UNIVERSIDAD LASALLISTA BENAVENTE

INGENIERÍA EN COMPUTACIÓN

Con estudios incorporados a la Universidad Nacional Autónoma de México
CLAVE: 8793-16

“Desarrollo de Sistema Multi-Negocio para el Control de Pedidos de Materiales de Construcción”

TESIS

Que para obtener el título de

INGENIERO EN COMPUTACIÓN

Presenta:

FRANCISCO MORENO ARRIAGA

Asesor: ING. ALEJANDRO GUZMÁN ZAZUETA

Celaya, Guanajuato.

AGRADECIMIENTOS PERSONALES.

Mi papá siempre me dijo ¿Cuándo te titulas? Y nunca le hice caso, pero dónde quiera que esté le gustará saber que por fin lo voy a hacer. Mis maestros siempre que me ven me preguntan que para cuándo eso y con el tiempo había adquirido la habilidad de dar excusas cada vez más elaboradas. Para que Rosa María, vea que siempre traté de ser una mejor persona y por supuesto se la dedico a mi mamá que me ha consentido desde el día en que nací.

ÍNDICE.

INTRODUCCIÓN.

INTRODUCCIÓN.

El campo del desarrollo de sistemas debe verse como una industria de vanguardia de rápido crecimiento que se ha extendido primordialmente gracias a los esfuerzos de muchos analistas y programadores que encauzan su energía creativa a la producción de programas o sistemas prácticos y útiles, que representen alternativas reales de solución a las problemáticas particulares de cada industria o de cada área del quehacer humano. Corresponde a estos mismos desarrolladores hacer que las organizaciones que adquieren estas herramientas abran sus mentes hacia el uso de las nuevas tecnologías de la información.

Todos los días, personas comunes, de los más diversos ámbitos realizan una gran variedad de tareas como parte de sus actividades cotidianas, esta rutina en la que inconsciente o involuntariamente se ven inmersos hace que en ocasiones pierdan la perspectiva y sean incapaces de concebir la idea de que existe o puede existir un sistema capaz de completar tales tareas, y no sólo eso, puede hacerlo de forma extraordinariamente mejor, en comparación como se hacia en el pasado, por consiguiente debe comprenderse plenamente que el verdadero propósito de sistema que se ofrece, no es únicamente completar las tares, sino hacer que los usuarios dediquen sus esfuerzos a otras más directivas que operativas y que desde sus diferentes puestos de trabajo, estos esfuerzos se traduzcan en una mayor productividad y traigan consigo beneficios tangibles a sus personas, de manera que procuren una mayor rentabilidad a sus organizaciones.

En lo particular, en el negocio donde se trabajó, cualquier inversión hecha se ve en términos de su rentabilidad, no interesa a los propósitos de este estudio revelar la remuneración recibida, pues el dueño sabe que el interés es exclusivamente académico; por su puesto esto no es de extrañarse, sucede en todas las organizaciones, excepto que en los negocios como éste, el impacto es más inmediato, no obstante, la

iniciativa vino del dueño y del contador de la empresa, y si bien, ellos tienen la responsabilidad de la aprobación económica también tienen a su cargo la determinación de las directrices de la empresa. Durante las múltiples visitas que se sucedieron en el lugar de trabajo, se les exponía en repetidas ocasiones una serie de disertaciones de cómo estarían constituidos los componentes lógicos del sistema, los recursos materiales, respecto al equipo, las impresoras, el mini printer y un pequeño cableado de red, así como los recursos humanos, es decir involucrar a todos los empleados en pro del éxito del sistema, se le hablo también de la inversión requerida para materializar los esfuerzos, los gastos de operación que se iban a necesitar, del margen de ganancias y el periodo de recuperación de la inversión, todo en términos económicos, además de las repercusiones en términos comerciales.

Desde un inicio se pensó en instalar el sistema únicamente en la matriz, para ver como funcionaba, la verdad es que el dueño estaba receloso, pero se hallaba en libertad de hacerlo, en un futuro próximo se piensa diseminar el sistema en la otra sucursal. Por supuesto esta previsión está contemplada en el sistema y los módulos respectivos podrían ser desarrollados próximamente. Desde luego se sabe del potencial de crecimiento del negocio y no es descabellado pensar en la creación de un área de sistemas. Factores como el equipo disponible, la infraestructura requerida y la disponibilidad económica son las limitantes actuales. Con la puesta en marcha del nuevo sistema se pretende llevar a la empresa a la consecución de una serie de propósitos. Según el nuevo ambiente de trabajo que representa este paradigma.

CAPÍTULO I

ANTECEDENTES

1.1. Antecedentes.

Hace veinte años el Sr. Rodrigo Chávez abrió un negocio de venta de materiales de construcción. Hoy día el negocio ha crecido y ahora se llama Prefabricados Ferretería y Construcción PreFeyCon; tiene una sucursal en Apaseo en Alto y la matriz en Apaseo el Grande, que compiten por ser reconocidas en la región, cuenta con 27 empleados en total, y su volumen de ventas se ha incrementado enormemente, igual que la cantidad de clientes, tanto así, que el contador ha demandado, junto con el propietario, la construcción e implementación de un sistema informático que venga a optimizar los procesos actuales y se vuelva una herramienta indispensable en la gestión y administración del negocio.

1.2 Empresa a la que se dirige.

La primera perspectiva es instalar el sistema en la matriz, y en un futuro inmediato diseminarla a la sucursal. Por supuesto esta previsión está contemplada en el sistema y los módulos respectivos serian desarrollados en el œorto plazo. No se descarta el potencial de crecimiento del negocio ni la creación de un área de sistemas que dé mantenimiento a la aplicación o de un desarrollo posterior que haga más robusto y potente al sistema. Factores como el equipo disponible, la infraestructura requerida y la disponibilidad económica son las limitantes actuales.

1.2.1 Misión y visión de la empresa.

Misión: Seguir siendo la comercializadora de materiales para la construcción número uno en ventas al mayoreo y menudeo de Apaseo y la región.

Visión: La comercializadora que ofrece los productos de mayor calidad, en el menor tiempo posible, a los precios más accesibles del mercado y con un trato humano al cliente.

1.2.2 Impacto tecnológico del sistema.

Con la puesta en marcha del nuevo sistema se pretende llevar a la empresa a la consecución de una serie de propósitos. Según el paradigma del nuevo ambiente de trabajo que representa el nuevo programa, ahora los clientes podrán ordenar sus pedidos en forma más rápida, cómoda, sencilla y practica que nunca. Se Integra un control de créditos, que incluye pagos parciales. Poseerá la habilidad de expedir recibos al cliente, con información detallada de los precios y los productos que adquiere. Integrara catálogos detallados de clientes, proveedores y productos. Generara reportes minuciosos de los viajes entregados por repartidor, del total de los pedidos hechos y de los productos enviados en cada pedido. Además de todo esto, se proyecta instaurar un mecanismo de control virtual de acceso de los usuarios. Y finalmente se podrán efectuar cortes diarios de pedidos.

1.3. Operación y procesos actuales.

Los principales procesos que se llevan a cabo dentro de las sucursales son:

Elaboración de pedidos. El cliente ordena los pedidos, en la medida de sus necesidades y sus posibilidades, y el negocio emite en consecuencia una nota, que representa el pedido que deberá ser entregado, y pagado en su debida oportunidad.

Control de entrega de mercancía. Las comisiones que obtiene los repartidores vienen dadas en proporción al volumen de pedidos que entreguen, estás comisiones se determinan manualmente.

Control de créditos. Comúnmente muchos de los pedidos son de contado, pero otros tantos son a crédito. El negocio otorga créditos a aquellos clientes con capacidad de pago o un historial de crédito positivo y lo hace en forma discrecional.

Reportes de los pedidos por día y por producto. Se elaboran cortes que contienen el total de los pedidos entregados por repartidor, otros que contiene los pedidos que fueron a crédito y los que fueron de contado y un concentrado de los productos totales entregados en una fecha en un periodo determinado.

1.4. Descripción de la problemática.

La problemática general del negocio se puede centralizar en lo que respecta a la elaboración de pedidos. El cliente ordena los pedidos, cada uno de estos últimos contiene una cantidad de productos significativamente dispar entre unos y otros. En el momento, en el negocio se elabora una nota por cada uno de ellos. Lógicamente por ser una actividad manual, hecha en medios tradicionales (papel, lápiz y calculadora) donde directamente está involucrado el factor humano, se presta a cometer errores ortográficos, pero sobre todo aritméticos. La otra gran vertiente se refiere a los cierres diarios. Los cortes de los pedidos se hacen todos lo días y para todos los repartidores, pero con la carga de trabajo, al final del mes los cierres que se han hecho no muestran con precisión los pedidos que cada chofer entregó. Se lleva dos o tres días más después de la fecha del corte en que esos reportes queden listos para el contador.

1.5. Solución propuesta.

En forma conjunta el sistema propone la siguiente solución; que se creé un mecanismo que restrinja el acceso de los usuarios al sistema, incluyendo la definición de niveles de usuario, perfiles de seguridad y menús disponibles. Implementar cortes electrónicos diarios de pedidos, que incluyan un control de chóferes repartidores. Que se facilite a los clientes el hacer sus pedidos de una forma más optima. Integrar un control de créditos, que incluya pagos parciales. Desarrollar herramientas para tener la habilidad de expedir recibos al cliente, con información detallada de los precios y los productos que adquiere. Permitir al contador, al propietario o a los encargados, modificar el precio según las políticas del negocio. Incorporar catálogos de clientes, proveedores y productos. Y que se generen reportes detallados de los viajes entregados por repartidor, del total de los pedidos hechos y de los productos enviados en cada pedido.

1.5. Nombre del sistema.

El presente sistema expone la tesis “desarrollo de sistema multi–negocio para el control de pedidos de materiales de construcción.” En lo sucesivo se llamará a ésta con la palabra sistema, y deberá entenderse por antonomasia.

1.6. Objetivo general del sistema.

Ofrecer una solución informática integral que resuelva la problemática persistente en la empresa, haciendo que sus procesos actuales tengan una equiparación con los del sistema, contribuir a la productividad de sus empleados y a la rentabilidad del negocio. Todo esto bajo un marco de apertura tecnológica, crecimiento comercial. Éste es el propósito general del sistema que será logrado gracias a la participación conjunta de todos los involucrados.

CAPÍTULO II

FUNDAMENTACIÒN

2.1. Introducción a la programación orientada a eventos.

Los lenguajes visuales orientados a eventos y con manejo de componentes dan al usuario que no cuenta con mucha experiencia en desarrollo, la posibilidad de construir sus propias aplicaciones utilizando interfaces gráficas sobre la base de ocurrencia de eventos.

Para soportar este tipo de desarrollo interactúan dos tipos de herramientas, una que permite realizar diseños gráficos y, un lenguaje de alto nivel que permite codificar los eventos. Con dichas herramientas es posible desarrollar cualquier tipo de aplicaciones basadas en este entorno.

Visual Basic es uno de los lenguajes de programación que más entusiasmo despierta entre los programadores de computadoras, tanto expertos como novatos. En el caso de los programadores expertos por la facilidad con la que desarrollan aplicaciones complejas en poquísimo tiempo (comparado con lo que cuesta programar en Visual C++, por ejemplo.) En el caso de los programadores novatos por el hecho de ver de lo que son capaces a los pocos minutos de empezar su aprendizaje.

Visual Basic es un lenguaje de programación visual, también llamado lenguaje de cuarta generación. Esto quiere decir que un gran número de tareas se realizan sin escribir código, simplemente con operaciones gráficas realizadas con el ratón sobre la pantalla.

Visual Basic es también un programa basado en objetos, aunque no orientado a objetos como Visual C++. La diferencia está en que Visual Basic utiliza objetos con propiedades y métodos, pero carece de los mecanismos de herencia y polimorfismo propios de los verdaderos lenguajes orientados a objetos como Java y C++.

2.1.1. Programas secuenciales y orientados a eventos.

Existen distintos tipos de programas. En los primeros tiempos de los ordenadores los programas eran de tipo secuencial (también llamados tipo batch.) Un programa secuencial es un programa que se arranca, lee los datos que necesita, realiza los cálculos e imprime o guarda en el disco los resultados. De ordinario, mientras un programa secuencial está ejecutándose no necesita ninguna intervención del usuario. A este tipo de programas se les llama también programas basados u orientados a procedimientos o algoritmos (procedural lenguajes.) Este tipo de programas siguen utilizándose ampliamente en la actualidad, aunque la difusión de los PC's ha puesto de moda otros tipos de programación.

Los programas interactivos exigen la intervención del usuario en tiempo de ejecución, bien para suministrar datos, bien para indicar al programa lo que debe hacer por medio de menús. Los programas interactivos limitan y orientan la acción del usuario.

Por su parte los programas orientados a eventos son los programas típicos de Windows, tales como Netscape, Word, Excel o PowerPoint. Cuando uno de estos programas ha arrancado, lo único que hace es quedarse a la espera de las acciones del usuario, que en este caso son llamadas eventos. El usuario dice si quiere abrir y modificar un archivo existente, o bien comenzar a crear un archivo desde el principio. Estos programas pasan la mayor parte de su tiempo esperando las acciones del usuario (eventos) y respondiendo a ellas. Las acciones que el usuario puede realizar en un momento determinado son variadísimas, y exigen un tipo especial de programación; la orientada a eventos. Este tipo de programación es sensiblemente más complicada que la secuencial y la interactiva, pero con los lenguajes visuales de hoy, se hace sencilla y agradable. Antes de continuar es necesario definir algunos conceptos de los elementos de la programación orientada a eventos.

2.1.2. Eventos.

Las acciones que el usuario produce sobre la aplicación se llaman eventos. Son eventos típicos el clic sobre un botón, el hacer doble clic sobre el nombre de un archivo para abrirlo, el arrastrar un icono, el pulsar una tecla o combinación de teclas, el elegir una opción de un menú, el escribir en una caja de texto o simplemente mover el ratón. Cada vez que se produce un evento sobre un tipo de control específico, Visual Basic arranca una determinada función o procedimiento que realiza la acción programada para ese evento en concreto. Estos procedimientos se llaman con un nombre que se forma a partir del nombre del objeto y el nombre del evento, separados por un guión bajo (_) como por ejemplo txtBox_click, que es el nombre del procedimiento que se ocupará de responder al evento clic en el objeto txtBox.

2.1.3. Propiedades.

Además de los eventos, la mayor parte de los objetos, como los formularios y los controles, son suministrados con propiedades.

Son conceptos fundamentales e importantes:

Propiedad. Es una asignación que describe algo sobre un objeto, como un formulario por ejemplo. Dependiendo de la propiedad de que se trate, se le puede asignar en tiempo de diseño usando la ventana propiedades y/o en tiempo de ejecución al momento de correr el programa.

A continuación se describen dos ejemplos de las propiedades del formulario de Visual Basic:

MinButton. Esta propiedad puede asignarse como true (verdadero) o false (falso.) Dependiendo de la asignación, el formulario tendrá o no un botón de minimizar.

BackColor. Asignando esta propiedad a un valor expresado como hexadecimal RGB (rojo, verde, azul) o como una constante, se cambia el color del fondo del formulario. Se pueden consultar las constantes usando el examinador de objetos (seleccione ver, examinador de objetos) y en la biblioteca VBRUN (columna clase) bajo "ColorConstants" y "SystemColorConstants."

2.1.4. Métodos.

Los métodos son funciones que también son llamadas desde programa, pero a diferencia de los procedimientos no son programadas por el usuario de VB, sino que vienen ya preprogramadas con el lenguaje. Los métodos realizan tareas típicas, previsibles y comunes para todas las aplicaciones, de ahí que vengan con el lenguaje y que se libere al programador de la tarea de construirlos. Cada tipo de objeto o de control tiene sus propios métodos.

En general sólo pueden ser corridos en tiempos de ejecución, no en tiempo de diseño. Algunos ejemplos de métodos de formularios son el método move, que mueve un formulario en un espacio de dos dimensiones en la pantalla.

Los métodos son invocados dando el nombre del objeto cuyo método se está llamando, listando el operador punto (.) y después listando el nombre del método. Como cualquier rutina los métodos pueden incorporar argumentos.

Por ejemplo:

Form1.Show 1

El método Show carga y muestra un formulario, dos acciones distintas que forman ambas partes del proceso de nacimiento o inicio de tal formulario (por ejemplo al ejecutar el formulario de inicio primero se carga y después se muestra dicho formulario.) El método Show puede ser invocado como no modal o modal. Modal significa que no se ejecuta ningún código posterior hasta que el formulario se oculte o se descargue. Cuando se muestra un formulario modal no se puede producir ninguna entrada de usuario (del teclado o del ratón) excepto para los objetos del formulario modal. Si se activa el estilo 1 (uno) es modal y 0 (cero) es no modal.

Como el nombre lo indica, una gran parte de la programación con Visual Basic se realiza visualmente. Esto significa que durante el tiempo de diseño usted tiene la capacidad de conocer la forma en que el programa se verá al ejecutarse. Esta es una gran ventaja sobre otros lenguajes de programación debido a que se tiene la capacidad de cambiar y experimentar con el diseño hasta que se esté satisfecho con los colores, proporciones e imágenes que incluya en el programa.

2.1.5. Programas para el entorno Windows.

Visual Basic está orientado a la realización de programas para Windows, pudiendo incorporar todos los elementos de este entorno informático, como ventanas, botones, cajas de diálogo y de texto, botones de opción y de selección, barras de desplazamiento, gráficos y menús. Prácticamente todos los elementos de interacción con el usuario de los que dispone Windows pueden ser programados en Visual Basic de un modo extraordinariamente sencillo. En ocasiones bastan unas pocas operaciones con el ratón y la introducción a través del teclado de algunas sentencias para disponer de aplicaciones con todas las características de Windows.

2.1.6. Modo de diseño y modo de ejecución.

La aplicación Visual Basic de Microsoft puede trabajar de dos modos distintos; en modo de diseño y en modo de ejecución. En modo de diseño el programador construye interactivamente la aplicación, colocando controles en el formulario, definiendo sus propiedades, desarrollando funciones para gestionar los eventos. La aplicación se prueba en modo de ejecución. En ese caso el usuario actúa sobre el programa (introduce eventos) y prueba cómo responde éste mismo. Hay algunas propiedades de los controles que deben establecerse en modo de diseño, pero muchas otras pueden cambiarse en tiempo de ejecución. Por el contrario también hay propiedades que sólo pueden establecerse en modo de ejecución y que no son visibles en modo de diseño.

2.1.7. Formularios y controles.

Cada uno de los elementos gráficos que pueden formar parte de una aplicación típica de Windows es un tipo de control; los botones, las cajas de diálogo y de texto, las cajas de selección desplegables, los botones de opción y de selección, las barras de desplazamiento horizontales y verticales, los gráficos, los menús, son controles para Visual Basic.

Cada control debe tener un nombre a través del cual se puede hacer referencia a él en el programa. Visual Basic proporciona nombres por defecto que el programador puede modificar.

En la terminología de Visual Basic se llama formulario (form) a una ventana. Un formulario puede ser considerado como una especie de contenedor para los controles. Una aplicación puede tener varios formularios, aunque un único formulario puede ser suficiente para las aplicaciones más sencillas. Los formularios deben también tener un nombre, que puede crearse siguiendo las mismas reglas que para los controles.

2.1.8. Objetos y propiedades.

Los formularios y los distintos tipos de controles son entidades genéricas de las que pueden haber varios ejemplares concretos en cada programa. En programación orientada a objetos (más bien basada en objetos, habría que decir) se llama clase a estas entidades genéricas, mientras que se llama objeto a cada ejemplar de una clase determinada.

Cada formulario y cada tipo de control tienen un conjunto de propiedades que definen su aspecto gráfico (por ejemplo el tamaño, color, posición en la ventana, tipo o tamaño de letra) y su forma de responder a las acciones del usuario (si está activo o no, por ejemplo.) Cada propiedad tiene un nombre que viene ya definido por el lenguaje.

Por lo general, las propiedades de un objeto son datos que tienen valores lógicos (true o false) o numéricos concretos, propios de ese objeto y distintos de las de otros de su clase. Así pues, cada clase, tipo de objeto o control tiene su conjunto de propiedades, y cada objeto o control concreto tiene valores determinados para las propiedades de su clase.

Casi todas las propiedades de los objetos pueden establecerse en tiempo de diseño y también -casi siempre- en tiempo de ejecución. En este segundo caso se accede a sus valores por medio de las sentencias del programa, en forma análoga a como se accede a cualquier variable en un lenguaje de programación. Para ciertas propiedades ésta es la única forma de acceder a ellas. Se puede acceder a una propiedad de un objeto por medio del nombre del objeto al que pertenece, seguido de un punto y el nombre de la propiedad, como por ejemplo optColor.objName.

Nombres de Objetos. En principio cada objeto de Visual Basic debe tener un nombre, por medio del cual se hace referencia a dicho objeto. El nombre puede ser el que el programador desee, e incluso Visual Basic

proporciona nombres por defecto para los diversos controles. Estos nombres por defecto hacen referencia al tipo de control y van seguidos de un número que se incrementa a medida que se van introduciendo más controles de ese tipo en el formulario (por ejemplo VScroll1 para una barra de desplazamiento vertical, HScroll1, para una barra horizontal.)

Los nombres por defecto no son adecuados porque hacen referencia al tipo de control, pero no al uso que de dicho control está haciendo el programador. Por ejemplo, si se utiliza una barra de desplazamiento para introducir una temperatura, conviene que su nombre haga referencia a la palabra temperatura, y así cuando haya que utilizar ese nombre se sabrá exactamente a qué control corresponde. Un nombre adecuado sería por ejemplo hsbTemp, donde las tres primeras letras indican que se trata de una barra de desplazamiento horizontal, y las restantes (empezando por una mayúscula) que servirá para definir una temperatura.

Existe una convención ampliamente aceptada que es la siguiente:

Se utilizan siempre tres letras minúsculas que indican el tipo de control, seguidas por otras letras (la primera mayúscula, a modo de separación) libremente escogidas por el usuario, que tienen que hacer referencia al uso que se va a dar a ese control.

La tabla 2.1 muestra las abreviaturas de los controles más usuales, junto con la nomenclatura inglesa de la que derivan.

Tabla 2.1 Convenciones utilizadas para nombrar los objetos de VB.

Abreviatura	Control	Abreviatura	Control
chk	Check box	cbo	Combo box
cmd	Command button	dir	Dir list box
drv	Drive list box	fil	File list box
frm	Form	fra	Frame
hsb	Horizontal scroll bar	img	Image
lbl	Label	lin	Line
lst	List	mnu	Menu
opt	Option button	pic	Picture
shp	Shape	txt	Text box
tmr	Timer	vsb	Vertical scroll bar

2.1.9. Orden de disparo de los eventos.

Para controlar con éxito la aparición y el comportamiento de los formularios (y también de los controles) en tiempo de ejecución, debe comprenderse en que orden se disparan los eventos. Las consideraciones del orden de disparo de los eventos se deciden generalmente por el usuario, donde debe ser colocada una parte determinada de código de respuesta de un evento. Los eventos de un formulario pueden ser divididos en los grupos siguientes:

Inicio.

Respuesta a una acción o evento producido por el usuario.

Vinculación.

Cierre.

