BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

"Aplicación Cliente-Servidor usando Código de Barras para el Inventario de una Cadena de Tiendas."

Presenta:

Bernardo José Flores Amaro

Tesis sometida como requisito para obtener el grado de

Licenciado en Ciencias de la Computación.

Asesor:

Dra. Bárbara Emma Sánchez Rinza

Puebla Pue; Septiembre de 2009

A mis padres,

por todo…

Las mentes inquietas no se conforman con conocimientos incompletos porque quieren llegar al fondo de sus problemas, porque sienten el reto de sus talentos, porque en síntesis, buscan afanosamente la solución a los problemas y el deseo de explorar la maraña de las relaciones entre los fenómenos.

Arias Galicia.

Tabla de contenido

Objetivo General

Crear una Aplicación Cliente/Servidor usando código de barras para el inventario de una cadena de tiendas (Tienda de deportes).

Objetivo Especifico De La Tesis

- La aplicación cliente servidor ayudara a llevar el control de los artículos vendidos y las que se tienen en existencia.
- Tener el control de la hora de entrada y salida de los empleados, así como la de los días laborados.

Introducción De La Tesis

En el medio del comercio hay ciertos problemas y los hay más cuando se tiene una cadena de tiendas de deporte, pues no se sabe como poder llevar el control de todos sus artículos, analizando detalle a detalle cada uno de los problemas se opto en crear una aplicación utilizando tecnología reciente (Php y MySql) el cual, sus características principales son los siguientes:

Crear una base de datos en Php junto con MySql para los artículos, empleados, proveedores, usuarios, ventas entre otros; en cada una de estas ramas que tendrá la base de datos se podrá hacer inserción, eliminación, consultas, y modificaciones, a lo largo de la tesis explicaremos a detalle las funciones y características de la aplicación.

Una de las características más importantes es que se creara una aplicación cliente-servidor, ya que con esto el empresario le será mucho más fácil ver las existencias de cada una de las tiendas, y en caso de que algún artículo este agotado este se le lleve y anexe lo propio.

Capítulo I.

Código de Barras

1.1. Introducción

En este capítulo se muestra los aspectos importantes de código de barras, además la documentación de la Base de Datos de la cadena de tiendas, así como una descripción de sus características y principios funcionales.

El **código de barras** es un código basado en la representación mediante un conjunto de líneas paralelas verticales de distinto grosor y espaciado que en su conjunto contienen una determinada información. De este modo, el código de barras permite reconocer rápidamente un artículo en un punto de la cadena logística y así poder realizar inventario o consultar sus características asociadas. Actualmente, el código de barras está implantado masivamente de forma global.

La correspondencia o mapeo entre la información y el código que la representa se denomina *simbología*. Estas simbologías pueden ser clasificadas en dos grupos atendiendo a dos criterios diferentes:

1) ***Continua o discreta*:** los caracteres en las simbologías continuas comienzan con un espacio y en el siguiente comienzan con una barra (o viceversa). Sin embargo, en los caracteres en las simbologías discretas, éstos comienzan y terminan con barras y el espacio entre caracteres es ignorado, ya que no es lo suficientemente ancho.
2) ***Bidimensional o multidimensional*:** las barras en las simbologías bidimensionales pueden ser anchas o estrechas. Sin embargo, las barras en las simbologías multidimensionales son múltiplos de una anchura determinada (X). De esta forma, se emplean barras con anchura X, 2X, 3X, y 4X.

1.2. Descripción

El código de barras es un sistema para identificar objetos y recoger datos automáticamente.

El código de barras es una tecnología que ha transformado la manera de hacer negocios. Desde su creación, ha facilitado muchas tareas tediosas de administración como son el levantamiento de inventarios, el registro de las mercancías vendidas y ha proporcionado más agilidad en la atención a los clientes. Al tener información confiable, recolectada en el punto de captura, los gerentes pueden tomar decisiones más oportunas sobre el manejo del negocio. Por ejemplo, que mercancía debe comprar, cuales Están obsoletas, cuales se deben de rematar.

Se ha convertido en una herramienta indispensable en la vida cotidiana. Gracias a la tecnología de código de barras se puede saber exactamente la localidad de un paquete durante todo el trayecto - Desde la recolección, pasando por todas las etapas de transporte, hasta la entrega final. Posiblemente, se ha dado cuenta, que en muchas empresas se registra el tiempo de los empleados por medio de tarjetas de identificación con código de barras. Esta tecnología es tan "invisible" que se ha involucrado en todas las actividades cotidianas. Ya la damos por hecho en nuestras vidas que es imposible pensar como viviríamos sin ella.

Los principales beneficios de esta tecnología son la sencillez de operación, la velocidad de captura, la confiabilidad de los datos, el uso de estándares establecidos y el bajo costo.

1.3. Nomenclatura Básica

- **Módulo:** Es la unidad mínima o básica de un código. Las barras y espacios están formados por un conjunto de módulos.
- **Barra:** El elemento (oscuro) dentro del código. Se hace corresponder con el valor binario 1.
- **Espacio:** El elemento (claro) dentro del código. Se hace corresponder con el valor binario 0.
- **Carácter:** Formado por barras y espacios. Normalmente se corresponde con un carácter alfanumérico.
- Densidad: Es la anchura del elemento (barra o espacio) más angosto dentro del símbolo de código de barras. Está dado en mils (milésimas de pulgada). Un código de barras no se mide por su longitud física sino por su densidad.
- WNR: (Wide to Narrow Ratio) Es la razón del grosor del elemento más angosto contra el más ancho. Usualmente es 1:3 o 1:2.
- Quiet Zone: Es el área blanca al principio y al final de un símbolo de código de barras. Esta área es necesaria para una lectura conveniente del símbolo.

ESTRUCTURA:

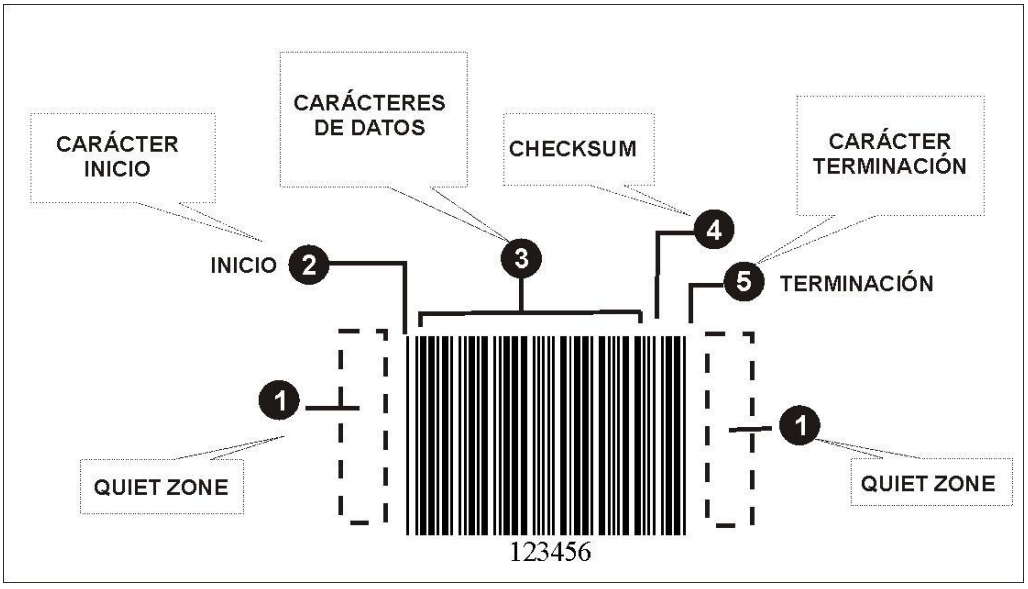

FIGURA 1.3.1. Estructura Código de Barras.

1.4. El Origen del Código de Barras

La primera patente de código de barras fue registrada por Joseph Woodland y Bernard Silver en 1952. En 1948 era estudiante en el Instituto Tecnológico Drexel de Filadelfia. Un empresario del sector de la distribución de alimentos había pedido al Instituto Drexel que desarrollara un sistema para leer automáticamente los datos de los productos. Bernard Silver y Norman Joseph Woodland trabajaron juntos en ese proyecto. Woodland inicialmente pensó utilizar una tinta sensible a los rayos ultravioleta e incluso desarrollaron un prototipo, pero consideraron que el resultado no era suficientemente satisfactorio porque era demasiado inestable y demasiado caro, por lo que volvieron a empezar con un nuevo desarrollo. En 1949 patentaron lo que denominaron "método de clasificación de productos mediante reconocimiento de formas". El código de barras se utilizó comercialmente por primera vez en 1966, pero pronto se dieron cuenta de que era necesario definir un estándar para el sistema. En 1970 se creó el primer estándar como código universal de identificación de los productos de las tiendas de alimentación. Luego el uso se extendió y se creó un estándar universal de identificación de productos.

Se estima que en el mundo se realizan cerca de 5 mil millones de lecturas de códigos de barra al día.

1.5. Un Poco de Historia

Actualmente, GS1 define a esta herramienta como "una identificación única y no ambigua de un producto, localización y/o servicio que se representa mediante un símbolo compuesto por barras oscuras y espacios claros paralelos entre sí y de anchos variables".

En concreto, hablamos de 3 grandes tipos códigos. **Los numéricos**, utilizados para identificar productos de venta directa al público; **los alfanuméricos**, compuestos por números y letras con el fin de entregar información adicional como el número de lote

o la fecha de elaboración y **los bidimensionales**, que han empezado a usarse en documentos que requieren el envío de mensajes más grandes, como un expediente clínico completo.

Pero el primer código de barra de la historia, patentado por Noman Woodland y Bernard Silver al finalizar la década del '40, poco se parecía a los actuales. Estaba hecho a base de círculos concéntricos y era leído por una foto detectora. Todavía no aparecía la identificación estándar (1972) capaz de contener una información única y restaban bastantes adelantos tendientes a perfeccionar los aparatos capaces de leer sin problemas la información que ellos contenían.

En 1961 apareció el primer escáner fijo de códigos de barras instalado por Sylvania General Telephone en Estados Unidos. Este aparato leía barras de colores rojo, azul, blanco y negro identificando vagones de ferrocarriles. Siete años después, la Asociación de Ferrocarriles de ese mismo país utilizó este sistema para el control de tránsito de embarques. El proyecto no duró mucho por falta de adecuado mantenimiento de las etiquetas conteniendo los códigos.

En 1969 se incorporó la tecnología láser y se hizo interactuar el lector de códigos con un computador. Los programas creados ejecutaban principalmente funciones de mantenimiento de inventarios. De este mismo período datan las primeras aplicaciones industriales, pero sólo para manejo de información.

En la siguiente década se crea el primer terminal portátil o lápiz de contacto, y el código Plessey hace su aparición en Inglaterra para control de archivos en organismos militares y control de documentos en bibliotecas.

Otros sistemas utilizados a principios de la década del '70 fueron el código Codabar (1971), que encontró su mayor aplicación en los bancos de sangre, y la identificación automática que utilizó la fábrica de automóviles Buick en sus operaciones de ensamble. En 1974 se inventa el primero de tipo alfanumérico y casi 10 años después se desarrolla el primer código bidimensional.

1.6. Código Estándar

Pero sin duda 1973 fue un hito en la historia de los códigos de barra. En ese año se creó en Estados Unidos el Universal Product Code (U.P.C), que se convertiría en el estándar de identificación más usado en ese país y Canadá. Este nuevo código que contaba con 12 dígitos surgió a través de un organismo llamado Uniform Code Council (UCC), que desarrolló un sistema para que fuera aplicado sólo a nivel nacional. Dado que los resultados fueron positivos, algunas entidades establecidas en Europa adoptaron el programa, pero agregándole un dígito más.

