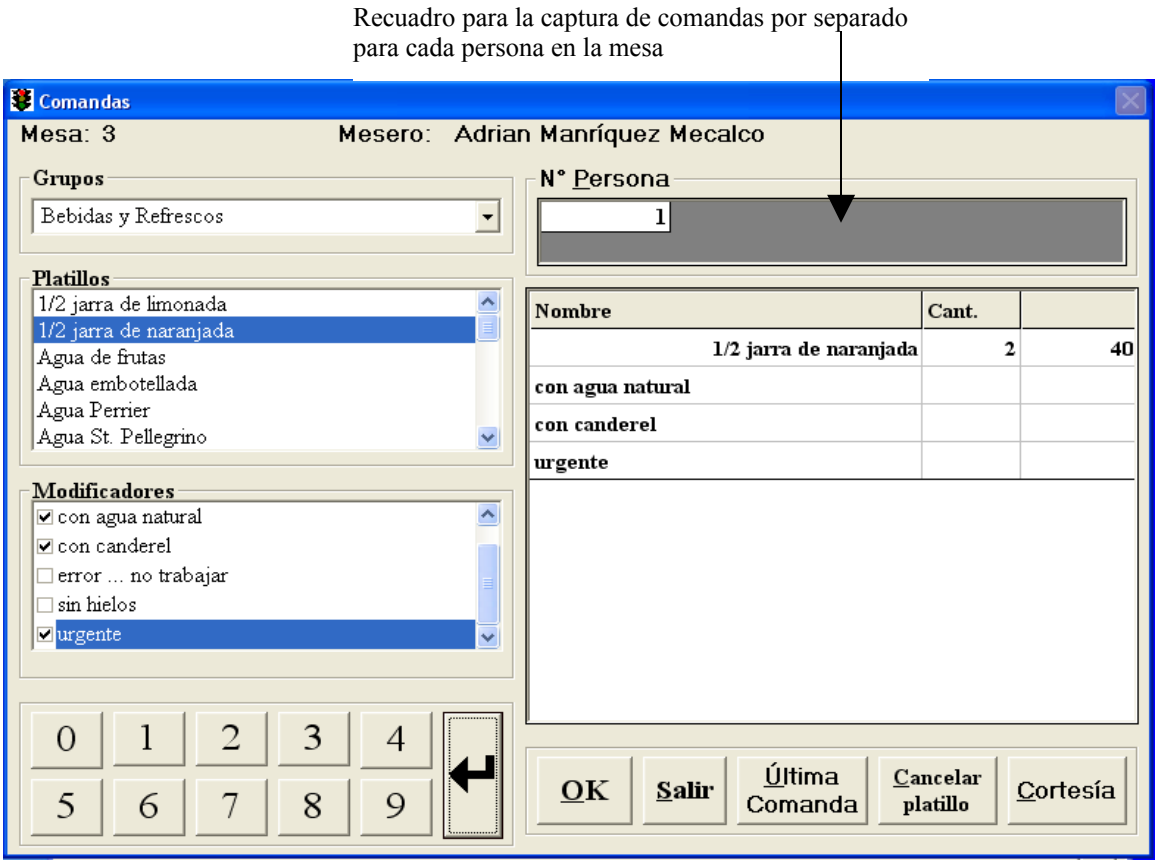


Figura 3.11 Nueva pantalla para la captura de comandas.

Los cambios más notorios se dan en la forma de presentar la información de los grupos, platillos y modificadores. Se optó por eliminar los grupos de botones que tenían originalmente y en su lugar se utiliza un combo box y dos cuadros lista para mostrar los datos; esto da una mejor apariencia y nos ayuda a presentar toda la información necesaria en una sola pantalla sin estar cambiando su aspecto como antes se hacía.

Con estas modificaciones, se respeta el movimiento lógico de la vista que es de izquierda a derecha y de abajo hacia arriba. [8]

Como puede notarse se presenta la misma cantidad de información pero ahora mejor agrupada y distribuida dentro de la pantalla. Por otro lado es más sencillo dar mantenimiento al código de esta pantalla que al de la pantalla original ya que esta no cambia de apariencia constantemente.
También se agrega en la parte superior un recuadro con el título No de persona para hacer la captura de una comanda por persona y posteriormente poder cobrar con cuentas separadas si así se requiere.

Otra pantalla modificada fué la utilizada para el cobro de una cuenta, la cual se muestra a en la figura 3.12:

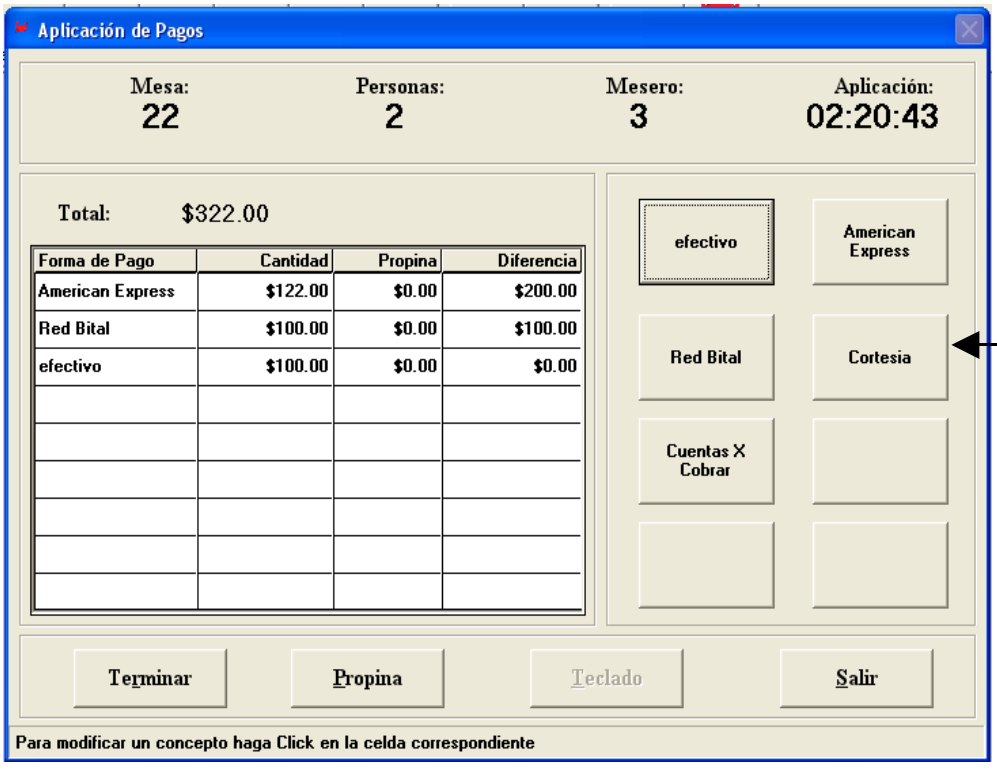


Figura 3.12 Pantalla de cobro de una cuenta.

[8] Libro: Análisis y Diseño de Sistemas, autores Kendall y Kendall, capítulos 14 y 15 "Diseño de una entrada eficaz".

En este caso la pantalla de cobros muestra las formas de pago usando un grupo de botones, cada vez que se pulsa un botón nuevamente la pantalla tiene que cambiar de aspecto como se muestra en la figura 3.13.

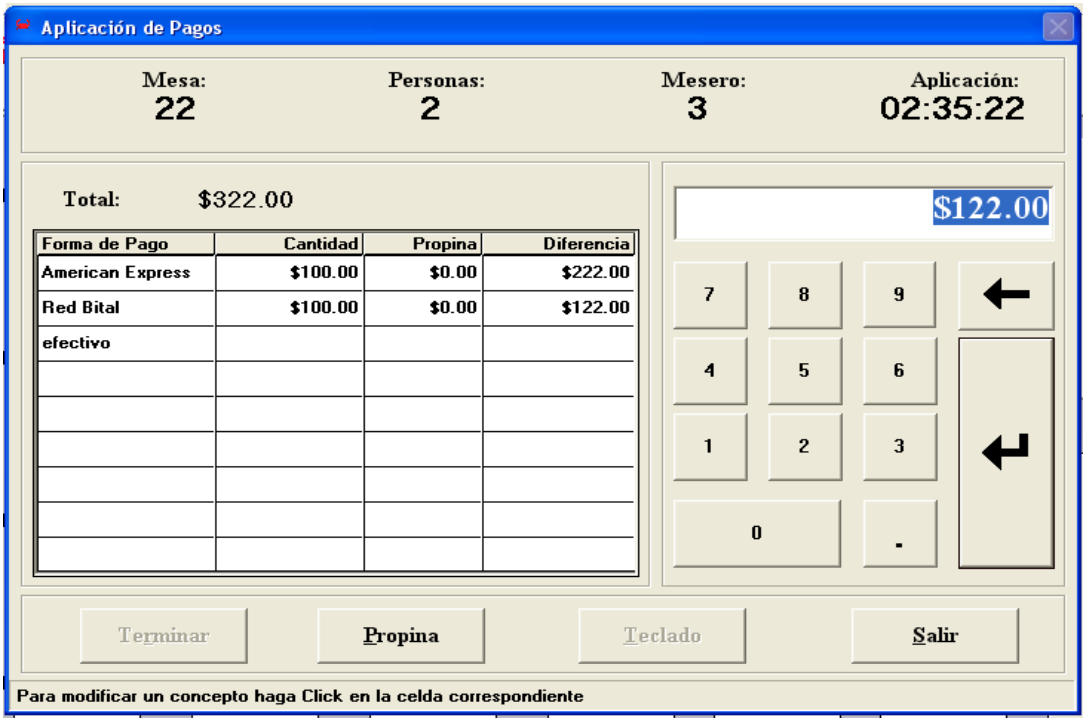


Figura 3.13 Nueva presentación de la pantalla de cobro de una cuenta.

Nuevamente este tipo de pantallas que tienen que cambiar constantemente de apariencia son difíciles de modificar en su código; también la secuencia de la vista no es el adecuado.[9]

[9] Libro: Análisis y Diseño de Sistemas, autores Kendall y Kendall, capítulos 14 y 15 "Diseño de una entrada eficaz".

La nueva pantalla propuesta tiene el aspecto mostrado en la figura 3.14:

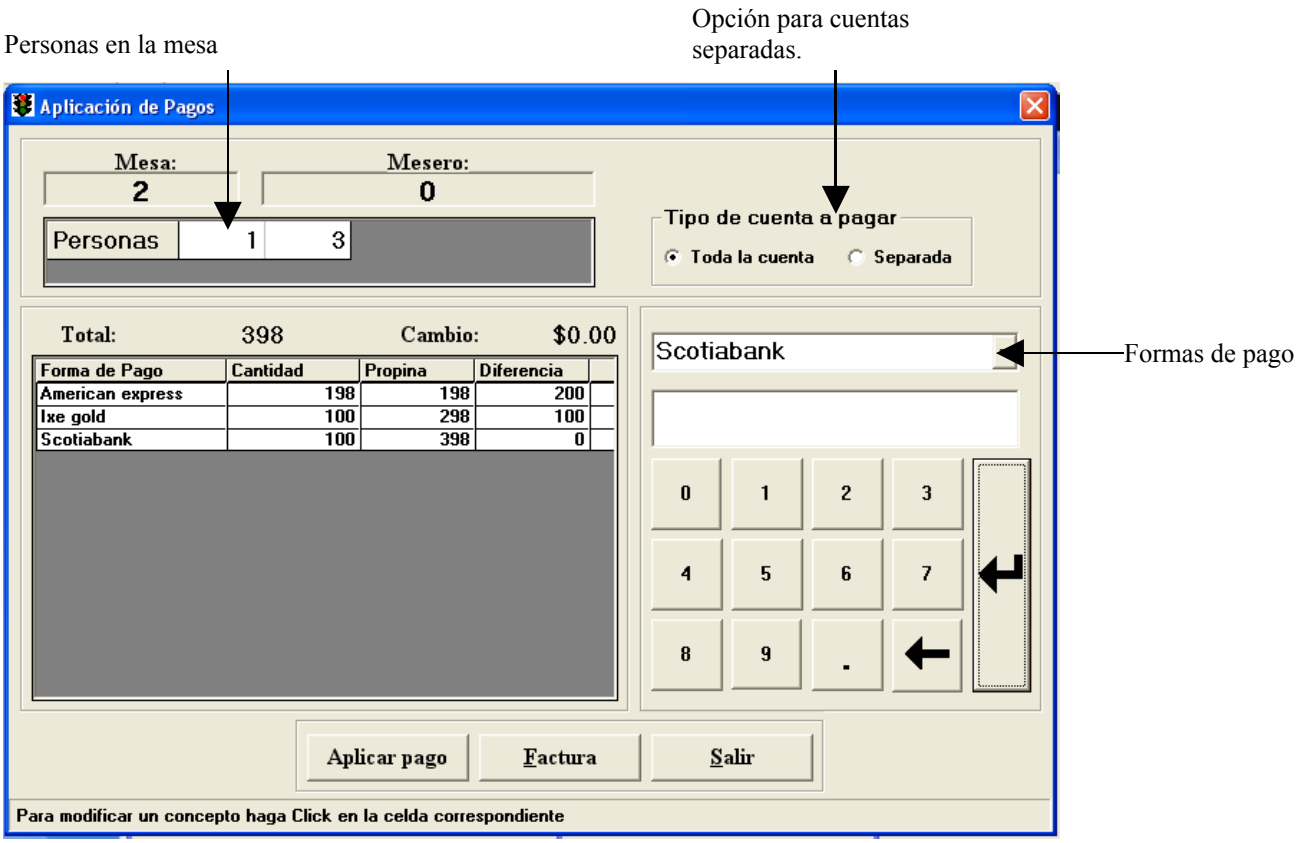


Figura 3.14 Pantalla propuesta para el cobro de una cuenta.

Ahora las formas de pago se muestran en un combo box lo que nos ahorra espacio y este espacio que sobra lo aprovechamos para mostrar el teclado numérico que aparecía en la segunda pantalla.

Otra modificación es la parte superior del grid, en el cual tenemos a las personas que están sentadas en la mesa y en el lado derecho dos opciones que nos permiten elegir entre pagar toda la cuenta o cobrar en cuentas separadas. Si se elige pagar la cuenta de forma separa se tendrá que escoger la persona a la que se le va a expedir el cheque (ticket) y el importe de las comandas de la persona seleccionada.

Otras pantallas que fueron simplificadas fueron las que forman parte de la facturación.

3.4 Cambios al código reportes.

En cuanto al código, este se redujo considerablemente al utilizar variables globales y la creación de un objeto activex para que reemplazará la representación de las mesas en la parte del control de las comandas.

El objeto activex incorpora mucho del código utilizado para el control de las mesas ahorrando así el control manual de estas y manejando mucha de la información de las mesas dentro de si. También permite representar la ubicación física de la mesa dentro del salón al tener la característica de poderse mover sobre la pantalla.