Es importante también comprender que con frecuencia un evento inicia automáticamente como consecuencia de otro, produciendo un efecto de cascada. Por ejemplo un evento KeyPress no puede ser disparado sin disparar también los eventos KeyUp y KeyDown. El secreto para trabajar con esta clase de situaciones es una comprensión clara de qué es lo que dispara cada evento en la secuencia; el peligro de la codificación es iniciar una cadena sin fin de llamadas a eventos circulares recursivos.

2.2. Introducción a la programación orientada a objetos.

La programación orientada a objetos (POO) nos permite escribir códigos menos propensos a fallos además de la reutilización de código de forma más conveniente.

Se presentan a continuación las características de la POO desde el punto de vista de los lenguajes de .NET Framework y cómo utilizar los distintos elementos que nos permitirán crear código más fácil de escribir y de mantener.

2.2.1. Los pilares de la POO.

Tres son las características principales de un lenguaje orientado a objetos, es decir, se considera que un lenguaje está totalmente orientado a objetos si es capaz de proveer estas características:

Encapsulación. Es la cualidad de unificar los datos y la forma de manipularlos, de esta manera es posible ocultar el funcionamiento de una clase y exponer sólo los datos que maneja (mediante propiedades), así como proveer de medios para poder manipular dichos datos (mediante métodos.) De esta forma sólo queda expuesta al mundo exterior la información y la manera de manipularla, ocultando los detalles usados para manejar esos datos y, lo que es más importante, evitando que nadie manipule de una forma no controlada dicha información.

Herencia. Es la cualidad de poder crear nuevas clases (o tipos) basadas en otras ya existentes, de forma que la nueva clase obtenga todas las características de la clase que ha heredado, tanto los datos que contiene como la forma de manipularlos, pudiendo añadir nuevas características e incluso cambiar el comportamiento de algunas de las incluidas en la clase base (siempre que así se haya previsto.) Mediante la herencia podemos crear de forma fácil una jerarquía de clases que comparten un mismo comportamiento básico pero que cada nueva generación puede tener (y de hecho tiene) un nuevo comportamiento.

Polimorfismo. Es la cualidad de implementar de forma particular algunas de las características que tienen las clases, de forma que cuando sea necesario usarlas no importe la implementación interna que cada una tenga, lo que realmente interesa o debe importar es que sean usadas esas características e incluso sean accedidas en forma anónima.

2.2.2. Las clases y sus estructuras.

En los lenguajes orientados a objetos, existe el concepto de clase. Cuando se hable de clase, también podrá ampliarse a estructuras, de hecho, para los programadores de C++ una clase no es más que una estructura que se comporta de forma diferente.

Una clase es una pieza de código en la que es posible definir una serie de datos y al mismo tiempo métodos (funciones o procedimientos) que permitirán acceder a esos datos.

Cuando una clase es definida, lo que se está haciendo es definir una plantilla, a partir de la cual es posible crear objetos en la memoria. Por tanto, la clase es el molde con el cual se pueden crear nuevos objetos. Para poder crear un objeto tangible a partir de una clase, se debe crear en la memoria un nuevo objeto del mismo tipo de la clase, en estos casos se dice que se instanció un nuevo objeto de la clase. A partir de ese momento se tendrá algo real con que trabajar; una instancia de la clase, es decir, la definición realizada en la clase se ha convertido en un objeto al que se puede acceder y comenzar a usar, dándole nuevos valores a los datos que maneja y usando las funciones que nos permiten manipular dichos datos.

La diferencia principal entre una clase y una estructura es la forma en que se crean los objetos que representan esos conceptos. Los objetos creados a partir de las clases son objetos por referencia, es decir, si una variable es declarada para manipular ese objeto, lo que se tiene es una referencia (o apuntador) a una dirección de memoria en la que realmente está el objeto; mientras que los objetos creados a partir de una estructura se almacenan de forma diferente, en lugar de apuntar a una dirección de memoria en la que se encuentra el objeto, es como si las variables declaradas como estructuras fuesen realmente el objeto pudiendo así

hacer ciertas operaciones y manipulaciones que los objetos obtenidos a partir de una clase no pueden realizar de la misma forma.

En .NET siempre se usará una clase para escribir cualquier tipo de código. Es decir, se haga lo que se haga, deberá hacerse dentro de una clase. Esto no quiere decir que siempre se deban usar las características de la POO, ya que si simplemente se desea hacer una aplicación que muestre un mensaje en pantalla, el código no tiene porqué usar la herencia, el polimorfismo o la encapsulación, simplemente se escribe el código que muestre el mensaje y con eso será suficiente.

2.2.3. Interfaces.

Cuando se habla de polimorfismo, inevitablemente se debe hablar de interfaces, ya que, principalmente permiten utilizar dicha característica de la POO. La pregunta es ¿Qué es una interfaz? No se trata aquí de las interfaces de usuario, es decir, lo que se mostrará al usuario final de aplicación, sino de una clase especial de interfaz en la que solamente se definen los métodos y propiedades que una clase que la implemente debe codificar. Las interfaces representan un contrato, de forma que cualquier clase que la implemente debe utilizar los miembros de la interfaz usando la misma forma en que ésta la ha descrito; mismo número de argumentos y mismo tipo de datos devueltos.

Gracias a la implementación de interfaces es posible crear relaciones entre clases que no estén derivadas de la misma clase base, pero que tengan métodos comunes, al menos en la forma, aunque no necesariamente en el fondo. Por ejemplo del método guardar, este método se puede definir en una interfaz, las clases que quieran implementar un método guardar estandarizado firmarán un contrato con la interfaz que lo especifica, aunque la forma interna de funcionamiento solo atañe al programador de la clase, lo importante es saber que cualquier clase que haya firmado ese contrato tendrá que seguir las condiciones

impuestas por la interfaz, de esta forma todas las clases tendrán un método guardar compatible, aunque el cómo se realice propiamente la acción de guardar no debe llegar a importar, simplemente se confía en que se ha implementado en forma adecuada para almacenar los datos que la clase manipula.

2.2.4. Constructores y destructores.

Cuando un objeto es creado a partir de una clase, se sigue un proceso, el cual comienza cuando se decide crear una nueva instancia de dicha clase. En estos casos, el compilador utiliza lo que se llama el constructor de la clase. Siempre que se crea un nuevo objeto en la memoria está involucrado el constructor de la clase.

Los constructores son procedimientos especiales (funciones que no devuelven un valor) en los que se escribe toda la lógica que se emplea para la creación del objeto. Por ejemplo, es posible inicializar las variables usadas y asignarles algunos valores predeterminados.

De forma análoga, cuando un objeto ya no se necesita más, se destruye mediante una llamada al destructor de la clase. En .NET la destrucción de los objetos suele hacerse de forma automática, es decir, a diferencia de lo que ocurre en otros entornos de programación, no es necesario destruir explícitamente un objeto para eliminarlo de la memoria, esa gestión de limpieza de objetos la realiza el recolector de basura Garbage Collector (GC) de .NET, el cual decide cuándo un objeto no se necesita más y en ese caso lo elimina dejando libre la memoria utilizada para otros usos.

2.2.5. Sobrecarga (Overload.)

Otra de las características que también nos ofrecen los lenguajes orientados a objetos es la posibilidad de definir varias funciones de una clase con un mismo nombre, pudiendo así crearse versiones diferentes, por ejemplo para que reciban argumentos de distintos tipos sin necesidad de cambiarle el nombre.

Supóngase que se desea hacer una función que realice cualquier tipo de operación sobre dos valores numéricos, sería lógico pensar que si esos valores son de tipo entero, el resultado que devuelva la función también debería ser de tipo entero, en caso de que los valores a usar en la operación sean de tipo flotante, el resultado podría devolverlo de ese mismo tipo.

En los lenguajes no orientado a objetos, se deberían crear dos funciones con nombres diferentes, sumaInt y sumaFloat por ejemplo. No obstante la sobrecarga permitirá crear dos funciones que se llamen suma y el compilador utilizará la que considere adecuada según el tipo de datos que se pasen como argumentos.

El único requisito para poder crear sobrecargas de métodos es que las versiones se distingan en los argumentos, ya sea que se trate de diferentes tipos de datos o que el número de argumentos usados sea diferente, de esa forma el compilador no tendrá ningún problema en saber cuál debe usar en cada ocasión. La sobrecarga se puede aplicar tanto a los constructores como a cualquier otro método de la clase.

2.2.6. Campos, propiedades y métodos miembros de la clase.

Las clases manejan datos y también proveen de funciones para acceder a esos datos. Para ser preciso, los datos se mantienen o almacenan internamente en los campos declarados en las clases. Los campos no son otra cosa que variables declaradas en la clase, por lo general en forma privada; pero ¿Por qué declaradas de forma privada? Precisamente para apegarse a la característica de encapsulación de la POO, es decir, los datos no deben exponerse de forma directa.

Si se deseara exponer los datos, podrían usarse las propiedades. Las propiedades son funciones especiales que permiten acceder a esos datos, aunque dicho de otro modo, en realidad las propiedades representan a los datos que una clase contiene, al menos en forma pública. De este modo es posible controlar la forma en que se leen o asignan dichos datos, ya que las propiedades realmente son funciones en las que se escribe código para controlar los valores asignados o leídos.

Los métodos permiten realizar acciones sobre los datos, por ejemplo devolver un rango de valores o simplemente dar formato a la información contenida. Debido a que algunas veces los métodos devolverán algo y otras no, se pueden usar tanto las funciones que devuelven valores como las que no.

Además de los campos, métodos y propiedades, las clases tienen otros miembros como los eventos y las enumeraciones. Éstos permitirán recibir notificaciones de cuando ocurra algún suceso (evento) o declarar ciertos valores constantes que pueden ser usados para restringir algunos valores asignados a las propiedades o que permitan seleccionar de forma coherente la información que se pretende obtener (enumeraciones.)

2.2.7. Ámbito de los miembros de la clase.

Los paradigmas de trabajo de las clases señalan que los campos deberán ser privados, con la idea de que no estén accesibles de forma externa. Del mismo modo también pudieran definirse los otros miembros de la clase, esto es especialmente útil cuando la funcionalidad es para uso exclusivo de los miembros de la clase. Por lo tanto cuando se quiera exponer la funcionalidad más allá de la clase deberá considerarse entonces el ámbito de los miembros de la clase.

El ámbito se emplea para permitir o restringir el acceso desde cualquier otro código fuera de la clase. Dependiendo de cómo sea el acceso a los miembros de la clase deberán emplearse distintos modificadores de ámbito, a saber:

Private. Cuando se declara un miembro de la clase como privado significa que sólo podrá ser accedido desde la propia clase. Éste es el más restrictivo de los modificadores de ámbito y el que se recomienda para los campos y las funciones de uso interno.

Protected. Los miembros declarados como protegidos sólo estarán accesibles, además de en la propia clase, desde cualquier clase derivada.

Friend. Para uso dentro de la propia aplicación. Cuando un miembro es declarado con este modificador, únicamente es posible acceder a él desde la propia clase o desde cualquier código que se encuentre en el mismo ensamblado (proyecto.)

Protected Friend. Se trata de una mezcla de Protected y Friend, es decir sólo accesible desde las clases derivadas o desde el mismo proyecto.

Public. Permite exponer abiertamente cualquier miembro de la clase, de forma que no haya restricciones para acceder a él.

2.3. Visual Basic.Net se basa en Framework.Net.

En la figura 2.1 se puede apreciar las distintas partes que componen la arquitectura del .NET, incluidas un entorno de ejecución de aplicaciones (CLR), un conjunto de bibliotecas de funcionalidad reutilizable (BCL) y los compiladores y herramientas de desarrollo de distintos lenguajes de programación. Todos estos componentes se montan en la familia de sistemas operativos de Windows.

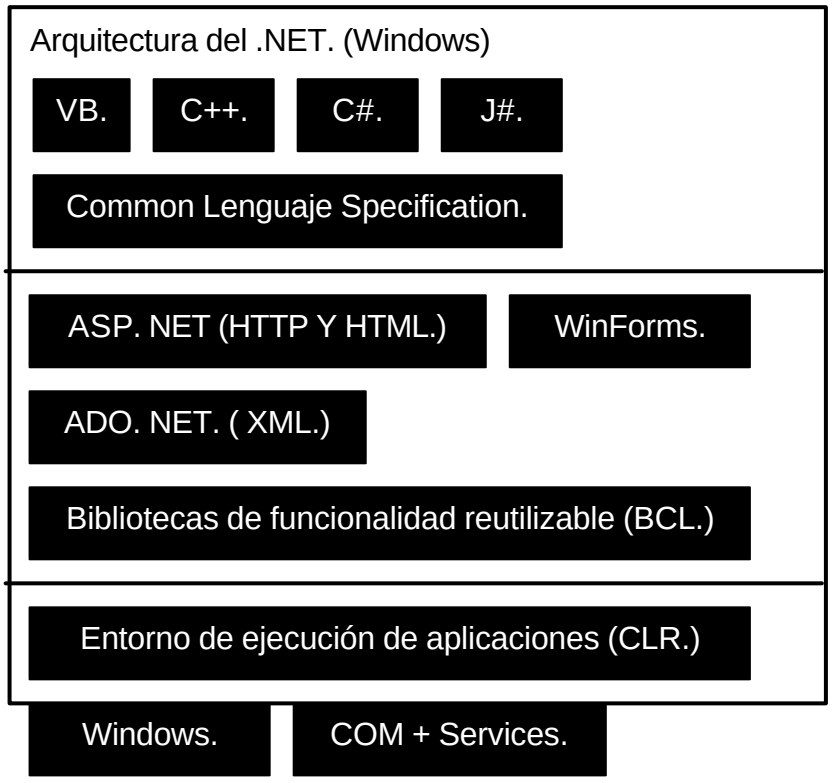

Fig. 2.1 Arquitectura del .NET.

Dentro del conjunto de las bibliotecas se distinguen cuatro subcomponentes principales:

ASP.NET. Constituye la tecnología para construir aplicaciones de usuario con interfaz tipo web (es decir, aplicaciones cuya lógica se encuentra centralizada en uno o varios servidores y que los clientes pueden ver utilizando un navegador mediante una serie de protocolos y estándares http y html.)

WinForms. Representa la tecnología que permite crear aplicaciones de usuario con interfaz tipo Windows basada en formularios y ventanas de funcionalidad rica y que ejecutan directamente los clientes.

ADO.NET. Contiene un conjunto de clases que permiten interactuar con bases de datos relacionales y documentos XML como repositorios de información persistente.

La biblioteca de funcionalidad reutilizable (BCL) que contiene las funcionalidades más utilizadas para el desarrollo de todo tipo de aplicaciones. Por ejemplo el manejo de colecciones, cadenas de texto, entradas y salidas, threading, operaciones matemáticas y dibujos en 2D.

2.4. Entorno de ejecución de aplicaciones (CLR.)

Ahora se describirá un poco el CLR es decir, el motor de ejecución de aplicaciones del .NET. Reconociendo algunos de los principales servicios que brinda a las aplicaciones que se ejecutan sobre él.

Compilación just in time. Se encarga de compilar las aplicaciones .NET a código máquina (nativo) según el sistema operativo y la plataforma de hardware en los que se esté ejecutando. Lo anterior lo hace sin intervención alguna del desarrollador o del usuario, y solamente en la medida que sea necesario.

Gestión automática de memoria. Abstrae a los desarrolladores de tener que pedir y liberar memoria explícitamente. Para esto, uno de sus componentes llamado garbage collector se encarga de liberar periódicamente la memoria que ya no está siendo utilizada por ninguna aplicación. Por otra parte, también les exenta del uso de apuntadores y del acceso a memoria de bajo nivel.

Gestión de errores consistente. Dado que las aplicaciones .NET no se ejecutan directamente contra el sistema operativo, cualquier error no manejado que ocurra en tiempo de ejecución será atrapado por el CLR en última instancia, no afectando a ninguna otra aplicación que se esté ejecutando ni teniendo efecto alguno en su estabilidad.

Ejecución basada en componentes. Todas las aplicaciones .NET son empaquetadas en componentes reutilizables denominados genéricamente assemblies, que el CLR se encarga de cargar en memoria y ejecutar.

Gestión de seguridad. El CLR provee una barrera más de contención a la hora de ejecutar aplicaciones manejadas, ya que permite establecer políticas de seguridad muy detalladas, que las aplicaciones .NET que se ejecutan en una determinada computadora deberán cumplir.

Threading. El CLR provee un entorno de ejecución multi hilos por encima de las capacidades del sistema operativo, así como también mecanismos para asegurar su sincronización y acceso concurrente a recursos compartidos.

El desarrollo de una aplicación .NET comienza con la escritura del código fuente en alguno de los distintos lenguajes de alto nivel soportados por ésta plataforma. Luego, este código es compilado obteniéndose así un ejecutable (que en Windows normalmente llevan la extensión .exe) o una biblioteca (que comúnmente lleva la extensión .dll.) A estos componentes resultantes se los denomina genéricamente assemblies.

Ahora bien, en lugar de contener código máquina específico para el sistema operativo y el hardware en el cual fueron compilados, los assemblies contienen un código denominado Microsoft intermediate language (MSIL.) EL MSIL es un set de instrucciones independientes de cualquier CPU. Incluye instrucciones para cargar, almacenar, inicializar e interactuar con objetos, sus atributos y sus métodos, así como también instrucciones aritméticas y lógicas, control de tráfico, acceso directo a memoria y manejo de errores. Por supuesto, antes de que el MSIL pueda ser ejecutado deberá ser convertido a código nativo específico para la CPU y el sistema operativo de que se trate, tarea que corre a cargo de los compiladores JIT incluidos en el CLR. Vea la figura 2.2.

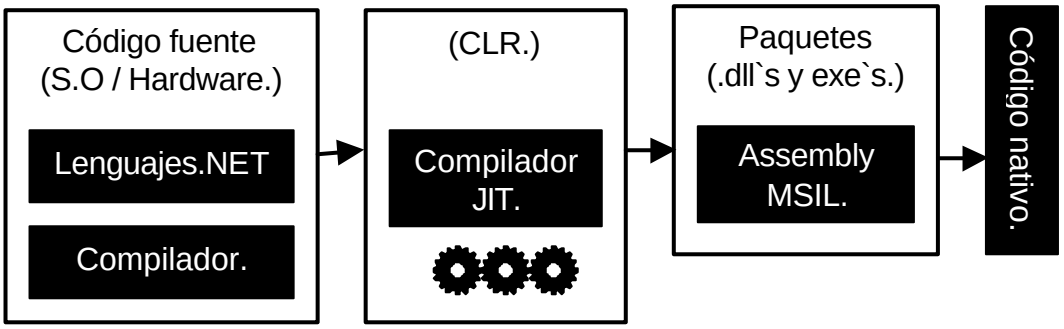

Fig. 2.2 Modelo de compilación y ejecución de aplicaciones .NET.

Una aplicación .NET se compone, entonces, de uno o más assemblies. La característica de los assemblies es que no necesitan estar registrados en Windows, como sus predecesores COM. De esta forma, instalar una aplicación .NET puede ser tan simple como copiar todos los assemblies necesarios en la maquina de destino es una ubicación específica, y bastaría con borrarlos a todos para tener una desinstalación limpia y completa.

Dado que .NET no depende del registro de Windows, y que cada assembly contiene información acerca de su versión y las versiones de los componentes de que depende; pueden coexistir múltiples versiones de assemblies sin ningún problema en la misma computadora.

Existen dos formas en que una aplicación pueda encontrar en tiempo de ejecución los assemblies de los que depende:

Ubicarlos en el mismo directorio. Esta es la opción preferida si dichos assemblies son utilizados por una única aplicación.

O ubicarlos en un repositorio centralizado de assemblies denominado global assembly cache (GAC) en el cual se instalan todos los assemblies que serán utilizados por múltiples aplicaciones en la misma computadora. Para registrar un assembly en el GAC es necesario utilizar una utilidad incluida en el SDK llamada gacutil.exe.

2.5. Modelo de ejecución del CLR.

La figura 2.3 representa el modelo de compilación y ejecución de aplicaciones .NET, al cual muchas veces se denomina de compilación diferida, o de compilación de dos etapas. Esto es así ya que el primer paso para poder ejecutar una aplicación dentro del CLR es compilar su código fuente para obtener un assembly con código MSIL. Este paso es realizado por cada uno de los compiladores de los distintos lenguajes de alto nivel soportados por .NET.

Luego, el CLR se encarga de compilar el código MSIL a código nativo que hace uso específico de los servicios del sistema operativo y la plataforma de hardware subyacente.

Todos los compiladores de los nuevos lenguajes .NET de Microsoft siguen este modelo de ejecución, con excepción de C++ .NET, que es el único lenguaje al que se le ha dejado la capacidad de emitir también código no manejado. Esto se debe a que ciertas aplicaciones, como los drivers de dispositivos, necesitan tener acceso a los recursos del sistema operativo a muy bajo nivel para lograr un rendimiento óptimo y mayor performance.

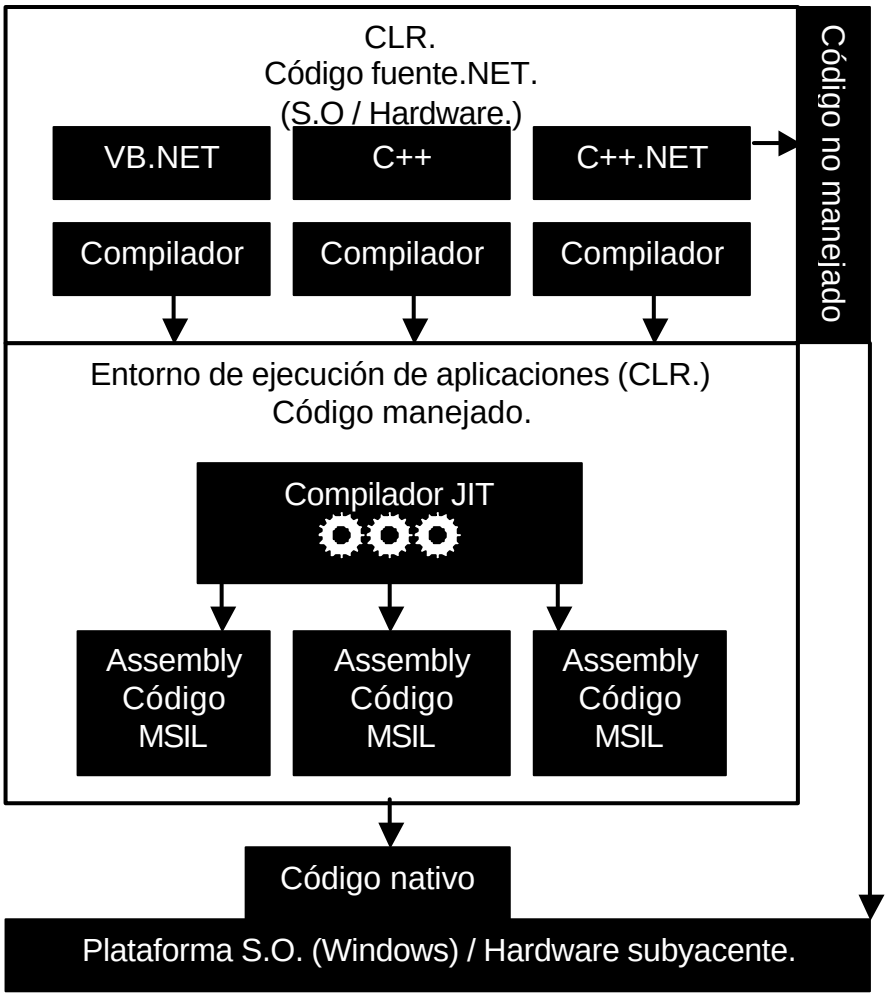

Fig. 2.3 Modelo de ejecución del CLR.