De ahí nació la idea de crear un organismo que liderara la difusión de un sistema estándar global, con el propósito de facilitar un lenguaje común para el comercio internacional. Gracias a ello nació EAN Internacional, (hoy día GS1) constituida en Bélgica en 1977.

Además de su trabajo en relación a los códigos de barras, EAN se preocupa del Intercambio Electrónico de Documentos (EDI), es decir, de la automática transmisión de documentos comerciales. Esto gracias a la interconexión de computadores entre clientes y proveedores.

1.7. La Información Disponible en un Sistema de Código de Barras

La información se procesa y almacena con base en un sistema digital binario donde todo se resume a sucesiones de unos y ceros. La memoria y central de decisiones lógicas es un computador electrónico del tipo estándar, disponible ya en muchas empresas comerciales y generalmente compatibles con las distintas marcas y modelos de preferencia en cada país. Estos equipos permiten también interconectar entre sí distintas sucursales o distribuidores centralizando toda la información. Ahora el distribuidor puede conocer mejor los parámetros dinámicos de sus circuitos comerciales,

permitiéndole mejorar el rendimiento y la toma de decisiones, ya que conocerá con exactitud y al instante toda la información proveniente de las bocas de venta esté o no en su casa central. Conoce los tiempos de permanencia de depósito de cada producto y los días y horas en que los consumidores realizan sus rutinas de compras, pudiendo entonces decidir en qué momento debe presentar ofertas, de qué productos y a qué precios.

1.8. Código de Barras en el Producto

Los códigos de barras se imprimen en los envases, embalajes o etiquetas de los productos. Entre sus requisitos básicos se encuentran la visibilidad y fácil legibilidad por lo que es imprescindible un adecuado contraste de colores. En este sentido, el negro sobre fondo blanco es el más habitual encontrando también azul sobre blanco o negro sobre marrón en las cajas de cartón ondulado. El código de barras lo imprimen los fabricantes (o, más habitualmente, los fabricantes de envases y etiquetas por encargo de los primeros) y, en algunas ocasiones, los distribuidores.

Para no entorpecer la imagen del producto y sus mensajes promocionales, se recomienda imprimir el código de barras en lugares discretos tales como los laterales o la parte trasera del envase. Sin embargo, en casos de productos pequeños que se distribuye individualmente no se puede evitar que ocupe buena parte de su superficie: rotuladores, barras de pegamento, etcétera.

1.9. Tipos de Códigos de Barras

1) Código De Barras De 1 Dimensión

Introducción al Código de Barras de 1 Dimensión

Al efectuar la lectura de éstos códigos se tiene en cuenta el ancho de las barras y los espacios entre ellas. La altura de las barras no otorga dato alguno. Sólo codifican a no más de una docena de caracteres, y representan la clave para acceder un registro de alguna base de datos en donde realmente reside la información, o sea, los símbolos no contienen información del producto o artículo.

Existen varias simbologías de código de barras, que se utilizan según el rubro. El código no contiene el precio del producto, sino una clave única que lo identifica.

De acuerdo al tipo de necesidades de identificación interna del negocio o a los requisitos que se deben cumplir para comerciar según las normas del mercado, se debe optar por el sistema de codificación adecuado.

Las simbologías más utilizadas en la Argentina son el EAN 13, Interlineado 2 de 5, el Código 39 y el Código 128.

El código de barras más conocido es el UPC (Universal Product Code) usado en la mayoría de los productos que se venden al consumidor en EEUU.

El sistema de codificación EAN es usado tanto en supermercados como en comercios. Es un estándar internacional, creado en Europa y de aceptación mundial. Identifica a los productos comerciales por intermedio del código de barras, indicando país-empresa-producto con una clave única internacional. Hoy en día es casi un requisito indispensable tanto para el mercado interno como internacional.
Más de 12.000 empresas en la Argentina ya han codificado de más de 350.000 productos.

El EAN-13 es la versión más difundida del sistema EAN y consta de un código de 13 cifras en la que sus tres primeros dígitos identifican al país, los seis siguientes a la empresa productora, los tres números posteriores al artículo y finalmente un dígito verificador, que le da seguridad al sistema.

Para artículos de tamaño reducido se emplea el código EAN-8.

A. EAN (European Article Numbering o Sistema de Numeración Europeo)

European Article Number es un sistema de códigos de barras adoptado por más de 100 países y cerca de un millón de empresas (2003). En el año 2005, la asociación EAN se ha fusionado con la UCC (*Uniform Code Council*) para formar una nueva y única organización mundial identificada como GS1, con sede en Bélgica.

Figura 1.9.1. Ejemplo EAN 13 (13 dígitos).

El Código EAN-8 es similar al UPC-E, codificando solo 8 dígitos.

Figura 1.9.2. Ejemplo Código EAN-8 (8 dígitos).

El código EAN más usual es EAN13, constituido por 13 dígitos y con una estructura dividida en cuatro partes:

- Los primeros dígitos del código de barras EAN identifican el país que otorgó el código, no el país de origen del producto. Por ejemplo, en Bolivia se encarga de ello una empresa responsable adscrita al sistema EAN y su código es el '777'.
- Referencia del ítem, compuesto de:
  - Código de empresa. Es un número compuesto por entre 5 y 8 dígitos, que identifica al propietario de la marca.
  - Código de producto. Completa los 12 primeros dígitos.
- Dígito de control. Para comprobar el dígito de control (por ejemplo, inmediatamente después de leer un código de barras mediante un escáner), **numeramos los dígitos de derecha a izquierda.** A continuación se suman los dígitos de las posiciones impares, el resultado se multiplica por 3, y se le suman los dígitos de las posiciones pares. Se busca decena inmediatamente superior y se le resta el resultado obtenido. **El resultado final es el dígito de control**. Si el resultado es múltiplo de 10 el dígito de control será 0.

- Por ejemplo, para 123456789041 el dígito de control será:
  - Numeramos de derecha a izquierda: 140987654321
  - Suma de los números en los lugares impares: 1+0+8+6+4+2 = 21
  - Multiplicado × 3: 21 × 3 = 63
  - Suma de los números en los lugares pares: 4+9+7+5+3+1 = 29
  - Suma total: 63 + 29 = 92
  - Decena inmediatamente superior = 100
  - Dígito de control: 100 - 92 = 8

1) El código quedará así: 1234567890418.

B. Implementación

Introducción.

A continuación se verá algunos códigos de diferentes lenguajes de programación, de cómo es que se calcula el digito de control EAN.

i. Perl

```
# Cálculo del dígito de control EAN
my $ean             = '123456789041';       # Valor de prueba
my $checksum        = 1000;
my $i               = 0;


                                    # Recorremos el $ean de forma inversa, dígito por dígito
for my $digit (split //, reverse $ean) {
        $checksum                   # Modificamos $checksum
        -= $i++ % 2                 # según la posición del dígito:
        ? $digit                    # posición impar
        : $digit*3                  # posición par
        ;
}

$checksum %= 10;                    # Ajustamos a la decena inmediatamente inferior

 print "Dígito de control: $checksum\n";

Print "Código EAN: $ean $checksum\n";
```

ii. C#

```
int iSum = 0;    // Cálculo del dígito de control EAN
int iDigit = 0;
string EAN = "123456789041";

for (int i = EAN.Length; i >= 1; i--)
{
 iDigit = Convert.ToInt32(EAN.Substring(i - 1, 1));
 if (i % 2 == 0)
 {
   iSum += iDigit;
 }
 else
 {
   iSum += iDigit * 3;
 }
}
int iCheckSum = (10 - (iSum % 10)) % 10;
Console.Write("Digito de control: " + iCheckSum.ToString());
```

iii. Python

```
iSum = 0
EAN = "123456789041"
 for i in range(len(EAN)-1,0,-1):
   if i%2==0:
      iSum += int(EAN[i])
   else:
      iSum += int(EAN[i])*3
```

```
iCheckSum = (10 - (iSum % 10)) % 10
print "Digito de control: %d" %iCheckSum
```

C. Código 128

Código 128 es un código de barras de alta densidad, usado ampliamente para la logística y paquetería. Utiliza 4 diferentes grosores para las barras y los espacios y tiene una densidad muy alta, ocupando en promedio sólo el 60% del espacio requerido para codificar información similar en Código 39. Puede codificar caracteres alfanuméricos o solo numéricos. Con este código es posible representar todos los caracteres de la tabla ASCII, incluyendo los caracteres de control.

Para comprender cómo se codifica este código, debemos tener en cuenta que cada ASCII se codifica mediante 11 barras.

Por ejemplo , el carácter ASCII <espacio> esto formado por:

- Dos barras negras
- Una barra blanca
- Dos barras negras
- Dos barras blancas
- Dos barras negras
- Dos barras blancas
- TOTAL= 11 Barras.

El código en realidad incluye seis zonas:

- A la izquierda, una zona en blanco que debería tener la longitud de dos caracteres.
- El carácter de inicio.
- Un número variable de caracteres ASCII y es lo más útil de este código.
- Un dígito para checkear la integridad de los datos.

- Un carácter de fin o "Stop character"
- A la derecha, una zona en blanco equivalente a dos caracteres.

Figura 1.9.3. Ejemplo Código 128.

D. Código 39

Código 39 es un código de barras que permite la codificación de caracteres numéricos, letras mayúsculas y algunos símbolos como -, ., $, /, +, % y "espacio". Se utilizan sólo dos grosores tanto para barras como para espacios. Posiblemente la mayor desventaja del código es su baja densidad de impresión (algunos lo consideran de densidad media). Esto significa que sería dificultoso etiquetar objetos demasiado pequeños con este código. A pesar de eso, este código es ampliamente utilizado y puede ser interpretado por casi cualquier lector de códigos de barras.

Esta simbología es actualmente la más usada para aplicaciones industriales y comerciales para uso interno.

No es un código denso y debido a que se requieren muchas barras y espacios para representar un solo carácter puede traer problemas con la longitud del símbolo que se incrementa proporcionalmente.

Figura 1.9.4. Ejemplo Código 39.

E. Código 93

El código de barras **Código 93** fue diseñado en 1982 por Intermec para lograr una mayor densidad de datos en el código Code 39. Es alfanumérico, de longitud variable. Code 93 primariamente fue usado por el servicio postal canadiense. Cada símbolo del código incluye dos caracteres de checksum.

F. Codabar

Codabar es un código de barras lineal desarrollado en el 1972 por Pitney Bowes. Fue especialmente diseñado para poder ser leído sin problemas aun si fuera impreso por una impresora de matriz de puntos. Además la nueva simbología permitía contener más información en el mismo tamaño de etiqueta. La simbología de longitud variable que codifica solo números. Utiliza dos tipos de grosores para barras y espacios y su densidad es similar a la del Código 39.

Figura 1.9.5. Ejemplo Código Codabar.

G. Intercalado 2 De 5

Se basa en la técnica de intercalar caracteres permitiendo un código numérico que utiliza dos grosores. El primer carácter se representa en barras, y el segundo por los espacios que se intercalan en las barras del primero. Es un código muy denso, aunque siempre debe haber una cantidad par de dígitos. La posibilidad de una lectura parcial es alta especialmente si se utiliza un lector láser. Por lo tanto, generalmente se toman

ciertas medidas de seguridad, como codificar un carácter de verificación al final del símbolo.

Figura 1.9.6. Ejemplo Código Intercalado 2 de 5.

H. UPC (Código Universal de Producto)

UPC es la simbología más utilizada en el comercio minorista de EEUU, pudiendo codificar solo números. El estándar UPC (denominado UPC-A) es un número de 12 dígitos. El primero es llamado "número del sistema". La mayoría de los productos tienen un "1" o un "7" en esta posición. Esto indica que el producto tiene un tamaño y peso determinado, y no un peso variable. Los dígitos del segundo al sexto representan el número del fabricante. Esta clave de 5 dígitos (adicionalmente al "número del sistema") es única para cada fabricante, y la asigna un organismo rector evitando código duplicado. Los caracteres del séptimo al onceavo son un código que el fabricante asigna a cada uno de sus productos, denominado "número del producto". El doceavo carácter es el "dígito verificador", resultando de un algoritmo que involucra a los 11 números previos.