A continuación en la figura 3.15, se muestra la forma en que se representan las mesas dentro del sistema y la información mostrada en cada mesa (circulo) :

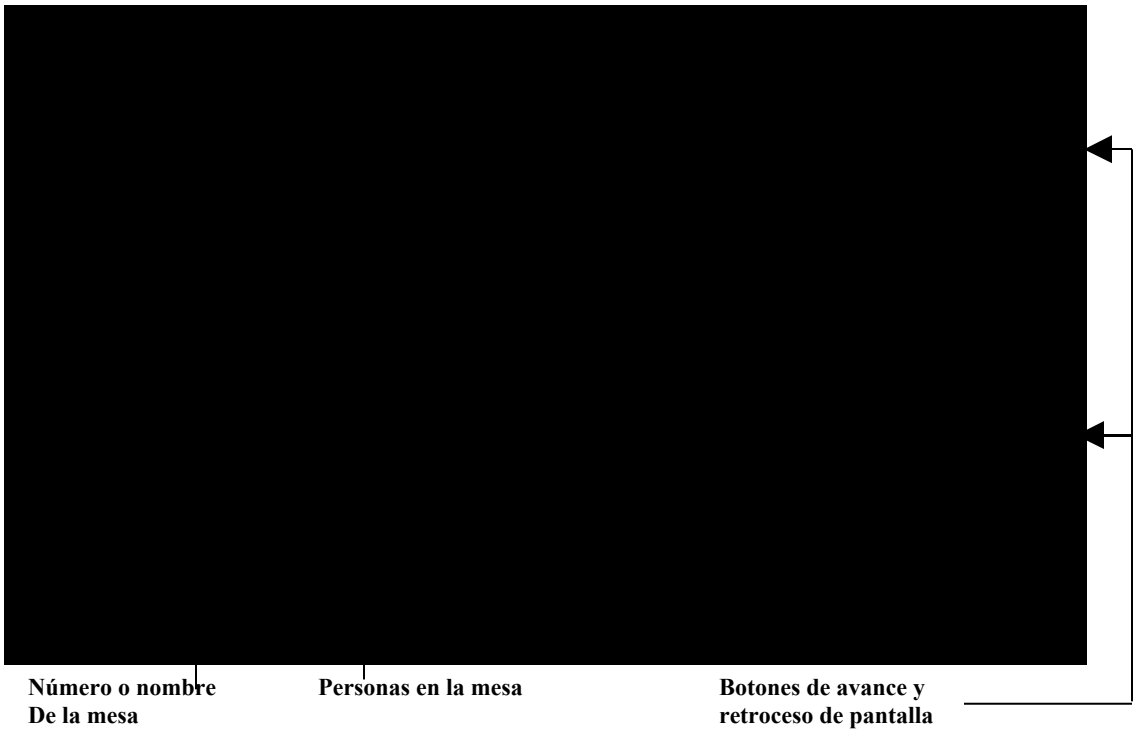


Figura 3.15 Representación de las mesas y su status.

Enseguida, la figura 3.16 presenta la nueva representación con el objeto activex.

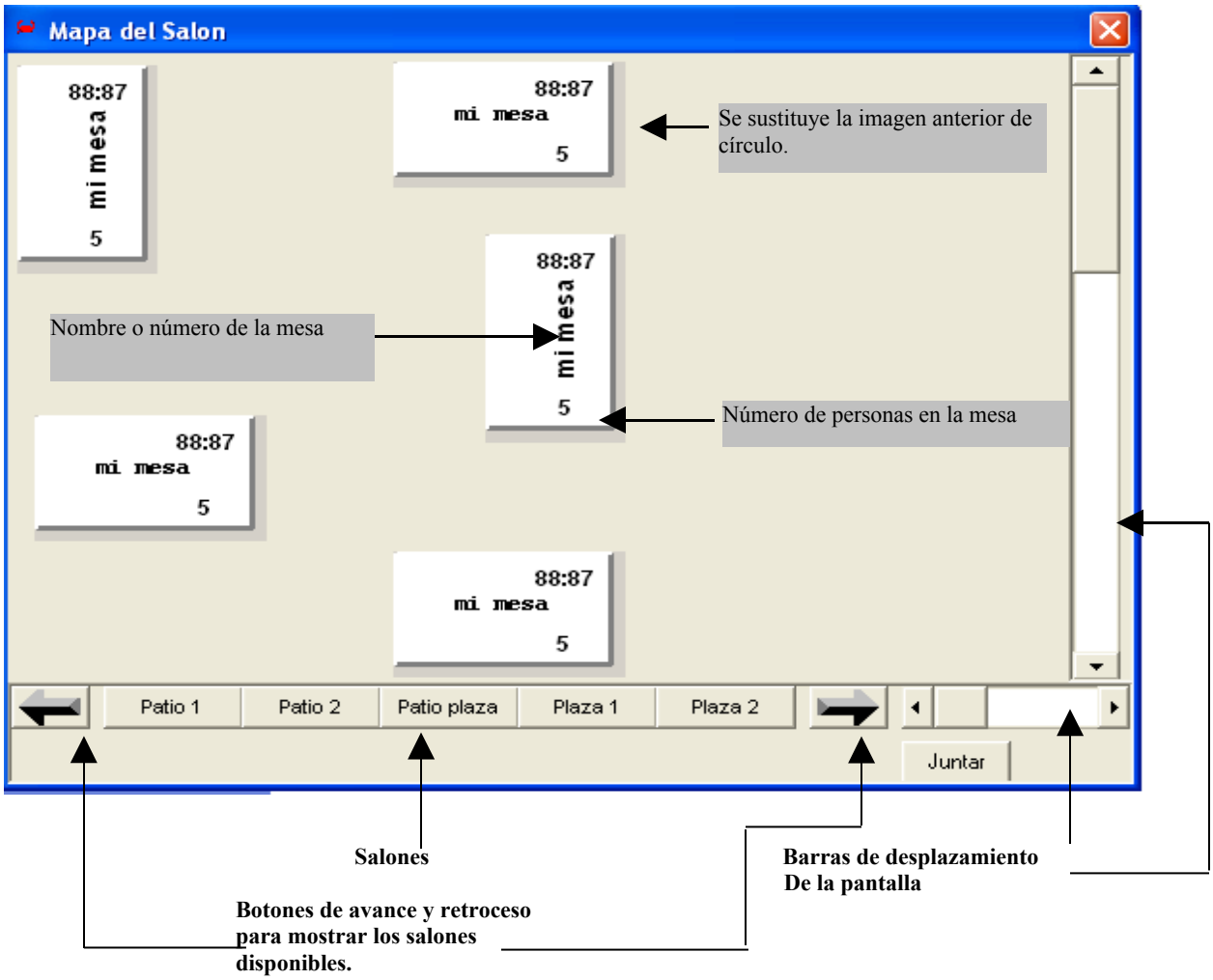


Figura 3.16 Nueva representación de las mesas y su status.

El código muestra del pintado de mesas viejo se encuentra en el anexo CIII.

La actualización de las mesas se lleva acabo eliminando los valores contenidos en cada mesa, posteriormente se extrae de la base de datos toda la información de cada mesa y se vuelve a asignar a cada una de las mesas los valores recuperados.

Para que una mesa muestre el color de la fase en la que se encuentra se hacen dos procedimientos pintado y repintado de mesas.

Ambos procedimientos recuperan la información de todas las mesas, muestran la información relacionada con cada una de ellas y visualizan a cada mesa con el color que corresponde a la fase en la que se encuentra. De esta forma da la impresión de que solamente se esta manipulando un solo objeto. Todo se realiza porque no se puede localizar directamente la mesa que fué seleccionada.

Con el nuevo objeto activex, es fácil saber que objeto (mesa) fué seleccionado dado que existe una variable global la cual indica que objeto estamos modificando, ese objeto tiene dentro de sus propiedades el valor *idmesa* de la mesa que esta representando por lo que ahora es fácil saber a que objeto hay que modificar sus valores en la tabla de la base como del programa. De esta manera, ya no es necesario recuperar la información de todas las mesas y colocarla en cada objeto de la mesa nuevamente.

La asignación del color se realiza en diferentes partes del programa y solamente tenemos que saber en que fase del ciclo de una mesa estamos, esto es fácil ya que el objeto activex tiene programada una propiedad llamada color, esta propiedad regresa un número que es el número de la fase de la mesa y así es mas sencillo controlar a una mesa.

Ahora ya no es necesario tener todo el código mostrado en el apartado CIII y solamente se hace necesario tener código para el control de la mesa como se muestra en el apartado CIV.

3.5 Cambios a reportes.

Ventajas de los nuevos formatos.

En la parte de reportes se eligió utilizar la herramienta gráfica Crystal Reports versión X, obteniéndose las siguientes ventajas:

- No es necesario crear tablas *como Cheq_omi. Cheq_ult, Fact_ult* que guarden los datos de posiciones para la impresión de campos e información de los documentos y también se ahorra espacio dentro de la base de datos.

- Se descartó el código de la aplicación que se había desarrollado para esta parte del sistema.

- Los reportes que forman parte del sistema adquieren una mejor apariencia, siendo ahora más agradables a la vista

- La modificación de estos documentos es más sencilla y versátil, lo cual nos permitirá adecuarlos a las necesidades particulares de algún cliente en especial.

En el anexo D apartados DI ad DIII se muestran algunos reportes muestra originales de la aplicación y en los apartados DIV al DVI los nuevos formatos para esos mismos reportes.

3.6 Requerimientos para su uso.

El nuevo punto de venta necesita los siguientes requerimientos para su correcto funcionamiento:

- Computadora servidor:

  Esta computadora deberá tener un procesador Pentium III o superior (incluido su similar amd), sistema operativo Linux o Windows y la base de datos Mysql 5; 128 mega bites de memoria o más y 10 gigabytes de disco duro para almacenar los respaldos que se realicen.

  En cuanto al resto de los equipos que funcionarán como terminales, los requerimientos son menores: 92 mega bites de memoria ram, sistema operativo Windows, instalar y configurar el conector Myodbc.

- Monitor:

  En lo que respecta al monitor, se pueden usar los mismos que se pedían anteriormente.

- Entorno de red a 100 mbps si es que se piensa utilizar este programa con varias terminales distribuidas en el negocio.

Algo que es importante resaltar es que se reduce la configuración de la instalación de esta aplicación en comparación con la original, anteriormente se tenía que configurar cada equipo terminal con un dispositivo de seguridad que se colocaba en el puerto de la impresora del equipo para que esta pudiera ejecutarse y se compartían recursos entre el servidor y las terminales como impresoras, carpetas compartidas, compartir la aplicación, etc. para que simularan que se estaba trabajando bajo un verdadero entorno de red.

En esta nueva aplicación ya no es necesario compartir recursos ya que cada computadora tendrá instalado el programa y aunado al uso del programa Myodbc, se tiene acceso a la información de la base de datos teniendo con

Conclusiones

A pesar de todos los defectos encontrados, Infocaja todavía es una aplicación redituable y que como se mencionó anteriormente vale la pena rescatarlo por medio de la Reingeniería de Software para que pueda seguir creciendo en cuanto a características.

Al cambiar el manejador de base de datos Access por Mysql, se pudo reducir el número de tablas que forman la base, también se eliminaron de estas las columnas redundantes y se crearon otras que ofrecen una mejor relación entre las tablas.

El uso de la herramienta Crystal Reports, permitió reducir el código y tener sentencias de SQL mucho más sencillas que las originales y principalmente agregarle funciones que no se tenían contempladas originalmente.

En cuanto a interfaces de usuario, la aplicación tiene una mejor presentación en sus pantallas y es más sencilla de usar que la versión original.

A nivel de código repetido, se reubicó poniéndolo en módulos compartidos, se utilizaron variables globales para disminuir el paso de parámetros entre los procesos, y se eliminó el uso de variables tipo variant.

Con lo mencionado anteriormente se ha conseguido una aplicación mejor organizada interiormente, esta nueva organización permitió agregarle nuevas funciones al sistema y como resultado ahora tenemos un sistema más completo que el Infocaja tomado para este trabajo.

Dentro de mi experiencia laboral he encontrado que las técnicas que nos ofrece la ingeniería de software, no se aplican en la mayoría de los departamentos de sistemas. De hecho la gran mayoría de los egresados de carreras informáticas ya insertados en el campo laboral, comentan que estas técnicas están fuera de la realidad, debido a que en la mayoría de las situaciones reales no suelen ser de mucha ayuda.

Debemos de estar consientes de que estas herramientas que se nos enseñan en la universidad cuentan con un respaldo científico y que son con las que cuenta el profesional del área de la informática. De ahí que sean enseñadas en casi todas las universidades que ofrecen carreras relacionadas con la computación, más que para llenar un plan de estudios y reprobar a los alumnos, sino que buscan eso: darnos herramientas científicas para afrontar los retos que el desarrollo de software y su mantenimiento implican.

También he podido observar que el principal obstáculo para no aplicar estas técnicas se debe a la exigencia que tienen los departamentos de sistemas de corregir lo más pronto posible cualquier error o anomalía que ocurra con los sistemas debido a que estos no pueden ser usados por el usuario final y por consecuencia no es productivo para la empresa.

En lo personal con este trabajo he podido constatar que estas técnicas no tienen como objetivo ser técnicas estándar de solución para todo tipo de problemas y entorno que rodea a cada situación; pero si ofrecen bases para que se pueda abordar una problemática con un método científico ya probado y del cual podemos hacer variantes y mezclar los diferentes métodos existentes.

Como se puede deducir, este trabajó buscó ser una aportación totalmente práctica sobre mucha de la enseñanza teórica que la carrera nos ofrece en el campo de las de base datos, programación estructurada, etc.

Anexo

A. Códigos de muestra completos.

A. I Código muestra para la impresión de una cuenta.

A. II Código muestra para el control de permisos de usuarios.

A. III Código muestra para el código del pintado de mesas viejo.

A. IV Código muestra para el código del pintado de mesas nuevo.

B. Reportes muestra.

B. I Reporte de ventas por grupo.

B. II Reporte de ventas por platillo.

B.III Reporte de tira de auditoría.

B.IV Nuevo reporte de ventas por grupo.

B.V Nuevo reporte de ventas por platillo.

B.VI Nuevo reporte de tira de auditoría.

Anexos

A Códigos muestra completos.