2.6. La base class library (BCL.)

De muy poco serviría a los desarrolladores el contar con una plataforma de ejecución de aplicaciones tan sofisticada y robusta como el CLR sin tener además un conjunto de funcionalidades y componentes empaquetados listos para aprovechar y utilizar en sus aplicaciones. Justamente ese es el propósito de class library que provee cientos de tipos básicos (clases e interfaces principalmente) orientados a objetos, extensibles mediante herencia, independientes del lenguaje de programación de alto nivel que se desee utilizar y organizados en namespaces jerárquicos, los principales namespaces son:

System. Raíz de todos los otros namespaces, y dentro del cual podemos encontrar la mayoría de los namespaces correspondientes a la base class library.

System.Data y System.XML. En conjunto, estos dos namespaces constituyen la tecnología conocida como ADO.NET.

System.Web. Dentro de éste se encuentran todos los tipos necesarios para programar aplicaciones y servicios web ASP.NET.

System.Windows.Forms. En éste se encuentran todos los tipos necesarios para programar aplicaciones de escritorio basadas en formularios y ventanas tipo Windows.

2.7. Common language specification (CLS) y elección del lenguaje.

La plataforma .NET es independiente del lenguaje de programación elegido para el desarrollo de aplicaciones. Para lograr esto se creó la common languaje specification (CLS) que define y estandariza un subconjunto de las características soportadas por el CLR y que son necesarias en la mayoría de las aplicaciones. Todos los componentes desarrollados y compilados de acuerdo con la especificación CLS pueden interactuar entre sí, independientemente del lenguaje de programación de alto nivel en el que fueron escritos.

Junto con la plataforma .NET, Microsoft provee implementaciones de cuatro lenguajes de alto nivel compatibles con la CLS, junto con sus compiladores:

Microsoft Visual Basic.NET.

Microsoft Visual C++.NET.

Microsoft Visual J++.NET.

Microsoft Visual C#.NET.

Esto quiere decir por ejemplo, que una aplicación escrita en Visual Basic.NET puede incorporar sin problemas nuevas partes escritas en C++ o C++.NET.

Un punto importante a destacar es que la elección del lenguaje de alto nivel en el que puede escribirse una aplicación .NET prácticamente ha sido reducida a una cuestión de gustos personales y comodidad con la sintaxis. No hay prácticamente motivos tecnológicos sobresalientes que inclinen la balanza hacia algún lenguaje en particular, al menos entre los ofertados por Microsoft todos utilizan el mismo entorno de ejecución de aplicaciones, todos utilizan el mismo conjunto de bibliotecas de la misma forma, no existen pues diferencias notorias de performance entre ellos, todos tienen el mismo potencial y todos tienen la misma capacidad de acceso a los recursos y servicios que ofrece el .NET.
De hecho, al cargar y ejecutar un assembly el CLR no sabe en qué lenguaje de programación de alto nivel ha sido escrito, ya que lo que recibe como entrada es código MSIL.

2.8. Ventajas del .NET

Estas son las ventajas que la plataforma Microsoft.NET ofrece:

Unificación de los modelos de programación, bibliotecas de funcionalidad y entornos de ejecución que existían anteriormente para distintos tipos de aplicaciones y distintos dispositivos. Antes del .NET existían lenguajes, bibliotecas, entornos de ejecución y herramientas de desarrollo distintas y específicas para cada tipo de aplicación y dispositivo (por ejemplo Visual Basic, Visual C++, ASP, VBScript o Embedded Visual C++) ahora .NET

unifica todos esos modelos de programación ofreciendo una única API, un único entorno de ejecución, un único conjunto de bibliotecas y una única herramienta de desarrollo para cualquier tipo de aplicación.

Ofrece un modelo de desarrollo simplificado, basado en objetos que utilizan un sistema unificado de tipos de datos y se empaquetan en componentes reutilizables y auto descriptivos (assemblies.)

Provee un entorno de ejecución de aplicaciones robusto y seguro (CLR), que proporciona servicios a las aplicaciones en ejecución y maneja su ciclo de vida reforzando la seguridad y abstrayendo a los programadores de optimizaciones y manejos de memoria de bajo nivel.

Otorga independencia al lenguaje de programación. Soporta múltiples lenguajes, lo cual acelera el aprendizaje de los desarrolladores permitiendo que cada uno elija basándose en sus gustos personales. Además, la posibilidad de utilizar las mismas herramientas de programación y tener las mismas capacidades de acceso a la plataforma independientemente del lenguaje le proporciona una flexibilidad sin precedentes.

Simplifica la instalación y administración de las aplicaciones gracias al uso de assemblies auto descriptivos, resolviendo gran parte de los problemas existentes en COM en lo que respecta al registro de componentes, manejo de múltiples versiones simultáneamente y compatibilidad de aplicaciones.

Posee extensibilidad. Todas las clases incluidas en el .NET son extensibles mediante los mecanismos de herencia propios del enfoque de programación orientada a objetos. Esto posibilita que funcionalidades o controles gráficos que no cumplan exactamente con una determinada necesidad puedan ser extendidos para agregarle o modificarle comportamiento sin tener que escribir todo el código nuevamente.

Proporciona un altísimo grado de interoperabilidad, tanto entre aplicaciones .NET escritas en distintos lenguajes como entre aplicaciones .NET y aplicaciones COM mediante un módulo llamado COM-Interop. Esto permite reutilizar y aprovechar aplicaciones o componentes existentes desarrollados sobre la plataforma COM (Visual Basic 6.0 por ejemplo.) Además de interoperabilidad entre aplicaciones .NET y múltiples tipos de aplicaciones desarrolladas sobre otras plataformas de software o hardware, incluso plataformas que no son de Microsoft.

2.9.1. Instalación del SQL Server.

La instalación del SQL Server 2000 require un sistema operativo compatible, como Windows 2000 Advanced Server, Windows 2000 Professional Edition, Windows 2000 Server, Windows Server 2003 o Windows XP Professional Edition.

Primero, de doble clic sobre el setup del CD o espere que comience el autorun, pulse a continuación sobre la opción SQL Server 2000 Components, lo que le llevará a la pantalla de instalación.

Luego siga estos pasos:

Paso 1. Seleccione la opción SQL Server 2000 Components.

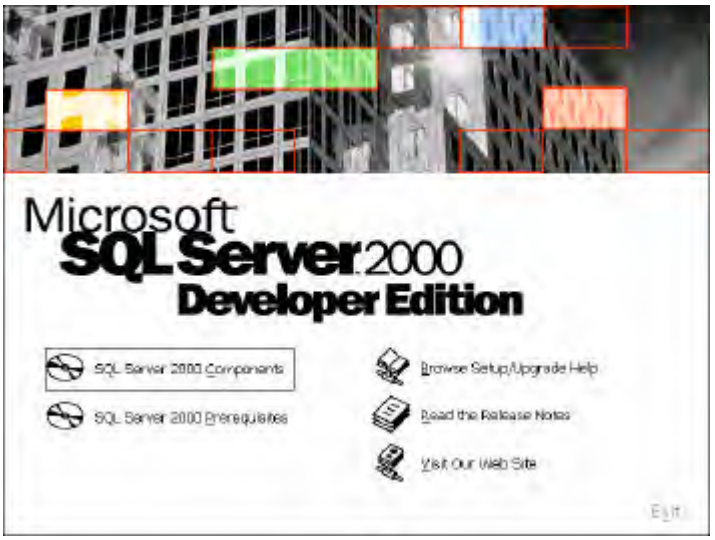

Fig. **2.4** Componentes de SQL Server 2000.

Paso 2. Seleccione la opción Install Database Server.

Fig. 2.5 Instalar servidor de base de datos.

A continuación se le presentaran diferentes ventanas, entre ellas la de acuerdo de licencia, presione Next en todas ellas hasta llegar a la ventana de Computer Name y seleccione la opción Local Computer. Esto significa que se instalará el gestor en la propia máquina y tomará por nombre el que tenga asignado el equipo.

Paso 3. Seleccione la opción Local Computer.

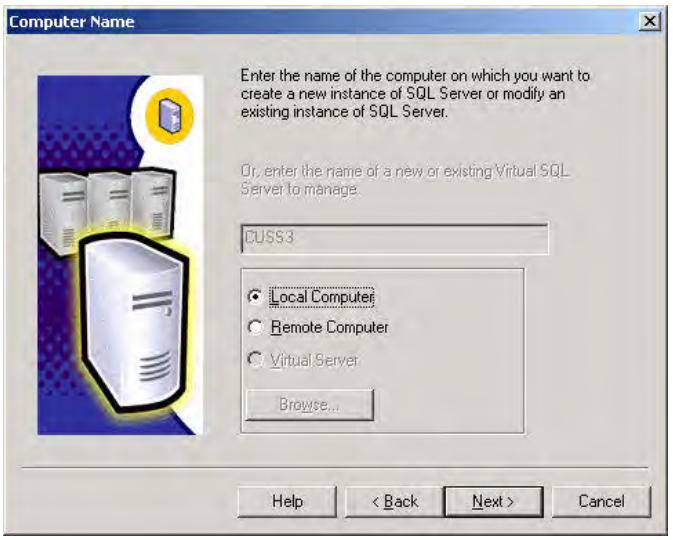

Fig. 2.6 Computadora local.

Paso 4. A continuación elija la opción Create a new instance of SQL Server.

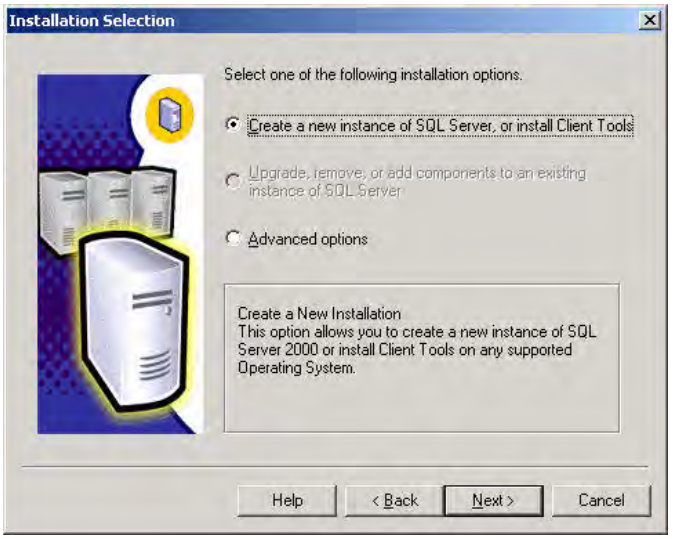

Fig. 2.7 Crear una nueva instancia de SQL Server.

Paso 5. Seleccione la opción Server and Client Tools. con la que se instalará el gestor y las herramientas cliente de enterprise manager y SQL analyzer.

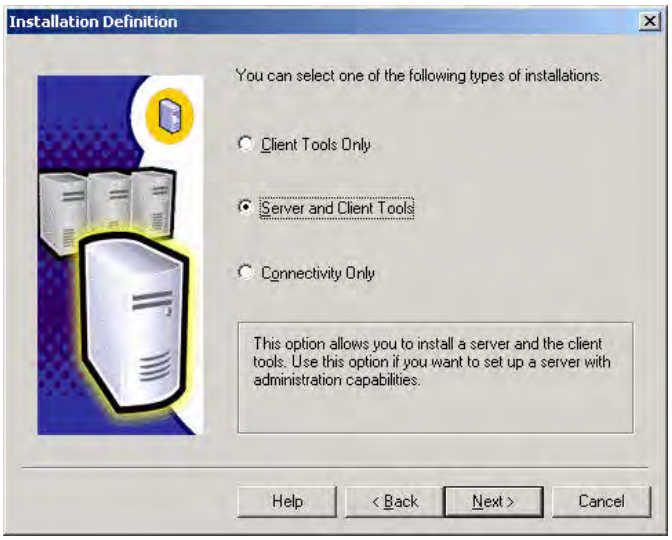

Fig. 2.8 Herramientas cliente-servidor.

Paso 6. Pulse el botón Next. Asegúrese de dejar seleccionada la opción Default.

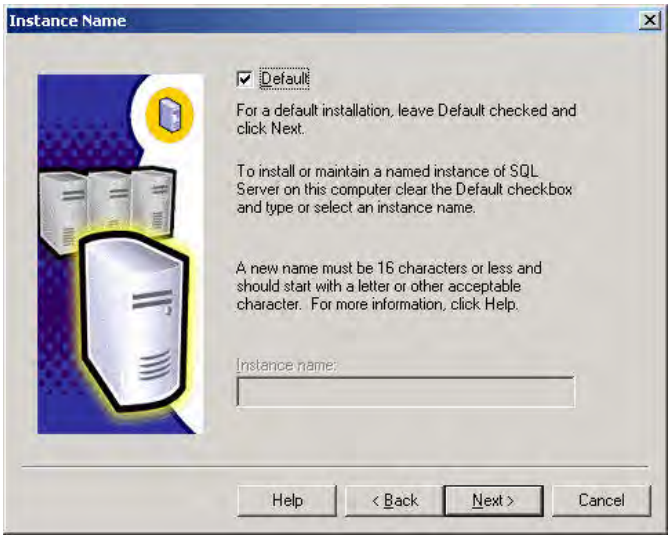

Fig. 2.9 Si lo desea puede asignar un nombre a la instancia que se va a crear.

Paso 7. Aparecerá la ventana en la que se establecerá la ubicación física donde se instalarán los archivos de programa y de datos (es decir, donde se almacenan las bases de datos que se creen.)

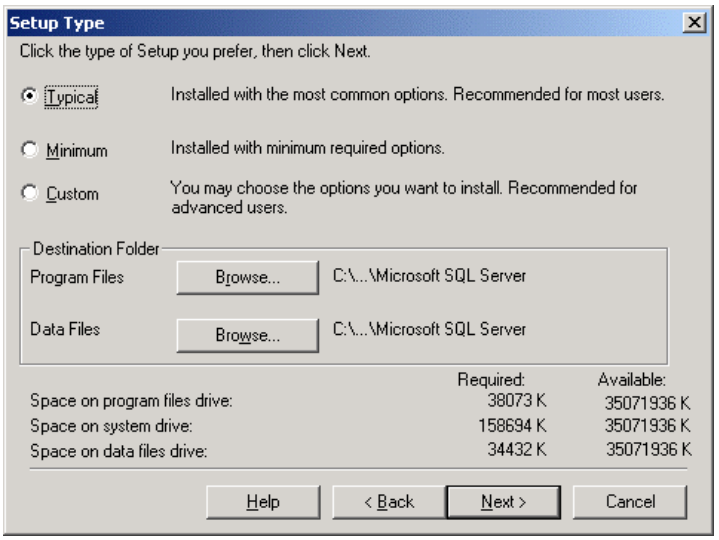

Fig. 2.10 Seleccione tipo de instalación deseada.

Paso 8. El SQL Server se instala como un servicio y, por lo tanto, es necesario asignar un usuario para que inicie dicho servicio y trabaje en un contexto de seguridad. En nuestro caso se trata de una instalación local, en un contexto cliente-servidor.

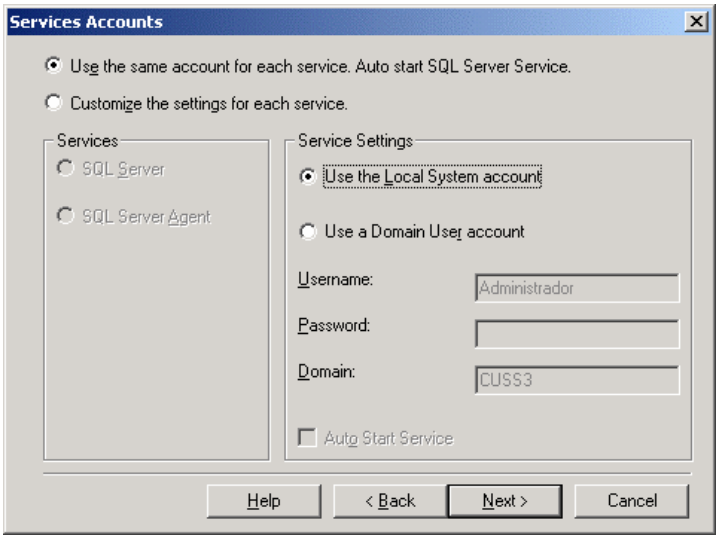

Fig. 2.11 Usar la misma cuenta para cada servicio o configurar en forma personalizada.

Paso 9. Para finalizar el proceso de instalación, se deben arrancar los servicios. Corra entonces el service manager que permite arrancar y parar los cuatro servicios que incluye el motor de base de datos; SQL Server, SQL Server Agent, Coordinador de transacciones y MSQL Server OLAP Service. Se recomienda que unicamente arranque el primero de ellos, pues los otros servicios consumen muchos recursos.

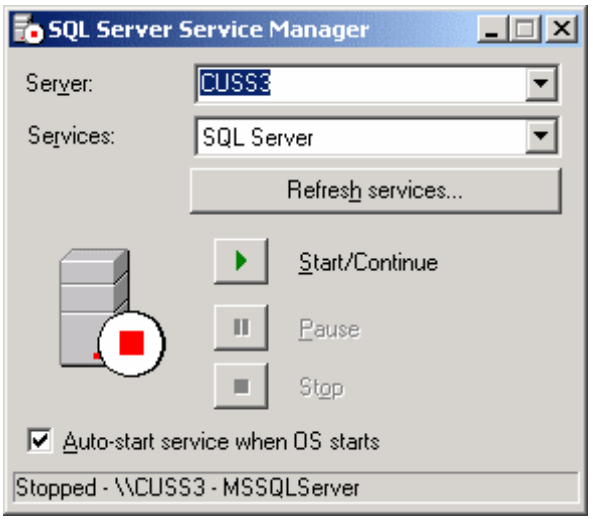

Fig. 2.12 El administrador de servicios.

2.10. Introducción a Crystal Report.

Crystal Reports le permite personalizar muchas de las opciones predeterminadas del programa para adaptarse a su manera de trabajar. Estas opciones afectan a aspectos tales como:

Su entorno de trabajo.

Su manera de seleccionar las bases de datos.

El acceso a SQL y ODBC.

La manera en que se da formato a diferentes tipos de datos, y las fuentes utilizadas para campos y texto.

2.10.1. Modelos de extracción e inserción.

Con el fin de ofrecer un acceso a base de datos cada vez más flexible para los programadores, los controladores de base de datos de Crystal Reports se han diseñado para proporcionar un modelo de extracción e inserción de acceso a datos.

2.10.2. Modelo de extracción.

Fig. 2.13 Modelo de extracción de Crystal Reports.

En un modelo de extracción, el controlador se conectará a la base de datos y extraerá datos siempre que sea necesario (ver figura 2.13.) Con este modelo Crystal Reports controla tanto la conexión a la base de datos como el comando SQL que se ejecuta para obtener los datos y no necesita ninguna otra codificación del programador. Si no se escribe ningún código especial en tiempo de ejecución, se utiliza el modelo de extracción.

2.10.3. Modelo de inserción.

Fig. 2.14 Modelo de inserción de Crystal Reports.

En contraposición, el modelo de inserción necesita que el programador escriba código para conectarse a la base de datos, debe ejecutarse un comando de SQL para crear un conjunto de registros o de datos que se ajusten a los campos del informe y enviar luego ese objeto al informe (ver figura 2.14.) Este método le permite crear recursos compartidos de conexión en la aplicación y filtrar los datos antes de que Crystal Reports los reciba.

2.10.4. Aspectos fundamentales de la elaboración de informes.

Diseño. Los informes son creados con la ayuda del Crystal Report Designer. Éste se inicia automáticamente al añadir un objeto de Crystal Reports al proyecto activo o al hacer doble clic en un objeto de Crystal Reports que ya exista en el proyecto.

Conexión con la base de datos. En Crystal Report Designer, seleccione primero el origen de datos al que hará referencia el informe. Recuerde que es posible utilizar varios orígenes de datos en un mismo informe. A continuación, seleccione las tablas de la base de datos que desee utilizar en el informe. Crystal Reports puede vincular las tablas automáticamente, o bien puede especificar el modo en el que desea vincularlas. Las tablas de bases de datos se vinculan para que los registros de una base de datos coincidan con los registros relacionados de otra tabla.

Objetos en los informes. Crystal Report Designer utiliza la función de arrastrar y colocar muy parecida a la de Visual Studio.NET, es decir, arrastre un objeto del informe hasta el diseñador (un campo de una tabla de la base de datos o un objeto de texto por ejemplo) utilice luego la ventana propiedades o el menú contextual para dar formato a tal objeto.

Los objetos que pueden ser agregados al informe y a los que se les podrá dar formato según las necesidades son:

Campos de las tablas de la bases de datos.

Campos de fórmula.

Campos de parámetro.

Campos de nombre de grupo.

Campos de total acumulado.

Campos de resumen.

Gráficos.

Subinformes.
Secciones de informes. Crystal Report Designer está dividido en secciones de informes, tales como encabezados, pies de página y detalles. Los objetos se arrastran a la sección de informe deseada.

Los datos que aparecen en el informe final dependen de las opciones de organización. En particular, los datos del informe varían según las secciones en las que desee insertar objetos de informe concretos. Por ejemplo, si se inserta un objeto de gráfico en la sección encabezado de informe, el gráfico sólo aparecerá una vez al principio del informe y resumirá los datos que contiene el informe. Además, si un objeto de gráfico se añade a la sección encabezado de grupo, aparecerá un gráfico individual al principio de cada grupo de datos y sólo se resumirán los datos relacionados con dicho grupo.

2.11. Introducción a la tecnología cliente-servidor.

Es un modelo para construir sistemas de información, que se sustenta en la idea de repartir el tratamiento de la información y los datos por todo el sistema informático, permitiendo mejorar el rendimiento del mismo sistema en forma global. El modelo cliente servidor se define como aquella tecnología que proporciona al usuario final un acceso transparente a las aplicaciones, datos, servicios o cualquier otro recurso del grupo de trabajo a través de la organización en múltiples plataformas. El modelo soporta un medio ambiente distribuido en el cual los requerimientos o peticiones de servicio hechos por estaciones de trabajo inteligentes, llamadas clientes, se traducen en un trabajo realizado por otros tipos de computadoras llamados servidores.

2.11.1. ¿Qué es una arquitectura?

Una arquitectura es un entramado de componentes funcionales que aprovechando diferentes estándares, convenciones, reglas y procesos, permite integrar una amplia gama de productos y servicios informáticos, de manera que pueden ser utilizados eficazmente dentro de la organización.

Debemos señalar que para seleccionar el modelo de una arquitectura, hay que partir del contexto tecnológico y organizativo del momento y, que la arquitectura cliente-servidor requiere una determinada especialización de cada uno de los diferentes componentes que la integran.