Figura 1.9.7. Ejemplo Código UPC

Para productos pequeños se utiliza el Código UPC-E.

Figura 1.9.8. Ejemplo Código UPC-E.

La industria editorial agregó suplementos de 2 y 5 dígitos al final de símbolo UPC-A, utilizados por lo general para la fecha de publicación o el precio.

Figura 1.9.9. Ejemplo Código UPC-A con 2 dígitos al final.

Figura 1.9.10.Ejemplo Código UPC-A con 5 dígitos al final.

I. Otros Códigos de Barras

PostNet, aparece en 1980 siendo usado por el Servicio Postal de los EEUU

Figura 1.9.11. Ejemplo Código PostNet.

2) Código De Barras De 2 Dimensiones

Introducción al Código de Barras de 2 Dimensión

Los datos están codificados en la altura y longitud del símbolo, y en éstos códigos la información no se reduce sólo al código del artículo, sino que puede almacenar gran cantidad datos. Un código del tamaño de una estampilla postal se puede almacenar más de mil caracteres alfanuméricos.

La principal ventaja de utilizar códigos de 2 dimensiones es que el código contiene una gran cantidad de información que puede ser leída de manera rápida y confiable, sin necesidad de acceder a una base de datos en donde se almacene dicha información (el caso de los códigos de 1 dimensión)

Los escaners empleados para leer los códigos 2D son diferentes a los utilizados para captar los símbolos tradicionales unidimensionales.

En la actualidad existen dos métodos de lectura:

1. El método más común utiliza un rayo láser que se mueve de izquierda a derecha y de arriba a abajo en lo que se conoce como un patrón de rastreo.

2. El segundo método aplica tecnología CCD. Los lectores CCD tienen un arreglo bidimensional de foto sensores que escanean la imagen en su totalidad.

La seguridad que es capaz de incorporar estos códigos los hace casi invulnerables a un sabotaje. Para estropear la legibilidad de un código unidimensional, basta con agregar otra barra al inicio o final del símbolo o trazar una línea paralela a las barras en cualquier lugar dentro del código. Los códigos de 2D se pueden construir con muchos grados de redundancia, duplicando así la información en su totalidad o sólo los datos vitales. La redundancia aumenta las dimensiones del símbolo pero la seguridad

del contenido se incrementa notablemente.

Se han hecho pruebas de resistencia a códigos bidimensionales perforándolos, marcándolos con tinta y maltratándolos. El símbolo es legible aún después de todos estos abusos.

Los códigos de 2D deben ser considerados como un complemento a la tecnología tradicional de códigos de 1D, no como su reemplazo; y las ventajas deben ser comparadas contra el incremento en costo.

Dentro de las simbologías bidimensionales hay dos categorías: de matriz o matrix y apilables o stacked.

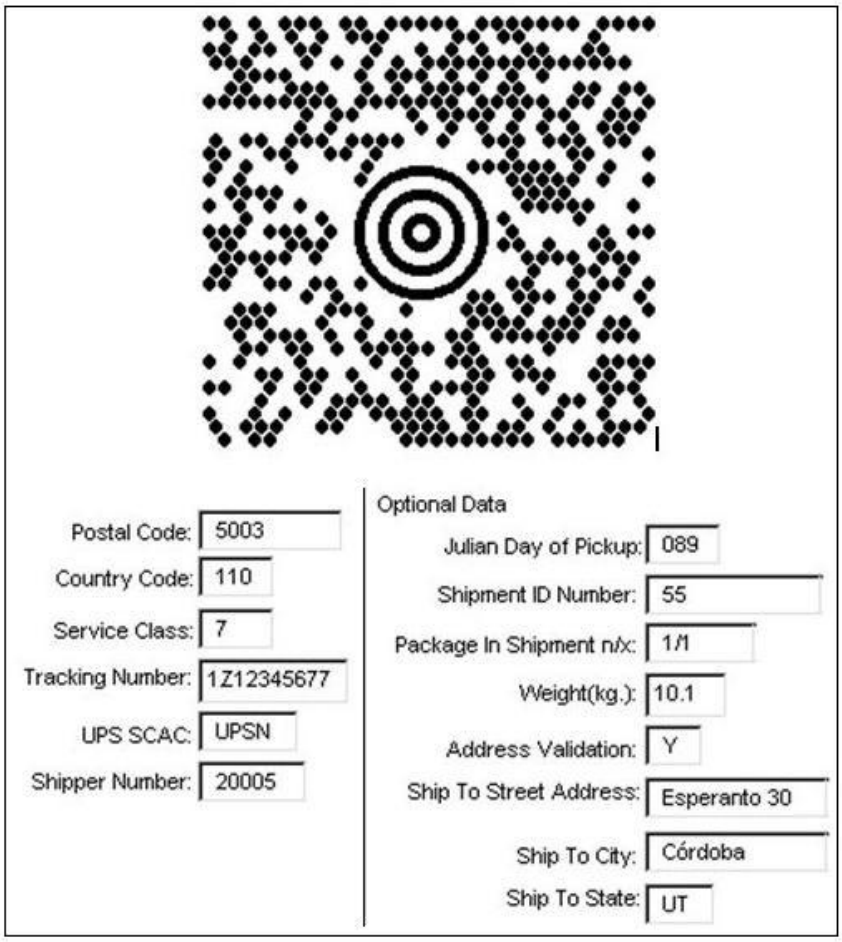

Figura 1.9.12. Ejemplo Codificado. (Códigos de 2D de Matriz “Matrix” MAXICODE)

Figura 1.9.13. Código 2D.

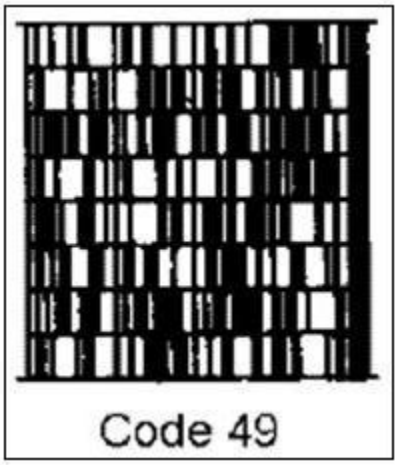

Figura 1.8.14. Código 2D.

a) Datamatrix

Desarrollado en 1989 por International Data Matrix Inc. La versión de dominio público es la ECC 200, desarrollada también por International Data Matrix en 1995.

i. Datos Generales

- Formato: Matriz
- Caracteres: ASCII, C40, binario, otros
- Capacidad alfanumérica: 2334 caracteres

ii. Aplicaciones

- Identificación y control de partes componentes (según AIAG: Automotive Industry Action Group).
- Control y prevención de productos en expiración o que han sido "recalled".
- Codificación de dirección postal en un símbolo bidimensional (usos en el servicio postal para automatizar ordenado del correo).
- Marcado de componentes para control de calidad.
- Los componentes individuales son marcados identificando al fabricante, fecha de fabricación y numero de lote, etc.
- Etiquetado de desechos peligrosos (radioactivos, tóxicos, etc.) para control y almacenamiento a largo plazo.
- Industria farmacéutica, almacenamiento de información sobre composición, prescripción, etc.
- Boletos de lotería, información específica sobre el cliente puede codificarse para evitar la posibilidad de fraude.
- Instituciones financieras, transacciones seguras codificando la información en cheques.

b) Maxicode

Es una simbología de alta densidad creada por UPS (United Parcel Service). En la actualidad esta simbología es de dominio público y está especificada bajo las normas ANSI (MH10.8.3M-1996).

i. Usos

Procesamiento de información a alta velocidad.

ii. Características

La estructura del Maxicode consiste de un arreglo de 866 hexágonos utilizados para el almacenamiento de datos en forma binaria. Estos datos son almacenados en forma seudo-aleatoria. Posee un blanco o "bull" utilizado para localizar a la etiqueta en cualquier orientación.

Es posible codificar hasta 100 caracteres en un espacio de una pulgada cuadrada. Este símbolo puede ser decodificado sin importar su orientación con respecto al lector óptico.

Para su impresión se requiere de impresoras laser o termales con una resolución mínima de 200dpi.

La simbología utiliza el algoritmo de Reed-solomon para corrección de error. Esto permite la recuperación de la información contenida en la etiqueta cuando hasta un 25 por ciento de la etiqueta este dañado.

c) Pdf-417

Es un archivo portátil de datos (Portable Data File), tiene una capacidad de hasta 1800 caracteres numéricos, alfanuméricos y especiales. El código contiene toda la información, no se requiere consultar a un archivo.

Cuenta con mecanismos de detección y corrección de errores: 9 niveles de seguridad lo que permite la lectura y decodificación exitosa aún cuando el daño del código llegue hasta un 40%.

i. Aplicaciones

- Industria en general.
- Sistemas de paquetería: cartas porte.
- Compañías de seguros: validación de pólizas.

- Instituciones gubernamentales: aduanas.
- Bancos: reemplazo de tarjetas y certificación de documentos.
- Transportación de mercadería: manifiestos de embarque.
- Identificación personal y foto credencial.
- Registros públicos de la propiedad.
- Testimonios notariales.
- Tarjetas de circulación.
- Licencias de manejo.
- Industria electrónica etc.

1.10. Beneficios

Es la mejor tecnología para implementar un sistema de colección de datos mediante identificación automática, además presenta muchos beneficios, entre otros.

- Virtualmente no hay retrasos desde que se lee la información hasta que puede ser usada
- Se mejora la exactitud de los datos
- Se tienen costos fijos de labor más bajos
- Se puede tener un mejor control de calidad, mejor servicio al cliente
- Se pueden contar con nuevas categorías de información.
- Se mejora la competitividad.

1.11. Aplicaciones

Las aplicaciones del código de barras cubren prácticamente cualquier tipo de actividad humana, tanto en industria, comercio, instituciones educativas, instituciones médicas, gobierno, etc.

- Control de material en proceso
- Control de inventario

- Control de tiempo y asistencia
- Punto de venta
- Control de calidad
- Control de inventario
- Embarques y recibos
- Control de documentos
- Facturación
- Bibliotecas
- Bancos de sangre
- Hospitales
- Control de acceso
- Control de tiempo y asistencia

1.12. Tipos de lectores

Los cuatro principales tipos de lectores son:

a) Lápiz Óptico O Wand

Debe ser deslizado haciendo contacto a lo ancho del código. Como se menciona anteriormente, envía una señal digital pura de las barras y espacios a una frecuencia igual a la velocidad con que se desliza el lápiz.

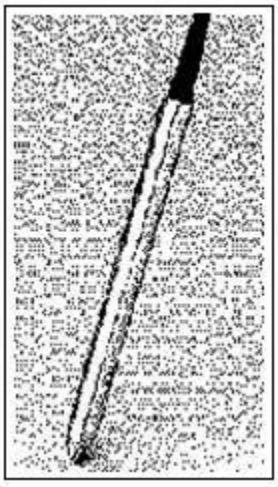

Figura 1.12.1. Lápiz Óptico O Wand

- Ventajas: es económico
- Desventajas: es lento, requiere que el usuario tenga práctica, tiene un bajo first read rate, requiere un decodificador de teclado, depende de la calidad de impresión del código.
- Precios: 100-150 dlls.

b) Laser De Pistola

Realiza un barrido mediante una luz laser y que genera una señal similar a la del lápiz óptico, pero a una mayor frecuencia. Esta señal es conocida como HHLC (Hand Held Laser Compatible)

- Ventajas: es rápido, puede no requerir decodificador de teclado, puede leer a distancia (estándar 5 a 30 cm, especial hasta 15m con etiquetas de papel retro-rreflectivo), tiene un alto FRR.