A. I Código muestra para la impresión de una cuenta.

```
Public Sub imp_cta(mmesa, mdesc, Mautoriza, Mforpago, Mcambio, Mimsn)
Dim Db As Database
ChDir App.Path
Set pdat_comandas = DBEngine.OpenDatabase("Infoset.mdb")
   '
   ' Abre tablas
   '
   Set prec_cheques = pdat_comandas.OpenRecordset("cheque")
   Set PREC_TabMesa = pdat_comandas.OpenRecordset("Mesa", dbOpenDynaset)
   Set prec_cobros = pdat_comandas.OpenRecordset("cobro")


 Comando = "SELECT Comanda.Cve_Mesa, Detalle.Desc_Producto, sum([cantidad]*[iva]) as ivaX,
sum([cantidad]*[impuesto]) As impX, first(Detalle.tipo_grupo) as grupo, first(comanda.horaini) as mhini,
First(Comanda.Personas) " & _
      " AS mper, First(Comanda.Id_Comanda) As ComandaId, First(Comanda.cve_mesero) AS mmes,
First(detalle.cve_Producto) AS Mcvep, Sum([cantidad]*[importe]) AS total, " & _
      " Sum(Detalle.Cantidad) AS Cantt FROM Comanda INNER JOIN Detalle ON Comanda.Id_Comanda =
Detalle.Comanda " & _
      " WHERE (((Comanda.Status)='I') AND ((Detalle.Status)<>'I') AND ((Detalle.Status_producto)='P') AND " &
_
      "((Detalle.cortesia)=False)) GROUP BY Comanda.Cve_Mesa, Detalle.Desc_Producto " & _
      " HAVING (((Comanda.Cve_Mesa) = '" & mmesa & "' ));"

   Set prec_good = pdat_comandas.OpenRecordset(Comando, 4)
   Comando = "SELECT Comanda.Cve_Mesa, Detalle.Desc_Producto, sum([cantidad]*[iva]) as ivaX,
sum([cantidad]*[impuesto]) As impX, first(Detalle.tipo_grupo) as grupo, first(comanda.horaini) as mhini,
First(Comanda.Personas) " & _
      " AS mper, First(Comanda.Id_Comanda) As ComandaId, First(Comanda.cve_mesero) AS mmes,
First(detalle.cve_Producto) AS Mcvep, Sum([cantidad]*[importe]) AS total, " & _
      " Sum(Detalle.Cantidad) AS Cantt FROM Comanda INNER JOIN Detalle ON Comanda.Id_Comanda =
Detalle.Comanda " & _
      " WHERE (((Comanda.Status)='I') AND ((Detalle.Status)<>'I') AND ((Detalle.Status_producto)='P') AND " &
_
      "((Detalle.cortesia)=True)) GROUP BY Comanda.Cve_Mesa, Detalle.Desc_Producto " & _
      " HAVING (((Comanda.Cve_Mesa) = '" & mmesa & "' ));"

Set prec_bad =
pdat_comandas.OpenRecordset(Comando, 4)
'
' Incrementa el numero de cheque en
  parametros
'
pdat_comandas.Execute ("UPDATE Parametro SET
Cheque = Cheque + Val(1)")

Set prec_paramet =  pdat_comandas.OpenRecordset(
_
   "SELECT Cheque FROM parametro")
```

```
        mcheq = prec_paramet!cheque - 1
prec_paramet.Close
    '
    ' Iva, nombre, rfc, domicilio, ticket o formato,
puerto de imp.,
    ' iva desglosado, presentar modif., ancho del ticket,
tamaño del font y negrita
    '
Set prec_paramet =
pdat_comandas.OpenRecordset("parametro")
    With prec_paramet
        mNOMR = .nombre
        Mrfcr = "Rfc : " & .RFC
        Mtele = "Tel.: " & .telefono
        mdomR1 = .Calle_Num & ", " & .Colonia
        mdomR2 = " C.P. " & .Cod_Pos & ", " &
.Poblacion
        mcual = .imp_che
        Mpto = .pto_che
        mddi = .c_iva
        Mcmo = .c_mod
        Manc = .anchoticket
        Mtam = .tam_font
        Mneg = .bold_font
        Mmen = .Mensaje
        Mlav = .Lineas_Avance
        DefaultEfectivo = .DefaultEfectivo
        Mimp1ley = .Nombre_Imp1
        Mimp2ley = .Nombre_Imp2
        c_imp1 = !c_imp1
    End With
    Mesp = Len(mNOMR)
    If Mesp > Manc Then
        mNOMR = Mid(mNOMR, 1, Manc)
    Else
        mNOMR = Space((Manc - Mesp) / 2) &
mNOMR
    End If
    Mesp = Len(mdomR1)
    If Mesp > Manc Then
        mdomR1 = Mid(mdomR1, 1, Manc)
    Else
        mdomR1 = Space((Manc - Mesp) / 2) &
mdomR1
    End If
    Mesp = Len(mdomR2)
    If Mesp > Manc Then
        mdomR2 = Mid(mdomR2, 1, Manc)
    Else
        mdomR2 = Space((Manc - Mesp) / 2) &
mdomR2
    End If
    Mesp = Len(Mtele)
    Mtele = Space((Manc - Mesp) / 2) & Mtele
    Mesp = Len(Mrfcr)
    Mrfcr = Space((Manc - Mesp) / 2) & Mrfcr
Select Case mcheq
Case Is < 10
    mscheq = "        " & Format(mcheq, "#")
Case Is < 100
    mscheq = "      " & Format(mcheq, "##")
Case Is < 1000
    mscheq = "     " & Format(mcheq, "###")
Case Is < 10000
    mscheq = "    " & Format(mcheq, "####")
Case Is < 100000
    mscheq = "   " & Format(mcheq, "#####")
Case Is < 1000000
    mscheq = "  " & Format(mcheq, "######")
Case Is < 10000000
    mscheq = " " & Format(mcheq, "#######")
Case Else
    mscheq = Format(mcheq, "########")
End Select
ImpCta.Label3.Caption = mscheq
'
' Imprime cuenta a puerto logico o físico
'
Dim X As Printer
N = 0
mpcaj = 0
If InStr("LPT COM", Mid(Mpto, 1, 3)) Then
    For Each X In Printers
     If InStr(X.Port, Mpto) <> 0 Then
        mpcaj = N
     End If
     N = N + 1
    Next
Else
    For Each X In Printers
     If InStr(X.DeviceName, Mpto) <> 0 Then
        mpcaj = N
     End If
     N = N + 1
    Next
End If
On Error GoTo NOSEIMP
Set Printer = Printers(mpcaj)
Printer.FontName = "courier New"
Printer.FontSize = Mtam
Printer.FontBold = Mneg
mlin = ""
hola = False
If prec_good.RecordCount() > 0 Then
    prec_good.MoveFirst
    mnper = prec_good.Fields("mper")
    mmese = prec_good.Fields("mmes")
    Mord = prec_good.ComandaId
    mhini = prec_good.mhini
    hola = True
Else
    prec_bad.MoveFirst
```

```
        mnper = prec_bad.Fields("mper")
        mmese = prec_bad.Fields("mmes")
        Mord = prec_bad.ComandaId
        mhini = prec_bad.mhini
    End If
    Mesp = Len(mmesa)
    smmesa = mmesa & Space(6 - Mesp)
    Mesp = Len(mnper)
    smnper = mnper & Space(6 - Mesp)
    Mesp = Len(mmese)
    smmese = mmese & Space(6 - Mesp)
    If mcual Then
        '
        '   Impresion de ticket
        '
        Msep = String(Manc, "-")
        Mlde = String(Manc - 19, "-")
        mlin = mNOMR & Chr(10) & mdomR1 &
Chr(10) & mdomR2 & Chr(10) & Mtele & Chr(10)
& Mrfcr & Chr(10) & Msep & Chr(10)
        If mmesa = "VC" Or mmesa = "VD" Then
            mlin = mlin & "Cheque:    " & Space(8 -
Len(mscheq)) & mscheq & "    Orden:  " & Space(8 -
Len(Mord)) & Mord & Chr(10)
 mlin = mlin & "Fecha:     " & Format(Date,
"dd/mm/yy") & "    Hora:   " & Space(8 -
Len(FormatDateTime(Time, vbShortTime))) &
FormatDateTime(Time, vbShortTime) & Chr(10)
        Else
            mlin = mlin & "Mesa  : " & smmesa &
Space(Manc - 31) & "Cheque:  " & mscheq &
Chr(10)
            mlin = mlin & "Mesero: " & smmese &
Space(Manc - 31) & "Fecha:   " & Format(Date,
"dd/mm/yy") & Chr(10)
            mlin = mlin & "Pers. : " & smnper &
Space(Manc - 31) & "Hora:       " & Format(Time,
"short time") & Chr(10)
        End If
        mlin = mlin & Msep & Chr(10)
        If hola Then
        mlin = mlin & "Cant.  Platillo" & Space(Manc -
25) & "Importe" & Chr(10) & "-----  " & Mlde & "
----------" & Chr(10)
        Mtot = 0
        Mivt = 0
        Mimpx = 0
    With prec_good
        While Not .EOF
            Mmcvep = .mcvep
            Miva = .Fields("ivaX")
            atot = .Fields("total")
            Mxxx = .Fields("impX")
            Mtot = Mtot + atot
            Mivt = Mivt + Miva
            If Impuesto1_Siempre = True Then
                Mimpx = Mimpx + Mxxx
            Else
                If Mimsn = "C" Then
                    Mimpx = Mimpx + Mxxx
                End If
            End If
            Mdes = Mid(.Fields("desc_producto"), 1,
Manc - 20)
            ms = Len(Mdes)
            maux = .Fields("cantt")
            Select Case maux
            Case Is = 0
                mcan = "     "
            Case Is < 10
                If maux = Int(maux) Then
                    mcan = "    " & Format(maux, "#")
                Else
                    If maux < 1 Then
                        mcan = "  " & Format(maux,
                                            ".00")
                    Else
                        mcan = " " & Format(maux,
                                            "#.00")
                    End If
                End If
            Case Else
                If maux = Int(maux) Then
                    mcan = "   " & Format(maux, "##")
                Else
                    mcan = Format(maux, "##.00")
                End If
            End Select
            mcan = mcan & Space(2)
            Mgem = atot
            If Impuesto1_Siempre = True Then
                If mddi Then
                    If c_imp1 Then
                        Mg2 = Miva + Mxxx
                    Else
                        Mg2 = Miva
                    End If
                Else
                    If c_ipm1 Then
                        Mg2 = Mxxx
                    End If
                End If
            Else
                Mg2 = Miva
            End If
            Mmeg = Mdes
            mlin = mlin & mcan
            If InStr(1, mdetcom, .ComandaId) = 0 Then
                mdetcom = mdetcom & .ComandaId & ", "
            End If
            If Mcmo Then
```

```
            Comando = "SELECT
Comanda.Cve_Mesa, Detalle.Desc_Producto,
sum([cantidad]*[iva]) As ivaX,
sum([cantidad]*[impuesto]) As impX,
first(Detalle.tipo_grupo) as grupo,
First(Comanda.Personas) " & _
                " AS mper, First(Comanda.Id_Comanda)
As ComandaId, First(Comanda.cve_mesero) AS
mmes, Sum([cantidad]*[importe]) AS total, " & _
                " Sum(Detalle.Cantidad) AS Cantt
FROM Comanda INNER JOIN Detalle ON
Comanda.Id_Comanda = Detalle.Comanda " & _
                " WHERE (((Comanda.Status)='I') AND
((Detalle.Status)<>'I') AND
 ((Detalle.status_producto)<>'P') AND" & _
                " ((Detalle.cortesia)=False) AND
((Detalle.Platillo_Modificador)= '" & Mmcvep & "' ))
GROUP BY Comanda.Cve_Mesa, " & _
                " Detalle.Desc_Producto HAVING
(((Comanda.Cve_Mesa) = '" & mmesa & "' ));"

        Else

            Comando = "SELECT
Comanda.Cve_Mesa, Detalle.Desc_Producto,
sum([cantidad]*[iva]) As ivaX,
sum([cantidad]*[impuesto]) As impX,
first(Detalle.tipo_grupo) as grupo,
First(Comanda.Personas) " & _
                " AS mper, First(Comanda.Id_Comanda)
As ComandaId, First(Comanda.cve_mesero) AS
mmes, Sum([cantidad]*[importe]) AS total, " & _
                " Sum(Detalle.Cantidad) AS Cantt
FROM Comanda INNER JOIN Detalle ON
Comanda.Id_Comanda = Detalle.Comanda " & _
                " WHERE (((Comanda.Status)='I') AND
((Detalle.Status)<>'I') AND
((Detalle.status_producto)='T') AND" & _
                " ((Detalle.cortesia)=False) AND
((Detalle.Platillo_Modificador)= '" & Mmcvep & "' ))
GROUP BY Comanda.Cve_Mesa, " & _
                " Detalle.Desc_Producto HAVING
(((Comanda.Cve_Mesa) = '" & mmesa & "' ));"

        End If
        Set prec_mod =
pdat_comandas.OpenRecordset(Comando, 4)
        With prec_mod
           If .RecordCount > 0 Then
              .MoveFirst
              While Not .EOF
                 atot = .Fields("total")
                 Miva = .Fields("ivaX")
                 Mxxx = .Fields("impX")
                 Mgem = Mgem + atot
                 If Impuesto1_Siempre = True Then
                    If mddi Then
                       If c_imp1 Then
                          Mg2 = Mg2 + Miva + Mxxx
                       Else
                          Mg2 = Mg2 + Miva
                       End If
                    Else
                       If c_ipm1 Then
                          Mg2 = Mg2 + Mxxx
                       End If
                    End If
                 Else
                    Mg2 = Mg2 + Miva
                 End If
                 Mtot = Mtot + atot
                 Mivt = Mivt + Miva
                 If Impuesto1_Siempre = True Then
                    Mimpx = Mimpx + Mxxx
                 Else
                    If Mimsn = "C" Then
                       Mimpx = Mimpx + Mxxx
                    End If
                 End If
                 Mdes =
prec_mod.Fields("desc_producto")
                 maux = prec_mod.Fields("cantt")
                 Select Case maux
                 Case Is = 0
                    mcan = "     "
                 Case Is < 10
                    If maux = Int(maux) Then
                       mcan = "    " & Format(maux,
"#")
                    Else
                       If maux < 1 Then
                          mcan = "  " & Format(maux,
".##")
                       Else
                          mcan = " " & Format(maux,
"#.##")
                       End If
                    End If
                 Case Else
                    If maux = Int(maux) Then
                       mcan = "   " & Format(maux,
"##")
                    Else
                       mcan = Format(maux, "##.##")
                    End If
                 End Select
                 mcan = "        "
                 Mmeg = Mmeg & ", " & Mdes
                 prec_mod.MoveNext
              Wend
           End If
        End With
```

```
        Mvec = Round((Len(Mmeg) / (Manc - 19)) +
0.5)
        If Mvec = 1 Then
           ms = Len(Mmeg)
           mlin = mlin & Mid(Mmeg, 1, Manc - 19) &
Space(Manc - 16 - ms)
        Else
           For Mcev = 1 To Mvec
              If Mcev = Mvec Then
                 ms = Len(Mid(Mmeg, ((Mcev - 1) *
(Manc - 19)) + 1, Manc - 19))
                 mlin = mlin & "       " & Mid(Mmeg,
((Mcev - 1) * (Manc - 19)) + 1, Manc - 19) &
Space(Manc - 16 - ms)
              Else
                 If Mcev = 1 Then
                    mlin = mlin & Mid(Mmeg, ((Mcev
- 1) * (Manc - 19)) + 1, Manc - 19) & Chr(10)
                 Else
                    mlin = mlin & "       " &
Mid(Mmeg, ((Mcev - 1) * (Manc - 19)) + 1, Manc -
19) & Chr(10)
                 End If
              End If
           Next
        End If
           Mgem = Mgem - Mg2
        Select Case Mgem
        Case Is < 10
           If Mgem = 0 Then
              ms = "        "
           Else
              ms = "    " & FormatCurrency(Mgem, 2)
           End If
        Case Is < 100
           ms = "   " & FormatCurrency(Mgem, 2)
        Case Is < 1000
           ms = "  " & FormatCurrency(Mgem, 2)
        Case Else
           ms = FormatCurrency(Mgem, 2)
        End Select
        mlin = mlin & ms & Chr(10)
        .MoveNext
     Wend
  End With
  msdesc = Str(mdesc * 100)
  Mivt = Round(Mivt, 2)
  Mimpx = Round(Mimpx, 2)
  If Impuesto1_Siempre = True Then
     Mtotaiva = Mtot - Mivt - Mimpx
  Else
     Mtotaiva = Mtot - Mivt
  End If
  Mtiva = Round(Mivt * (1 - mdesc), 2)
  mimpu = Round(Mimpx * (1 - mdesc), 2)
  mdesc = Round(Mtotaiva * mdesc, 2)
  Mtot = Mtotaiva - mdesc + Mtiva + mimpu