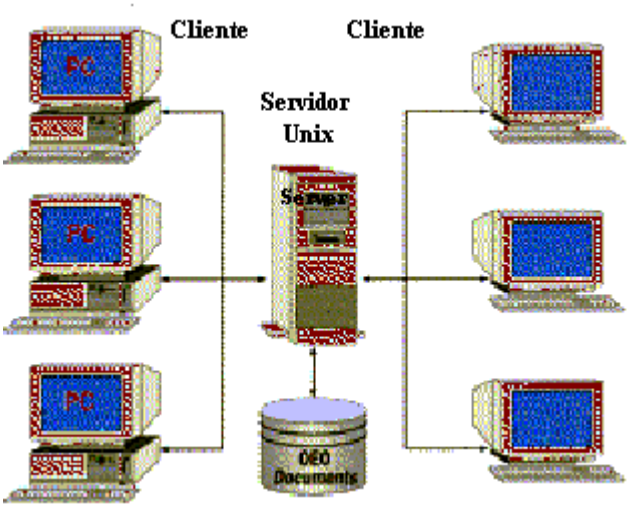

Fig. 2.15 Arquitectura cliente-servidor.

2.11.2. ¿Qué es un cliente?

Es la estación de trabajo inteligente que inicia un requerimiento o petición de servicio. El requerimiento inicial puede derivar en múltiples requerimientos de trabajo a través de redes LAN o WAN. La ubicación de los datos o de las aplicaciones es totalmente transparente para el cliente.

2.11.3. ¿ Qué es un servidor?

Es cualquier recurso de cómputo dedicado a dar respuesta a los requerimientos del cliente. Los servidores pueden estar conectados a los clientes a través de redes LAN o WAN para proveer de múltiples servicios a los clientes y ciudadanos tales como impresión, acceso a bases de datos, fax o procesamiento de imágenes.

2.11.4. ¿ Qué es un proceso distribuido?

Es un modelo de sistemas y/o de aplicaciones, en el cual las funciones y los datos pueden estar distribuidos a través de múltiples recursos de cómputo, conectados en un ambiente de redes LAN o WAN.

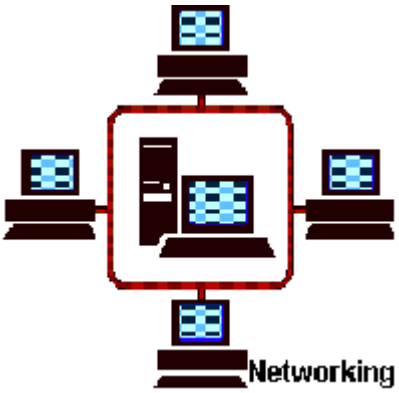

Fig. 2.16 Proceso Distribuido.

2.11.5. Elementos de la arquitectura cliente-servidor.

En esta aproximación, y con el objetivo de definir y delimitar el modelo de referencia de una arquitectura cliente-servidor, deberán ser identificados los componentes que permitan articular dicha arquitectura, considerando que toda aplicación o sistema de información está caracterizado por tres componentes básicos:

Presentación-captación de la Información.

Procesos.

Almacenamiento de la Información.

Los cuales se suelen distribuir tal como se presenta en la figura 2.17.

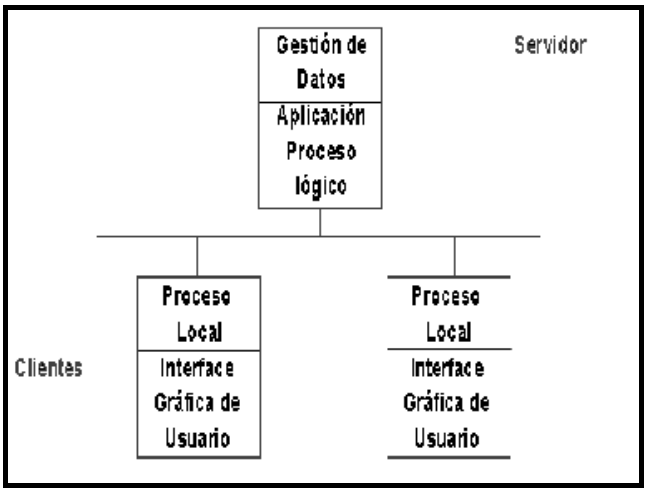

Fig. 2.17 Elementos de la arquitectura cliente-servidor.

Y se integran en una arquitectura cliente-servidor basándose en los elementos que caracterizan dicha arquitectura, es decir; puesto de trabajo o cliente, comunicaciones y servidores (back-end.) Lo anterior se representa en la figura 2.18.

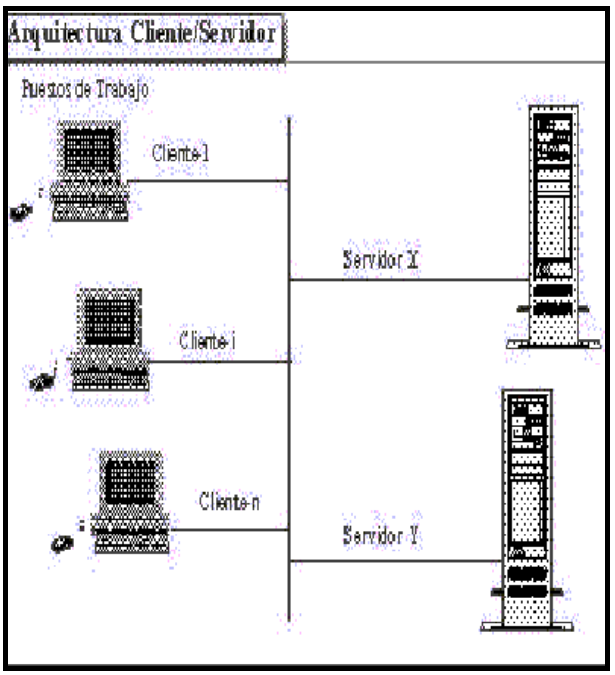

Fig. 2.18 Elementos de la arquitectura cliente-servidor.

Puesto de trabajo o cliente. Se trata de una computadora (de escritorio o portátil) con acceso al sistema y conectada a una red, que le permite gestionar una serie de recursos, el cual se perfila como una estación de trabajo universal, en la que se realiza la mayor parte de los procesos.

Fig. 2.19 Puesto de trabajo o cliente.

Debemos destacar que el puesto de trabajo basado en una computadora conectada a una red, favorece la flexibilidad y el dinamismo en las organizaciones. Dado que permite trasladar la ubicación de los puestos de trabajo, gracias a las bondades de la red.

Comunicaciones. Sus dos vertientes son:

Infraestructura de redes. Se refiere a los componentes de hardware y software que garantizan la conexión física y la transferencia de datos entre los distintos equipos de la red.

Infraestructura de comunicaciones. Es decir los componentes de software y hardware que permiten la comunicación y la gestión, entre los clientes y los servidores.

Servidores (back-end.) Se trata de uno o varios equipos que suministran una serie de servicios como bases de datos, archivos o comunicaciones.

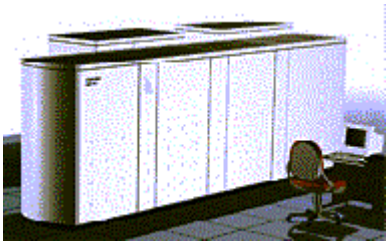

Fig. 2.20 Servidores (back-end.)

Los servidores, según la especialización y los requerimientos de los servicios que debe suministrar pueden ser:

Mainframes.

Miniordenadores.

Especializados (dispositivos de red o imagen.)

2.11.6. Ventajas de la tecnología cliente–servidor.

El uso de esquemas cliente–servidor se ha promovido gracias a la existencia de plataformas de hardware cada vez más baratas. Ésta constituye a su vez la ventaja más palpable de este esquema; la posibilidad de utilizar máquinas considerablemente más baratas que las requeridas por una solución centralizada, basada en sistemas más grandes.

Es posible utilizar componentes, tanto de software como de hardware de distintos fabricantes, lo cual contribuye considerablemente a la reducción de costos y favorece la flexibilidad en la implementación y actualización de las soluciones.

Este esquema facilita la integración entre sistemas diferentes y permite compartir información, permitiendo por ejemplo que las máquinas que ya se tienen puedan usarse, pero empleando interfaces más amigables al usuario. De esta manera, podemos integrar equipos en sistemas medianos y grandes, sin necesidad de que todos tengan el mismo sistema operativo.

Al proliferar el uso de interfaces gráficas, los sistemas construidos bajo este esquema tienen una mayor interacción, cada vez más intuitiva para el usuario. El uso de este tipo de interfaces, en este tipo de esquemas representa una ventaja respecto a un esquema centralizado, ya que no siempre es necesario transmitir información gráfica a través de la red pues ésta puede residir en el cliente, lo cual permite aprovechar el ancho de banda de la propia red.

En el uso del esquema cliente–servidor el mantenimiento y el desarrollo de aplicaciones resulta más rápido, pues se pueden emplear los recursos con los que ya se cuenta (por ejemplo los servidores de SQL o las herramientas de más bajo nivel como los sockets o el RPC.)

La estructura inherentemente modular que posee el esquema facilita la integración de nuevas tecnologías y el crecimiento de la infraestructura, permitiendo así que las soluciones puedan escalarse.

El esquema contribuye además, a proporcionar a los diferentes departamentos de una organización, soluciones locales, pero permitiendo la integración de la información relevante a escala global.

2.11.7. Desventajas de la tecnología cliente–servidor.

El esquema cliente–servidor cuenta con muy pocas herramientas para la administración y ajuste del desempeño de los sistemas. En el desarrollo de aplicaciones se deben tener en cuenta los aspectos, que se mencionan a continuación:

Existe la implicación que tanto los clientes como los servidores utilicen los mismos mecanismos generales (sockets o RPC) en diferentes plataformas.

Se debe contar con estrategias pare el manejo de errores y para mantener la consistencia de los datos. La seguridad en este esquema es de suma importancia. Deberán hacerse verificaciones periódicas tanto en el cliente como en el servidor. Podrá recurrirse también técnicas como el encriptamiento.

El desempeño es otro de los factores a considerar en este tipo de esquemas. Dado que puede derivarse en problemas de congestión o embotellamientos en el tráfico de la red. Consecuentemente se debe planear cómo distribuir los datos en la red. En el caso de una organización, por ejemplo, esta distribución puede ser hecha por departamentos, en forma geográfica o lógica. Se deberá tener presente que en algunos casos, por razones de confiabilidad o eficiencia, se hace

necesario tener datos replicados, y que las actualizaciones se hagan en forma simultánea.

En las organizaciones por ejemplo, se hace necesario la toma de decisiones respecto a la disyuntiva entre comprar o desarrollar los diferentes componentes que se desean implementar.

2.11.8. Ventajas para las organizaciones.

En la sección 2.11.6 se encuentran las ventajas del esquema cliente–servidor, el cual permite que el usuario invoque servicios de forma totalmente transparente, haciendo énfasis en los aspectos técnicos. Se describe ahora como estas ventajas tienen un impacto positivo en la organización.

Este esquema reduce considerablemente los costos de producción de software y disminuye sustancialmente los tiempos requeridos para ello. Esto es así, pues, para la construcción de una nueva aplicación pueden usarse los servidores con los que ya se disponga, dedicándose el desarrollo únicamente a la elaboración de los procesos del cliente, según los requerimientos que este haya deseado.

Por consecuencia se disminuyen directamente los costos del departamento de sistemas. Además, es posible llegar a reducir los costos del hardware requerido, migrando paulatinamente las aplicaciones hacia plataformas más baratas, es decir haciendo uso de las nuevas tecnologías de la información, aprovechando el potencial de los diferentes elementos de la red, y facilitando la interacción entre las distintas aplicaciones que posea la organización, para hacerlas más especializadas.

El esquema es capaz de disminuir los costos de capacitación del personal, dado que las interfaces gráficas cada vez resultan más

intuitivas, amigables y hasta personalizadas para el usuario final, al cual destinadas, traduciéndose lo anterior en una mayor productividad.

Las interfaces gráficas que ahora se emplean contribuyen al suministro de información a los usuarios en forma remota y por medios no tradicionales (en papel o en discos.) Esta información proporcionada es consistente, significativa, puntual y representativa para el usuario, es decir, los datos ahora representan información veraz y oportuna, un cliente puede incluso recolectar información específica desde diferentes sitios de la red y organizarla en el sentido que desee, todo lo anterior gracias a que se cuenta con un control centralizado de los elementos compartidos.

Se posee la habilidad inherente de integrar sistemas heterogéneos. Es decir, tanto clientes como servidores pueden coexistir en múltiples plataformas y tener acceso a los datos desde las más diversas ubicaciones en la red.

Gracias a que hace algunos años los fabricantes de equipo de cómputo decidieron la implementación de estándares en la fabricación de sus productos, para hacerlos más abiertos y facilitar así la correcta gestión y administración de los distintos componentes de software y hardware con los que cuentan las empresas y los individuos, y debido también a la gran cantidad de firmas que incursionan en el mercado con soluciones cada vez más modernas y sofisticadas; ahora cuando una organización decide adquirir o actualizar un producto, servicio o tecnología, tiene a su disposición una gran variedad de fabricantes y compañías que a su vez ofrecen una variedad de estos nuevos productos o servicios. Cualquier desarrollo posterior a la primera compra no implica adquirirlo del mismo proveedor, en virtud del enorme factor de compatibilidad que ahora existe. Paradójicamente esta razón aumente la competitividad, pues las grandes marcas ya no acaparan el mercado y por consecuencia las organizaciones pueden cambiar de proveedor según sus necesidades y sus posibilidades.

Con el establecimiento de estándares aparecieron los sistemas abiertos. Un sistema abierto es un medio en el cual se pueden intercambiar componentes de software y hardware, dando a un usuario mayor posibilidad de escoger productos de acuerdo a sus necesidades y fomentando la competencia entre proveedores, que deben mejorar sus servicios para ganar clientes.

Los equipos o tecnologías abiertas permiten el desarrollo y la implementación de aplicaciones distribuidas, en virtud de que es posible combinar equipos con distintas características y diferentes sistemas operativos.

CAPÍTULO III

METODOLOGÍA DE IMPLEMENTACIÓN.

3.1. Introducción a la metodología del análisis de sistemas.

En orden de desarrollar un sistema que se considere optimo, la primera etapa a seguir, es la realización de un análisis del sistema ya que éste es la base del éxito o fracaso de un proyecto. Un analista de sistemas generalmente evalúa la manera en que funcionan los negocios, examina la entrada de datos, el procesamiento que se les da a dichos datos y las salidas de información que se provocan, todo esto con el propósito de mejorar los procesos actuales de la organización. Un análisis de sistema debe cumplir con los siguientes parámetros:

Comience por definir los objetivos generales del proyecto. Identifique las necesidades reales del cliente, distinga entre los requerimientos y las expectativas. Sección 3.2. Proceda a diseñar y a aplicar entrevistas. Sección 3.3. A continuación deberá elaborar y presentar a los ejecutivos del negocio un estudio de rentabilidad, el cual estará motivado según las necesidades del usuario y los objetivos del proyecto. Sección 3.4. Si el estudio procede, presente el sistema en forma gráfica, es decir elabore un diagrama de flujo de datos. Secciones 3.5 y 3.6. Sobre la base de bs objetivos originales ya planteados, defina específicamente los requerimientos que el sistema deberá satisfacer. Sección 3.7. De forma similar a como presentó el estudio de rentabilidad, presente ahora un estudio de factibilidad. Sección 3.8. Si nuevamente el estudio procede, construya ahora un diccionario de datos para definir todas las entidades que poseerá el sistema. Sección 3.9. Finalmente construya un DER para mostrar en su totalidad la estructura y funcionalidad del programa. Sección 3.10.

Identifique las necesidades reales del cliente. Reúnase con el cliente (puede ser un ejecutivo, un jefe de departamento, un operativo o cliente en particular) e identifiquen juntos los objetivos generales del proyecto. Opcionalmente el cliente puede preparar un documento conceptual del proyecto, aunque es recomendable que éste se elabore durante la reunión

con el analista, ya que de hacerlo el cliente solo, de todas maneras tendría que ser modificado, durante la identificación de las necesidades.

3.2. Objetivos generales del proyecto.

El analista debe determinar con precisión cuáles son los objetivos generales que persigue el proyecto, para lo cual es necesario identificar plenamente las necesidades reales de los usuarios. Dichos objetivos deben ser claros, precisos y concisos, pues de éstos depende en gran medida el éxito o fracaso del producto final, ya que si éstos no fueren los más adecuados o estuvieren mal interpretados el producto final se presenta como inapropiado en cuanto no satisface las necesidades originales de los usuarios y en vano resulta todo el proceso de desarrollo.

3.3. Entrevista con los usuarios.

La entrevista con los usuarios es la herramienta más importante para la determinación de los objetivos generales del proyecto, pues por medio de ésta el analista tiene conocimiento de la manera en que funciona actualmente el negocio y de cuales son las expectativas del usuario del sistema. La forma más atinada para la realización de entrevistas es aplicarlas de arriba hacia abajo, es decir, comenzar por los niveles gerenciales y terminar con los trabajadores operativos, que manejan el sistema que actualmente se está estudiando.

Antes de realizar cualquier entrevista es necesario solicitar autorización explicita del jefe inmediato de la persona que se desea entrevistar, y de ser posible se debe hacer extensiva para ambos.

Toda entrevista debe ser planeada con anterioridad, con el fin de tener bien claro cuáles son les objetivos que se persiguen con ésta, es decir, realizar un

esbozo de las preguntas que se van a hacer al entrevistado, así como el enfoque que se le va a dar a cada una de ellas; sin embargo el analista se debe permitirse ciertos desvíos del plan, pues en ocasiones las respuestas obtenidas indican que es necesario ahondar en ciertos temas o quizá haya algunos otros que se tornen de poco o nulo interés, e incluso hay ocasiones en las que debe darse completamente un giro a lo que se tenía planeado. En realidad, no siempre es necesario determinar con anterioridad las preguntas que se van a plantear, pues a veces el analista decide que es preferible realizar una entrevista menos formal en la que solamente se vaya dirigiendo al entrevistado hacia el tema o los temas que se desean conocer. La decisión de realizar una entrevista bien planeada o una menos formal se toma basándose en la labor que el entrevistado realiza dentro de la organización y la información que se desea obtener.

Si se desea éxito en la entrevista, se debe buscar un horario oportuno en el que el entrevistado no se encuentre distraído, ni conteste todas las preguntas rápidamente sin analizarlas por tener trabajo pendiente, o se presente cualquier otra situación fortuita por la que no preste la atención debida. Por lo anterior se recomienda hablar antes con el entrevistado y darle la oportunidad de determinar el día y la hora en prefiere ser entrevistado, aclarándole que necesitamos de su completa colaboración.

El analista está obligado a causar una buena impresión al entrevistado, debe ser cortés, nunca prepotente, debe transmitir confianza al entrevistado para que éste no sienta que se le está juzgando o fiscalizando; el analista debe presentarse vestido de una manera formal, pues de otra manera es difícil que alguien acceda a cooperar.

Se deben evitar en la medida de lo posible, las entrevistas largas, pues llega un momento en que el entrevistado se cansa y comienza a distraerse; por lo cual es preferible tener una serie de entrevistas cortas en lugar de una larga.

Lo recomendable es propiciar un ambiente adecuado de tal manera que la persona a entrevistar se sienta bien, para ello es necesario considerar ciertos aspectos tales como el dejar hablar al entrevistado, no interrumpirlo a menos que se esté desviando demasiado del tema. El analista no debe suponer nada ni dar algo por hecho, pues el entrevistado se puede confundir o sentir mal. Hay que respetar a cualquier persona sea cual fuere su puesto dentro de la organización. Es imprescindible también tratar de hablarle a cada entrevistado usando su propio argot.

El analista debe concretarse al tema, no empezar a tratar asuntos que no tienen nada que ver con el estudio del sistema.

Deberá también tener despiertos sus cinco sentidos, estar pendiente de lo que pasa a su alrededor y del lenguaje corporal del entrevistado, pues en muchas ocasiones estos aspectos revelan cosas importantes.

Inmediatamente después de terminada la entrevista ésta se debe transcribir y documentar, incluso se pueden elaborar diagramas que contribuyan a una mejor comprensión de lo investigado. Una grabación de la entrevista es de gran ayuda para la documentación, pero ésta se debe hacer sólo con el consentimiento del entrevistado. Un resumen y auto evaluación podría servir de gran ayuda para entrevistas posteriores.

3.3.1. Preguntas en la entrevista.

Todas las preguntas de la entrevista deben estar encaminadas a conocer perfectamente todas las actividades que realiza el sistema. Úsese la tabla 3.1 (si lo desea puede agregar las preguntas que considere necesario.)

Tabla 3.1 Formato de evaluación de actividades.

ENTEVISTA **Formato de evaluación de actividades**		
Nombre de la Empresa:		
Nombre del Proyecto:		
Nombre del entrevistado:		
Puesto:	**Nombre del jefe inmediato:**	
Departamento:		
Lugar:	**Fecha:**	**Clave:**
Actividad:		
1. ¿Cuál es el nombre completo de la actividad?		
2. ¿Cuál es el puesto y el nombre de la persona que la realiza?		
3. Describa cómo realiza la actividad.		
4. ¿Cada cuándo la lleva a cabo, dónde la realiza y cuál es el propósito?		
5. ¿Cuánto tiempo le lleva completar esta actividad?		
6. ¿Qué decisiones toma cuando se encuentra haciendo esta actividad?		
7. ¿Qué es lo que acostumbra hacer en el sistema cuando realiza esta actividad?		
8. ¿Cómo sabe que la actividad que ha desarrollado se completó correctamente?		
Nombre del entrevistador:		
Observaciones: (para uso exclusivo del entrevistador)		
Nombre del entrevistado:	**Firma:**	

Del mismo modo debe estudiar las entradas requeridas. Empléese la tabla 3.2.

Tabla 3.2 Formato para evaluación de entradas.

ENTEVISTA **Formato de evaluación de entradas**		
Nombre de la Empresa:		
Nombre del Proyecto:		
Nombre del entrevistado:		
Puesto:	**Nombre del jefe inmediato:**	
Departamento:		
Lugar:	**Fecha:**	**Clave:**
Actividad:		
9. ¿Qué es lo que el sistema pide para comenzar a realizar esta actividad?		
10. ¿Quién o qué departamento le proporciona lo que necesita para comenzar?		
11. ¿Cuándo puede solicitarlas y cuánto tiempo se tardan en proporcionarlas?		
12. ¿En que formato le son proporcionadas (en pantalla, impresos o en archivo)?		
13. ¿Específicamente qué hace con los datos?		
14. ¿ Cómo sabe que los datos que se ingresan son correctos?		
Nombre del entrevistador:		
Observaciones: (para uso exclusivo del entrevistador)		
Nombre del entrevistado:	**Firma:**	

Debe determinar las salidas producidas. Aplíquese la tabla 3.3.

Tabla 3.3 Formato de evaluación de salidas.