- Desventajas: es relativamente caro (aunque existen modelos de 545 dlls), puede presentar problemas de durabilidad debido a sus partes móviles (espejos giratorios), puede tener problemas para leer con demasiada luz ambiental.
- Precios: 500-1500 dlls.

Figura 1.12.2. Laser De Pistola.

c) CCD (Charge Coupled Device)

Mediante un arreglo de fotodiodos toma una 'foto' del símbolo de código de barras y la traduce a una señal, que puede ser similar a la enviada por el laser (HHLC) o a la del lápiz óptico.

- Ventajas: es rápido, es económico, es muy durable por no tener partes móviles, puede no necesitar decodificador de teclado, tiene un alto FRR.
- Desventajas: requiere estar muy cerca del código (0-1.5cm), no puede leer símbolos que rebasen el ancho de su ventana.
- Precios: 200-400dlls

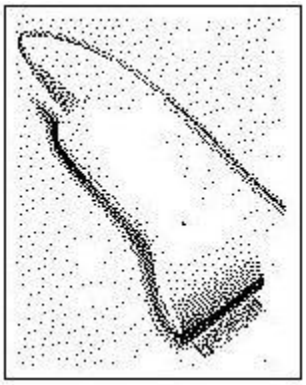

Figura 1.12.3. CCD.

d) Laser Omnidireccional

Es un lector que envía un patrón de rayos laser y que permite leer un símbolo de código de barras sin importar la orientación del mismo.

- Ventajas: Todas las ventajas del laser de pistola más un FRR de prácticamente 100%.
- Desventajas: es caro (aquí no hay modelos económicos), el operador requiere que los artículos etiquetados no sean muy voluminosos pues el scanner se monta en posición fija.
- Precios: 1200-2700 dlls.

Figura 1.12.4. Laser Omnidireccional.

1.13. Ventajas Del Código De Barras

Entre las primeras justificaciones de la implantación del código de barras se encontraron la necesidad de agilizar la lectura de los artículos en las cajas y la de evitar errores de digitación. Otras ventajas que se pueden destacar de este sistema son:

- Agilidad en etiquetar precios pues no es necesario hacerlo sobre el artículo sino simplemente en el lineal.
- Rápido control del stock de mercancías.
- Estadísticas comerciales. El código de barras permite conocer las referencias vendidas en cada momento pudiendo extraer conclusiones de mercadotecnia.
- El consumidor obtiene una relación de artículos en el ticket de compra lo que permite su comprobación y eventual reclamación.

Entre las pocas desventajas que se le atribuyen se encuentra la imposibilidad de recordar el precio del producto una vez apartado del lineal.

Capítulo II

Análisis

2.1. Introducción

En el presente capítulo se muestra los aspectos más importantes referentes al análisis, además hablaremos acerca de la arquitectura Cliente/Servidor, ya que para nuestra aplicación utilizamos ésta arquitectura.

En el mundo de TCP/IP las comunicaciones entre computadoras se rigen básicamente por lo que se llama modelo Cliente-Servidor, éste es un modelo que intenta proveer usabilidad, flexibilidad, interoperabilidad y escalabilidad en las comunicaciones.

El término Cliente/Servidor fue usado por primera vez en 1980 para referirse a PC's en red.

Este modelo Cliente/Servidor empezó a ser aceptado a finales de los 80's. Su funcionamiento es sencillo: se tiene una máquina cliente, que requiere un servicio de una máquina servidor, y éste realiza la función para la que está programado (nótese que no tienen que tratarse de máquinas diferentes; es decir, una computadora por sí sola puede ser ambos cliente y servidor dependiendo del software de configuración).

2.2. El Modelo Cliente/Servidor

El modelo Cliente/Servidor es una abstracción que nos permite separar conceptualmente los mecanismos y procesos que realizan aplicaciones que de manera cooperativa resuelven una serie de problemas no locales. En el modelo cliente servidor, el cliente envía un mensaje solicitando un determinado servicio a un servidor (hace una petición), y este envía uno o varios mensajes con la respuesta (provee el servicio) (Ver Figura 2.2.1). En un sistema distribuido cada máquina puede cumplir el rol de servidor para algunas tareas y el rol de cliente para otras.

Figura 2.2.1. Modelo Cliente/Servidor.

La idea es tratar a una computadora como un instrumento, que por sí sola pueda realizar muchas tareas, pero con la consideración de que realice aquellas que son más, adecuadas a sus características. Si esto se aplica tanto a clientes como servidores se entiende que la forma más estándar de aplicación y uso de sistemas cliente/Servidor es mediante la explotación de las PC's a través de interfaces gráficas de usuario; mientras que la administración de datos y su seguridad e integridad se deja a cargo de computadoras centrales tipo mainframe. Usualmente la mayoría del trabajo pesado se hace en el proceso llamado servidor y el o los procesos cliente sólo se ocupan de la interacción con el usuario (aunque esto puede variar). En otras palabras la arquitectura Cliente/Servidor es una extensión de programación modular en la que la base fundamental es separar una gran pieza de software en módulos con el fin de hacer más fácil el desarrollo y mejorar su mantenimiento. Esta arquitectura permite distribuir físicamente los procesos y los datos en forma más eficiente lo que en computación distribuida afecta directamente el tráfico de la red, reduciéndolo grandemente.

En los sistemas aislados se ha explotado este modelo para controlar y administrar algunos recursos, como ejemplo podemos citar algunos componentes del SO (Sistema Operativo): el administrador de proceso, el manejador de ventanas en ambientes orientados a ventanas en ambientes grafico y el gestor de medios de almacenamiento masivo (discos duros, CD's, DVD's, etc.). Donde debemos recalcar que a pesar de tratar el caso de sistemas modernos mono usuario, ya que estos ofrecen la opción de la multitarea, es necesario controlar las acciones de los procesos que el usuario ejecuta en primer o segundo plano para evitar conflictos e inconsistencias.

Para regular las relaciones entre el cliente y el servidor normalmente se definen regla, las cuales se encapsulan en librerías que permiten la operación entre las entidades involucradas. Estas reglas conforman los denominados **protocolos** de interacción. Y los procedimientos públicos habilitados para la interacción se organizan en **Interfaces de Programación de aplicaciones** (API). Este binomio permite a los desarrolladores conocer cómo utilizar los métodos de trabajo, las variables involucradas y sus tipos, así como el alcance y restricciones de explotación.

En los sistemas de computo conectados en la red local (LAN) o en la redes amplias (WAN), además se deben tomar en cuenta las facilidades que el sistema operativo de red nos ofrece así como las restricciones que impone, de aquí se puede definir que es importante conocer y controlar no solo la dinámica de lo que sucede en cada nodo de la red, sino como debe coordinar los diferentes nodos y medios de enlace entre ellos, es decir se deben considerar cada uno de los elementos involucrados en el sistema: locales, remotos e intermediarios. Un ejemplo de aplicaciones cliente/servidor corresponde a los denominados Sitios de Conversación (Chat).

2.3. Elementos básicos de un Sistema Cliente/Servidor

En general puede haber muchos clientes operando, en este caso el servidor requiere emprender acciones de conciliación, secuenciación y bloqueo, entre otras, para que los clientes sean atendidos de manera oportuna y expedita. Cuando en el sistema opera un solo servidor y un solo cliente se dice que tenemos una Aplicación Servidor Simple. Este modelo se puede desarrollar incluso en una misma estación de trabajo. En particular en el caso de un Manejador de Base de Datos (DBMS) que se ejecuta en una maquina y en la misma máquina se ejecuta un programa que interactúa con el DBMS realizando una o varias tareas. En este caso en este modelo le llamaremos "local", la aplicación saca provecho de la capacidad de realizar transacciones del DBMS y se centra en la lógica requerida para resolver los problemas particulares del problema específico. En general en este enfoque la aplicación brinda una interface simple y amigable al usuario para realizar sus tareas, y del lado del desarrollador, este se orienta a desarrollar la(s) interface(s) de la aplicación de manera manual o mediante una herramienta de desarrollo, de tal manera que las necesidades del usuario final queden atendidas. Probablemente este esquema sea el más difundido en el mercado de las pequeñas aplicaciones de BD. Considerándose como la meta del desarrollador y el producto esperado por el cliente. En este ejemplo podemos identificar como "servidor" al DBMS y como "cliente" a la aplicación del usuario.

Mas en general el asunto no es tan simple como se planteo en el párrafo anterior, en general las necesidades en el proceso de negocios y servicios son más complejos y requieren de una arquitectura ampliada, tanto a nivel software como hardware . Este modelo extendido en el ámbito de la Computación y la Informática se denomina Modelo de Bases de Datos Distribuidas. En la siguiente sesión se presentas un panorama general del tema.

2.4. Elementos de Bases de Datos Distribuidos

En el procesamiento distribuido una maquina, conectada en una red (LAN) o amplia (WAN) a otras maquinas, puede realizar una tarea de procesamiento de datos en ella y sobre el resto de las maquinas de la red, de tal manera que los procesos y los datos se ejecuten de manera paralela, coherente y mediante ciertas reglas en uno o varios nodos de a red.

Si los nodos se encuentran localizados en un sistema físico de cómputo compacto, tendremos un esquema "estrictamente paralelo" y si los nodos están dispersos geográficamente entonces nos encaramos a un Sistema Distribuido. La comunicación entre los nodos se garantiza mediante un SO de red o bien con un componente de software de comunicaciones.

Una organización diferencial posible para los Sistemas Distribuidos se compone por cuatro categorías a saber:

1) **Sistema Dorsal o Arquitectura Centralizada.** Hay un solo servidor y varios clientes, en el servidor se encuentra instalado el DBMS y reside las BD para diversas aplicaciones. Los clientes mediante un poder local de cómputo, presentan las interfaces de usuarios y preparan las transacciones solicitadas, las cuales son enviadas al servidor para su procesamiento. En este modelo tiene la ventaja de repartir el trabajo en partes (en paralelo), lo cual mejora el tiempo de respuesta y reduce la lactancia en las comunicaciones.
    a) El servidor puede ser una maquina especializada de alto rendimiento para el manejo de BD, por ejemplo contener un arreglo de discos duros de alta velocidad y rendimiento (SCSI) y un sistema operativo estable multitarea, multiusuario y con seguridad.
    b) El cliente puede ser una estación de trabajo personal (PC) con software y hardware adaptado a las necesidades del usuario final, con alta disponibilidad, respuesta rápida y autonomía local para procesos internos.

De esta manera varias maquinas cliente podrán acceder a la misma máquina que juega el rol de servidor. Por lo cual una base de datos puede ser compartida entre distintos clientes.

Este modelo se acopla a términos formales al mecanismo que utilizan las empresas e instituciones. Poe ejemplo es común en una empresa opere muchas computadoras, donde algunas juegan el papel se servidores y el resto (la mayoría) de clientes. Mas no se descarta que algunas de ellas jueguen el doble rol, es decir ser servidor para algunas y cliente para otras. Podemos decir que al menos en los servidores se pueden afirmar que cada uno de ellos soporta *un sistema de bases de datos complejo*, en el sentido de las BD distribuidas.

2) **Arquitectura Distribuida.** El sistema se conforma por varios servidores y varios clientes. Cada servidor contiene un DBMS y varias BD's. los clientes dependiendo de las necesidades del usuario se conectan a unos o varios servidores para satisfaces sus demandas. El acceso puede ser proporcionado de dos maneras:
    a) Un cliente puede acceder a cualquier servidor, pero solo a uno a la vez. De tal forma que el cliente conoce donde debe solicitar la información. En este esquema no es posible combinar datos de dos o más servidores en una misma petición.
    b) El cliente puede acceder a varios servidores a la vez, de tal forma que una petición de base de datos puede combinar información de varios servidores, este modelo tiene la ventaja de que el usuario no tiene que saber en qué servidor se encuentran ubicados los datos.