  If mdesc > 0 Then
     Select Case mdesc
     Case Is < 10
        ms = 7
     Case Is < 100
        ms = 6
     Case Is < 1000
        ms = 5
     Case Else
        ms = 3
     End Select
     mlin = mlin & Space(Manc - 26) & " - Desc.("
& msdesc & "%)" & Space(ms) &
FormatCurrency(mdesc, 2) & Chr(10)
  End If
  mlin = mlin & Msep & Chr(10)

  If mddi Then
     If c_imp1 Then
        maux = Mtotaiva - mdesc
     Else
        maux = Mtotaiva - mdesc + mimpu
     End If
     Select Case maux
     Case Is < 10
        ms = 7
     Case Is < 100
        ms = 6
     Case Is < 1000
        ms = 5
     Case Else
        ms = 3
     End Select
     mlin = mlin & Space(Manc - 23) & " Sub-
A/Iva:" & Space(ms) & FormatCurrency(maux, 2) &
Chr(10)
     Select Case Mtiva
     Case Is < 10
        ms = 7
     Case Is < 100
        ms = 6
     Case Is < 1000
        ms = 5
     Case Else
        ms = 3
     End Select
     mlin = mlin & Space(Manc - 23) & "     + Iva:"
& Space(ms) & FormatCurrency(Mtiva, 2) &
Chr(10)
  Else
     If Mimpx > 0 And c_imp1 Then
        maux = Mtotaiva + Mtiva
        Select Case maux
        Case Is < 10
           ms = 7
```

```
          Case Is < 100
             ms = 6
          Case Is < 1000
             ms = 5
          Case Else
             ms = 3
          End Select
          mlin = mlin & Space(Manc - 23) & " Sub-
Total:" & Space(ms) & FormatCurrency(maux, 2) &
Chr(10)
       End If
    End If
    Select Case Mtot
    Case Is < 10
       ms = 7
    Case Is < 100
       ms = 6
    Case Is < 1000
       ms = 5
    Case Else
       ms = 3
    End Select
    mban = 0
    If Mimpx = 0 Or c_imp1 = False Then
       mlin = mlin & Space(Manc - 23) & "     Total:"
& Space(ms) & FormatCurrency(Mtot, 2) & Chr(10)
& Chr(10)
    Else
       Select Case mimpu
       Case Is < 10
          ms = 7
       Case Is < 100
          ms = 6
       Case Is < 1000
          ms = 5
       Case Else
          ms = 3
       End Select
       Mnespl = 15 + Len(Mimp1ley)
       mlin = mlin & Space(Manc - Mnespl) & "+ " &
Mimp1ley & ":" & Space(ms) &
FormatCurrency(mimpu, 2) & Chr(10)
'       mlin = mlin & Space(Manc - 23) & "+
Impuesto:" & Space(ms) & FormatCurrency(mimpu,
2) & Chr(10)
       Select Case Mtot
       Case Is < 10
          ms = 7
       Case Is < 100
          ms = 6
       Case Is < 1000
          ms = 5
       Case Else
          ms = 3
       End Select
       mlin = mlin & Space(Manc - 23) & "     Total:"
& Space(ms) & FormatCurrency(Mtot, 2) & Chr(10)
& Chr(10)
    End If
       Mtotl = Mtot
       TotalCero = Mtotl
       Letras = NumALetras(Mtotl)
       If Len(Letras) > Manc Then
          let1 = Mid(Letras, 1, Manc)
          let2 = Mid(Letras, Manc + 1, Manc)
          mlin = mlin & let1 & Chr(10)
          mlin = mlin & let2 & Chr(10) & Chr(10)
       Else
          mlin = mlin & Letras & Chr(10) & Chr(10)
       End If
    End If
  With prec_bad
       If .RecordCount() > 0 Then
          mlin = mlin & " Cortesias:" & Chr(10) &
                 "Cant.  Platillo" & Chr(10) &
                 "-----  " & Mlde & Chr(10)
           .MoveFirst
           While Not .EOF
              Mmcvep = .mcvep
              Mdes =
                    Mid(.Fields("desc_producto"),
                          1, Manc - 19)
              ms = Len(Mdes)
              Mdes = Mdes + Space(Manc - 16 –
                                    ms)
              maux = .Fields("cantt")
              Select Case maux
              Case Is = 0
                 mcan = "     "
              Case Is < 10
                 If maux = Int(maux) Then
                    mcan = "    " & Format(maux,
                                        "#")
                 Else
                    If maux < 1 Then
                       mcan = "  " & Format(maux,
                                         ".##")
                    Else
                       mcan = " " & Format(maux,
                                         "#.##")
                    End If
                 End If
              Case Else
                 If maux = Int(maux) Then
                    mcan = "   " & Format(maux,
                                       "##")
                 Else
                    mcan = Format(maux, "##.##")
                 End If
              End Select
              mcan = mcan & Space(2)
              mlin = mlin & mcan & Mdes & Chr(10)
```

```
                If Mcmo Then
                Comando = "SELECT
Comanda.Cve_Mesa, Detalle.Desc_Producto,
first(Detalle.tipo_grupo) as grupo,
First(Comanda.Personas) " & _
                " AS mper,
First(Comanda.Id_Comanda) As ComandaId,
First(Comanda.cve_mesero) AS mmes,
Sum([cantidad]*[importe]) AS total, " & _
                " Sum(Detalle.Cantidad) AS Cantt
FROM Comanda INNER JOIN Detalle ON
Comanda.Id_Comanda = Detalle.Comanda " & _
                " WHERE (((Comanda.Status)='I')
AND ((Detalle.Status)<>'I') AND
((Detalle.Status_producto)<>'P') AND" & _
                " ((Detalle.cortesia)=True) AND
((Detalle.Platillo_Modificador)= '" & Mmcvep & "' ))
GROUP BY Comanda.Cve_Mesa, " & _
                " Detalle.Desc_Producto HAVING
(((Comanda.Cve_Mesa) = '" & mmesa & "' ));"
                Else
                Comando = "SELECT
Comanda.Cve_Mesa, Detalle.Desc_Producto,
first(Detalle.tipo_grupo) as grupo,
First(Comanda.Personas) " & _
                " AS mper,
First(Comanda.Id_Comanda) As ComandaId,
First(Comanda.cve_mesero) AS mmes,
Sum([cantidad]*[importe]) AS total, " & _
                " Sum(Detalle.Cantidad) AS Cantt
FROM Comanda INNER JOIN Detalle ON
Comanda.Id_Comanda = Detalle.Comanda " & _
                " WHERE (((Comanda.Status)='I')
AND ((Detalle.Status)<>'I') AND
((Detalle.Status_producto)='T')  AND" & _
                " ((Detalle.cortesia)=True) AND
((Detalle.Platillo_Modificador)= '" & Mmcvep & "' ))
GROUP BY Comanda.Cve_Mesa, " & _
                " Detalle.Desc_Producto HAVING
(((Comanda.Cve_Mesa) = '" & mmesa & "' ));"
End If

Set prec_mod =  dat_comandas.OpenRecordset
(Comando, 4)
                I
if prec_mod.RecordCount > 0 Then
     prec_mod.MoveFirst
     While Not prec_mod.EOF
        If mddi Then
             atot =  prec_mod.Fields("total") /
                  ((100 + Miva) / 100)
        Else
             atot = prec_mod.Fields("total")
        End If
        Mtot = Mtot + atot
        Mdes =   mid_mod.Fields  _
                  ("desc_producto"), 1, Manc - 19)
        ms = Len(Mdes)
        Mdes = Mdes + Space(Manc - 16 - ms)
        maux = prec_mod.Fields("cantt")
        Select Case maux
        Case Is = 0
             mcan = "     "
        Case Is < 10

    If maux = Int(maux) Then
        mcan = "    " & Format(maux, "#")
    Else
        If maux < 1 Then
            mcan = "  " & Format(maux, ".##")
        Else
            mcan = " " & Format(maux, "#.##")
        End If
     End If
 Case Else
      If maux = Int(maux) Then
          mcan = "   " & Format(maux, "##")
       Else
          mcan = Format(maux, "##.##")
       End If
 End Select
 mcan = mcan & Space(2)
 Select Case atot
 Case Is < 10
    If atot = 0 Then
          ms = "        "
    Else
          ms = "    " & FormatCurrency(atot, 2)
    End If
 Case Is < 100
          ms = "   " & FormatCurrency(atot, 2)
  Case Is < 1000
          ms = "  " & FormatCurrency(atot, 2)
  Case Else
          ms = FormatCurrency(atot, 2)
  End Select
  mlin = mlin & mcan & Mdes & ms & Chr(10)
  prec_mod.MoveNext
Wend
End If
.MoveNext
Wend
mlin = mlin & " " & Chr(10)
End If
End With

 If mmesa = "VD" Then
     With FrmBuscaClientes
          mlin = mlin & "Preguntar por: " &
                .Txt(1).Text & Chr(10) & _
                "Telefono: " & .Txt(2).Text &
               Chr(10) & "Domicilio: " &
```

```
                .Txt(3).Text
            mlin = mlin & Chr(10) & "Entre: " &
                .Txt(8) & Chr(10) & "Forma de
                                    pago: " & _
                Mforpago
            If Mcambio <> "" Then
                mlin = mlin & Chr(10) & "Cambio
                                        de: " &
Mcambio
              End If
            End With
        Else
            mlin = mlin & Mmen & Chr(10)
        End If
        If Mcorte = "" Or IsNull(Mcorte) Then
            For i = 1 To Mlav - 1
               Polluken$ = Polluken$ & vbCrLf  'Chr(10)
            Next i
            Polluken$ = Polluken$ & "_" & Chr(10)
            mlin = mlin + Polluken$
            Printer.Print mlin
        Else
            Printer.Print mlin
            Printer.FontSize = 10
            Printer.FontName = "control"
            Printer.Print Mcorte
        End If
    Else
    '
    '   Impresion de cheque bajo formato predefinido
    '
        '
        '   Encabezado
        '
        mini = 1
        mtip = "E"
        Set prec_forma =
pdat_comandas.OpenRecordset("select * from
cheq_ult where tipo = '" & mtip & "'")
        With prec_forma
            .MoveFirst
            mbin = 0
            Do While Not .EOF
               no_lin = .no_lin
               no_col = .no_col
               campo = .campo
               linsig = no_lin
                  .MoveNext
               If Not .EOF Then
                  linsig = .no_lin
               End If
               .MovePrevious
               If Mid(campo, 1, 1) = "*" Then
                  campo = Mid(campo, 2, Len(campo) - 1)
               Else
                  Select Case campo
                     Case Is = "folio"
                        campo = mscheq
                     Case Is = "fecha"
                        campo = Date
                     Case Is = "mesa"
                        campo = mmesa
                  Case Is = "mesero"
                     campo = mmese
                  Case Is = "personas"
                     campo = mnper
                  Case Else
                     Mlen = Len(campo)
                     campo = Mid(campo, 2, Mlen - 1)
                  End Select
               End If
               Mlen = Len(campo)
               msigue = 1
               If mbin = 0 Then
                  lin0 = no_lin
                  col0 = no_col - 1
                  col1 = col0
                  mlen0 = Mlen
                  mlen1 = mlen0
                  mbin = 1
               Else
                  If no_lin = lin0 Then
                     col0 = no_col - 1 - col1 - mlen1
                     If col0 < 0 Then col0 = 1
                        col1 = no_col - 1
                        mlen1 = Mlen
                     Else
                        col0 = no_col - 1
                        col1 = col0
                        lin0 = no_lin
                        mlen0 = Mlen
                        mlen1 = mlen0
                     End If
                     If no_lin < linsig Then
                        msigue = 0
                     End If
                End If

                If mini < no_lin Then
                   inda = mini
                   For k = inda To no_lin - 1
                      mlin = mlin & " " & Chr(10)
                      mini = mini + 1
                   Next
                End If
                mlin = mlin & Space(col0) & campo
                If msigue = 0 Then
                   mlin = mlin & Chr(10)
                   mini = mini + 1
                End If
                .MoveNext
               Loop
```

```
   mlin = mlin & Chr(10)
   mini = mini + 1
End With
'
'   Detalle
'
 mtip = "D"
 i = 1
 mbpu = 0
 mbal = 0
 Set prec_forma =
        pdat_comandas.OpenRecordset( _
         "select * from cheq_ult where tipo = '" &
          mtip & "'")
  With prec_forma
     .MoveFirst
        mlini = .no_lin
        mlfin = .no_lin
        Do While Not .EOF
           xd(i) = .no_col
           If .no_lin < mlini Then
              mlini = .no_lin
           End If
           If .no_lin > mlfin Then
              mlfin = .no_lin
           End If
           Select Case .campo
              Case Is = "p.u."
                 mbpu = 1
              Case Is = "alim-beb"
                 mbal = 1
           End Select
           i = i + 1
       .MoveNext
     Loop
     End With
     If mini < mlini Then
        inda = mini
        For k = inda To mlini - 1
           mlin = mlin & " " & Chr(10)
           mini = mini + 1
        Next
      End If
      Mldes = xd(3) - xd(2) - 2
      Mtot = 0
      mlindet = 0
      Mtali = 0
      Mtbeb = 0
      With prec_good
         While Not .EOF
            Mmcvep = .mcvep
            atot = .Fields("total")
            If atot > 0 And .Fields("cantt") > 0
            Then
              apu = Round(atot / .Fields("cantt"), 2)
            Else
              apu = 0
            End If
            Mtot = Mtot + atot
            Mdes = Mid(.Fields("Desc_Producto"),
                       1, Mldes)
            ms = Len(Mdes)
Mdes = Mdes + Space(Mldes - ms)
maux = .Fields("cantt")
Select Case maux
Case Is = 0
    mcan = "     "
Case Is < 10
    If maux = Int(maux) Then
        mcan = "    " & Format(maux, "#")
    Else
        mcan = "  " & Format(maux, "#.#")
    End If
 Case Is < 100
    If maux = Int(maux) Then
        mcan = "   " & Format(maux, "##")
    Else
        mcan = " " & Format(maux, "##.#")
    End If
 Case Else
    If maux = Int(maux) Then
        mcan = "  " & Format(maux, "###")
    Else
        mcan = Format(maux, "###.#")
    End If
 End Select
 Mesp = xd(2) - xd(1) - 5
 If Mesp < 0 Then
    MsgBox ("favor de revisar el formato
                del detalle")
     Exit Sub
 End If
 mcan = mcan & Space(Mesp)
 Select Case atot
 Case Is < 10
       ms = "    " & FormatCurrency(atot, 2)
 Case Is < 100
        ms = "   " & FormatCurrency(atot, 2)
 Case Is < 1000
      ms = "  " & FormatCurrency(atot, 2)
 Case Else
        ms = FormatCurrency(atot, 2)
 End Select
 Select Case apu
 Case Is < 10
     ms1 = "    " & FormatCurrency(apu, 2)
 Case Is < 100
     ms1 = "   " & FormatCurrency(apu, 2)
 Case Is < 1000
      ms1 = "  " & FormatCurrency(apu, 2)
 Case Else
      ms1 = FormatCurrency(apu, 2)
```

```
        End Select
        If mbpu = 1 Then
             mhol = Mdes & ms1
        Else
             mhol = Mdes
        End If
    If mbal = 1 Then
       If .Fields("grupo") = "A" Then
            mlin = mlin & mcan & mhol & ms &