ENTEVISTA **Formato de evaluación de salidas**		
Nombre de la Empresa:		
Nombre del Proyecto:		
Nombre del entrevistado:		
Puesto:	**Nombre del jefe inmediato:**	
Departamento:		
Lugar:	**Fecha:**	**Clave:**
Actividad:		
15. ¿Qué es lo que el sistema arroja después de haber realizado esta actividad?		
16. ¿Quién o qué departamento necesita lo que el sistema arrojó para seguir trabajando?		
17. ¿Cada cuándo puede obtener estos datos y cuánto tiempo se tarda en lograrlo?		
18. ¿En que formato los presenta el sistema (en pantalla, impresos o en archivo)?		
19. ¿Específicamente a dónde o a quién le son enviados los datos?		
20. ¿ Cómo sabe que los datos que se sacan son correctos?		
Nombre del entrevistador:		
Observaciones: (para uso exclusivo del entrevistador)		
Nombre del entrevistado:	**Firma:**	

Estudie los recursos materiales y la infraestructura disponible. Utilícese la tabla 3.4.

Tabla 3.4 Formato de equipo de procesamiento de datos.

ENTEVISTA **Formato de equipo de procesamiento de datos**		
Nombre de la Empresa:		
Nombre del Proyecto:		
Nombre del entrevistado:		
Puesto:	**Nombre del jefe inmediato:**	
Departamento:		
Lugar:	**Fecha:**	**Clave:**
Actividad:		
21. Describa las características del equipo con que cuenta para realizar esta actividad.		
22. ¿Cuáles los programas o aplicaciones que están instalados en el equipo?		
23. ¿Quiénes pueden entrar, usar o configurar el equipo y quién es responsable de ello?		
24. ¿Qué hace o a quién llama cuando una aplicación o el propio equipo fallan?		
25. ¿Funcionan todos los componentes, le faltan, responden siempre, se traban?		
26. ¿ El equipo trabaja en red o se encuentra aislado?		
Nombre del entrevistador:		
Observaciones: (para uso exclusivo del entrevistador)		
Nombre del entrevistado:	**Firma:**	

Una vez que se conoce cómo trabaja el sistema actual, se procede a investigar los posibles cambios. Empléese la tabla 3.5.

Tabla 3.5 Formato de mejoramiento de los procesos actuales.

ENTEVISTA **Formato de mejoramiento de los procesos actuales**		
Nombre de la Empresa:		
Nombre del Proyecto:		
Nombre del entrevistado:		
Puesto:	**Nombre del jefe inmediato:**	
Departamento:		
Lugar:	**Fecha:**	**Clave:**
Actividad:		
27. ¿Cuáles son los problemas más comunes del sistema actual con esta actividad?		
28. ¿Qué le hace falta al sistema para mejorar la realización de la presente actividad?		
29. ¿ Qué le sobra al sistema para optimizar la realización de la presente actividad?		
30. ¿Qué cosas cambiaria de lo que hace el sistema respecto a esta actividad y por qué?		
Nombre del entrevistador:		
Observaciones: (para uso exclusivo del entrevistador)		
Nombre del entrevistado:	**Firma:**	

3.4. Estudio de rentabilidad.

Una vez que se han terminado las entrevistas y se conocen las necesidades del usuario y los objetivos del proyecto, el analista estará en condiciones de plantear en forma escrita los objetivos generales del nuevo sistema.

Se procede entonces a la realización y presentación de un estudio de rentabilidad, según los objetivos generales descritos. Este estudio debe ser revisado y autorizado por los ejecutivos de alto nivel de los distintos departamentos involucrados o por los propietarios de la organización que hayan ordenado el desarrollo e implementación del sistema o por aquellos cuya responsabilidad sea la aprobación económica de los nuevos proyectos o la determinación de las directrices de la empresa. El estudio debe contener una disertación completa de los componentes lógicos del sistema, los recursos materiales y humanos involucrados, la inversión requerida para materializarlos, los gastos de operación, el margen de ganancias y el periodo de recuperación de la inversión, todo en términos económicos, además de las repercusiones en términos comerciales.

3.5. Diagramas de flujo.

Los diagramas de flujo son una herramienta muy útil en la concepción y visualización del sistema. Se proponen dos tipos de diagramas:

Diagramas de flujo de datos.

Diagramas de flujo del programa.

3.6. Diagramas de flujo de datos.

Este tipo de diagramas muestran cuáles son las entradas, salidas y procesos que emplea el sistema, así como los orígenes y destinos de los datos.

Es recomendable no hacer muy detallado ese tipo de diagramas, pues el objetivo principal es tener una visión general del sistema sin profundizar mucho en cada una de las operaciones que este realiza. Se puede elaborar un diagrama en el que se observe cómo está ubicado el sistema dentro de la organización y las relaciones que tiene con los demás sistemas en caso de que existan y otro en el que se muestre cómo intercalan los diferentes módulos que forman parte del sistema.

Los diagramas de flujo de datos cuentan con cuatro tipos de gráficos, cuya descripción se realiza a continuación:

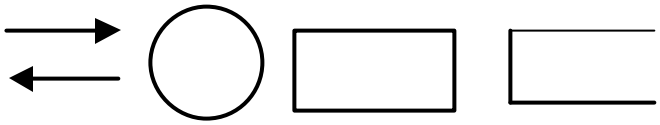

Fig. 3.1 Símbolos utilizados en los diagramas de flujo de datos.

Flujo. Se representa con una flecha con punta que simboliza los datos que se comunican entre las diferentes operaciones del sistema. Junto a la flecha debe colocarse un nombre con que el se identifica a los datos.

Proceso. Se dibuja dentro de un ovalo y representa la conversión de datos de entrada en datos de salida. Cada proceso debe poseer un nombre.

Origen o destino de datos. Se rotula con un rectángulo y simboliza una persona, organización u otro agente que no interesa al estudio del sistema, pero que opera como un origen o destino de los datos.

Archivo. Se traza empleando un rectángulo abierto en uno de sus lados y hace alusión al almacenamiento de datos. Cuando una flecha de flujo de datos apunta hacia el símbolo de archivo, se indica que se esta almacenando información, pero si la flecha sale de él, es indicio de que se está obteniendo información del usuario.

Las figuras 3.2 y 3.3 contienen ejemplos de diagramas de flujo de datos.

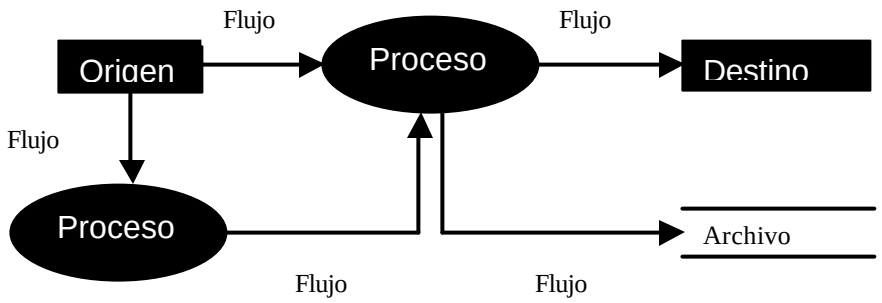

Fig. 3.2 Diagrama de flujo de datos.

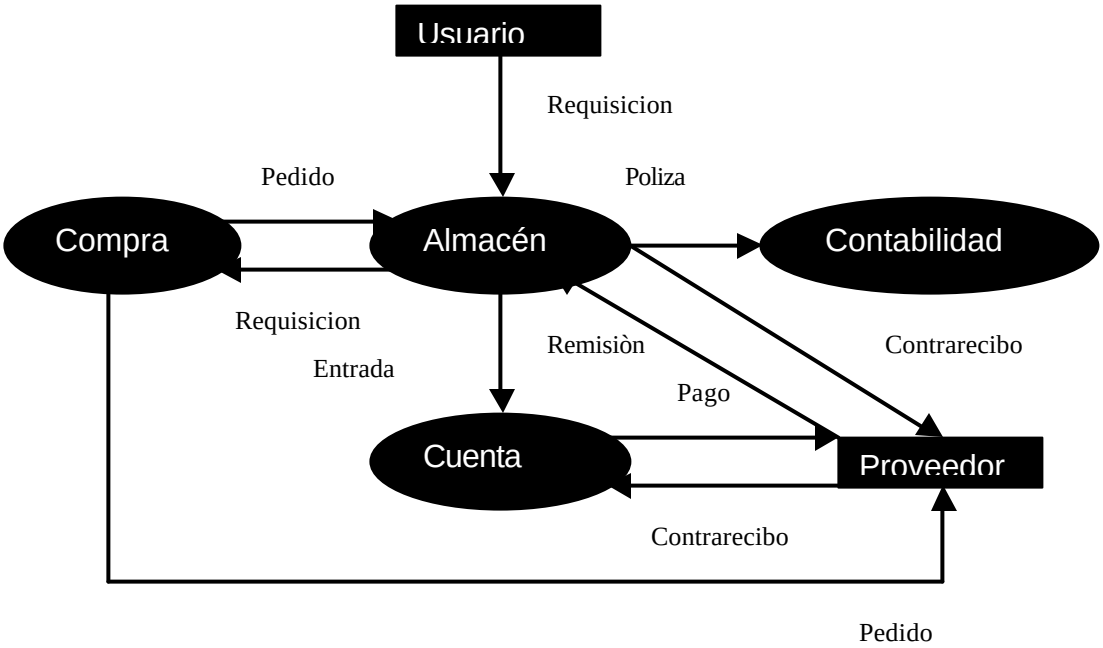

Fig. 3.3 Diagrama de flujo de datos.

En su momento, cuando se actualizó el sistema anterior o en lo sucesivo cuando deba actualizarse la nueva aplicación o alguno de sus componentes estructurales, el nuevo desarrollador suele tener serias dificultades para entender la lógica del programador anterior, ya que por lo común los programadores son contratados por un periodo de tiempo y por un proyecto especifico, salvo en aquellas organizaciones que tienen su propio departamento de desarrollo y que sin embargo no están exentas de que sus desarrolladores les abandonen en algún momento.

Para poder implementar de forma ideal todos los cambios exigidos al nuevo sistema se hace necesario comprender perfectamente toda la lógica de los módulos que lo integran, no obstante no es recomendable que se realice un diagrama de flujo por cada uno de los módulos del sistema, pues habrá algunos de ellos que por su facilidad de comprensión no será necesario representarlos gráficamente, y aunque lo ideal sería que existiera un diagrama de flujo para cada uno de ellos, habrá ocasiones en las que esto es casi imposible dadas las limitaciones descritas en la planificación temporal.

Los símbolos más empleados en este tipo de diagramas se muestran en la figura 3.4.

Entrada–salida**.**

Entrada–salida impresa.

Operación.

Inicio o fin.

Condición.

Conector.

Leer o escribir.

Fig. 3.4 Simbología de los diagramas de flujo de datos.

3.7. Determinación de los requerimientos.

El objeto de esta etapa es definir con mayor precisión las habilidades que poseerá el sistema, para ello se hizo necesario determinar y evaluar las entradas, las salidas requeridas, el procesamiento de datos y los recursos necesarios para tales procesamientos y satisfacer así las necesidades de la organización. No obstante, debe quedar bien claro que esta etapa no se trata de la reingeniería del sistema, sino de establecer cuáles son los criterios generales de su funcionamiento, mismos que en su oportunidad, nos ayudarán en el rediseño del sistema.

Para la determinación de los requerimientos el analista debe basarse principalmente en los datos obtenidos en las etapas de planteamiento de los objetivos generales y análisis preliminar del sistema, pues éstas nos dicen cuáles son las necesidades reales de los usuarios.

Dado que los sistemas poseen un periodo de vida determinado, se deben integrar como requerimientos, las expectativas de crecimiento de la empresa, las estimaciones en el incremento de la de demanda de los usuarios, el crecimiento esperado en el nivel de las transacciones y la posibilidad en el uso de tecnologías futuras, para lograr que el nuevo sistema logre abarcar dichas planificaciones.

La determinación de los requerimientos del nuevo sistema debe conservar el siguiente orden:

Salidas esperadas. Debe especificarse el tipo y la forma de salida, puede ser informes, reportes, cortes, estados, recibos, formularios, notas, facturas, ya sea en pantalla, en forma impresa en archivo.

Entradas requeridas. Se refieren a aquellas que son capaces de producir las salidas esperadas. En forma similar que en las salidas se debe especificar el

tipo y la forma, incluyendo si deben ser tomadas en forma automática por el propio sistema o si pueden ser alimentadas manualmente por el usuario.

Procesamientos de datos necesarios. Deberán determinarse las operaciones de procesamiento a las que serán sometidos los datos, para que el sistema esté en condiciones de producir las salidas esperadas.

Recursos materiales y la infraestructura disponible. Se especifican los equipos de procesamiento de datos con los que se cuenta. Los recursos y la infraestructura necesaria para la operabilidad del sistema, los recursos materiales y humanos implicados, la inversión destinada para gastos de operación, la rentabilidad ofrecida y el periodo de recuperación de la inversión

Controles. Determinan la disponibilidad de las salidas para cualquier nivel de usuario. Otras salidas estarán reservadas exclusivamente para usuarios que las necesiten para la realización cotidiana de sus actividades y otras más estarán destinadas o usuarios con niveles direccionales para quienes resultan significativas en un modo tal que les faciliten la toma de decisiones. Controles similares se aplican para las entradas, es decir, puntualizar que entidades o elementos de la organización son directamente responsables de proveer las entradas que demanda el sistema, en tiempo y forma, y de prever también los alcances y limitaciones que tendrá el sistema para hacerse de ellas, y de las facultades que se le otorgaran para someter a los datos a procesamientos continuos hasta obtener informaciones relevantes y completar las tareas para las cuales fue concebido, así como señalar con precisión las configuraciones de seguridad y las cuentas de acceso de los usuarios, la planificación del mantenimiento preventivo y correctivo de la infraestructura y cuales quiera otros mecanismos destinados al observancia de los requerimientos inicialmente estipulados.

Existe por su parte una nomenclatura ampliamente utilizada dentro del análisis de sistemas la cual consiste en especificar los documentos ya sean

de entrada o salida con una letra antecediendo al requerimiento que sea para indicar el tipo de documento que se está utilizando.

A continuación se especifican los valores y su significado:

O : Original.

N : Nuevo.

C: Cambios.

B : Baja.

Figúrese un sistema capaz de imprimir facturas, a modo de ejemplo se tiene:

Requerimiento 1: O : Emisión de factura.

Se requiere que el sistema imprima una factura, en original y copia, la original es entregada al cliente y la otra se la queda el negocio, con el objetivo de optimizar el control de las mismas. El cliente deberá proporcionar sus datos fiscales, la primera vez o actualizarlos si es necesario, antes que pueda ser emitida.

La factura se imprimiría de acuerdo al siguiente formato:

Fig. 3.5 Formato de factura.

3.8. Estudio de factibilidad.

En este punto ya se definieron completamente las características del sistema, las partes que lo integran, las habilidades que posee y las tareas que es capaz de completar. Se procederá entonces a la realización de un estudio de factibilidad, que a diferencia del de rentabilidad que presenta una serie de tópicos que deberán satisfacerse a cambio de gozar de los beneficios del sistema; este otro estudio determina las posibilidades reales de adquirir el sistema haciendo uso de todos los recursos presentes con los que cuenta la empresa y en su caso si tal sistema satisface en verdad las necesidades especificas que pretenden ser subsanadas y que estaban presentes en la determinación de los objetivos del sistema.

3.9. Diccionario de datos.

Es un catálogo de datos que contiene las descripciones detalladas de cada una de las entidades y sus atributos que definen la composición de la estructura de la base de datos o de la metabase de datos (véase la tabla 3.6.) Los objetivos de este inventario de datos son:

Apoyar el diseño y la creación de la base de datos.

Facilitar el control y las relaciones de cada una de las entidades y sus respectivos atributos, que forman parte de la estructura de la base de datos en la que se apoya sistema.

Controlar dinámicamente la estructura de las ventanas la interfase al usuario, para las diferentes pantallas del sistema.

Tabla 3.6 Diccionario de datos del sistema.

DICCIONARIO DE DATOS						
Nombre del proyecto:						
Dirección física del archivo:						
N	**NOMBRE**	**DESCRIPCIÓN**	**TIPO**	**L**	**NULO**	
Nombre de la base de datos:			**Nombre del archivo:**			
Nombre de la entidad: Empresa			**Atributos: 1:6**			
1.	EmpresaPK	**¶ Clave primaria de la tabla Empresa.**	Integer **(int)**	**2**	NOT NULL	**%**
2.	NomEmpresa	**Nombre completo o razón social de la empresa.**	String **(sng)**	**40**	NULL	**$**
3.	DomEmpresa	**Dirección o domicilio fiscal de la empresa.**	String **(sng)**	**60**	NULL	**$**
4.	RFCEmpresa	**Registro federal de contribuyentes de la empresa.**	String **(sng)**	**13**	NULL	**$**
5.	TelEmpresa	**Número de teléfono (con extensión) de la empresa.**	String **(sng)**	**40**	NULL	**$**
6.	Curp	**Clave única de registro de población de la empresa.**	String **(sng)**	**18**	NULL	**$**
7.	ContraAut	**Contraseña autorizada para el cambio de precios.**	String **(sng)**	**20**	NULL	**$**
Nombre de la entidad: Clasif			**Atributos: 1:2**			
8.	ClasifPK	**¶ Clave primaria de la tabla Clasificación.**	Integer **(int)**	**2**	NOT NULL	**%**
9.	NomClasif	**Nombre o descripción de las Clasificaciones.**	String **(sng)**	**40**	NULL	**$**
10.	EmpresaPK	**Clave primaria de la tabla Empresa.**	Integer **(int)**	**2**	NOT NULL	**%**
Nombre de la entidad: Producto			**Atributos: 2:10**			
11.	EmpresaPK	**¶ Clave primaria de la tabla Producto.**	Integer **(int)**	**2**	NOT NULL	**%**
12.	CveProducto	**¶ Clave del producto. Llave primaria de la tabla.**	Integer **(int)**	**2**	NOT NULL	**%**

13.	NomProducto	**Nombre o descripción detallada del producto.**	String **(sng)**	**40**	NULL	**$**
14.	Marca	**Marca comercial o nombre de la firma del producto.**	Long Integer **(int)**	**4**	NULL	**$**
15.	UM	**Unidad de medida o presentación del producto.**	String **(sng)**	**40**	NULL	**$**
16.	Existencia	**Nivel de existencias disponibles o stock del producto.**	Integer **(int)**	**2**	NULL	**%**
17.	Máximo	**Número máximo del nivel de existencia o stock overload.**	Long Integer **(int)**	**4**	NULL	**%**
18.	Mínimo	**Número mínimo del punto de reorden o stock reload.**	Long Integer **(int)**	**4**	NULL	**%**
19.	ClasifPK	**Clave primaria de la tabla Clasificación.**	Integer **(int)**	**2**	NOT NULL	**%**
20.	ProveedorPK	**Clave primaria de la tabla Proveedor.**	Integer **(int)**	**2**	NOT NULL	**%**
21.	PrecioMax	**Precio máximo autorizado o precio de venta al menudeo.**	Long Integer **(int)**	**4**	NULL	**$**
22.	PrecioMin	**Precio mínimo autorizado o precio de venta al mayoreo.**	Long Integer **(int)**	**4**	NULL	**$**
Nombre de la entidad: Proveedor		**Atributos: 1:5**				
23.	ProveedorPK	**¶ Clave primaria de la tabla Proveedor.**	Integer **(int)**	**2**	NOT NULL	**%**
24.	NomProveedor	**Nombre completo o razón social del proveedor.**	String **(sng)**	**40**	NULL	**$**
25.	DomProveedor	**Dirección o domicilio fiscal del proveedor.**	String **(sng)**	**60**	NULL	**$**
26.	RFCProveedor	**Registro federal de contribuyentes del proveedor.**	String **(sng)**	**13**	NULL	**$**
27.	TelProveedor	**Número de teléfono (con extensión) del proveedor.**	String **(sng)**	**40**	NULL	**$**
28.	EmpresaPK	**Clave primaria de la tabla Empresa.**	Integer **(int)**	**2**	NOT NULL	**%**
Nombre de la entidad: Pedido		**Atributos: 3:10**				

29.	EmpresaPK	**¶ Clave primaria de la tabla Empresa.**	Integer **(int)**	**2**	NOT NULL	**%**
30.	Letra	**¶ Número de letra o pago parcial de los créditos.**	Integer **(int)**	**2**	NOT NULL	**%**
31.	FolPedido	**¶ Número de folio del pedido que hace el cliente.**	Integer **(int)**	**2**	NOT NULL	**%**
32.	FecPedido	**Fecha en que se realiza el pedio de un cliente especifico.**	Date **(dtm)**	**8**	NULL	
33.	PedCred	**Pedido a crédito o de contado que realiza el cliente.**	Boolean **(bin)**	**2**	NOT NULL	
34.	ClientePK	**¶ Clave primaria de la tabla Cliente.**	Integer **(int)**	**2**	NOT NULL	**%**
35.	Observacion	**Observación o comentarios acerca del pedido del cliente.**	String **(sng)**	**40**	NULL	**$**
36.	Repartidor	**Repartidor o nombre del chofer que entrega el pedido.**	String **(sng)**	**40**	NULL	**$**
37.	NoViajes	**Número de viajes que realiza un repartido en un periodo.**	Integer **(int)**	**2**	NOT NULL	**%**
38.	EstadoVts	**Estado de la ventas que se han realizado a los clientes.**	Integer **(int)**	**2**	NOT NULL	**%**
39.	Saldo	**Saldo de la cantidad total a pagar por un pedido hecho.**	Long Integer **(int)**	**4**	NULL	**$**
40.	TotImporte	**Total de la cantidad a pagar por un pedido hecho.**	Long Integer **(int)**	**4**	NULL	**$**
41.	ImporteLetra	**Importe en letra de la cantidad total a pagar por pedido.**	String **(sng)**	**40**	NULL	**$**
Nombre de la entidad: DetallePedido		**Atributos: 3:4**				
42.	EmpresaPK	**¶ Clave primaria de la tabla Empresa.**	Integer **(int)**	**2**	NOT NULL	**%**
43.	Letra	**¶ Número de letra o pago parcial de los créditos.**	Integer **(int)**	**2**	NOT NULL	**%**
44.	FolPedido	**¶ Número de folio del pedido que hace el cliente.**	Integer **(int)**	**2**	NOT NULL	**%**