El soporte de una base de datos distribuida implica que las aplicaciones deben operar de manera transparente sobre los datos que estén dispersos sobre la red, donde es posible que los manejadores de BD locales (en cada servidor) utilicen una representación diferente para los datos, es decir los DBMS no tienen que ser iguales, así como los sistemas operativos (SO) en cada servidor. Al referirnos a "transparencia" debemos entender que la aplicación opera como si los datos fueran manejados por un solo

DBMS y en una sola maquina donde reside un servidor virtual. Una característica extra de este modelo es que un servidor puede atender a muchos clientes o bien servidores a la vez.

Para que un sistema distribuido trabaje adecuadamente además es necesario contar con un sistema de comunicaciones (red) robusto, estable y oportuno. Lo cual incluye un ingrediente más a la arquitectura de los sistemas distribuidos.

En la figura (2.4.2.) se muestra la topología de los sistemas antes mencionados.

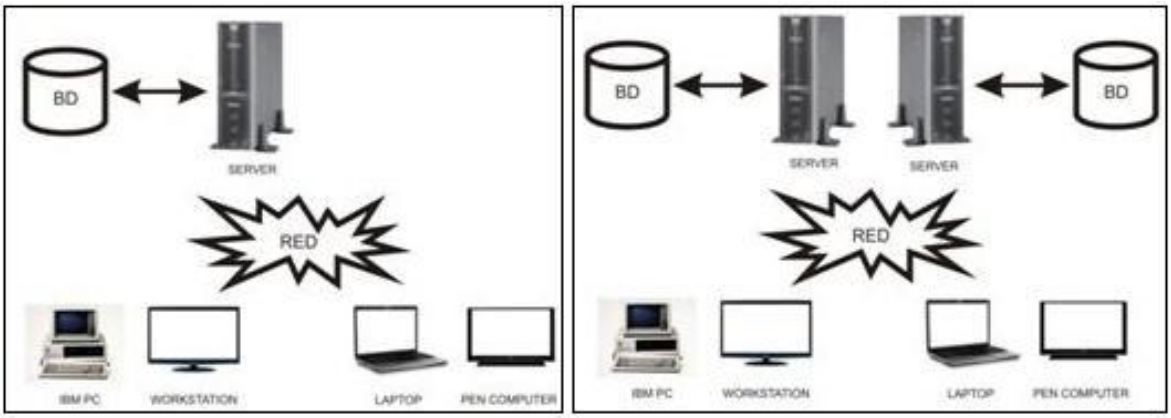

Figura 2.4.2. Esquema de Arquitecturas Centralizada (izquierda) y

Distribuida (derecha).

El principio fundamental de una Base de Datos Distribuida (BDD) es que: ante un usuario, un sistema distribuido se comporta exactamente igual que uno no distribuido. Entonces los problemas de los sistemas distribuidos son problemas internos o en el nivel de implementación, no externos o en el nivel del usuario. [4].

Los sistemas de BDD operan sobre 12 reglas u objetivos interrelacionados, a continuación se presenta un resumen de estos, para mayor detalle ver [4].

1) **Autonomía Local.** Los sitios deben ser autónomos, es decir las operaciones en un sitio X están controladas por él; de tal suerte que la operación satisfactoria de X no depende de otro sitio Y. Esto implica que los datos locales son propiedad del sitio y son administrados por él. Debido a esto la seguridad, integridad

y representación de almacenamiento de los datos locales permanecen bajo el control y jurisdicción del sitio local.

2) **Independencia de un Sitio Central.** No debe haber un sitio "maestro" central para algún servicio central de tal manera que todo el sistema depende de ese sitio central. La existencia de sitios maestros conlleva al problema de vulnerabilidad de todo el sistema por falla de este.
3) **Operación Continua.** El sistema no debe interrumpir de manera aleatoria sus servicios. Esto se refleja en la **confiabilidad y disponibilidad**: la primera implica que la respuesta del sistema debe ser sostenida a pesar de uno o varios sitios fallen; y la segunda debe garantizar que casi todo el tiempo el sistema responda a las solicitudes. Estos objetivos se alcanzan mediante la replicación y la redundancia, juntos forman el esquema de **tolerancia a fallas.**
4) **Independencia de la Ubicación.** Esta también se conoce como **transparencia de ubicación** y consiste en el hecho de que los usuarios no tiene que saber donde se encuentran almacenados físicamente los datos, sino que deben ser capases de comportarse, al menos desde el punto de vista lógico, como si estuviera almacenados de forma local.
5) **Independencia de Fragmentación.** También llamada **transparencia de fragmentación.** Los datos a nivel de registros o tablas, pueden estar fragmentados, es decir distribuidos en diferentes sitios para efecto de almacenamiento físico. Normalmente la fragmentación se provoca para localizar los datos en los sitios donde son utilizados con mayor frecuencia, esto reduce el uso de la red para realizar las transacciones y aumentar la velocidad de respuesta del sistema de BDD. Existen dos tipos de fragmentación: **Horizontal y vertical**. La primera se refiere a partir las tablas en partes con registros íntegros; y la segunda corresponde a separar los campos de los registros en dos o más sub-tablas, donde cada una de estas contiene las llaves para la localización de registro. Un principio básico que se debe cumplir es que todos los fragmentos deben ser independientes, es decir que ninguno de ellos podrá ser derivado a partir de los

otros ni existen restricciones o proyecciones que puedan ser derivadas a partir de otros.

6) **Independencia de Replicación.** Esto significa que una tabla o fragmento distribuido de ella que se encuentra almacenado en algún sitio puede ser representado por otras copias (replicas) almacenadas en diferentes sitios. Esta propiedad se denomina **redundancia controlada**. La principal desventaja de la replicación radica en que al hacer un cambio en algún registro, este se debe propagar a los demás sitios que contengan copias sin violar la integridad del sistema completo de la BDD. En general será responsabilidad del encargado del sistema de definir cuales replicas deben ser accedidas físicamente para satisfacer una solicitud de usuario dada.
7) **Procesamiento Distribuido de Consultas.** Este criterio implica que el sistema debe optimizar las transacciones de tal manera que reduzca el tráfico de la red y se acceda a la réplica más cercana o bien al sitio de menor tiempo de respuesta. Esto implica que el modelo a utilizarse para realizar las transacciones deben
8) ser menos relacional.
9) **Administración de Transacciones Distribuidas.** Dada la característica distribuida del sistema se deben tomar las previsiones necesarias para garantizar el control de la recuperación de datos y el control de concurrencia. El mecanismo natural que se utiliza es de **agentes,** es decir dado que se pueden realizar varias transacciones de forma concurrente en diferentes sitios, los agentes se encargaran de realizar las transacciones en cada sitio representando las transacciones originales del usuario.
10) **Independencia de Hardware.** Esta regla define que debe ser posible ejecutar el mismo DBMS en diferentes plataformas de hardware, de tal manera que cada máquina participe como un socio igualitario en el sistema distribuido.
11) **Independencia del Sistema Operativo.** Este criterio implica que cada sitio puede funcionar con un sistema operativo diferente (tanto de versión como de tecnología). Y a pesar de esto el DBMS operar en cada sitio de la misma manera.

12) **Independencia de la Red.** Esto implica que algunos nodos pueden trabajar sobre redes de comunicación diferentes, pero el sistema en general puede comunicar a un nodo con otro en todo momento. Esto se logra mediante dispositivos de traducción e interface, los cuales hacen compatibles los mensajes de entrada y salida.
13) **Independencia del DBMS.** En todos los sitios de tipo servidor no necesariamente opera el mismo DBMS, sino que se requiere que cada DBMS local soporte la misma interface para las transacciones. Este principio completa el esquema heterogéneo de un sistema de BDD.

En general el problema principal que se debe resolverse para la operación correcta de un sistema de BDD es la **reducción del uso de la red**, es decir se debe minimizar la cantidad y volumen de los mensajes entre sitios. Los aspectos donde impacta más este aspecto son los siguientes:

- Procesamiento de consultas
- Administración de catálogos
- Propagación de las actualizaciones
- Control de recuperación
- Control de concurrencia
- Procesos de mantenimiento.

En particular cada DBMS para ambientes distribuidos ofrece diferentes mecanismos para atender estas problemáticas.

2.5. Tipos de fragmentación de datos

Existen tres tipos de fragmentación:

- Fragmentación horizontal
- Fragmentación vertical
- Fragmentación híbrida [5]

1) Fragmentación horizontal primaria

Consiste del particionamiento en tuplas de una relación global en subconjuntos, donde cada subconjunto puede contener datos que tienen propiedades comunes y se puede definir expresando cada fragmento como una operación de selección sobre la relación global. [5]

Ejemplo 3. Considere la relación global

SUPPLIER (SNUM, NAME, CITY)

Entonces, la fragmentación horizontal puede ser definida como:

SUPPLIER1 = SLcity == "SF"SUPPLIER

SUPPLIER1 = SLcity == "LA"SUPPLIER

a) Esta fragmentación satisface la condición de completes si "SF" y "LA" son solamente los únicos valores posibles del atributo CITY.
b) La condición de reconstrucción se logra con:

SUPPLIER = SUPPLIER1 unión SUPPLIER2

c) La condición de disjuntos se cumple claramente en este ejemplo.

2) Fragmentación horizontal derivada

La fragmentación derivada horizontal se define partiendo de una fragmentación horizontal.

En esta operación se requiere de Semi-junta (Semi-Join) el cual nos sirve para derivar las tuplas o registros de dos relaciones. [5]

Ejemplo 4. Las siguientes relaciones definen una fragmentación horizontal derivada de la relación SUPPLY.

SUPPLY1 = SUPPLYSJsnum == snumSUPPLIER1

SUPPLY2 = SUPPLYSJsnum == snumSUPPLIER2

3) Fragmentación vertical

La fragmentación vertical es la subdivisión de atributos en grupos. Los fragmentos se obtienen proyectando la relación global sobre cada grupo. La fragmentación es correcta si cada atributo se mapea en al menos un atributo del fragmento. [5]

Ejemplo 5. Considere la siguiente relación global:

EMP (empnum, name, sal, tax, mgrnum, depnum)

Una fragmentación vertical de esta relación puede ser definida como:

EMP1 = PJempnum, name, mgrnum, depnum EMP

EMP2 = PJempnum, sal, tax EMP

La reconstrucción de la relación EMP puede ser obtenida como:

EMP = EMP1 (JN empnum) EMP2 porque empnum es una clave de EMP

4) Fragmentación híbrida

En la que respecto a la fragmentación híbrida, esta consiste en aplicar la fragmentación vertical seguida de la fragmentación horizontal o viceversa. [5]

2.6. Sistema Cliente/Servidor

Como es natural ahora podemos afirmar que los sistemas cliente/servidor (CS) son un caso particular de un sistema distribuido. De la manera que el sistema distribuido:

a) Algunos sitios juegan el papel de **servidores** y otros de **clientes**.
b) Todos los datos residen en los sitios servidores.

c) Las aplicaciones de usuario son ejecutadas en los sitios clientes.
d) No existe una independencia total de ubicación.

Donde en los servidores (S) se encuentran los datos y su correspondiente DBMS y en los clientes (C) las aplicaciones o interfaces. Podemos definir un sistema CS mediante las siguientes reglas [5]:

a) Un sistema CS es una relación entre procesos corriendo en maquinas separadas:

EL SERVIDOR ES UN PROOVEDOR DE SERVICIOS.

EL CLIENTE ES UN CONSUMIDOR DE SERVICIOS.

b) El cliente y el servidor interactúan mediante un mecanismo de paso de mensajes:

PETICION DE SERVICIO.
RESPUESTA A LA PETICION.