                   Chr(10)
           Mtali = Mtali + atot
       Else
            mlin = mlin & mcan & mhol &
                   Space(9) & ms & Chr(10)
           Mtbeb = Mtbeb + atot
       End If
   Else
       If atot > 0 Then
            mlin = mlin & mcan & mhol & ms &
                   Chr(10)
       Else
            mlin = mlin & mcan & mhol &
                   Chr(10)
        End If
   End If
   If Mcmo Then

     Comando = "SELECT
Comanda.Cve_Mesa, Detalle.Desc_Producto,
first(Detalle.tipo_grupo) as grupo,
First(Comanda.Personas) " & _
                " AS mper, First(Comanda.Id_Comanda)
As ComandaId, First(Comanda.cve_mesero) AS
mmes, Sum([cantidad]*[importe]) AS total, " & _
                " Sum(Detalle.Cantidad) AS Cantt
FROM Comanda INNER JOIN Detalle ON
Comanda.Id_Comanda = Detalle.Comanda " & _
                " WHERE (((Comanda.Status)='I') AND
((Detalle.Status)<>'I') AND
((Detalle.Status_producto)<>'P') AND" & _
                " ((Detalle.cortesia)=False) AND
((Detalle.Platillo_Modificador)= '" & Mmcvep & "' ))
GROUP BY Comanda.Cve_Mesa, " & _
                " Detalle.Desc_Producto HAVING
(((Comanda.Cve_Mesa) = '" & mmesa & "' ));"

            Else

              Comando = "SELECT
Comanda.Cve_Mesa, Detalle.Desc_Producto,
first(Detalle.tipo_grupo) as grupo,
First(Comanda.Personas) " & _
                " AS mper, First(Comanda.Id_Comanda)
As ComandaId, First(Comanda.cve_mesero) AS
mmes, Sum([cantidad]*[importe]) AS total, " & _
                " Sum(Detalle.Cantidad) AS Cantt
FROM Comanda INNER JOIN Detalle ON
Comanda.Id_Comanda = Detalle.Comanda " & _
                " WHERE (((Comanda.Status)='I') AND
((Detalle.Status)<>'I') AND
((Detalle.Status_producto)='T') AND" & _
                " ((Detalle.cortesia)=False) AND
((Detalle.Platillo_Modificador)= '" & Mmcvep & "' ))
GROUP BY Comanda.Cve_Mesa, " & _
                " Detalle.Desc_Producto HAVING
(((Comanda.Cve_Mesa) = '" & mmesa & "' ));"
End If
Set prec_mod =   _
pdat_comandas.OpenRecordset(Comando, 4)

If prec_mod.RecordCount > 0 Then
   prec_mod.MoveFirst
   While Not prec_mod.EOF
       atot = prec_mod.Fields("total")
       If atot > 0 And prec_mod.Fields("cantt")
          > 0 Then

            apu = Round(atot /
                  prec_mod.Fields("cantt"), 2)
       Else
            apu = 0
       End If
       Mtot = Mtot + atot
       Mdes =  Mid(prec_mod.Fields
                  ("Desc_Producto"),
                   1, Mldes)
       ms = Len(Mdes)
       Mdes = Mdes + Space(Mldes - ms)
       maux = prec_mod.Fields("cantt")
       Select Case maux
       Case Is = 0
            mcan = "     "
       Case Is < 10
            If maux = Int(maux) Then
               mcan = "    " & Format(maux, "#")
            Else
               mcan = "  " & Format(maux, "#.#")
            End If
       Case Is < 100
            If maux = Int(maux) Then
               mcan = "   " & Format(maux, "##")
            Else
              mcan = " " & Format(maux, "##.#")
            End If
        Case Else
            If maux = Int(maux) Then
               mcan = "  " & Format(maux, "###")
            Else
               mcan = Format(maux, "###.#")
            End If
       End Select
       Mesp = xd(2) - xd(1) - 5
```

```
      If Mesp < 0 Then
           MsgBox ("favor de revisar el formato
                    del detalle")
           Exit Sub
      End If
      mcan = mcan & Space(Mesp)
      Select Case atot
      Case Is < 10
           ms = "    " & FormatCurrency(atot, 2)
      Case Is < 100
           ms = "   " & FormatCurrency(atot, 2)
      Case Is < 1000
           ms = "  " & FormatCurrency(atot, 2)
      Case Else
           ms = FormatCurrency(atot, 2)
      End Select
      Select Case apu
      Case Is < 10
          ms1 = "    " & FormatCurrency(apu, 2)
      Case Is < 100
          ms1 = "   " & FormatCurrency(apu, 2)
      Case Is < 1000
           ms1 = "  " & FormatCurrency(apu, 2)
      Case Else
           ms1 = FormatCurrency(apu, 2)
      End Select

      If mbpu = 1 Then
           mhol = Mdes & ms1
      Else
           mhol = Mdes
      End If
      If mbal = 1 Then
         If prec_mod.Fields("grupo") = "A" Then
            mlin = mlin & mcan & mhol & ms &
                   Chr(10)
            Mtali = Mtali + atot
         Else
            mlin = mlin & mcan & mhol &
                   Space(9) & ms & Chr(10)
            Mtbeb = Mtbeb + atot
         End If
      Else
          If atot > 0 Then
             mlin = mlin & mcan & mhol & ms
                    & Chr(10)
          Else
             mlin = mlin & mcan & mhol &
                    Chr(10)
          End If
       End If
       prec_mod.MoveNext
   Wend
 End If
mlindet = mlindet + 1
mini = mini + 1
 .MoveNext
 Wend
 End With
      '
      '   Totales
      '
      If mini < mlfin - 1 Then
         inda = mini
         For k = inda To mlfin - 3
            mlin = mlin & " " & Chr(10)
            mini = mini + 1
         Next
      End If
      msdesc = Str(mdesc * 100)
      Mtotaiva = Mtot / ((100 + Miva) / 100)
      mdesc = Round(Mtotaiva * mdesc, 2)
      Mtot = (Mtotaiva - mdesc) * (100 + Miva) / 100
      Mtotl = Mtot
      TotalCero = Mtotl
      Mtiva = Round((Mtotaiva - mdesc) * Miva /
100, 2)
      mimpu = Round(Mtot * Mimpx / 100, 2)
      Select Case mdesc
      Case Is < 10
         ms = "    " & FormatCurrency(mdesc, 2)
      Case Is < 100
         ms = "   " & FormatCurrency(mdesc, 2)
      Case Is < 1000
         ms = "  " & FormatCurrency(mdesc, 2)
      Case Else
         ms = FormatCurrency(mdesc, 2)
      End Select
      If mdesc <> 0 Then
         mlin = mlin & "Descuento  (" & msdesc &
"%):" & Space(8) & ms & Chr(10)
      Else
         mlin = mlin & " " & Chr(10)
      End If
      mtip = "T"
      Set prec_forma =
pdat_comandas.OpenRecordset("select * from
cheq_ult where tipo = '" & mtip & "'")
      With prec_forma
         .MoveLast
         lin = .RecordCount
         .MoveFirst
         i = 0
         Do While Not .EOF
            i = i + 1
            no_lin = .no_lin
            no_col = .no_col
            campo = .campo
            mini = no_lin + 1
            If i <= lin - 1 Then
               .MoveNext
               no_sig = .no_lin
```

```
        no_olc = .no_col
        .MovePrevious
     Else
    no_sig = .no_lin + 1
    no_olc = .no_col
End If
If i > 1 Then
    .MovePrevious
    no_ant = .no_lin
    no_loc = .no_col
    .MoveNext
Else
     no_ant = no_lin
     no_loc = 0
End If
If no_lin > no_ant + 1 Then
     For k = no_ant + 1 To no_lin - 1
          mlin = mlin & " " & Chr(10)
     Next
End If
Select Case campo
Case Is = "folio"
       campo = mscheq
Case Is = "total"
       Select Case Mtot
       Case Is < 10
            ms = "    " &
            FormatCurrency(Mtot, 2)
       Case Is < 100
            ms = "   " &
             FormatCurrency(Mtot, 2)
       Case Is < 1000
         ms = "  " & FormatCurrency(Mtot, 2)
       Case Else
          ms = FormatCurrency(Mtot, 2)
       End Select
       campo = ms
Case Is = "total+impuesto"
       Select Case Mtot + mimpu
       Case Is < 10
          ms = "    " &
          FormatCurrency(Mtot + mimpu, 2)
       Case Is < 100
           ms = "   " & FormatCurrency(Mtot
                                  + mimpu, 2)
       Case Is < 1000
          ms = "  " & FormatCurrency(Mtot
                                  + mimpu, 2)
       Case Else
            ms = FormatCurrency(Mtot +
                                  mimpu, 2)
       End Select
       campo = ms
Case Is = "total en letra"
       campo = NumALetras(Mtotl)
Case Is = "total+impuesto en letra"
            campo = NumALetras(Mtotl +
                          mimpu)
       Case Is = "total antes iva"
    Select Case Mtotaiva
    Case Is < 10
        ms = "    " & FormatCurrency(Mtotaiva, 2)
    Case Is < 100
         ms = "   " & FormatCurrency(Mtotaiva, 2)
    Case Is < 1000
        ms = "  " & FormatCurrency(Mtotaiva, 2)
    Case Else
         ms = FormatCurrency(Mtotaiva, 2)
    End Select
    campo = ms
Case Is = "iva"
    Select Case Mtiva
    Case Is < 10
         ms = "    " & FormatCurrency(Mtiva, 2)
    Case Is < 100
         ms = "   " & FormatCurrency(Mtiva, 2)
    Case Is < 1000
         ms = "  " & FormatCurrency(Mtiva, 2)
    Case Else
         ms = FormatCurrency(Mtiva, 2)
    End Select
    campo = ms
Case Is = "impuesto"
     Select Case mimpu
     Case Is < 10
        ms = "    " & FormatCurrency(mimpu, 2)
     Case Is < 100
        ms = "   " & FormatCurrency(mimpu, 2)
     Case Is < 1000
        ms = "  " & FormatCurrency(mimpu, 2)
     Case Else
        ms = FormatCurrency(mimpu, 2)
     End Select
     campo = ms
 Case Is = "descuento monto"
       Select Case mdesc
       Case Is < 10
          ms = "    " & FormatCurrency(mdesc, 2)
       Case Is < 100
           ms = "   " & FormatCurrency(mdesc, 2)
       Case Is < 1000
           ms = "  " & FormatCurrency(mdesc, 2)
       Case Else
           ms = FormatCurrency(mdesc, 2)
       End Select
       campo = ms
Case Is = "descuento clave"
        campo = "Descuento  (" & msdesc & "%):"
Case Else
        Mlen = Len(campo)
        campo = Mid(campo, 2, Mlen - 1)
 End Select
```

```
If no_lin > no_ant Then
      Mdes = Space(no_col - 1) & campo
      no_col = 0
      Else
          Mdes = campo
      End If
      If no_lin = no_sig Then
          lent = no_olc - no_col
          Mdes = Mid(Mdes, 1, lent)
          lenr = Len(Mdes)
         If lenr < lent Then Mdes = Mdes +
                              Space(lent - lenr)
      End If
      If no_lin < no_sig Then
         mlin = mlin & Mdes & Chr(10)
      Else
         mlin = mlin & Mdes
      End If
      .MoveNext
   Loop
   End With
   '
   '   desprendible
   '
   mtip = "S"
   Set prec_forma =
          pdat_comandas.OpenRecordset(
          "select * from cheq_ult where tipo = '"
           & mtip & "'")
    With prec_forma
       If .RecordCount > 0 Then
       .MoveFirst
       mbin = 0
       Do While Not .EOF
          no_lin = .no_lin
          no_col = .no_col
          campo = .campo
          Mband = 0
          If Mid(campo, 1, 1) = "*" Then
             campo = Mid(campo, 2,
                          Len(campo) - 1)
          Else
             Select Case campo
             Case Is = "fecha"
                campo = Date
             Case Is = "folio"
                campo = mscheq
             Case Is = "total"
                campo = FormatCurrency(Mtot,
                                       2)
             Case Is = "total en letra"
                campo = NumALetras(Mtot)
             End Select
          End If
          Mlen = Len(campo)
          msigue = 1
           If mbin = 0 Then
              lin0 = no_lin
              col0 = no_col - 1
          col1 = col0
          mlen0 = Mlen
          mlen1 = mlen0
          mbin = 1
       Else
          If no_lin = lin0 Then
             col0 = no_col - 1 - col1 - mlen1
             col1 = no_col - 1
             mlen1 = Mlen
          Else
             col0 = no_col - 1
             col1 = col0
             lin0 = no_lin
             mlen0 = Mlen
             mlen1 = mlen0
             msigue = 0
          End If
       End If
       If mini < no_lin Then
          inda = mini
          For k = inda To no_lin - 1
              mlin = mlin & " " & Chr(10)
              mini = mini + 1
          Next
       End If
       mlin = mlin & Space(col0) & campo
       If msigue = 0 Then
          mlin = mlin & Chr(10)
          mini = mini + 1
       End If
       .MoveNext
    Loop
    mlin = mlin & Chr(10)
    mini = mini + 1
End If
End With
Printer.Print mlin
End If
Printer.EndDoc
'
' Alta del cheque
'
    With prec_cheques
       .AddNew
       pollin = FrmBuscaClientes.Txt(1).Text
       mimpu = IIf(IsEmpty(mimpu), 0, mimpu)
       .Fields("num_cheque") = mcheq
       .Fields("monto") = Round(Mtot, 2)
       .Fields("iva") = Round(Mtiva, 2)
       .Fields("impuesto") = IIf(IsEmpty(mimpu), 0,
                                 mimpu)
       .Fields("status") = "C"
       .Fields("mesa_cheque") = mmesa
```

```
    If mcual Then
        If mddi Then
         .Fields("descuento") = Round(mdesc, 2)
      Else
         .Fields("descuento") = Round(mdesc /
                              ((100 + Miva) / 100), 2)
    End If
Else
        .Fields("descuento") = Round(mdesc, 2)
 End If
 If Bandera_AutorizaImpresion = True Then
       .Fields("Autoriza") = Mautoriza
       Bandera_AutorizaImpresion = False
       ImpCta.LblDerechotes.Caption = ""
 End If
    .Fields("Usuario") =
     Frmderechos.Label1(38).Caption
    .Fields("nopersonas") = mnper
    .Fields("hora_cierre") = Time()
    .Fields("hora_inic") = mhini
    .Fields("Num_factura") = Null
    .Fields("Num_ticket") = Null
    If mmesa = "VD" Then
        holita = FrmBuscaClientes.Txt(0).Text
        .Fields("cliente") =
                    FrmBuscaClientes.Txt(0).Text
    End If
    .Update
End With
'
' Aplica el pago en cxc en venta a domicilio
'
If mmesa = "VD" Then
    With prec_cobros
        If Not AplicoPag Then
           AplicoPag = True
           .AddNew
           .Fields("f_pago") = "cuentas por
                               cobrar"
           .Fields("Importe") = Mtot + Mivt +
                               mimpu
           .Fields("num_cheque") = mcheq
           .Fields("fecha") = Date
           .Fields("hora") = Time()
           Mconspag = !id_cobro
           .Update
        Else
           .Index = "id_cobro"
           .Seek "=", Mconspag
           .Edit
           .Fields("num_cheque") = mcheq
           .Update
        End If
     End With
     With prec_paramet
        .Edit
        .Fields("ticket") = !Ticket + 1
        .Update
     End With
  End If
  '
  ' Cierra tablas
  '
  prec_cheques.Close
  prec_cobros.Close
  prec_good.Close
  Exit Sub
  '
NOSEIMP:
  MsgBox "favor de revisar que la impresora esté :"
& vbCr & vbCr & "   - prendida" & vbCr & vbCr & "
- en linea" & vbCr & vbCr & "   - ó conectada al cpu"
& vbCr
  If mmesa = "VD" Or mmesa = "VC" Then
     ImpCta.Command1.Visible = True
     ImpCta.Command1.Enabled = True
  Else
     ImpCta.Command2.Visible = True
     ImpCta.Command2.Enabled = True
  End If
End Sub
```