45.	CveProducto	**¶ Clave del producto. Llave primaria de la tabla.**	Integer **(int)**	**2**	NOT NULL	**%**
46.	CantPed	**Cantidad pedida de un producto que aparece en la nota.**	Integer **(int)**	**2**	NOT NULL	**%**
47.	PrecioPed	**Precio del producto pedido que aparece en la nota.**	Long Integer **(int)**	**4**	NULL	**$**
48.	ImportePed	**Importe total del pedido que aparece en la nota.**	Long Integer **(int)**	**4**	NULL	**$**
49.	Reqist	**Registro de la nota de venta o tikect del cliente.**	Integer **(int)**	**2**	NOT NULL	**%**
Nombre de la entidad: Cliente		**Atributos: 2:3**				
50.	EmpresaPK	**¶ Clave primaria de la tabla Empresa.**	Integer **(int)**	**2**	NOT NULL	**%**
51.	ClientePK	**¶ Clave primaria de la tabla Cliente.**	Integer **(int)**	**2**	NOT NULL	**%**
52.	NomCliente	**Nombre completo o razón social del cliente.**	String **(sng)**	**40**	NULL	**$**
53.	DomCliente	**Dirección o domicilio fiscal del cliente.**	String **(sng)**	**60**	NULL	**$**
54.	RFCliente	**Registro federal de contribuyentes del cliente.**	String **(sng)**	**13**	NULL	**$**
55.	TelCliente	**Número de teléfono (con extensión) del cliente.**	String **(sng)**	**40**	NULL	**$**
Nombre de la entidad: Saldos		**Atributos: 3:6**				
56.	ArioRes	**¶ Folio de los pagos restantes hechos al saldo insoluto.**	Integer **(int)**	**2**	NOT NULL	**%**
57.	EmpresaPK	**¶ Clave primaria de la tabla Empresa.**	Integer **(int)**	**2**	NOT NULL	**%**
58.	CveProducto	**¶ Clave del producto. Llave primaria de la tabla.**	Integer **(int)**	**2**	NOT NULL	**%**
59.	SaldoIniaial	**Saldo inicial después de un corte de mes para un cliente.**	Long Integer **(int)**	**4**	NULL	**$**
60.	SaldoEntradas	**Saldo de las entradas de nuevos pedidos de un cliente.**	Long Integer **(int)**	**4**	NULL	**$**

61.	DevEntradas	**Devoluciones de las entradas de nuevos pedidos.**	Integer **(int)**	**2**	NOT NULL	**%**
62.	SaldoSalidas	**Saldo de las salidas de nuevos pedidos de un cliente.**	Long Integer **(int)**	**4**	NULL	**$**
63.	DevSalidas	**Devoluciones de las salidas de nuevos pedidos.**	Integer **(int)**	**2**	NOT NULL	**%**
64.	SaldoFinal	**Saldo final antes de un corte de mes para un cliente.**	Long Integer **(int)**	**4**	NULL	**$**
Nombre de la entidad: Pago		**Atributos: 4:2**				
65.	EmpresaPK	**¶ Clave primaria de la tabla Empresa.**	Integer **(int)**	**2**	NOT NULL	**%**
66.	Letra	**¶ Número de letra o pago parcial de los créditos.**	Integer **(int)**	**2**	NOT NULL	**%**
67.	FolPedido	**¶ Número de folio del pedido que hace el cliente.**	Integer **(int)**	**2**	NOT NULL	**%**
68.	PagoPK	**¶ Clave primaria de la tabla Pago**	Integer **(int)**	**2**	NOT NULL	**%**
Nombre de la entidad: Precio		**Atributos: 3:2**				
69.	EmpresaPK	**¶ Clave primaria de la tabla Empresa.**	Integer **(int)**	**2**	NOT NULL	**%**
70.	CveProducto	**¶ Clave del producto. Llave primaria de la tabla.**	Integer **(int)**	**2**	NOT NULL	**%**
71.	PrecioPK	**¶ Llave secundaria de la tabla Precio.**	Integer **(int)**	**2**	NOT NULL	**%**
72.	Precio	**Precio de lista de los producto disponibles al cliente.**	Long Integer **(int)**	**4**	NULL	**$**
73.	DescPrecio	**Descripción general del precio al publico o especial.**	String **(sng)**	**40**	NULL	**$**
Nombre de la base de datos:		**Nombre del archivo:**				
Nombre de la entidad: Usuario		**Atributos: 1:4**				
74.	UsuarioPK	**¶ Llave primaria de la tabla Usuario.**	Integer **(int)**	**2**	NOT NULL	**%**

75.	NomUsuario	**Nombre completo o razón social del usuario.**	String **(sng)**	**40**	NULL	**$**
76.	PassWord	**Contraseña de acceso al sistema del usuario.**	String **(sng)**	**40**	NULL	**$**
77.	TipoSeguridadPK	**¶ Llave primaria de la tabla TipoSeguridad.**	Integer **(int)**	**2**	NOT NULL	**%**
78.	FotoUsuario	**Fotografia del usuario para una identificación visual.**	Usuario **(udt)**		NULL	**$**
Nombre de la entidad: TipoSeguridad		**Atributos: 1:1**				
79.	TipoSeguridadPK	**¶ Llave primaria de la tabla TipoSeguridad.**	Integer **(int)**	**2**	NOT NULL	**%**
80.	DescTipoSeguridad	**Descripción del tipo de seguridad o nivel de usuario.**	String **(sng)**	**40**	NULL	**$**
Nombre de la entidad: Perfil		**Atributos: 2:0**				
81.	MenuPK	**¶ Llave primaria de la tabla de la tabla Perfil.**	Integer **(int)**	**2**	NOT NULL	**%**
82.	TipoSeguridadPK	**¶ Llave primaria de la tabla TipoSeguridad.**	Integer **(int)**	**2**	NOT NULL	**%**
Nombre de la entidad: Menu		**Atributos: 1:3**				
83.	MenuPK	**¶ Llave primaria de la tabla de la tabla Perfil.**	Integer **(int)**	**2**	NOT NULL	**%**
84.	OpcMenu	**Opciones de menu disponibles para un perfil de usuario.**	Integer **(int)**	**2**	NOT NULL	**%**
85.	DescMenu	**Descripción del menu disponibles para un perfil.**	String **(sng)**	**40**	NULL	**$**

Fecha:	**Reviso:**	**Autorizo:**

3.10. DER.

Fue propuesto por Peter Chen en 1976. Es usado como el modelo sobre el cual se soporta el diseño de una base de datos permitiendo crear un esquema de datos en términos de entidades, sus atributos y las relaciones existentes entre dichas entidades.

Los componentes de este modelo son:

Entidades. Son objetos o unidades funcionales especificas, que resultan de interés y que describen en forma gráfica la composición del sistema. Se dibujan en forma de bloque y se recomienda que el nombre que las describa sea en singular (figura 3.6.) Por ejemplo, los departamentos de una empresa, las áreas de un negocio o las etapas de un proceso de manufactura. En el sistema que se presenta en este trabajo las entidades funcionales son, empresa, clasificación (de productos), producto, proveedor, pedido, detalle del pedido, cliente, saldo, pago y precio.

Fig 3.6 Entidad.

Atributos. Son aquellas características o cualidades que describen a las entidades. Un atributo identificador es aquel que determina de manera única a cada ocurrencia de la entidad. Mientras que un atributo descriptor constituye una característica o cualidad de la entidad. En el sistema, la entidad que se refiere a los clientes del negocio, los atributos son, cliente PK (identificador o llave primaria), nombre del cliente, domicilio, RFC y teléfono (figura 3.7.)

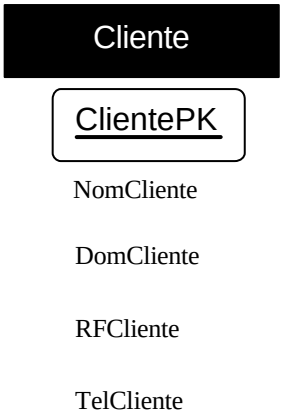

Fig. 3.7 Atributos.

Relaciones. Son los vínculos que existen entre las entidades. Representan la forma en que están involucradas en el contexto de un modelo gráfico. Se producen cuando cuales quiera dos entidades comparten alguna correspondencia. Poseen un nombre que las identifica en forma univoca y una propiedad llamada obligatoriedad que describe el sentido en forma cuantitativa de dicha correspondencia. Existen los tipos, uno a uno (el menos común), de uno a muchos (o viceversa), de muchos a muchos y dos tipos especiales, uno llamado opcional y el otro unitario.

Modelación gráfica de datos. El propósito de un diagrama entidad–relación (DER) es mostrar en forma esquemática las entidades que componen la estructura del sistema y de señalar cómo éstas se hallan relacionadas entre sí.

Para construir un DER, la pregunta inicial es ¿Cuáles son las entidades que resultan de interés para el sistema? Una vez contestada ésta, la siguiente pregunta es ¿Qué relaciones existen entre cuales quiera dos entidades, de las que definieron en la pregunta inicial?

Para definir plenamente la relación existente entre dos entidades específicas se deben hacer un par de preguntas; la primera de izquierda a derecha y la segunda de derecha a izquierda.

Por ejemplo, en un sistema que exista la relación cliente–factura (figura 3.8), las preguntas que definen esta relación son ¿Un cliente, cuantas facturas puede tener? Muchas, y ¿Una factura, cuantos clientes puede tener? Uno. Queda claro que esta relación es del tipo uno a muchos.

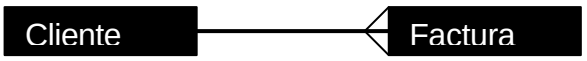

Fig. 3.8 Relación uno a muchos.

Después se deberá analizar si la relación es obligatoria (figura 3.9.) Se evalúa en ambos sentidos a través de dos preguntas, primero ¿Una factura puede emitirse sino existe el diente? No, es decir, es obligatorio que exista el cliente, ¿Puede un cliente existir sin que se le hayan emitido facturas? Si, es decir, no es obligatorio que todos los clientes tengan facturas. La obligatoriedad se representa así:

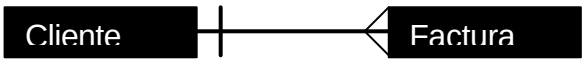

Fig. 3.9 Obligatoriedad.

Para ilustrar ahora un ejemplo de una relación del tipo muchos a muchos, se propone la relación producto–factura (figura 3.10.) Del mismo modo que en el ejemplo anterior, se hacen un par de preguntas ¿Un producto, en cuántas facturas puede aparecer? En muchas, y ¿En una factura, cuantos productos pueden aparecer? Muchos. Ahora, para determinar la obligatoriedad, se hacen otro par de preguntas ¿Una factura puede existir sin productos? No, porque existe obligatoriedad, es decir, se hace necesario que exista un producto y ¿Un producto puede existir sin que exista la factura? Sí, ya que no existe obligatoriedad.

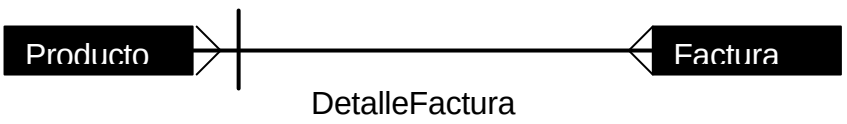

Fig. 3.10 Relación muchos a muchos.

Para continuar con este mismo tipo de relaciones, se expone a continuación la relación producto–proveedor (figura 3.11.) Háganse las siguientes preguntas ¿Un producto, cuantos proveedores puede tener? Muchos, y ¿Un proveedor, cuantos productos puede tener? Muchos. En orden de determinar la obligatoriedad, respóndanse estas dos preguntas ¿Un proveedor puede existir sin producto? Sí, ya que no existe obligatoriedad, y ¿Un producto puede existir sin un proveedor? Sí, ya que no existe obligatoriedad.

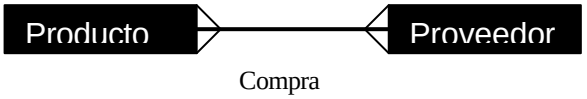

Fig. 3.11 Relación muchos a muchos.

El diagrama de la figura 3.12 ejemplifica las dos últimas relaciones descritas.

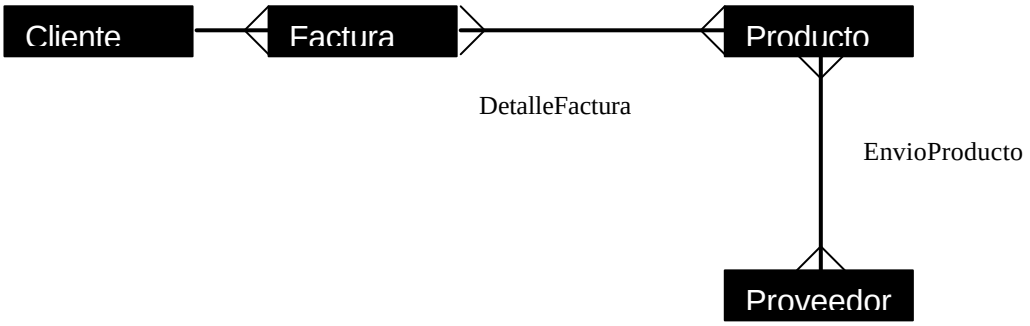

Fig. 3.12 Relaciones de muchos a muchos.

Ahora bien, según el paradigma de construcción de un DER, para romper una relación del tipo de muchos a muchos, se ordena crear una tercera entidad entre ambas, que adoptará el nombre que tenia la relación, y deberá cumplir con la siguiente condición; la nueva entidad creada tendrá, una relación de uno a muchos con las entidades originales (el lado de muchos estará al lado de la nueva entidad y el lado de uno tendrá que ser obligatorio con las entidades originales.) La condición que se acaba de describir se ilustra en la figura 3.13.

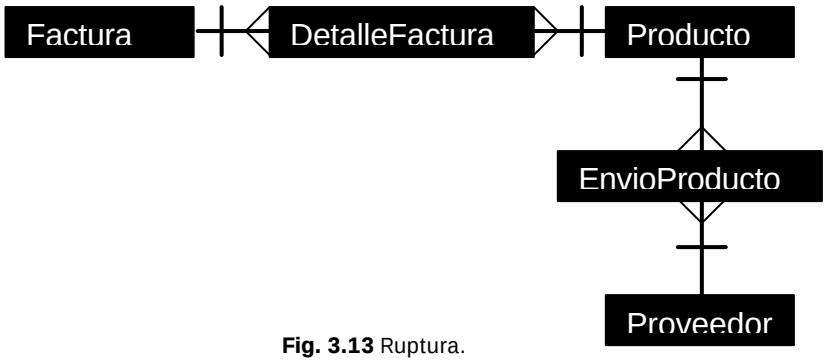

Fig. 3.13 Ruptura.

El diagrama de la figura 3.14 muestra todas las entidades y relaciones hasta ahora descritas.

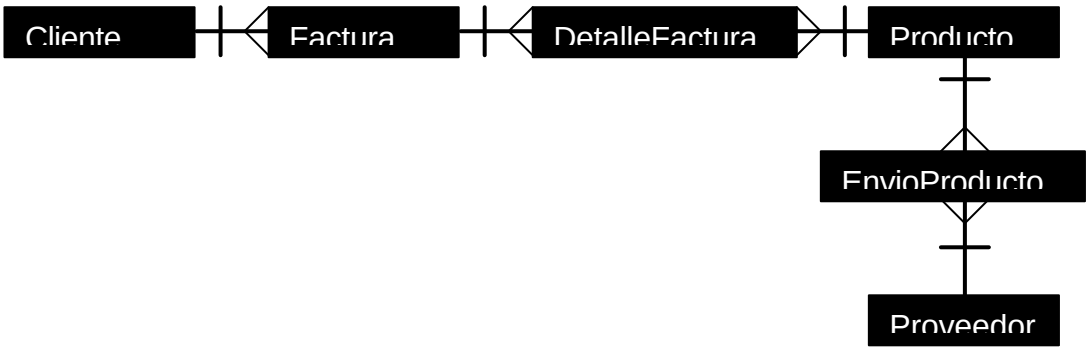

Fig. 3.14 Diagrama entidad–relación.

Existe también un tipo de relación llamada opcional. Se llama así, aquella en la que dos entidades están relacionadas, en todas, en una o ninguna de sus ocurrencias. Por ejemplo, considérese un sistema en el que existe la relación empleado–proyecto (figura 3.15.) Es posible entonces describir esta relación diciendo que un empleado puede estar asignado a muchos proyectos, a uno sólo o en su defecto, a ninguno, y un proyecto puede tener asignados muchos empleados, uno sólo o ninguno.

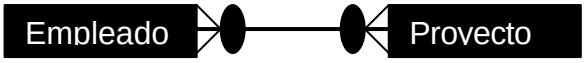

Fig. 3.15 Relación opcional.

Se describirá a continuación un tipo de relación llamada de uno a uno. Ésta se describe como aquella donde el atributo identificador es el mismo para ambas entidades relacionadas. Figúrese que existiera la relación división–gerente (figura 3.16), ésta podría ser descrita diciendo que una división es atendida sólo por un gerente, del mismo modo que un gerente atiende únicamente a una división.

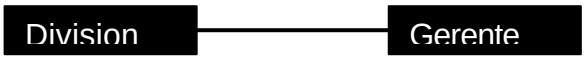

Fig. 3.16 Relación de uno a uno.

También es posible definir un tipo especial de relación muy poco común, llamada relación unitaria. Ésta se describe como aquella en la cual una entidad cualquiera está relacionada consigo misma. Analícese la entidad empleado (figura 3.17) en forma aislada, estará de acuerdo, que un empleado es jefe de cero o más empleados, y cualquier empleado tiene al menos un jefe.

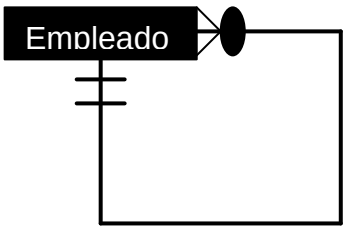

Fig. 3.17 Relación unitaria.

Por todo lo anteriormente expuesto, es posible concluir que el DER, apoya el análisis de sistemas, ayuda a definir los requerimientos de la empresa, describe la información acerca de las entidades y las relaciones requeridas para modelar estos mismos requerimientos, ayuda a determinar los tipos de transacciones que se busca ejecutar en la base de datos, permite conceptuar el diseño del nuevo sistema, también mejora las habilidades del diseñador de la base de datos y facilita el precisar los requerimientos de información del mundo real de una manera más precisa. Sin mencionar que define la semántica de las relaciones entre los datos.

CAPÍTULO IV

IMPLEMENTACIÓN DEL SISTEMA

4.1. Descripción.

El sistema ha sido diseñado para cumplir con una serie de propósitos generales. Primeramente posee un mecanismo de control de acceso restringido a los usuarios, incluyendo la definición de niveles de usuario, perfiles de seguridad y menús disponibles. Efectúa cortes diarios de pedidos. Implementa un control de chóferes repartidores. Permite que los clientes hagan sus pedidos en forma más rápida, cómoda y practica. Integra un control de créditos, que incluye pagos parciales. Posee la habilidad de expedir recibos al cliente, con información detallada de los precios y los productos que adquiere. Permite modificar el precio de los productos según las tendencias del mercado y las condiciones de crédito. Integra los catálogos de clientes, proveedores y productos. Y además genera reportes detallados de los viajes entregados por repartidor, del total de los pedidos hechos y de los productos enviados en cada pedido.

4.2. Diagrama de flujo.

La figura 4.1 muestra el diagrama de flujo de información en cual está fundamentado el desarrollo del proyecto. Nótese que existe una relación directa entre el sistema y el cliente, que es quien ordena los pedidos y obtiene a cambio un recibo, para comprobar dicha transacción. Por otro lado, es el contador de la empresa, quien ahora tiene la posibilidad de obtener cortes de pedidos, en el momento que lo desee y para el periodo que lo requiera, sin devolver nada cambio, en virtud que su puesto es de tipo gerencial y la información obtenida se torna significativa en el momento que se vuelve una herramienta valiosa para la gestión y administración del negocio.

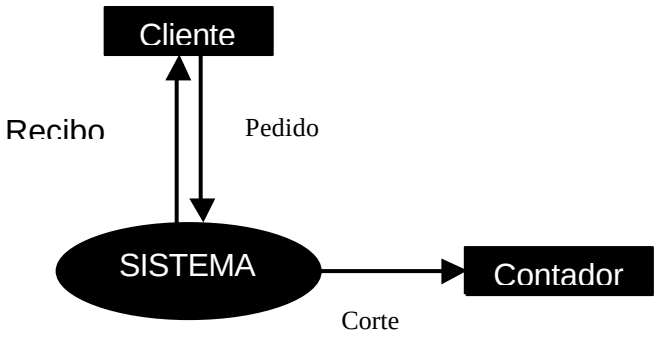

Fig.4.1 Diagrama de flujo de información del sistema.

4.3. Base de datos.

En la figura 4.2 se presenta el modelo conceptual del diseño de la base de datos a la que tendrá acceso el sistema, misma que está desarrollada en Microsoft SQL. Véase que existe una tabla llamada cliente, en ella se irá almacenando e integrado lo que se conoce como catálogo de clientes. Se creó de modo ex profeso una tabla llamada clasificaciones, puesto que se hacia necesario que al momento de ingresar un producto, éste pudiera pertenecer a una categoría específica, misma que tiene una estrecha relación con la tabla precios. En forma parecida a la tabla clientes, existe una tabla productos, misma donde se guardara y generara el catálogo de productos, que al igual que aquel eventualmente irá creciendo. Existe también la tabla pedidos que contendrá un registro histórico de todos los pedidos realizados. Los distintos tipos de precios se guardan en una tabla llamada precios. Por su parte la tabla pago, contendrá un registro de los abonos realizados para liquidar el monto total a pagar de los pedidos o para disminuir el saldo pendiente de los mismos.

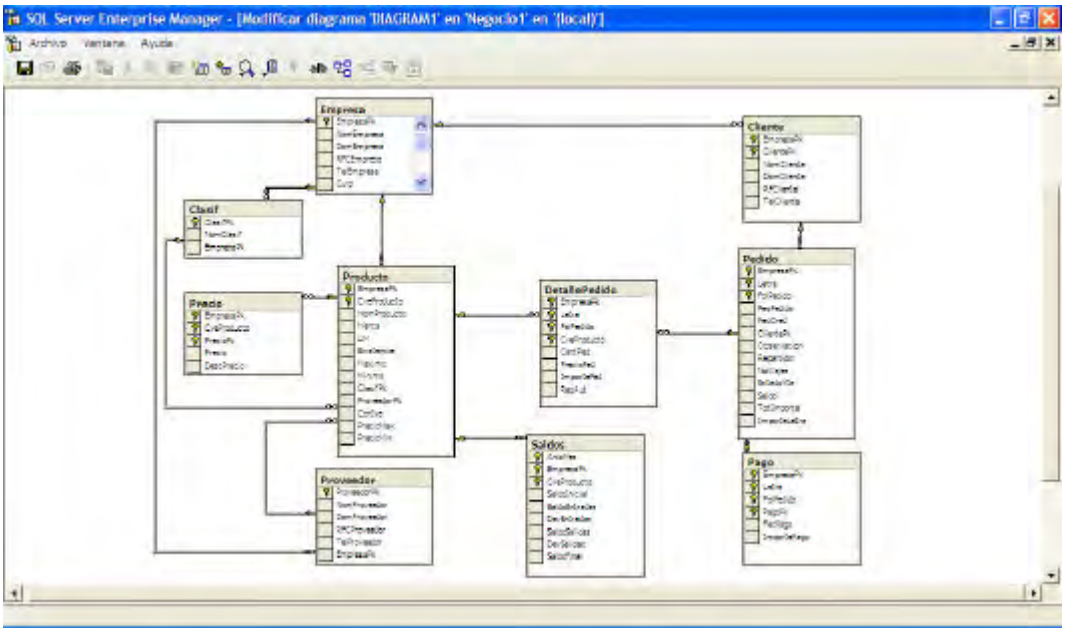

Fig.4.2 Modelo conceptual del diseño de la base de datos del sistema.