Retomando las doce características de los sistemas distribuidos, mínimamente un sistema CS debe tener las siguientes propiedades:

- **Control de los Recursos.** Esto implica que un S puede atender a muchos C y mantener el control local de sus recursos.
- **Transparencia.** Debe haber transparencia de ubicación, hardware y plataforma (SO).
- **Encapsulamiento.** Una petición hecha por el C o S indicará claramente **qué** servicio se desea y entonces un S se encargara de **cómo** resolverla. En general deberá ser posible modificar los S sin afectar a los C.
- **Escalabilidad.** Se pueden realizar acciones de escalamiento (upgrade) sin afectar a otros componentes, se manejan dos tipos:

Horizontal: se agregan otros C y S.

Vertical: se cambia un S por otro más potente o se distribuye su trabajo entre varios.

- **Integridad.** Las funciones y datos del S son manejadas en forma centralizada. Eso beneficia el mantenimiento e integridad de los datos.

A pesar de que el modelo de sistema distribuido es más poderoso que el cliente/servidor, a nivel comercial se utiliza mas el segundo, pero a mediano plazo estos serán reemplazados por los primero. Dependiendo de la complejidad y tipo de aplicación a desarrollarse los sistemas CS se organiza en tres clases, a saber:

1) **Modelo de Dos Capas.** Un cliente se localiza en un sitio físico bien definido y lo mismo sucede con el servidor. Donde la lógica de aplicación puede organizarse en tres formas:

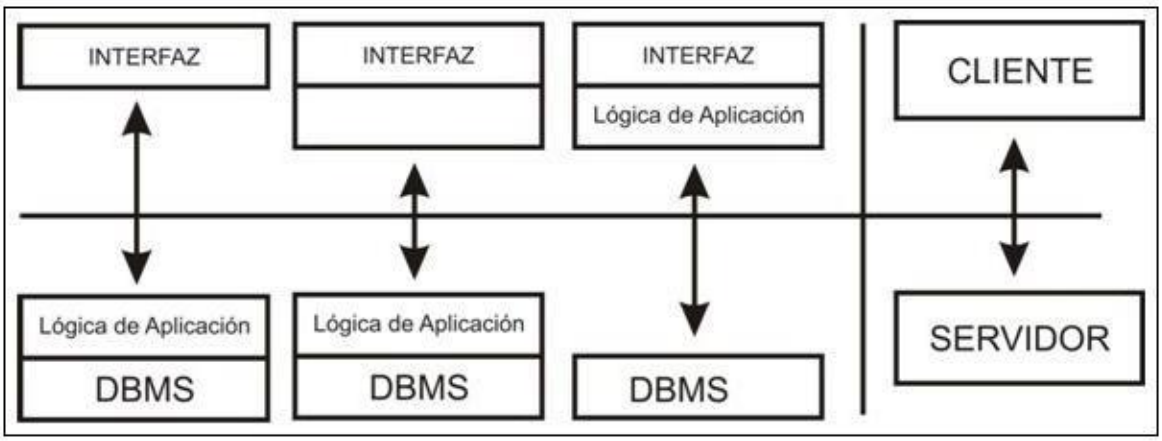

Figura 2.6.3. Modelo Sistema Cliente/Servidor

En el primer modelo, el cliente juega un papel de manejar la interfase se la aplicación simplemente y la lógica de la aplicación y el DBMS residen en el servidor. En el segundo caso la lógica de la aplicación se comparte entre el cliente y el servidor; y en el último caso la lógica de la aplicación se localiza en el lado del cliente. Esta división del

trabajo imprime las siguientes características dependiendo donde se ubica la lógica de la aplicación:

En el lado del Cliente
La semántica está dividida entre los programas.
Cada programa solo maneja una parte del problema.
Los controles están embebidos en el código de los programas.
Existen fuertes problemas de mantenimiento, duplicidad de esfuerzos y de integridad.

Tabla 2.6.1. Características Cliente.

En el lado del Servidor
La lógica está centralizada y en general es consistente.
Aumenta la seguridad e integridad.
Algunos controles se pueden expresar de manera simple en SQL y otros pueden ser rutinas complejas (procedimientos almacenados).
Aumenta el trabajo en el servidor y su complejidad.

Tabla 2.6.2. Características Servidor.

2) **Modelo de Tres Capas**. En este modelo se separa la lógica de la aplicación de la interfase ubicada en el lado del cliente y del DBMS situado en el lado del servidor. El esquema se muestra en el diagrama adjunto. Este modelo se aplica en los siguientes casos:

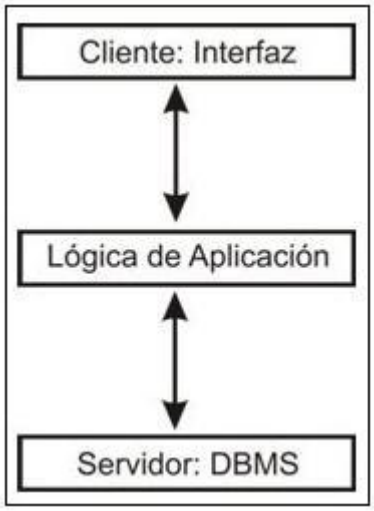

Figura 2.6.4. Modelo de Tres capas.

a) Cuando se ofrecen muchos servicios del lado del servidor.
b) Si las aplicaciones se ejecutan en el servidor se operan con diferentes lenguajes (tipo intérprete).
c) Se manejan sistemas de BD heterogéneos.
d) Hay muchas transacciones por unidad de tiempo.
e) Hay muchos usuarios conectados simultáneamente.
f) Existe mucha comunicación entre aplicaciones.

3) **Modelo de N Capas.** Este esquema se utiliza principalmente cuando la operación se realiza mediante la colaboración de varios servidores de software en el lado del server o bien los clientes acceden a varios servicios según sea su necesidad. En el siguiente diagrama (2.6.3.) se muestra un ejemplo de esta arquitectura:

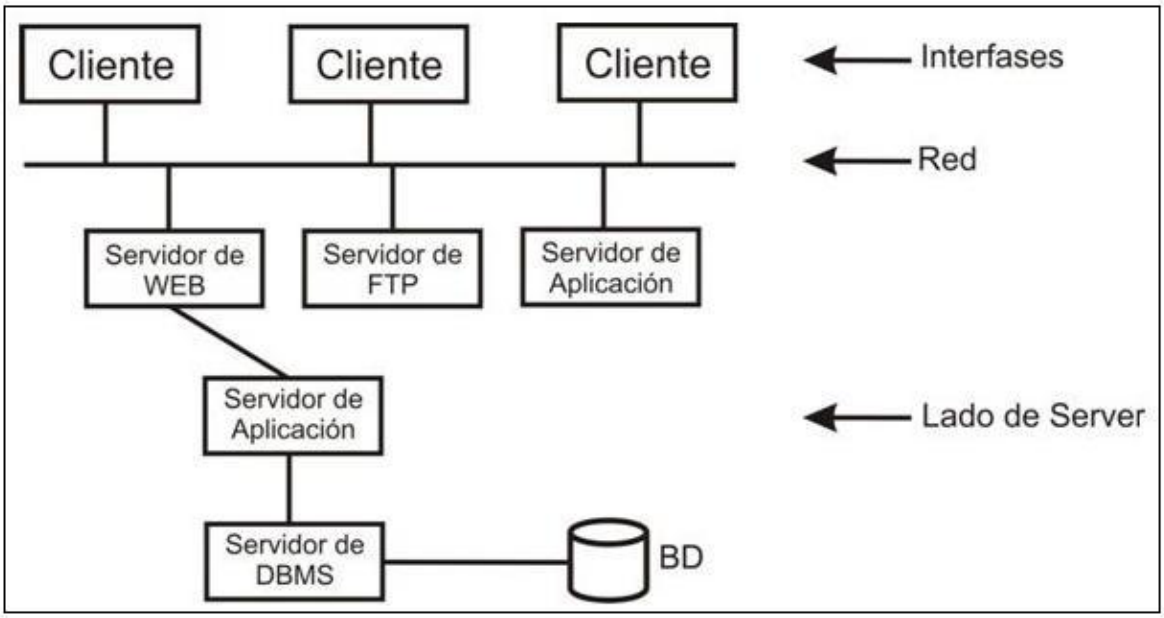

Figura 2.6.5. Esquema del Modelo de N Capas.

Este modelo se utiliza cuando se tiene una arquitectura compleja compuesta por varios servicios serializados que de manera conjunta resulten algunos problemas específicos. Tenemos como ejemplos de esta arquitectura:

- Sistemas de acceso a BD mediante interfases WEB.
- Sistemas que utilizan Prexies y firewalls.
- Sistemas de manejo remoto de archivos con derecho a ejecución.

2.7. Cliente

El cliente es el proceso que permite al usuario formular los requerimientos y pasarlos al servidor, se le conoce con el término **front-end**.

El Cliente normalmente maneja todas las funciones relacionadas con la manipulación y despliegue de datos, por lo que están desarrollados sobre plataformas que permiten construir interfaces gráficas de usuario (GUI), además de acceder a los servicios distribuidos en cualquier parte de una red.

Las funciones que lleva a cabo el proceso cliente se resumen en los siguientes puntos:

- Administrar la interfaz de usuario.
- Interactuar con el usuario.
- Procesar la lógica de la aplicación y hacer validaciones locales.
- Generar requerimientos de bases de datos.
- Recibir resultados del servidor.
- Formatear resultados.

2.8. Servidor

Es el proceso encargado de atender a múltiples clientes que hacen peticiones de algún recurso administrado por él. Al proceso servidor se le conoce con el término back-end. El servidor normalmente maneja todas las funciones relacionadas con la mayoría de las reglas del negocio y los recursos de datos.

Las funciones que lleva a cabo el proceso servidor se resumen en los siguientes puntos:

- Aceptar los requerimientos de bases de datos que hacen los clientes.
- Procesar requerimientos de bases de datos.
- Formatear datos para trasmitirlos a los clientes.
- Procesar la lógica de la aplicación y realizar validaciones a nivel de bases de datos.

2.9. Características de la arquitectura Cliente/Servidor

Las características básicas de una arquitectura Cliente/Servidor son:

- Combinación de un cliente que interactúa con el usuario, y un servidor que interactúa con los recursos compartidos. El proceso del cliente proporciona la interfaz entre el usuario y el resto del sistema. El proceso del servidor

actúa como un motor de software que maneja recursos compartidos tales como bases de datos, impresoras, módems, etc.

- Las tareas del cliente y del servidor tienen diferentes requerimientos en cuanto a recursos de cómputo como velocidad del procesador, memoria, velocidad y capacidades del disco y input-output devices.
- Se establece una relación entre procesos distintos, los cuales pueden ser ejecutados en la misma máquina o en máquinas diferentes distribuidas a lo largo de la red.
- Existe una clara distinción de funciones basada en el concepto de "servicio", que se establece entre clientes y servidores.
- La relación establecida puede ser de muchos a uno, en la que un servidor puede dar servicio a muchos clientes, regulando su acceso a recursos compartidos.
- Los clientes corresponden a procesos activos en cuanto a que son éstos los que hacen peticiones de servicios a los servidores. Estos últimos tienen un carácter pasivo ya que esperan las peticiones de los clientes.
- No existe otra relación entre clientes y servidores que no sea la que se establece a través del intercambio de mensajes entre ambos. El mensaje es el mecanismo para la petición y entrega de solicitudes de servicio.
- El ambiente es heterogéneo. La plataforma de hardware y el sistema operativo del cliente y del servidor no son siempre la misma. Precisamente una de las principales ventajas de esta arquitectura es la posibilidad de conectar clientes y servidores independientemente de sus plataformas.
- El concepto de escalabilidad tanto horizontal como vertical es aplicable a cualquier sistema Cliente/Servidor. La escalabilidad horizontal permite agregar más estaciones de trabajo activas sin afectar significativamente el rendimiento. La escalabilidad vertical permite mejorar las características del servidor o agregar múltiples servidores.