A.II Código muestra para el control de permisos de usuarios.

```
Set Db = OpenDatabase("infocaja.mdb")
Comando_SQL$ = "Select * " _
          & "From  Derecho inner join Usuario " _
          & "On    Derecho.Cve_Usuario = Usuario.Cve_Usuario " _
          & "Where Usuario.nombre = '" & Text1.Text & "'" _
          & "And   Usuario.cve_usuario = '" & Text2.Text & "'"
 Set Rst = Db.OpenRecordset(Comando_SQL$, dbOpenDynaset)
 With Rst
   'operacion, reservacion d_2
   If !d_2 Then
     LstOp(1).AddItem "reservacion"
   Else
     LstOp(0).AddItem "reservacion"
   End If
   'operacion, reservacion d_3
   If !d_3 Then
     LstOp(1).AddItem "liberar mesa"
   Else
     LstOp(0).AddItem "liberar mesa"
   End If
   'operacion, reservacion d_4

If !d_4 Then
      LstOp(1).AddItem "abrir mesa"
Else
      LstOp(0).AddItem "abrir mesa"
End If
'operacion, reservacion d_5
If !D_5 Then
     LstOp(1).AddItem "cambio de mesa"
Else
      LstOp(0).AddItem "cambio de mesa"
End If
'operacion, reservacion d_6
If !d_6 Then
      LstOp(1).AddItem "cancelar mesa"
Else
      LstOp(0).AddItem "cancelar mesa"
End If
'operacion, reservacion d_7
If !d_7 Then
      LstOp(1).AddItem "alta de comanda"
Else
      LstOp(0).AddItem "alta de comanda"
End If
'operacion, reservacion d_8
If !d_8 Then
      LstOp(1).AddItem "cancelar comanda"
Else
      LstOp(0).AddItem "cancelar comanda"
End If
'operacion, reservacion d_9
If !d_9 Then
      LstOp(1).AddItem "cortesia"
Else
      LstOp(0).AddItem "cortesia"
End If
'operacion, reservacion d_10
If !d_10 Then
      LstOp(1).AddItem "pagar mesa"
Else
      LstOp(0).AddItem "pagar mesa"
End If
'operacion, reservacion d_11
If !d_11 Then
      LstOp(1).AddItem "aplicacion de
                        descuentos"
Else
      LstOp(0).AddItem "aplicacion de
                        descuentos"
End If
'operacion, reservacion d_12
If !d_12 Then
      LstOp(1).AddItem "imprimir cuenta"
Else
      LstOp(0).AddItem "imprimir cuenta"
End If
'operacion, vistas x status d_39
If !d_39 Then
      LstOp(1).AddItem "vistas x status"
Else
      If FrmUsuarios.Nel = False Then
          LstOp(0).AddItem "vistas x status"
      End If
```

```
End If
.Close
 End With
 Db.Close
 PrimeraVezOp = False
 End If

'Habilitar
If LstOp(0).ListCount = 0 Then
        Command1.Enabled = False
Else
        Command1.Enabled = True
End If
If LstOp(1).ListCount = 0 Then
        Command3.Enabled = False
Else
        Command3.Enabled = True
End If

Case "catalogos"
   LstOp(0).Visible = False
   LstOp(1).Visible = False
   LstCat(0).Visible = True
   LstCat(1).Visible = True
   LstRep(0).Visible = False
   LstRep(1).Visible = False
   LstOt(0).Visible = False
   LstOt(1).Visible = False
   LstVen(0).Visible = False
   LstVen(1).Visible = False
   If PrimeraVezCat Then
      ChDir App.Path
      Set Db = OpenDatabase("proto.mdb")
      Comando_SQL$ = "Select * " _
          & "From  Derecho inner join Usuario " _
          & "On    Derecho.Cve_Usuario =
                   Usuario.Cve_Usuario " _
          & "Where Usuario.nombre = '" &
                   Text1.Text & "'" _
          & "And   Usuario.cve_usuario = '" &
                  Text2.Text & "'"
        Set Rst = Db.OpenRecordset( _
              Comando_SQL$, dbOpenDynaset)
        With Rst
          'catalogos, parametros d_14
          If !d_14 Then
             LstCat(1).AddItem "parametros"
          Else
       LstCat(0).AddItem "parametros"
     End If
     'catalogos, mesas d_15
     If !d_15 Then
        LstCat(1).AddItem "mesas"
     Else
        LstCat(0).AddItem "mesas"
     End If
     'catalogos, meseros d_16
     If !d_16 Then
        LstCat(1).AddItem "meseros"
     Else
        LstCat(0).AddItem "meseros"
     End If
     'catalogos, platillos d_17
     If !d_17 Then
        LstCat(1).AddItem "platillos"
     Else
       LstCat(0).AddItem "platillos"
     End If
     'catalogos, grupos d_18
     If !d_18 Then
        LstCat(1).AddItem "grupos"
     Else
        LstCat(0).AddItem "grupos"
     End If
     'catalogos, modificadores d_19
     If !d_19 Then
        LstCat(1).AddItem "modificadores"
     Else
        LstCat(0).AddItem "modificadores"
     End If
     'catalogos, formas de pago d_20
     If !d_20 Then
        LstCat(1).AddItem "formas de pago"
     Else
        LstCat(0).AddItem "formas de pago"
     End If
     'catalogos, unidades d_21
     If !d_21 Then
        LstCat(1).AddItem "unidades"
     Else
        LstCat(0).AddItem "unidades"
     End If
     'catalogos, usuarios d_22
     If !d_22 Then
        LstCat(1).AddItem "usuarios"
     Else
        LstCat(0).AddItem "usuarios"
     End If
     'catalogos, clientes d_47
     If !d_47 Then
        LstCat(1).AddItem "clientes"
     Else
       LstCat(0).AddItem "clientes"
     End If
      'catalogos, colonias d_48
      If !d_48 Then
          LstCat(1).AddItem "colonias"
      Else
           LstCat(0).AddItem "colonias"
      End If
      'catalogos, Tipos de grupo d_54
      If !d_54 Then
```

```
            LstCat(1).AddItem "tipos de grupo"
        Else
            LstCat(0).AddItem "tipos de grupo"
        End If
        .Close
    End With
    Db.Close
    PrimeraVezCat = False
    End If

    'Habilitar
    If LstCat(0).ListCount = 0 Then
        Command1.Enabled = False
    Else
        Command1.Enabled = True
    End If
    If LstCat(1).ListCount = 0 Then
        Command3.Enabled = False
    Else
        Command3.Enabled = True
    End If
Case "reportes"
    LstOp(0).Visible = False
    LstOp(1).Visible = False
    LstCat(0).Visible = False
    LstCat(1).Visible = False
    LstRep(0).Visible = True
    LstRep(1).Visible = True
    LstOt(0).Visible = False
    LstOt(1).Visible = False
    LstVen(0).Visible = False
    LstVen(1).Visible = False
    If PrimeraVezRep Then
        ChDir App.Path
        Set Db = OpenDatabase("proto.mdb")
        Comando_SQL$ = "Select * " _
          & "From  Derecho inner join Usuario " _
          & "On    Derecho.Cve_Usuario =
                        Usuario.Cve_Usuario " _
          & "Where Usuario.nombre = '"  _
          & Text1.Text & "' _
          & "And   Usuario.cve_usuario = '" _
          & Text2.Text & "'"
        Set Rst = Db.OpenRecordset( _
              Comando_SQL$, dbOpenDynaset)
        With Rst


    'reportes, cierre de caja d_24
    If !d_24 Then
        LstRep(1).AddItem "cierre de caja"
    Else
        LstRep(0).AddItem "cierre de caja"
    End If
    'reportes, formas de pago d_25
    If !d_25 Then
        LstRep(1).AddItem "formas de pago"
    Else
        LstRep(0).AddItem "formas de pago"
    End If
    'reportes, por grupo d_26
    If !d_26 Then
        LstRep(1).AddItem "por grupo"
    Else
        LstRep(0).AddItem "por grupo"
    End If
    'reportes, por platillo d_27
    If !d_27 Then
        LstRep(1).AddItem "por platillo"
    Else
        LstRep(0).AddItem "por platillo"
    End If
    'reportes, por mesero d_28
    If !d_28 Then
        LstRep(1).AddItem "por mesero"
    Else
        LstRep(0).AddItem "por mesero"
    End If
    'reportes, por propinas d_29
    If !d_29 Then
        LstRep(1).AddItem "por propina"
    Else
        LstRep(0).AddItem "por propina"
    End If
    'reportes, mesas canceladas d_30
    If !d_30 Then
        LstRep(1).AddItem "mesas canceladas"
    Else
        LstRep(0).AddItem "mesas canceladas"
    End If
    'reportes, mesas asignadas d_31
    If !d_31 Then
        LstRep(1).AddItem "mesas asignadas"
    Else
        LstRep(0).AddItem "mesas asignadas"
    End If
    'reportes, tira de auditoria d_32
    If !d_32 Then
        LstRep(1).AddItem "tira de auditoria"
    Else
        LstRep(0).AddItem "tira de auditoria"
    End If


    'reportes, platillos cancelados d_41
    If !d_41 Then
        LstRep(1).AddItem "platillos cancelados"
    Else
        LstRep(0).AddItem "platillos cancelados"
    End If
    'reportes de Ventas de platillos por mesero
    If !d_56 Then
```

```
            LstRep(1).AddItem "platillos por mesero"
        Else
            LstRep(0).AddItem "platillos por mesero"
        End If
        .Close
    End With
    Db.Close
    PrimeraVezRep = False
End If

'Habilitar
If LstRep(0).ListCount = 0 Then
        Command1.Enabled = False
Else
        Command1.Enabled = True
End If
    If LstRep(1).ListCount = 0 Then
            Command3.Enabled = False
    Else
            Command3.Enabled = True
    End If
    Case "otros"
        LstOp(0).Visible = False
        LstOp(1).Visible = False
        LstCat(0).Visible = False
        LstCat(1).Visible = False
        LstRep(0).Visible = False
        LstRep(1).Visible = False
        LstOt(0).Visible = True
        LstOt(1).Visible = True
        LstVen(0).Visible = False
        LstVen(1).Visible = False
        If PrimeraVezOt Then
            ChDir App.Path
            Set Db = OpenDatabase("proto.mdb")
            Comando_SQL$ = "Select * " _
              & "From  Derecho inner join Usuario " _
              & "On    Derecho.Cve_Usuario =
                       Usuario.Cve_Usuario " _
              & "Where Usuario.nombre = '" &
                        Text1.Text & "'" _
              & "And   Usuario.cve_usuario = '" &
                        Text2.Text & "'"
            Set Rst = Db.OpenRecordset( _
                  Comando_SQL$, dbOpenDynaset)



      With Rst
        'otros, respaldo d_34
        If !d_34 Then
            LstOt(1).AddItem "respaldo"
        Else
            LstOt(0).AddItem "respaldo"
        End If
        'otros, cierre d_35
        If !d_35 Then
            LstOt(1).AddItem "cierre"
        Else
            LstOt(0).AddItem "cierre"
        End If
        'otros, inicializar d_36
        If !d_36 Then
            LstOt(1).AddItem "inicializar"
        Else
            LstOt(0).AddItem "inicializar"
        End If
        'otros, restaurar d_37
        If !d_37 Then
            LstOt(1).AddItem "restaurar"
        Else
            LstOt(0).AddItem "restaurar"
        End If
        'otros, cambio de forma de pago d_40
        If !d_40 Then
            LstOt(1).AddItem "cambio de forma de
                              pago"
        Else
            LstOt(0).AddItem "cambio de forma de
                              pago"
        End If
        'otros, formato de factura d_42
        If !d_42 Then
            LstOt(1).AddItem "formato de factura"
        Else
            LstOt(0).AddItem "formato de factura"
        End If
        'otros, formato de cheque d_43
        If !d_43 Then
            LstOt(1).AddItem "formato de cheque"
        Else
            LstOt(0).AddItem "formato de cheque"
        End If
        'otros, facturacion d_44
        If !d_44 Then
            LstOt(1).AddItem "facturacion"
        Else
            LstOt(0).AddItem "facturacion"
        End If
        'otros, retiro de efectivo d_45
        If  !d_45 Then
            LstOt(1).AddItem "retiro de efectivo"
        Else
            LstOt(0).AddItem "retiro de efectivo"
        End If
        'otros, corte de fondos d_46
        If !d_46 Then
            LstOt(1).AddItem "corte de fondos"
        Else
            LstOt(0).AddItem "corte de fondos"
        End If
        'otros, Retiros varios d_52
```

```
        If !d_52 Then
            LstOt(1).AddItem "retiros varios"
        Else
            LstOt(0).AddItem "retiros varios"
        End If
        'otros, corte de retiros varios d_53
        If !d_53 Then
            LstOt(1).AddItem "corte de retiros
                                varios"
        Else
            LstOt(0).AddItem "corte de retiros
                                varios"
        End If
        .Close
    End With
    Db.Close
    PrimeraVezOt = False
End If

'Habilitar
If LstOt(0).ListCount = 0 Then
    Command1.Enabled = False
Else
    Command1.Enabled = True
End If
If LstOt(1).ListCount = 0 Then
    Command3.Enabled = False
Else
    Command3.Enabled = True
End If
Case "tipos de venta"
    LstOp(0).Visible = False
    LstOp(1).Visible = False
    LstCat(0).Visible = False
    LstCat(1).Visible = False
    LstRep(0).Visible = False
    LstRep(1).Visible = False
    LstOt(0).Visible = False
    LstOt(1).Visible = False
    LstVen(0).Visible = True
    LstVen(1).Visible = True
    If PrimeraVezVen Then
        ChDir App.Path
        Set Db = OpenDatabase("proto.mdb")
        Comando_SQL$ = "Select * " _
        & "From  Derecho inner join Usuario " _
        & "On    Derecho.Cve_Usuario =
                 Usuario.Cve_Usuario " _
 & "Where Usuario.nombre = '" & Text1.Text & "'"
& "And    Usuario.cve_usuario = '" & Text2.Text &
"'"

  Set Rst = Db.OpenRecordset(Comando_SQL$,
dbOpenDynaset)
With Rst
   If VSalon Then
      'tventas, venta salon d_49
      If !d_49 Then
         LstVen(1).AddItem "venta salon"
      Else
         LstVen(0).AddItem "venta salon"
      End If
   End If
   If VCalle Then
      'tventas, venta calle d_50
      If !d_50 Then
         LstVen(1).AddItem "venta calle"
      Else
          LstVen(0).AddItem "venta calle"
      End If
   End If
 If VDom Then
     'tventas, venta domicilio d_51
      If !d_51 Then
          LstVen(1).AddItem "venta domicilio"
       Else
          LstVen(0).AddItem "venta domicilio"
       End If
 End If
.Close
End With
Db.Close
PrimeraVezVen = False
End If

'Habilitar
If LstVen(0).ListCount = 0 Then
    Command1.Enabled = False
Else
     Command1.Enabled = True
 End If
 If LstVen(1).ListCount = 0 Then
```