4.4. Base de datos para la seguridad.

En forma análoga a como se mostró el diseño de la base de datos del sistema y dado que estarán íntimamente relacionadas, la figura 4.3 presenta el modelo conceptual del diseño de la base de datos de la seguridad del sistema. Entiéndase pues que todas las habilidades que el sistema posee en lo relativo al control de acceso, la definición de niveles de usuario y determinación de perfiles de seguridad serán controladas gracias a los registros que existan en esta base de datos.

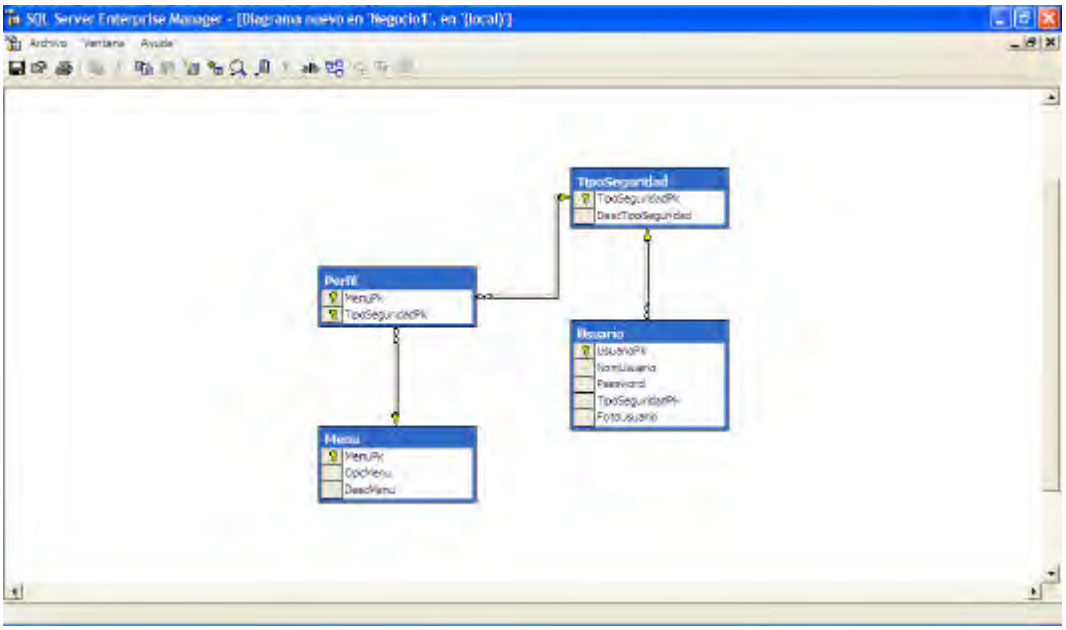

Fig.4.3 Modelo conceptual de la base de datos para el manejo de la seguridad en el sistema.

4.5. Validación de entrada.

El sistema cuenta con un control de acceso restringido a usuarios o dicho de otro modo, con una validación de entrada al sistema, cuyo propósito principal es hacer que cada usuario ingrese con su nombre y contraseña, de manera que se cargue en forma automática el perfil de trabajo reservado para éste. Opcionalmente podrá seleccionar la sucursal a la que pertenece. En este momento esta opción no tiene repercusión alguna, pero se planea que en un futuro el sistema sea diseminado y esté en condiciones de distinguir entre una sucursal y otra (vea la figura 4.4.)

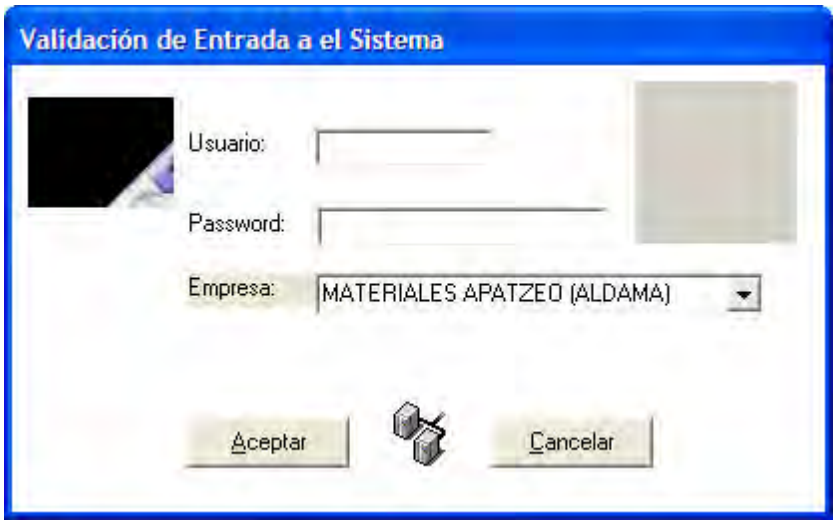

Fig.4.4 Validación de entrada al sistema.

4.6. Catálogo de tipos de seguridad.

Este catálogo sirve para clasificar los distintos tipos de seguridad (desde capturista hasta administrador del sistema) de acuerdo al nivel de funcionalidad y operabilidad que se les haya asignado dentro del esquema de trabajo del sistema, y de acuerdo con esta clasificación les serán otorgados los permisos necesarios, para tener disponibles las distintas opciones del menú que existen. (Figura 4.5.)

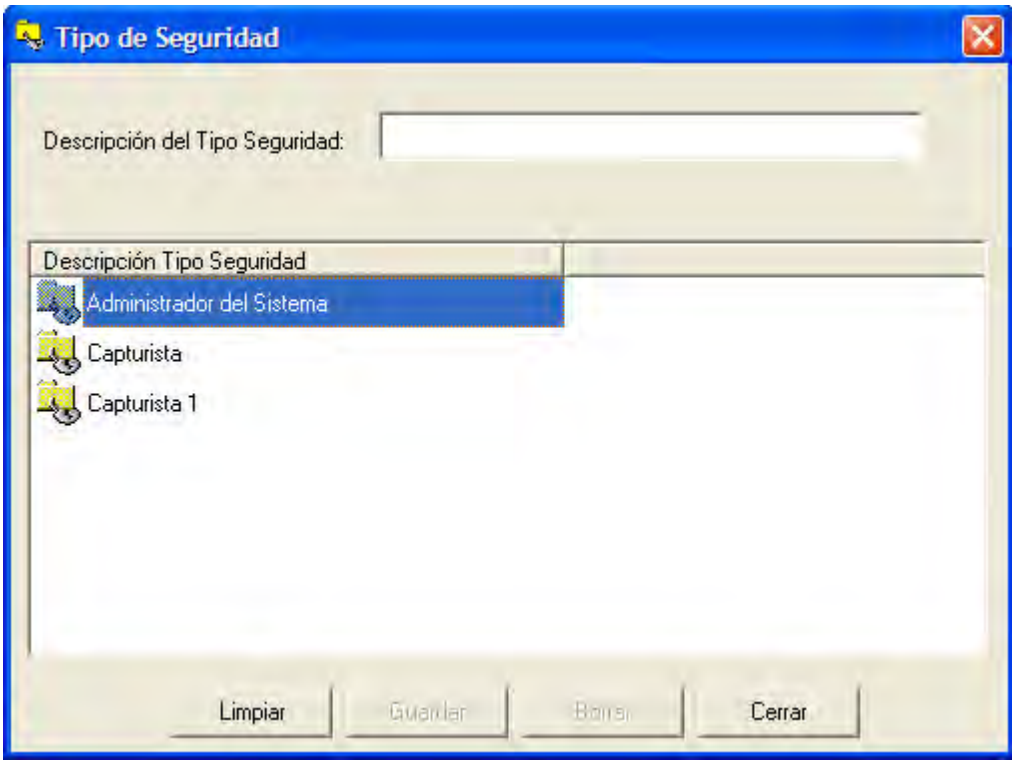

Fig.4.5 Catálogo de tipos de seguridad.

4.7. Catálogo de usuarios.

El presente catálogo permite dar de alta los usuarios, asignándoles un nombre, una contraseña y un tipo de seguridad, para de este modo hacer que se carguen los permisos necesarios relacionados con cada uno de los tipos preestablecidos. (Obsérvese la figura 4.6.) Adicionalmente a cada usuario se le podrá asignar una fotografía como parte de la personalización de sus cuentas.

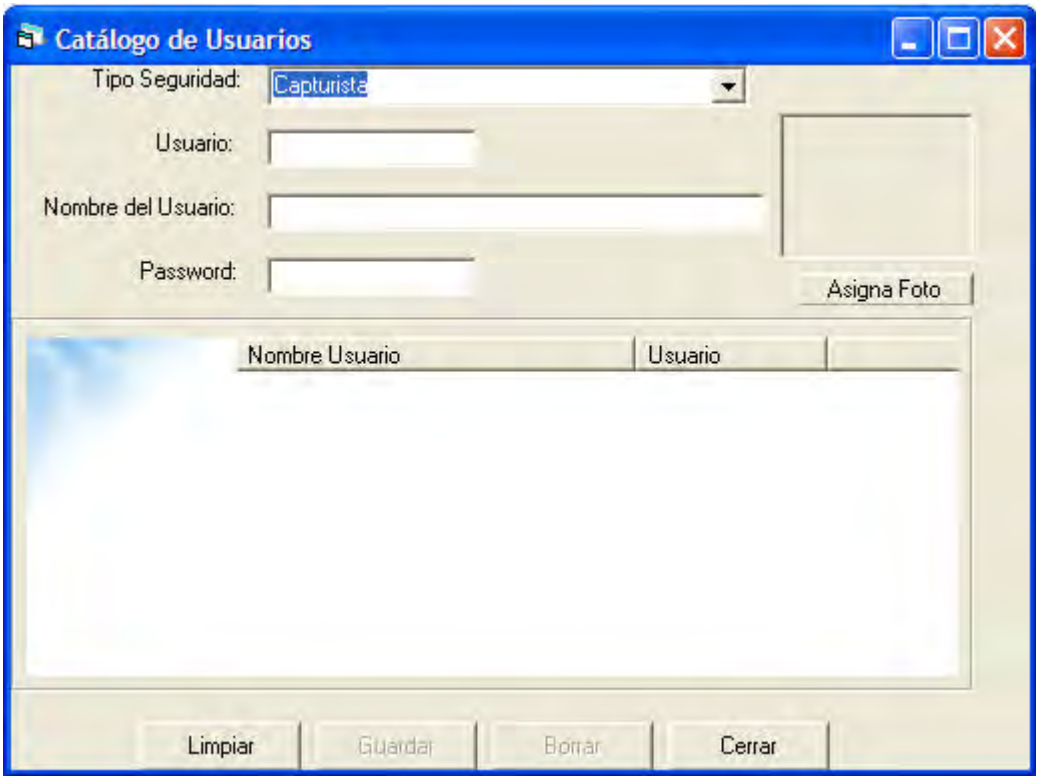

Fig.4.6 Catálogo de usuarios.

.

4.8. Catálogo de menús.

Es facultad exclusiva del administrador del sistema dar de alta las diferentes opciones del menú con las que cuenta el sistema. Cada opción corresponde a un menú diferente que estará disponible según los tipos distintos de seguridad que se hayan creado anteriormente. (Figura 4.7.)

Fig.4.7 Catálogo de menús.

4.9. Catálogo de perfiles.

La figura 4.8 muestra como es posible definir el perfil para cada uno de los tipos de seguridad que se hayan creado de acuerdo con la sección 4.6. Primero, seleccione un tipo de seguridad en el combo de la parte superior de la ventana, arrastre alguna de las opciones disponibles en la lista de la parte izquierda y déjelas en la derecha.

Fig.4.8 Catálogo de perfiles de tipos de seguridad.

4.10. Catálogo de datos de la empresa.

En función de que una de las tareas para las que fue desarrollado el sistema es la de generar reportes de pedidos e imprimir recibos para el cliente, el catálogo de la figura 4.9 deberá contener los datos actualizados de la empresa, es decir, nombre, domicilio, RFC, teléfono y CURP. Adicionalmente el catálogo es capaz de validar una contraseña que permite cambiar los precios de los productos en forma discrecional.

Fig.4.9 Catálogo de datos de la empresa.

4.11. Catálogo de clasificaciones.

El propósito del catálogo que se observa en la figura 4.10 es el de crear, modificar o eliminar las distintas categorías o familias de productos que existen, cuya repercusión se vera en el catalogo de precios de la figura 4.12

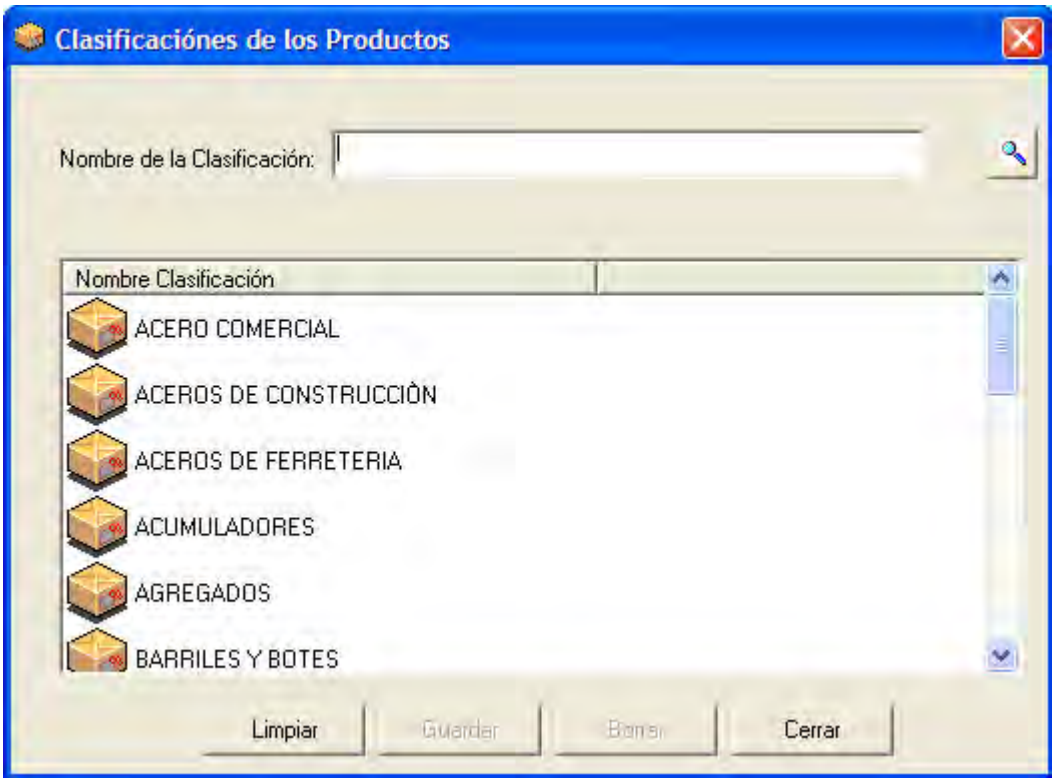

Fig.4.10 Catálogo de clasificaciones de los productos.

4.12. Catálogo de productos.

Eventualmente el catálogo de productos debe estar lo más completo posible. Véase la figura 4.11, en ella se muestra el catálogo que emplea el sistema, en éste mismo es posible crear, modificar o eliminar cualquier entrada de producto. Seleccione la clasificación deseada y el proveedor que la distribuye, asigne una clave al producto y una descripción al mismo, ingrese la marca y llene el campo de la unidad de medida, y si lo prefiere seleccione la casilla de IVA, para que aparezca esa leyenda en el precio del producto. Dé clic en el botón precios si desea ver los detalles relativos al precio y si su perfil se lo permite, podría hasta cambiar los distintos precios del producto en cuestión. (véase la figura 4.12.) Finalmente presione el botón de guardar. Los botones con binoculares sirven para hacer búsquedas más detalladas de las diferentes opciones en las que se encuentran.

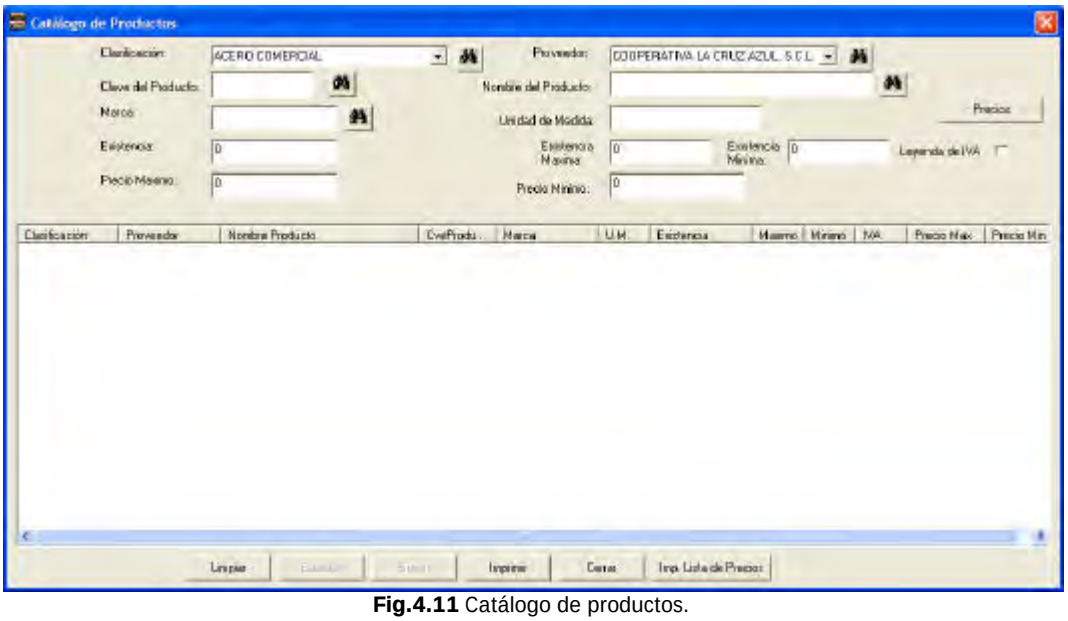

Fig.4.11 Catálogo de productos.

La figura 4.12 muestra la ventana de precios de un producto. Los distintos precios de un producto podrán ser asignado en forma discrecional, y en atención a las reservas de seguridad del sistema. La volatilidad de los precios podría verse afectada por algunos factores como las tendencias del mercado, el tiempo de traslado de los productos y las estrategias de mercadeo. Seleccione alguna de las opciones de precio que aparecen en la rejilla de la parte inferior o en su defecto llene las cajas de texto de la parte superior y presione el botón de guardar.

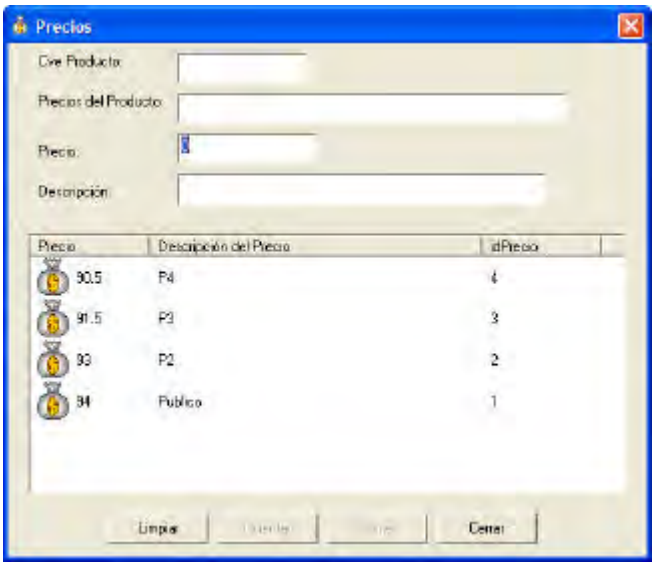

Fig.4.12 Precios.

4.13. Pedidos.

La columna vertebral del sistema, es el módulo de pedidos (figura 4.13.) Primeramente el sistema asigna un folio único a cada pedido, mismo que se renueva cada mes y que está integrado por una letra y un número consecutivo. Se muestra por defecto la fecha actual y queda seleccionada la opción de pedido, si se trata de un pedido a crédito, seleccione entonces la otra opción que dice crédito, ambas son mutuamente excluyentes. Comience a agregar los productos que ha ordenado el cliente y que se irán listando en la rejilla de la parte central de la ventana, llene el cuadro de texto de la parte inferior, para que en la nota de pedido aparezca la dirección donde los repartidores deberán entregar los productos. Automáticamente aparecerá la cantidad total a pagar. Presione F2 o F7 para ver o modificar los precios de los productos, (requiere contraseña.) Se imprimirá automáticamente el pedido.

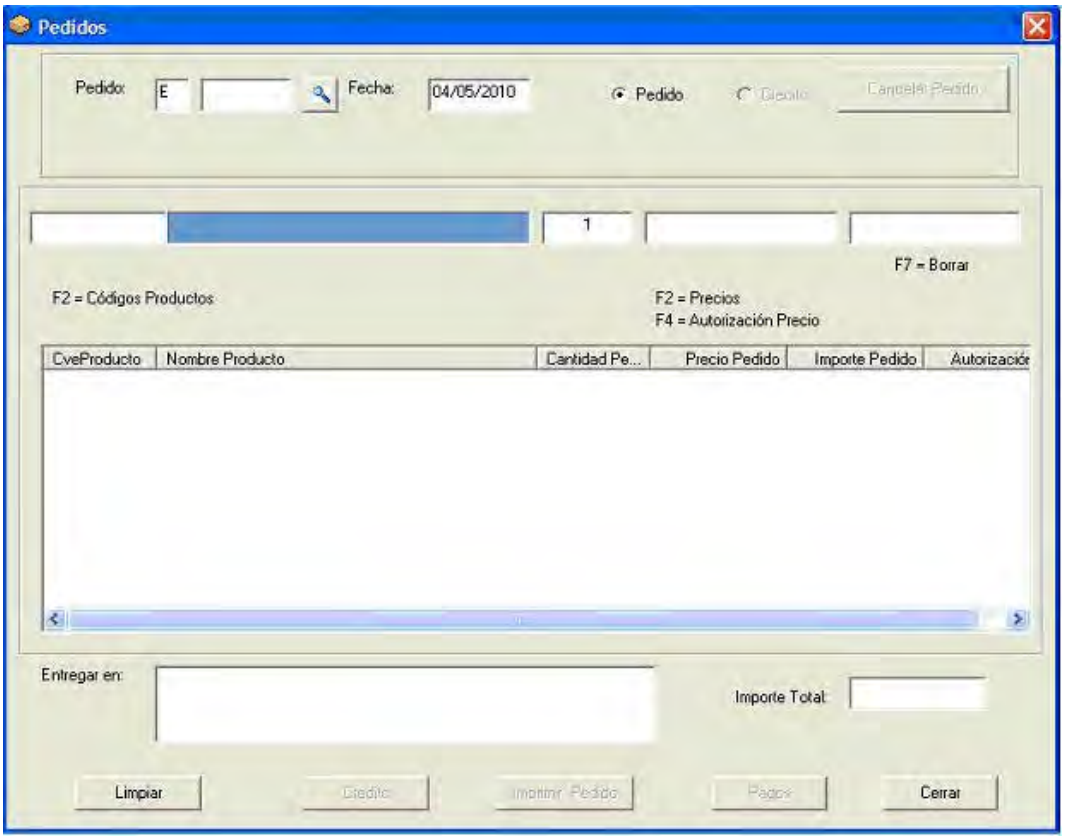

Fig.4.13 Pedidos.