2.10. Ventajas del esquema Cliente/Servidor

Entre las principales ventajas del esquema Cliente/Servidor están:

- Uno de los aspectos que más ha promovido el uso de sistemas Cliente/Servidor, es la existencia de plataformas de hardware cada vez más baratas. Esta constituye a su vez una de las más palpables ventajas de este esquema, la posibilidad de utilizar máquinas considerablemente más baratas que las requeridas por una solución centralizada, basada en sistemas grandes. Además, se pueden utilizar componentes, tanto de hardware como de software, de varios fabricantes, lo cual contribuye considerablemente a la reducción de costos y favorece la flexibilidad en la implantación y actualización de soluciones.
- El esquema Cliente/Servidor facilita la integración entre sistemas diferentes y comparte información permitiendo, por ejemplo que las máquinas ya existentes puedan ser utilizadas pero utilizando interfaces mas amigables al usuario. De esta manera, podemos integrar PCs con sistemas medianos y grandes, sin necesidad de que todos tengan que utilizar el mismo sistema operacional.
- Al favorecer el uso de interfaces gráficas interactivas, los sistemas Construidos bajo este esquema tienen mayor interacción y más intuitiva con el usuario. En el uso de interfaces gráficas para el usuario, el esquema Cliente/Servidor presenta la ventaja, con respecto a uno centralizado, de que no es siempre necesario transmitir información gráfica por la red pues esta puede residir en el cliente, lo cual permite aprovechar mejor el ancho de banda de la red.
- Una ventaja adicional del uso del esquema Cliente/Servidor es que es más rápido el mantenimiento y el desarrollo de aplicaciones, pues se pueden emplear las herramientas existentes (por ejemplo los servidores de SQL o las herramientas de más bajo nivel como los sockets o el RPC).

- La estructura inherentemente modular facilita además la integración de nuevas tecnologías y el crecimiento de la infraestructura computacional, favoreciendo así la escalabilidad de las soluciones.
- El esquema Cliente/Servidor contribuye además, a proporcionar, a los diferentes departamentos de una organización, soluciones locales, pero permitiendo la integración de la información relevante a nivel global.

2.11. Desventajas del esquema Cliente/Servidor

Entre las principales desventajas del esquema Cliente/Servidor están:

- El mantenimiento de los sistemas es más difícil pues implica la interacción de diferentes partes de hardware y de software, distribuidas por distintos proveedores, lo cual dificulta el diagnóstico de fallas.
- Se cuenta con muy escasas herramientas para la administración y ajuste del desempeño de los sistemas.
- Es importante que los clientes y los servidores utilicen el mismo mecanismo (por ejemplo sockets o RPC), lo cual implica que se deben tener mecanismos generales que existan en diferentes plataformas.
- Además, hay que tener estrategias para el manejo de errores y para mantener la consistencia de los datos.
- La seguridad de un esquema Cliente/Servidor es otra preocupación importante. Por ejemplo, se deben hacer verificaciones en el cliente y en el servidor.
- El desempeño es otro de los aspectos que se deben tener en cuenta en el esquema Cliente/Servidor. Problemas de este estilo pueden presentarse por congestión en la red, dificultad de tráfico de datos, etc.

Capítulo III

Diseño de la Aplicación

3.1. Introducción

Se verá lo que es el diseño de la aplicación y cada una de las partes que tendrá la base de datos tanto en la parte del cliente así como la del servidor.

3.2. Diagrama Entidad-Relación

El modelo de datos entidad/relación (E-R) está basado en una percepción del mundo real que consta de un conjunto de objetos básicos llamados entidades y de relaciones entre estos objetos.

Una entidad es una <<cosa>> u <<objeto>> en el mundo real que es distinguible de otros objetos. Por ejemplo para el caso que nos ocupa un inmueble es una entidad. Las entidades se describen en una base de datos mediante un conjunto de atributos. Una relación es una asociación entre varias entidades.

El método E-R parte del hecho de que uno de los resultados del análisis del sistema es la compresión clara de cuáles son las entidades incluidas.

Un modelo E-R asiste al diseñador para definir y entender las cosas significativas acerca de cuál información necesita ser conocida o manejada y también las relaciones entre estas cosas.

Un grupo de entidades similares forman un conjunto de entidades. Las propiedades de las entidades son llamadas atributos, los valores para esos atributos están en un dominio de valores.

Un atributo o conjunto de atributos cuyos valores identifican únicamente cada entidad en un conjunto de entidades es llamada una “llave” o “clave”.

El modelo E/R categoriza todos los elementos de un sistema como una entidad (una persona, lugar o cosa) o una relación entre entidades. Ambas construcciones son representadas por la misma estructura, una tabla.

La aplicación del modelo E/R requiere los siguientes pasos:

- Identificar las entidades del sistema y construir una tabla para representar cada entidad.
- Identificar las relaciones entre las entidades y las tablas actuales extendidas o crear tablas nuevas para representar estas relaciones. [6]

3.3. Relaciones

Las relaciones representan las reglas y la información que el negocio necesita.

- Una relación es una asociación importante entre dos entidades.
- Una relación es "que tiene que ver una cosa con otra".
- Una relación es bi-direccional, asociación importante entre dos entidades.

Los tipos de relaciones pueden ser:

1. **Relaciones Muchos a Uno**. Estas son las más comunes y muestra que una relación tiene un grado de uno o más en una dirección y uno y solamente uno en la otra dirección.
2. **Relaciones Muchos a Muchos**. Hay un grado de uno o más en ambas direcciones y también son muy comunes. Usualmente son opcionales en ambas direcciones.
3. **Relaciones Uno a Uno**. Hay un grado de uno y solo uno en ambas direcciones. Este tipo de relaciones es bastante raro, comúnmente indican que las dos entidades son realmente la misma entidad en términos de negocio. [7]

Las entidades se describen en una base de datos mediante un conjunto de atributos.

Los tipos de relaciones se expresan en el modelo entidad/relación, como lo indica la figura 3.3.1.

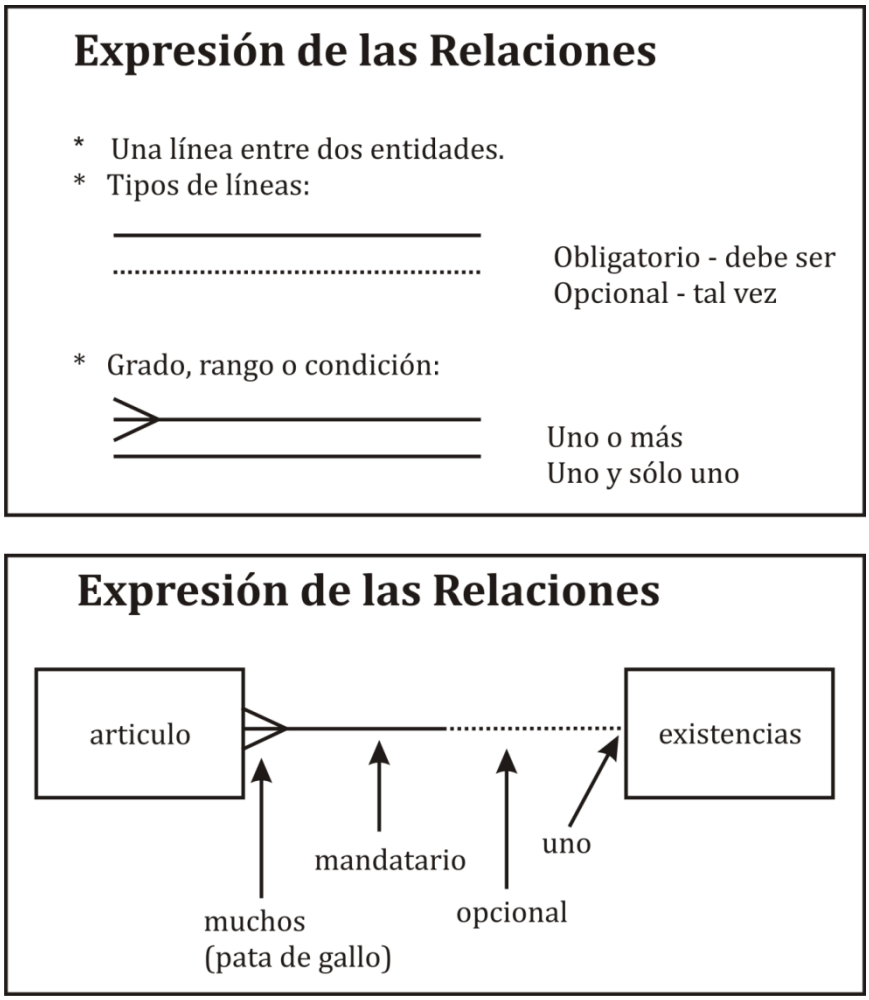

Figura 3.3.1. Representación de los tipos de relaciones en el modelo entidad/relación.

3.4. Atributos.

Los atributos son información acerca de una entidad que necesita ser conocida o descrita. Los atributos describen una entidad por que la:

- Habilitan.
- Identifican.
- Clasifican.
- Numeran.
- Expresa su estado.

En la Figura 3.4.2. Se muestra como cada una de las entidades del sistema, además de la relación entre ellas

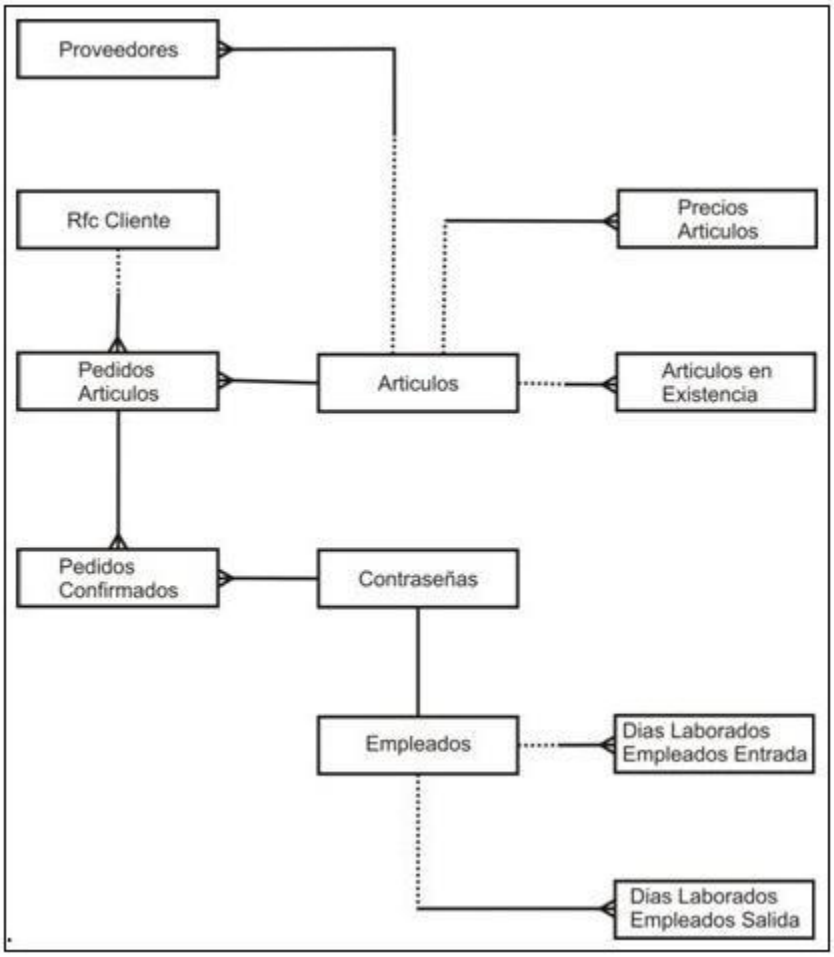

Figura 3.4.2. Diagrama entidad-relación.

3.5. Modelo Relacional Normalizado

Para el experto diseñador de bases de datos, derivar entidades o registros de tipo conceptual de un grupo de datos se puede hacer intuitivamente. Sin embargo, tal intuición no siempre surge espontáneamente en los principiantes, especialmente cuando el diseño es muy complejo.