… entre otras líneas

A.III Código muestra para el código del pintado de mesas viejo.

```
Private Sub Lb_Mesa_Click(Index As Integer)
   On Error Resume Next
   Mesa_Activa = Mesa(Index).Caption
   Repintado_de_Mesas Mesa(Index).Caption, Val(Index)
   Select Case Shape1(Index).BackColor
   Case &H80000005 ' Mesa en Blanco
      If Frmderechos.Label1(4).Caption = "no" Then
         FrmTeclado.FmeAutorizacion.Visible = True
         FrmTeclado.Caption = "Necesita de Autorizacion para realizar esta acción"
         FrmTeclado.LblProblema.Caption = "D_4"
      End If
      frmMain.Enabled = False
      FrmTeclado.LblQuienLlama.Caption = "operacion"
      FrmTeclado.LblAComandas(1).Caption = Mesa(Index).Caption
      If frmMain.sbStatusBar.Panels(2).Tag <> "" Then
         FrmTeclado.LblAComandas(5).Caption = frmMain.sbStatusBar.Panels(2).Tag
         FrmTeclado.LblAComandas(6).Caption = frmMain.sbStatusBar.Panels(1).Tag
      End If
      QuienHabla = "FrmOperacion"
      Load FrmTeclado
      FrmTeclado.Show vbModal, frmMain
   Case &HC0FFC0 'Mesa en Verde
      frmMain.Enabled = False
      FrmDesicionMAbierta.LblDMesa.Caption = Mesa(Index).Caption
      Load FrmDesicionMAbierta
      FrmDesicionMAbierta.Show
   Case &H80FF&    'Mesa en Naranja
      FrmDesicionCheque.LblDMesa = Mesa(Index).Caption
      frmMain.Enabled = False
      Load FrmDesicionCheque
      FrmDesicionCheque.Show
   Case &HFF8080 ' Mesa en Azul
      frmMain.Enabled = False
      FrmDesicionCerrada.LblDMesa.Caption = Mesa(Index).Caption
      Load FrmDesicionCerrada
      FrmDesicionCerrada.Show
   Case &HFFFF& ' Mesa en Amarillo
      ChDir App.Path
      Set PDAT_BaseInfoset = OpenDatabase("Infoset.mdb")
      comanmesita$ = "select * from Mesa where cve_mesa ='" & Mesa(Index).Caption & "'"
      Set PREC_TabMesa = PDAT_BaseInfoset.OpenRecordset(comanmesita$, dbOpenDynaset)
      PREC_TabMesa.Edit
      PREC_TabMesa!Status = "V"
      PREC_TabMesa!VieneDe = ""
      PREC_TabMesa.Update
      PREC_TabMesa.Close
      PDAT_BaseInfoset.Close
      Shape1(Index).BackColor = &H80000005
      Pintar_Mesas
   Case &HFF& ' Mesa en Rojo
      frmMain.Enabled = False
      Load FrmDesicion
```

```
      FrmDesicion.Show
      FrmDesicion.LblDMesa.Caption = Mesa(Index).Caption
   End Select
End Sub


Public Sub Pintar_Mesas()
On Error Resume Next
ChDir App.Path
Set PDAT_BaseInfosetPm = OpenDatabase("Infoset.mdb")
If Si_es_Mesero = True Then
   If frmOperacion.PBOL_OpDinamica Then
      Comando_Mesitas$ = "SELECT Mesa.Cve_Mesa, Mesa.Mesero, Mesa.Status" & _
                  " From Mesa WHERE (Mesa.Mesero = '" &
Inner_Mesero(Frmderechos.Label1(38).Caption) & "' ) " & _
                  "OR (Mesa.Status = 'V' ) order by " & OrdenMesas & " val(mesa.cve_mesa),
CVar(mesa.cve_mesa)"
   Else
      Comando_Mesitas$ = "SELECT Mesa.Cve_Mesa, Mesa.Mesero, Mesa.Status" & _
                  " From Mesa WHERE " & NumeroTurnos & "order by " & OrdenMesas & "
val(mesa.cve_mesa), CVar(mesa.cve_mesa)"
   End If
Else
   Comando_Mesitas$ = "select * from mesa " & PSTR_QueSEs & "order by val(mesa.cve_mesa),
CVar(mesa.cve_mesa)"
End If
For i = 0 To 23
   frmOperacion.Lb_Mesa(i).Caption = ""
   frmOperacion.Mesa(i).Caption = ""
   frmOperacion.Shape1(i).BackColor = &H80000005
   frmOperacion.LblDExtras(i).Caption = ""
   frmOperacion.LblChkOut(i).Caption = ""
Next i
Set PREC_TabMesitas = PDAT_BaseInfosetPm.OpenRecordset(Comando_Mesitas$, dbOpenDynaset)
With PREC_TabMesitas
   If .RecordCount > 0 Then
      .MoveLast
      Mesitas = .RecordCount
      .MoveFirst
      If Mesitas <= 24 Then
         For m = 0 To Mesitas - 1
            frmOperacion.Lb_Mesa(m).Caption = ""
            frmOperacion.Mesa(m).Caption = !Cve_Mesa
            If PREC_TabMesitas!Status = "A" Then
               If TienePersonas Then
                  SQLDatos = "SELECT Comanda.Id_Comanda, Comanda.Personas, " & _
                     "Mesa.Capacidad FROM Comanda INNER JOIN Mesa ON " & _
                     "Comanda.Cve_Mesa = Mesa.Cve_Mesa WHERE Mesa.Cve_Mesa " & _
                     "= '" & frmOperacion.Mesa(m).Caption & "' AND Comanda.Status = 'A'"
                  Set PREC_DatosExtras = PDAT_BaseInfosetPm.OpenRecordset(SQLDatos)
                  frmOperacion.LblDExtras(m).Caption = PREC_DatosExtras!Personas & _
                     "/" & PREC_DatosExtras!Capacidad
               Else
                  SQLDatos = "select Id_Comanda, Cve_Mesa, Personas, Status from Comanda where " & _
                     "Cve_Mesa = '" & frmOperacion.Mesa(m).Caption & "' and " & _
                     "Status = 'A' order by Id_Comanda"
```

```
                Set PREC_DatosExtras = PDAT_BaseInfosetPm.OpenRecordset(SQLDatos)
                frmOperacion.LblDExtras(m).Caption = PREC_DatosExtras!Personas
              End If
              PREC_DatosExtras.Close
              frmOperacion.Shape1(m).BackColor = &HC0FFC0
            ElseIf PREC_TabMesitas!Status = "V" Then
              frmOperacion.Shape1(m).BackColor = &H80000005
            ElseIf PREC_TabMesitas!Status = "R" Then
              frmOperacion.Shape1(m).BackColor = &HFF&
            ElseIf PREC_TabMesitas!Status = "M" Then
              If TienePersonas Then
                SQLDatos = "SELECT Comanda.Id_Comanda, Comanda.CheckOut, Comanda.Personas, " &
_
                  "Mesa.Capacidad FROM Comanda INNER JOIN Mesa ON " & _
                  "Comanda.Cve_Mesa = Mesa.Cve_Mesa WHERE Mesa.Cve_Mesa " & _
                  "= '" & frmOperacion.Mesa(m).Caption & "' AND (Comanda.Status = 'I' " & _
                  "OR Comanda.Status = 'N' OR Comanda.Status = 'A')"
                Set PREC_DatosExtras = PDAT_BaseInfosetPm.OpenRecordset(SQLDatos)
                Capa = PREC_DatosExtras!Capacidad
                PREC_DatosExtras.MoveLast
                frmOperacion.LblChkOut(m).Caption = Format(PREC_DatosExtras!CheckOut, "HH:mm")
                frmOperacion.LblDExtras(m).Caption = PREC_DatosExtras!Personas & _
                  "/" & Capa
              Else
                SQLDatos = "select Id_Comanda, Cve_Mesa, CheckOut, Personas, Status from Comanda
where " & _
                  "Cve_Mesa = '" & frmOperacion.Mesa(m).Caption & "' and " & _
                  "(Status = 'I' or Status = 'N' OR Comanda.Status = 'A') order by Id_Comanda"
                Set PREC_DatosExtras = PDAT_BaseInfosetPm.OpenRecordset(SQLDatos)
                PREC_DatosExtras.MoveLast
                frmOperacion.LblChkOut(m).Caption = Format(PREC_DatosExtras!CheckOut, "HH:mm")
                frmOperacion.LblDExtras(m).Caption = PREC_DatosExtras!Personas
              End If
              PREC_DatosExtras.Close
              frmOperacion.Shape1(m).BackColor = &H80FF&
            ElseIf PREC_TabMesitas!Status = "O" Then
              frmOperacion.Shape1(m).BackColor = &HFFFF&
            ElseIf PREC_TabMesitas!Status = "C" Then
              SQLDatos = "select Num_Cheque, Mesa_Cheque, Hora_Cierre, Status from Cheque where " & _
                  "Mesa_Cheque = '" & frmOperacion.Mesa(m).Caption & "' and " & _
                  "Status = 'C' order by Num_Cheque"
              Set PREC_DatosExtras = PDAT_BaseInfosetPm.OpenRecordset(SQLDatos)
              frmOperacion.LblDExtras(m).Caption = Format(PREC_DatosExtras!hora_cierre, "HH:mm")
              PREC_DatosExtras.Close
              frmOperacion.Shape1(m).BackColor = &HFF8080
            End If
            .MoveNext
          Next m
          frmOperacion.Command2.Enabled = False
        ElseIf Mesitas > 24 Then
          For m = 0 To 23
            frmOperacion.Lb_Mesa(m).Caption = ""
            frmOperacion.Mesa(m).Caption = !Cve_Mesa
            If PREC_TabMesitas!Status = "A" Then
              If TienePersonas Then
                SQLDatos = "SELECT Comanda.Id_Comanda, Comanda.Personas, " & _
```

```
                    "Mesa.Capacidad FROM Comanda INNER JOIN Mesa ON " & _
                    "Comanda.Cve_Mesa = Mesa.Cve_Mesa WHERE Mesa.Cve_Mesa " & _
                    "= '" & frmOperacion.Mesa(m).Caption & "' AND Comanda.Status = 'A'"
                Set PREC_DatosExtras = PDAT_BaseInfosetPm.OpenRecordset(SQLDatos)
                frmOperacion.LblDExtras(m).Caption = PREC_DatosExtras!Personas & _
                    "/" & PREC_DatosExtras!Capacidad
            Else
                SQLDatos = "select Id_Comanda, Cve_Mesa, Personas, Status from Comanda where " & _
                    "Cve_Mesa = '" & frmOperacion.Mesa(m).Caption & "' and " & _
                    "Status = 'A' order by Id_Comanda"
                Set PREC_DatosExtras = PDAT_BaseInfosetPm.OpenRecordset(SQLDatos)
                frmOperacion.LblDExtras(m).Caption = PREC_DatosExtras!Personas
            End If
            PREC_DatosExtras.Close
            frmOperacion.Shape1(m).BackColor = &HC0FFC0
        ElseIf PREC_TabMesitas!Status = "V" Then
            frmOperacion.Shape1(m).BackColor = &H80000005
        ElseIf PREC_TabMesitas!Status = "R" Then
            frmOperacion.Shape1(m).BackColor = &HFF&
        ElseIf PREC_TabMesitas!Status = "M" Then
            If TienePersonas Then
                SQLDatos = "SELECT Comanda.Id_Comanda, Comanda.CheckOut, Comanda.Personas, " &
_
                    "Mesa.Capacidad FROM Comanda INNER JOIN Mesa ON " & _
                    "Comanda.Cve_Mesa = Mesa.Cve_Mesa WHERE Mesa.Cve_Mesa " & _
                    "= '" & frmOperacion.Mesa(m).Caption & "' AND (Comanda.Status = 'I' " & _
                    "OR Comanda.Status = 'N' OR Comanda.Status = 'A')"
                Set PREC_DatosExtras = PDAT_BaseInfosetPm.OpenRecordset(SQLDatos)
                Capa = PREC_DatosExtras!Capacidad
                PREC_DatosExtras.MoveLast
                frmOperacion.LblChkOut(m).Caption = Format(PREC_DatosExtras!CheckOut, "HH:mm")
                frmOperacion.LblDExtras(m).Caption = PREC_DatosExtras!Personas & _
                    "/" & Capa
            Else
                SQLDatos = "select Id_Comanda, Cve_Mesa, CheckOut, Personas, Status from Comanda
where " & _
                    "Cve_Mesa = '" & frmOperacion.Mesa(m).Caption & "' and " & _
                    "(Status = 'I' or Status = 'N' OR Comanda.Status = 'A') order by Id_Comanda"
                Set PREC_DatosExtras = PDAT_BaseInfosetPm.OpenRecordset(SQLDatos)
                PREC_DatosExtras.MoveLast
                frmOperacion.LblChkOut(m).Caption = Format(PREC_DatosExtras!CheckOut, "HH:mm")
                frmOperacion.LblDExtras(m).Caption = PREC_DatosExtras!Personas
            End If
            PREC_DatosExtras.Close
            frmOperacion.Shape1(m).BackColor = &H80FF&
        ElseIf PREC_TabMesitas!Status = "O" Then
            frmOperacion.Shape1(m).BackColor = &HFFFF&
        ElseIf PREC_TabMesitas!Status = "C" Then
            SQLDatos = "select Num_Cheque, Mesa_Cheque, Hora_Cierre, Status from Cheque where " & _
                    "Mesa_Cheque = '" & frmOperacion.Mesa(m).Caption & "' and " & _
                    "Status = 'C' order by Num_Cheque"
            Set PREC_DatosExtras = PDAT_BaseInfosetPm.OpenRecordset(SQLDatos)
            frmOperacion.LblDExtras(m).Caption = Format(PREC_DatosExtras!hora_cierre, "HH:mm")
            PREC_DatosExtras.Close
            frmOperacion.Shape1(m).BackColor = &HFF8080
        End If
```

```
            .MoveNext
          Next m
          frmOperacion.Command2.Enabled = True
          frmOperacion.PaginaMesitas = 1
        End If
        frmOperacion.Mesitas = Mesitas
        PREC_TabMesitas.Close
        PDAT_BaseInfosetPm.Close
    Else
        If Not Si_es_Mesero Then
          MsgBox Laultimaynosvamos(PSTR_QueSEs)
          CualStatus_Va
          PREC_TabMesitas.Close
          PDAT_BaseInfosetPm.Close
          If Lobo_Dudoso < 4 Then
            Lobo_Dudoso = Lobo_Dudoso + 1
            Pintar_Mesas
          End If
        End If
    End If
End With
    Lobo_Dudoso = 0
    frmOperacion.Command1.Enabled = False
    Invalida_MesaPm
End Sub



Private Sub Repintado_de_Mesas(Meseta As String, What As Integer)
    Dim PDAT_DB As Database
    Dim PREC_RS As Recordset
    ChDir App.Path
    Set PDAT_DB = OpenDatabase("infoset.mdb")
    SQL = "select Cve_Mesa, Status, Mesero from Mesa where Cve_Mesa = '" & Meseta & "'"
    Set PREC_RS = PDAT_DB.OpenRecordset(SQL)
    Select Case PREC_RS!Status
    Case "V"
        Shape1(What).BackColor = &H80000005
    Case "A"
        Shape1(What).BackColor = &HC0FFC0
    Case "M"
        Shape1(What).BackColor = &H80FF&
    Case "C"
        Shape1(What).BackColor = &HFF8080
    Case "O"
        Shape1(What).BackColor = &HFFFF&
    Case "R"
        Shape1(What).BackColor = &HFF&
    End Select
    PREC_RS.Close
    PDAT_DB.Close
End Sub
```