4.14. Clientes, créditos y pagos.

Cuando se trate de pedidos a crédito se deberá dar de alta el cliente, e ir creando así un catalogo de clientes (figura 4.14.) Para agregar un cliente, llene las cajas de texto de nombre, domicilio, población y teléfono y presione el botón de guardar. También es posible eliminar un cliente, basta con seleccionarlo de la lista que aparece y dar clic en el botón de borrar. Recuerde que esta ventana está relacionada con el módulo de pedidos, en aquella presione el botón que dice crédito.

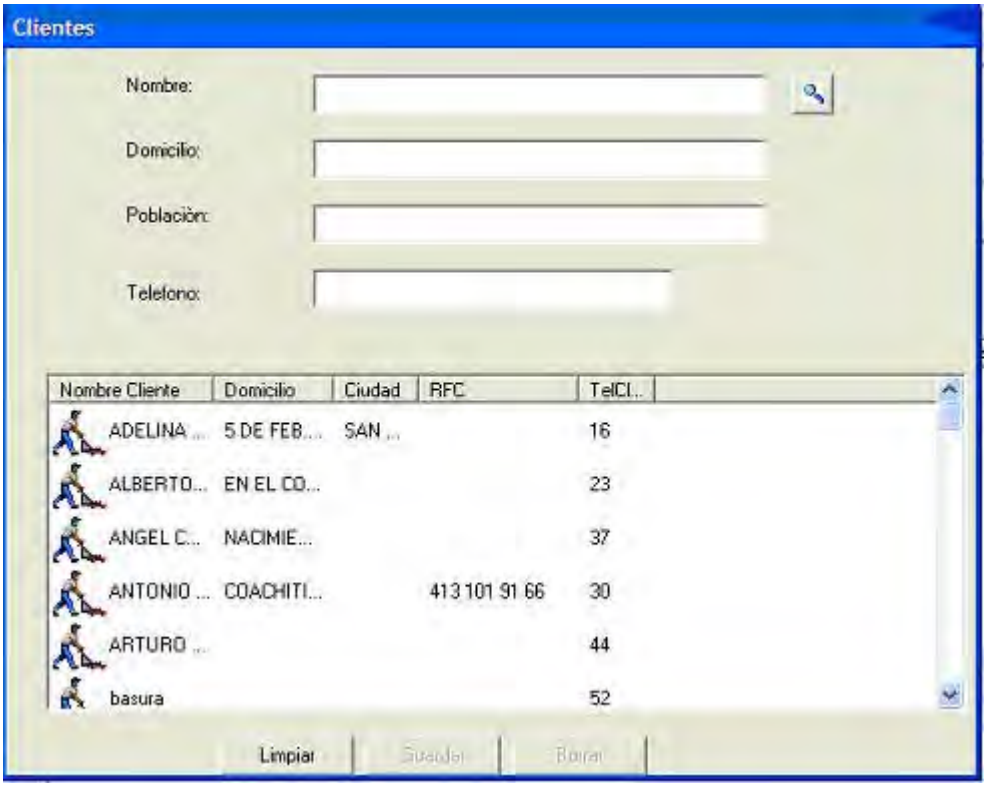

Fig.4.14 Clientes.

En el supuesto de que el cliente se encuentre dado de alta ya, dé doble clic sobre botón que se halla delante del nombre del cliente (figura 4.14), y aparecerá la ventana de la figura 4.15, donde se muestra una lista de los créditos y el folio de los pedidos que le pertenecen. Se agregan otros detalles, como la fecha, el importe original y el saldo.

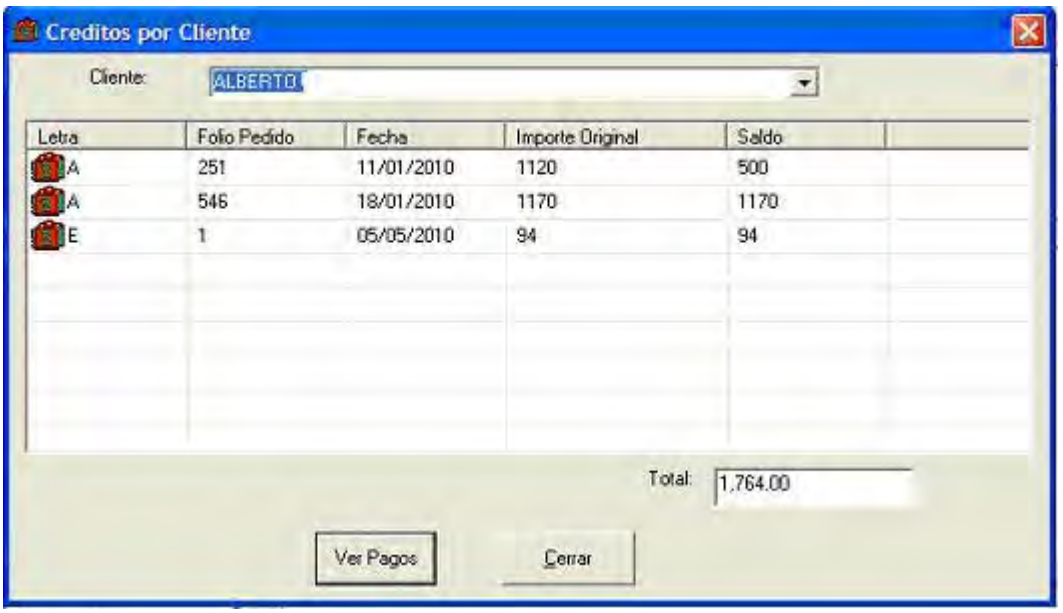

Fig.4.15 Créditos por cliente.

Seleccione el número de folio deseado y presione el botón ver pagos (figura 4.15), y se abrirá la ventana de la figura 4.16, donde se muestra el detalle de los pagos efectuados a un determinado pedido.

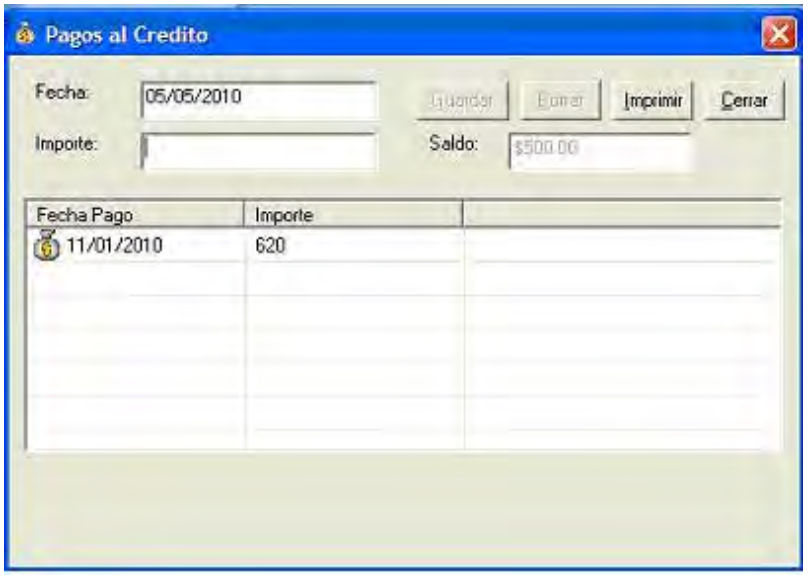

Fig.4.16 Pagos efectuados a los pedidos a crédito.

4.14. Viajes.

La comisión de los repartidores crece en proporción directa con la cantidad de pedidos repartidos. La ventana que aparece en la figura 4.17 sirve para generar una actualización diaria de la cantidad de notas de pedido que cada chofer efectuó y la comisión respectiva. Por defecto aparece la fecha actual, puede seleccionar otra fecha si lo desea, presione el botón que está delante del combo de fecha. Lo descrito hace un momento, en el sistema se conoce como actualización de viajes de repartidores. Presione por último el botón de imprimir, se obtendrá el reporte de la figura 4.18, donde aparecen agrupados, el nombre del repartidor, los folios de los pedidos que entregó, el número de viajes efectuados y la suma total de ellos. Como puede verse está hecho en Crystal Reports.

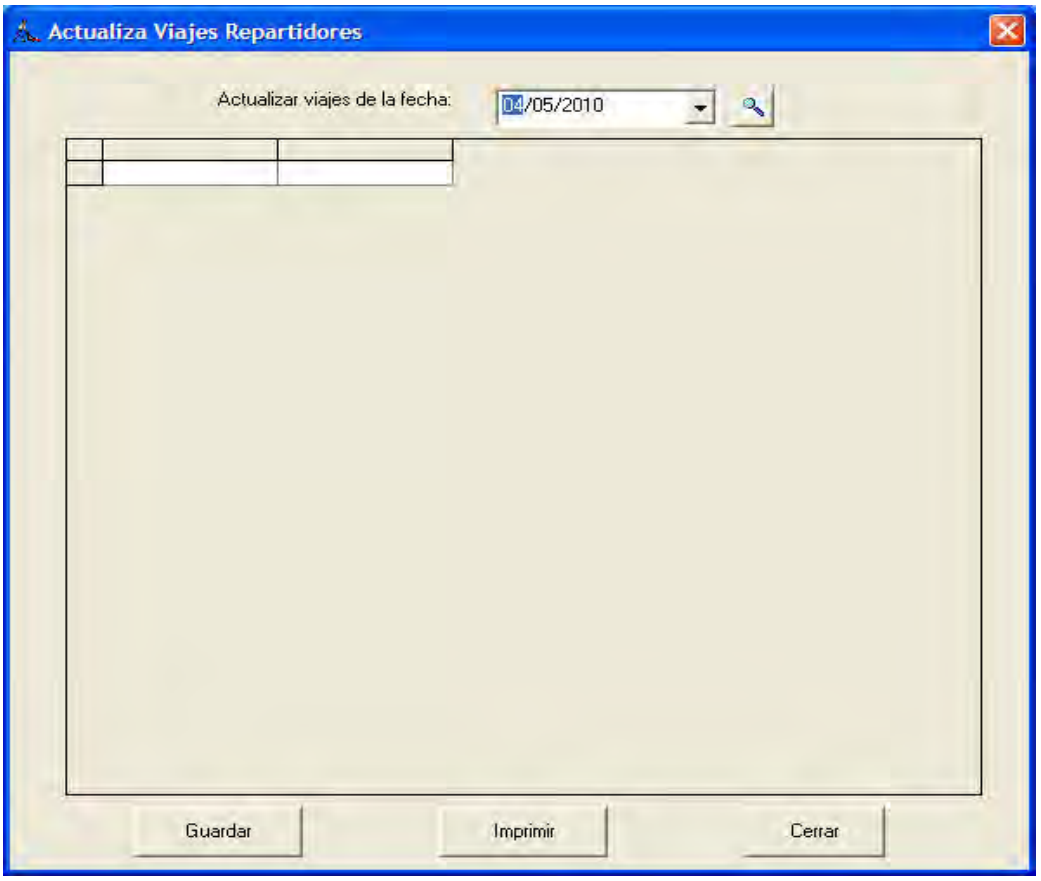

Fig. 4.17 Actualización de viajes de repartidores.

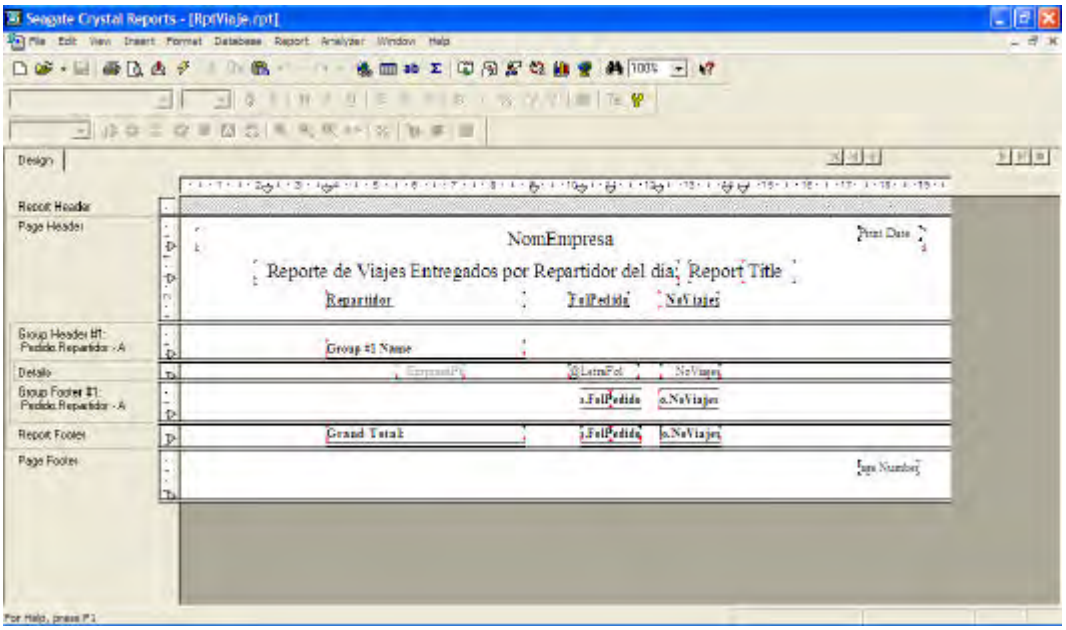

Fig. 4.18 Reporte de viajes entregados por repartidor.

4.15 Reporte de pedidos.

De forma parecida al reporte de la figura 4.18, el reporte de la figura 4.19 muestra el total de los pedidos efectuados, agrupados por número de folio y por fecha, separando los pedidos de crédito de los de contado.

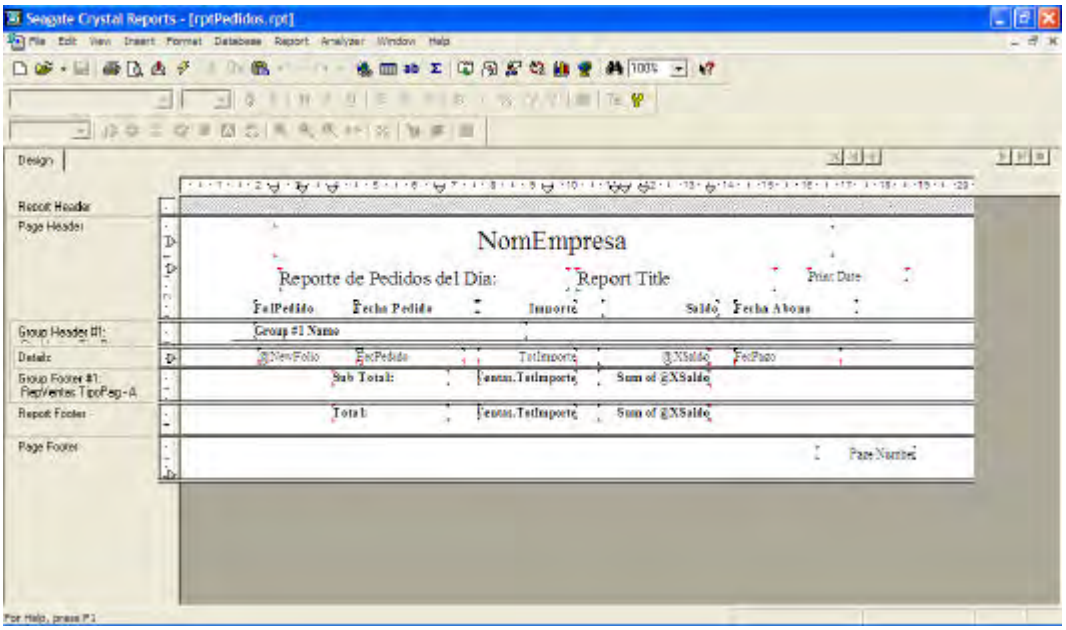

Fig. 4.19 Reporte de pedidos.

4.16. Reporte de pedidos por producto.

Igual como se presentaron los reportes de las figuras 4.18 y 4.19, se presenta ahora el reporte de la figura 4.20 donde se detallan el total de los pedidos realizados en un periodo determinado, pero agrupado por familias (clasificaciones) y por nombre del producto.

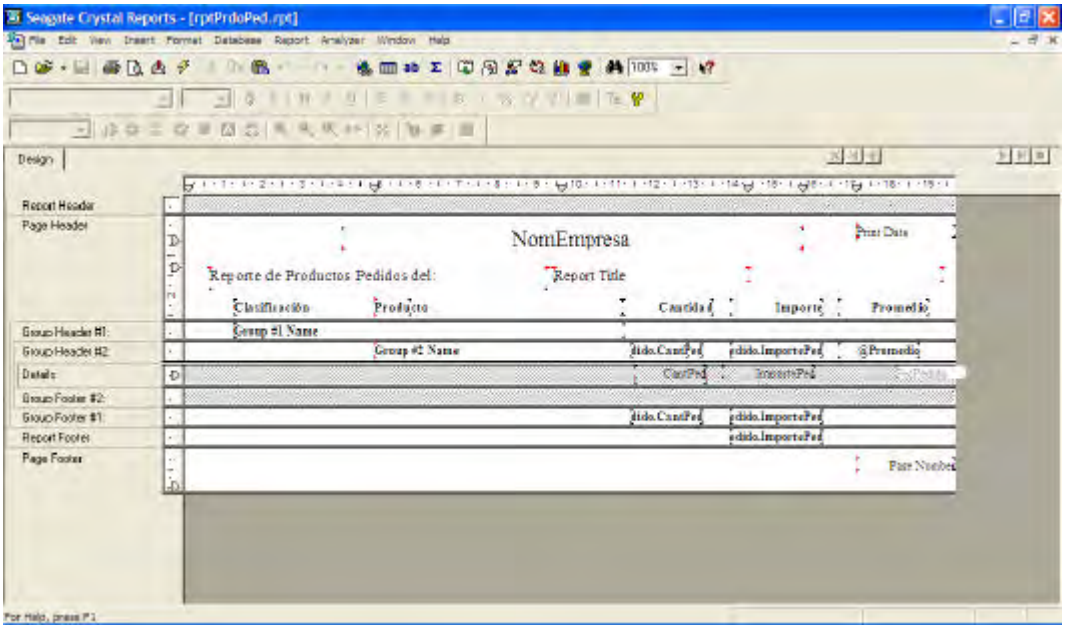

Fig. 4.20 Reporte de pedidos por producto**.**

CONCLUSIONES.

El tiempo es un factor primordial en todas las organizaciones y en todos los proyectos que éstas mismas desarrollan en un afán por conseguir el éxito al que todas ellas están obligadas. En el campo del desarrollo de sistemas el tiempo debe verse como un agente decisivo. La planificación temporal que en su momento realizan los analistas o los programadores resulta influenciada por la carga de trabajo de las organizaciones y la vorágine de operaciones diarias que se realizan, mismas que hacen imposible esperar meses o años a que una solución particular que se haya implantado comience a rendir frutos, de hecho, el periodo de transición de un nuevo sistema debe ser lo más breve posible, pues ésta es la característica que puede significar su adopción o su completo rechazo. En el negocio donde se trabajó, se le dijo al dueño que los pedidos se iban a comenzar a tomar a partir de la fecha de implantación.

Según la experiencia, las múltiples tareas que corresponden a un analista o a un programador de sistemas no debieran centrarse en la creación del código que ejecutarán las aplicaciones que lleguen a desarrollarse, sino encausar sus esfuerzos a las etapas de diseño, y no viceversa, como sucede comúnmente, dado que existe la provocativa tendencia de dedicarse inmediatamente a la codificación del sistema, restando importancia al análisis previo, lo que a la postre se traduce en un retrazo considerable por carecer de la visión global que un estudio de sistemas aporta. Este sistema revela que el código no debe ser medido en términos cuantitativos, sino cualitativos.

Cuando se está desarrollando una aplicación, debe hacerse uso de todas las tecnologías de software y hardware disponibles en el mercado y de aquellas que estén al alcance, según la medida de las posibilidades, tanto del programador, pero principalmente del negocio para el cual se está trabajando. La presente aplicación fue desarrollada en Visual Basic 6.0,

las bases de datos están hechas en SQL Server 2000 y los reportes se generaren Crystal Reports.

La práctica nos dice que Microsoft Visual Basic 6.0 permite crear, sin mayores problemas, aplicaciones realmente competitivas y soluciones a la medida. Se presenta entonces como la mejor y más extendida herramienta de programación. Está pensada para solucionar todos los requisitos que la actualidad demanda. VB desde su primera versión ha copado la inmensa mayoría de los desarrollos de sistemas del mundo, actualmente se puede decir que no sólo es un lenguaje de programación, sino una plataforma idónea para el mantenimiento y ampliación de todos los instrumentos que un sistema contiene. Gestión de base de datos (prácticamente cualquier tipo), animaciones graficas y aplicaciones multimedia. Ofrece la capacidad de simplificar el desarrollo de sistemas, por su característica de estar orientado tanto a objetos como a eventos. Por todo lo anteriormente expuesto se decidió que la opción mas viable para el negocio era trabajar en VB 6.0.

Cuando se trabaja en un proyecto como éste, es importante seguir una metodología de análisis y diseño. Es recomendable y permisible rediseñar el sistema cuantas veces sea necesario, apegándose siempre de forma estricta a las metodologías de trabajo existentes, en virtud de que el valor de éstas ha sido plenamente probado con anterioridad por otros autores, por lo tanto, deberá continuarse hasta que no quede completamente satisfecho el dueño del negocio que ordeno el sistema, es decir, se debe experimentar con diferentes soluciones para escoger aquella que mejor se adapte completamente a las necesidades persistentes de la empresa.

BIBLIOGRAFÍA.

1. Charte, Francisco, “Visual Studio.NET”, Ed. Anaya Multimedia, México, 2003, 819 pp.

2. Date, C.J., “Sistemas de bases de datos”, Ed. Addison Wesley Longman, México, 1993, 860 pp.

3. Fitzgerald, Jerry, “Fundamentos de Análisis de Sistemas”, Ed. Continental, México, 1989, 558 pp.

4. Gerez, Víctor, “Desarrollo y Administración de Programas de Computadora”, Ed. Continental, México, 1984, 299 pp.

5. Ramírez R., José Felipe, “Aprenda Visual Basic practicando”, Ed. Pearson Prentince Hall, México, 2001, 820 pp.

6. Ramírez R., José Felipe, “Arquitectura de Aplicaciones para .NET”, Ed. Microsoft, México, 2001, 819 pp.

7. Senn, James A., “Análisis y Diseño de Sistemas de Información”, Ed. McGraw Hill, México, 1990, 558 pp.

OTRAS FUENTES.

8. http://www.inei.gob.pe/biblioineipub/bancopub/Inf/Lib5038/indice.html.

9. http://www.lawebdelprogramador.com.mx.

10. http://www.microsoft.com.ar/argentina/technet.

11. http:// www.msdn.microsoft.com.es/es.

12. http://www.msdn.microsoft.com.es/es/library/aa291593(vs.71).aspx.

13. http://www.personales.unican.es/zorrillm/pdfs/docencia/basesdatos/instalar%20sqlserver.pdf.