A.IV Código muestra para el pintado de mesas.

```
Private Sub Mesa_Click(Index As Integer, boton As Integer, Shift As Integer, X As Single, Y As Single)
  If boton <> vbLeftButton Then Exit Sub
  Objeto = Index
  mesa_seleccionada = Index
  Select Case mesa(Index).Color
  Case 0  'Mesa reservada                    rojo
  Case 1  'Mesa vacía                        blanco
    Load frmdatosiniciales
    frmdatosiniciales.lblsalon2.Caption = BtnSE(salon).Caption
    frmdatosiniciales.lblmesa2.Caption = mesa(Index).mesa
    frmdatosiniciales.lblmesa2.Tag = mesa(Index).Caption
    frmdatosiniciales.Show vbModal
  Case 2  'Mesa abierta (se esta tomando la orden)      verde claro
    Load frmdatosiniciales2
    frmdatosiniciales2.Show vbModal
  Case 3  'mesa modificada                   naranja
    Load frmdatosiniciales2
    frmdatosiniciales2.Show vbModal
  Case 4  'Mesa pagada                       azul claro
    dbbase.Execute "UPDATE mesas " & _
      "SET status = 'vacia'" & _
      "WHERE idmesa = " & mesa(Index).Caption
    mesa(Index).Color = 1
  Case 6  'Mesa desactivada                  gris

  End Select
End Sub
```

B. Reportes de muestra

B.I Reporte de ventas por grupo.

Operadora Tamaulipas 130 SA de CV
Av. Tamaulipas # 130 Col. Condesa, Condesa, CP 6140
Reporte de Ventas por Grupo al: 15/06/2008

D e s c r i p c i o n	Clave	Monto	%
**** alimentos ****			
*** Sopas	200	$40.00	2.29
*** Aves	600	$75.00	4.29
*** Pescados y Mari	700	$334.00	19.11
*** Paninos y Crois	900	$300.00	17.16
*** Sugerencias	1100	$399.00	22.83
**** Total alimentos		$1,148.00	65.68
**** Gran Total		$1,748.00	

B.II Reporte de ventas por platillo.

Operadora Tamaulipas 130 SA de CV
Av. Tamaulipas # 130 Col. Condesa, Condesa, CP 6140
Reporte de Ventas por Platillo al: 15/06/2008

D e s c r i p c i o n	Clave	Cant	Monto	%
*** alimentos				
** Sopas				
* Sopa Minestrone	202	1	$40.00	4.0%
**Sub-Total Sopas		1	$40.00	4.0%
** Aves				
* Pechuga Corregidora	606	1	$75.00	7.5%
**Sub-Total Aves		1	$75.00	7.5%
** Pescados y Mariscos				
* Filete de Huachinango Fonduta	702	2	$204.00	20.4%
* 1/2 Trucha en ensalada	706	2	$130.00	13.0%
**Sub-Total Pescados y Mariscos		4	$334.00	33.5%
** Paninos y Croissants				
* Pa niño Pobre	905	2	$150.00	15.0%
**Sub-Total Paninos y Croissants		2	$150.00	15.0%
** Sugerencias				
* Camarones al Tequila	1103	3	$399.00	40.0%
**Sub-Total Sugerencias		3	$399.00	40.0%
***Total alimentos		11	$998.00	100.0%
**** Gran Total			$998.00	

B.III Reporte de tira de auditoría.

```
  Operadora Tamaulipas 130 SA de CV
Av. Tamaulipas # 130 Col. Condesa,
Tira de Auditoria, del: 15/06/2008
□Comanda  Cant  Producto       Monto
Cant  Producto       Monto    Canc.
-----------------------------------
#Comanda: 209
    1 Bife de Chori   $150.00
      con pure de p
      con papa horn
#Comanda: 210
    2 Bife de Chori   $300.00
      con papa fran
      con chiles to
-----------------------------------
#Ch: 2207   Mesa:12
#Comanda: 211
    1 Fresas con Cr    $40.00
      con helado
-----------------------------------
#Ch: 2210   Mesa:22       pagado
#Comanda: 212
    2 Pa niño Pobre   $150.00
    2 1/2 Trucha en   $130.00
-----------------------------------
#Ch: 2211   Mesa:21       no pagado
#Comanda: 215
    2 Pa niño Pobre   $150.00
-----------------------------------
#Ch: 2212   Mesa:22       pagado
#Comanda: 216
    1 Pechuga Corre    $75.00
-----------------------------------
#Ch: 2213   Mesa:23       pagado
#Comanda: 217
    2 Filete de Hua   $204.00
-----------------------------------
#Ch: 2214   Mesa:200      pagado
#Comanda: 218
    1 Sopa Minestro    $40.00
    3 Camarones al    $399.00
      error ... no
-----------------------------------
Cheques Can.: 2206
-----------------------------------
Totales:
Folio Inicial: 2206
Folio Final  : 2214
  #  Vta Bruta  Descuentos Vta Neta
-----------------------------------
C   1
P   4   $998.00     $0.00   $998.00
S   1   $150.00     $0.00   $150.00
T   6 $1,148.00     $0.00 $1,148.00
C = Cancelados
P = Pagados
S = Cerrados pero sin pagar
T = Totales
```

B.IV Nuevo reporte de ventas por grupo.

Reporte de ventas por grupo

Bosque de duraznos 127 piso 14 col. Bosques de las lomas c.p. 11700
deleg. Miguel Hidalgo México D.F. tel: 50-05-00-50 R.F.C.

Folio inicial:	22
Folio final:	30

Clave	Nombre del grupo	Total
6	Aves	$136.00
20	Bebidas Alcoholicas	$90.00
31	Bot. 1/2 Vino Tinto Imp.	$280.00
33	Bot. 1/2 Vino Blanco Imp.	$250.00
36	Almuerzos	$266.00

Total vendido: $1,022.00

B.V Nuevo reporte ventas por platillo.

Reporte ventas por platillo

Bosque de duraznos 127 piso 14 col. Bosques de las lomas c.p. 11700
deleg. Miguel Hidalgo México D.F. tel: 50-05-00-50 R.F.C.

Aves	cantidad	precio	Total
Pechuga al tamarindo	1	$ 72.00	$ 72.00
Pechuga plancha	1	$ 64.00	$ 64.00
	Total por grupo:		$136.00
Bebidas Alcoholicas	**cantidad**	**precio**	**Total**
Anis chinchón dulce	1	$ 42.00	$ 42.00
Teq. cazadores	1	$ 48.00	$ 48.00
	Total por grupo:		$90.00
Bot. 1/2 Vino Tinto Imp.	**cantidad**	**precio**	**Total**
Bot. 1/2 Chianti	1	$ 155.00	$ 155.00
Bot. 1/2 Valpoli	1	$ 125.00	$ 125.00
	Total por grupo:		$280.00
Bot. 1/2 Vino Blanco Imp.	**cantidad**	**precio**	**Total**
Bot. 1/2 Undurraga bco.	1	$ 130.00	$ 130.00
Bot. 1/2 viña Carrasco bco.	1	$ 120.00	$ 120.00
	Total por grupo:		$250.00
Almuerzos	**cantidad**	**precio**	**Total**
Chilaquiles tradicionales	1	$ 43.00	$ 43.00
Enchiladas suizas	3	$ 46.00	$ 138.00
Huevos motuleños	1	$ 45.00	$ 45.00
Omelette dietético	1	$ 40.00	$ 40.00
	Total por grupo:		$266.00
	Total:		$1,022.00

B.VI Nuevo reporte de tira de auditoría.

Tira de auditoría

Bosque de duraznos 127 piso 14 col. Bosques de las lomas c.p. 11700
deleg. Miguel Hidalgo México D.F. tel: 50-05-00-50 R.F.C. eino50628t61

Cheque:	1	Status:	facturado

Comanda:	23	Mesa:	2

Platillo	Cantidad	Monto
Huevos motuleños sin cebolla sin picante	1	$45.00
Bot. 1/2 Valpoli	1	$125.00

Comanda:	25	Mesa:	2

Platillo	Cantidad	Monto
Bot. 1/2 viña Carrasco	1	$120.00

Cheque:	2	Status:	pagado

Comanda:	29	Mesa:	2

Platillo	Cantidad	Monto
Bot. 1/2 Chianti	1	$155.00
Pechuga al tamarindo con papa francesa	1	$72.00

Bibliografía.

Libros.

- Ingeniería del software: un enfoque práctico
  Autor: Roger S. Pressman
  Editorial Mc Graw Hill tercera edición.

- Ingeniería de software
  autor: Richar Fairley
  editorial Mc Graw Hill

- Software Engineering
  Autor Ian Sommerville
  Editorial Pearson – Addison Wesley
  Seventh edition

- Software Engineering a practitioners approach
  Autor Roger S. Pressman
  Editorial Mc Graw Hill international edition
  Sixth edition

- Análisis y diseño de sistemas
  Autor Kendall y Kendall
  Editorial Prentice Hall Hispanoamericana

- MySQL edición especial
  Autor Paul DoBois
  Editorial Prentice may

- MySQL para windows y linux
  Autor César Pérez
  Editorial Alfaomega, Ra-ma

- Fundamentos de bases de datos
  Autores Abraham Silberschatz, Henry F. Korth y S. Sudarshan
  editorial Mc Graw Hill

- Linux edición especial sexta edición
  Autores David Bandel y Robert Napier
  Editorial prentice hall

- Java cómo programar
  Autor Harvey M. Deitel y Paul J. Deitel
  Editorial Pearson – Prentice hall

- Quinta edición.
  Libro visual basic 6, distribuited applications
  Serie MCDS (microsoft certified solution developer)
  Editorial Osborne Mc Graw Hill
  Sin autores.

Direcciones en internet.

- www.mysql.com
- http://es.wikipedia.org/wiki/Industria_del_software
- http://www.monografias.com/trabajos/computacion/computacion.shtml
- http://es.wikibooks.org/wiki/Evoluci%C3%B3n_del_software
- http://www.idg.es/computerworld/articulo.asp?id=165543
- http://www.elrinconcito.com/articulos/Reingenieria/Articulo.htm
- http://www.saber.ula.ve/db/ssaber/Edocs/pubelectronicas/visiongerencial/ano4num1/articulo1.pdf
- www.alarcos.inf-cr.uclm.es/per/fruiz/cur/mso/trans/s6.pdf
- www.gtd.es/website/esp/sobre/
- www.monografias.com/trabajos14

- Resultados de la comparación entre MySQL 4, Oracle 9i, DB2 7.2, SQL Server 2000 y ASE 12.5 de PC Computing:
  http://www.eweek.com/article2/0,3959,293,00.asp