La teoría de la normalización es una ayuda que proporciona un procedimiento riguroso para el diseño de base de datos. Una base de datos mal diseñada puede funcionar

inicialmente pro puede mostrar anomalías en el almacenamiento debidas al agrupamiento indiscriminado de los campos cuando se efectúan en los archivos las operaciones de inserción, actualización o eliminación. La teoría de la normalización ayuda a reconocer las cualidades no deseadas en un archivo y la forma de corregirlas.

Con el procedimiento de normalización, un archivo conceptual se representa como una tabla de dos dimensiones llamada relación: la forma más simple para representar datos mediante una tabla.

El método basado en la normalización supone que el análisis se sistemas produce una lista de los campos de datos de la aplicación y las relaciones entre ellas para que posteriormente sea el proceso de normalización el que separe los campos que identifican identidades de aquellos que solo las describen. Una relación no-normalizada es una relación que contiene varias ocurrencias de algunos valores en cualquiera de los campos. Por otro lado, una relación normalizada solo permite una ocurrencia de un valor en cada campo. Las relaciones normalizadas se agrupan en cuatro categorías llamadas formas normales FN siendo cada nivel una descomposición más completa de una relación que la del nivel anterior.

La meta final del proceso de normalización es la agrupación de todos los atributos (o campos) de una base de datos en relaciones adecuadas para que la base se pueda almacenar con el mínimo de datos redundantes.

El propósito de este proceso es quitar las cualidades indeseables de una relación que puedan causar anomalías en el almacenamiento cuando se efectúen operaciones de actualización en la base de datos.

El proceso de normalización empieza con la combinación de todos los datos de la base en una relación, la que a su vez se descompone en dos o más relaciones más pequeñas. Se efectúan descomposiciones sucesivas de las relaciones intermedias hasta que todas las relaciones obtenidas pertenecen a la cuarta forma normal (4FN).

Antes de describir el proceso de la normalización, se debe describir la manera de determinar la forma normal de una relación a partir de su relación de atributos conocida como “dependencia funcional”. [8]

3.6. Dependencia Funcional.

El análisis de una relación de dependencia funcional entre los campos de una relación permite clasificar la relación en una de las cuatro formas normales. El concepto de dependencia funcional (DF), se tomo de las matemáticas elementales. Se dice que Y es función de X, Y = f(X), si el valor de Y esta siempre determinado por el valor de X. si se aplica la misma terminología a una relación, la dependencia funcional entre los atributos A y B en una relación se define como sigue: El atributo A es funcionalmente dependiente del atributo B si el valor de A esta determinado por el valor de B.

Tal dependencia se simboliza:

El modelo relacional normalizado muestra cada una de las tablas con todos los campos a utilizar, teniendo en cuenta que cada una de las tablas están relacionadas por medio de los campos llave (primary key) véase en la Figura 3.6.3. Cada una de las entidades normalizadas.

Tabla Artículos	
# *	id_articulo
*	Nombre
*	Marca
*	Modelo
*	Proveedor
*	Código
*	Imagen

Tabla Empleados	
# *	id_empleado
*	nombre
*	apellido_paterno
*	apellido_materno
*	dirección
*	telefono_casa
*	telefono_cell
o	e_mail
*	Cargo

Tabla Días Laborados Empleado Entrada	
# *	id_empleado
# *	nombre_empleado
*	Fecha
*	hora_entrada

Tabla Días Laborados Empleado Salida	
# *	id_empleado
*	hora_entrada

Tabla Artículos en Existencia	
# *	id_articulo
# *	no_sucursal
*	talla
*	color
*	existencias

Tabla Proveedor	
# *	id_proveedor
# *	nombre_articulo
*	nombre
*	apellido_paterno
*	apellido_materno
*	dirección
*	telefono_casa
*	telefono_cell
o	e_mail

Tabla Contraseñas	
# *	id_usuario
*	nombre_usuario
*	password_usuario

Tabla Precios Artículos	
# *	id_articulo
# *	precio_compra
*	precio_mayoreo
*	precio_menudeo

Tabla Pedidos Artículos	
# *	id_pedido_a
# *	id_articulo
*	existencias

Tabla RFC Clientes	
# *	id_cliente
*	cliente
*	r.f.c.
*	dirección
*	ciudad
*	cod_Postal

Tabla Pedidos Confirmados	
# *	id_pedido_a
*	id_usuario
*	fecha_p
*	fin_p

Nota:

 # Llave

*** Obligatorio**

o Opcional

Figura 3.6.3. Entidades Normalizadas

A continuación, se encuentra un listado de las tablas que forman parte de la base de datos para utilizar en la aplicación cliente/servidor, con sus respectivas estructuras.

Con el signo **#** hemos representado a las llaves (claves primarias), mientras que hemos utilizado ***** para los campos obligatorios, y **O** para los campos opcionales:

Tabla Artículos				
	Nombre del Campo	**Tipo Dato**	**Tama-ño/Formato**	**Descripción**
# *	id_articulo	int	6	Clave única que identifica a cada artículo.
*	nombre	Varchar	50	Nombre articulo.
*	Marca	Varchar	25	Marca articulo.
*	modelo	Varchar	25	Modelo articulo.
*	proveedor	Varchar	25	Proveedor articulo.
*	código	Varchar	25	Código articulo.
*	imagen	Blob		Imagen articulo.

Tabla 3.6.1. Estructura de la tabla Artículos.

Tabla Artículos en Existencia				
	Nombre del Campo	**Tipo Dato**	**Tama-ño/Formato**	**Descripción**
# *	id_articulo	Int	6	Clave única que identifica a cada artículo.
# *	no_sucursal	Int	6	Número sucursal.
*	Talla	Varchar	25	Talla articulo.
*	Color	Varchar	25	Color articulo.
*	existencias	Int	6	Existencias artículos.

Tabla 3.6.2. Estructura de la tabla Artículos en Existencia.

Precios Artículos				
	Nombre del Campo	**Tipo Dato**	**Tama-ño/Formato**	**Descripción**
# *	id_articulo	Int	6	Clave única que identifica a cada artículo.
# *	precio_compra	Float	4,2	Precios articulo.
*	precio_mayoreo	Float	4,2	Precio mayoreo artículo.
*	precio_menudeo	Float	4,2	Precio menudeo artículo.

Tabla 3.6.3. Estructura de la tabla Precios Artículos.

Tabla Empleados				
	Nombre del Campo	**Tipo Dato**	**Tamaño/Formato**	**Descripción**
# *	id_empleado	Int	6	Clave única que identifica a cada empleado.
*	nombre	Varchar	25	Nombre empleado.
*	apellido_paterno	Varchar	25	Apellido paterno empleado.
*	apellido_materno	Varchar	25	Apellido materno empleado.
*	dirección	Varchar	50	Dirección empleado.
*	telefono_casa	Varchar	25	Teléfono casa empleado.
*	telefono_cell	Varchar	25	Celular empleado.
o	e_mail	Varchar	50	E-mail empleado.
*	Cargo	Varchar	25	Cargo empleado.

Tabla 3.6.4. Estructura de la tabla Empleados.

Tabla Días Laborados Empleado Entrada				
	Nombre del Campo	**Tipo Dato**	**Tamaño/Formato**	**Descripción**
# *	id_empleado	Int	6	Clave única que identifica a cada empleado.
# *	nombre_empleado	Int	6	Número sucursal.
*	Fecha	Date	dd-mm-aaaa	Fecha de día laborado.
*	hora_entrada	Time	hh:mm:ss	Hora de entrada.

Tabla 3.6.5. Estructura de la tabla Días Laborados Empleado Entrada.

Tabla Días Laborados Empleado Salida				
	Nombre del Campo	**Tipo Dato**	**Tamaño/Formato**	**Descripción**
# *	id_empleado	Int	6	Clave única que identifica a cada empleado.
*	hora_entrada	Time	hh:mm:ss	Hora de entrada.

Tabla 3.6.6. Estructura de la tabla Días Laborados Empleado Salida.

Tabla Proveedor				
	Nombre del Campo	**Tipo Dato**	**Tamaño/Formato**	**Descripción**
#*	id_proveedor	Int	6	Clave única que identifica a cada empleado.
#*	nombre_articulo	Varchar	25	Nombre articulo.
*	Nombre	Varchar	25	Nombre proveedor.
*	apellido_paterno	varchar	25	Apellido paterno proveedor.
*	apellido_materno	Varchar	25	Apellido materno proveedor.
*	Dirección	Varchar	50	Dirección proveedor.
*	telefono_casa	Varchar	25	Teléfono cada proveedor.
*	telefono_cell	Varchar	25	Celular proveedor.
o	e_mail	Varchar	50	E-mail proveedor.

Tabla 3.6.7. Estructura de la tabla Proveedor.

Tabla RFC Clientes				
	Nombre del Campo	**Tipo Dato**	**Tamaño/Formato**	**Descripción**
#*	id_cliente	Int	6	Clave única que identifica a cada cliente.
*	Cliente	Varchar	50	Nombre cliente.
*	r.f.c.	Varchar	25	R.f.c. cliente.
*	dirección	Varchar	100	Dirección cliente.
*	ciudad	Varchar	25	Ciudad cliente.
*	cod_Postal	Varchar	5	Código postal cliente.

Tabla 3.6.8. Estructura de la tabla RFC Clientes.

Tabla Contraseñas				
	Nombre del Campo	**Tipo Dato**	**Tamaño/Formato**	**Descripción**
#*	id_usuario	Int	6	Clave única que identifica a cada usuario.
*	nombre_usuario	Varchar	80	Nickname del usuario.
*	password_usuario	Varchar	80	Password del usuario.

Tabla 3.6.9. Estructura de la tabla Contraseñas.

Tabla Pedidos Artículos				
	Nombre del Campo	**Tipo Dato**	**Tama-ño/Formato**	**Descripción**
# *	id_pedido_a	Int	6	Clave única que identifica a cada pedido.
# *	id_articulo	Int	6	Clave única que identifica a cada artículo.
*	existencias	Smallint	4	Cantidad de discos en el pedido.

Tabla 3.6.10. Estructura de la tabla Pedidos Artículos.

Tabla Pedidos Confirmados				
	Nombre del Campo	**Tipo Dato**	**Tama-ño/Formato**	**Descripción**
# *	id_pedido_a	Int	6	Clave única que identifica a cada pedido.
*	id_usuario	Int	6	Clave única que identifica a cada usuario.
*	fecha_p	dateti-me	dd-mm-aaaa	Fecha en que se realizo el pedido.
*	fin_p	Char	1	Indica si el pedido se confirmo (s) o no (n).

Tabla 3.6.11. Estructura de la tabla Pedidos Confirmados.

3.7. Diagrama de Casos de Uso

Pasamos ahora a dar una visión general de cómo de cómo se desarrolla el trabajo a través de todos los flujos de trabajo. Nos centraremos en el hecho de que está dirigido por casos de uso.

El administrador tendrá acceso a todos los nodos de la BDD, así como a toda la base de datos, el administrador podrá hacer lo siguiente:

- Altas de Artículos
- Bajas de Artículos
- Consulta General de Artículos
- Modificaciones de Artículos, Empleados y Proveedores
- Crear Registros Empleados y Proveedores

- ❖ Eliminar Registros Empleados y Proveedores
- ❖ Consultas a Empleados y Proveedores

Ver la figura 3.7.4. Las acciones que el administrador podrá hacer.

El empleado de cada una de las sucursales podrá hacer lo siguiente:

- ❖ Registrar a la BD los artículos vendidos por Días
- ❖ Consultar BD (cuestión de existencias)
- ❖ Facturar (Nota ó Factura)
- ❖ Checar su hora de Entrada y Salida

Ver figura 3.7.4. Las acciones que el empleado podrá hacer.

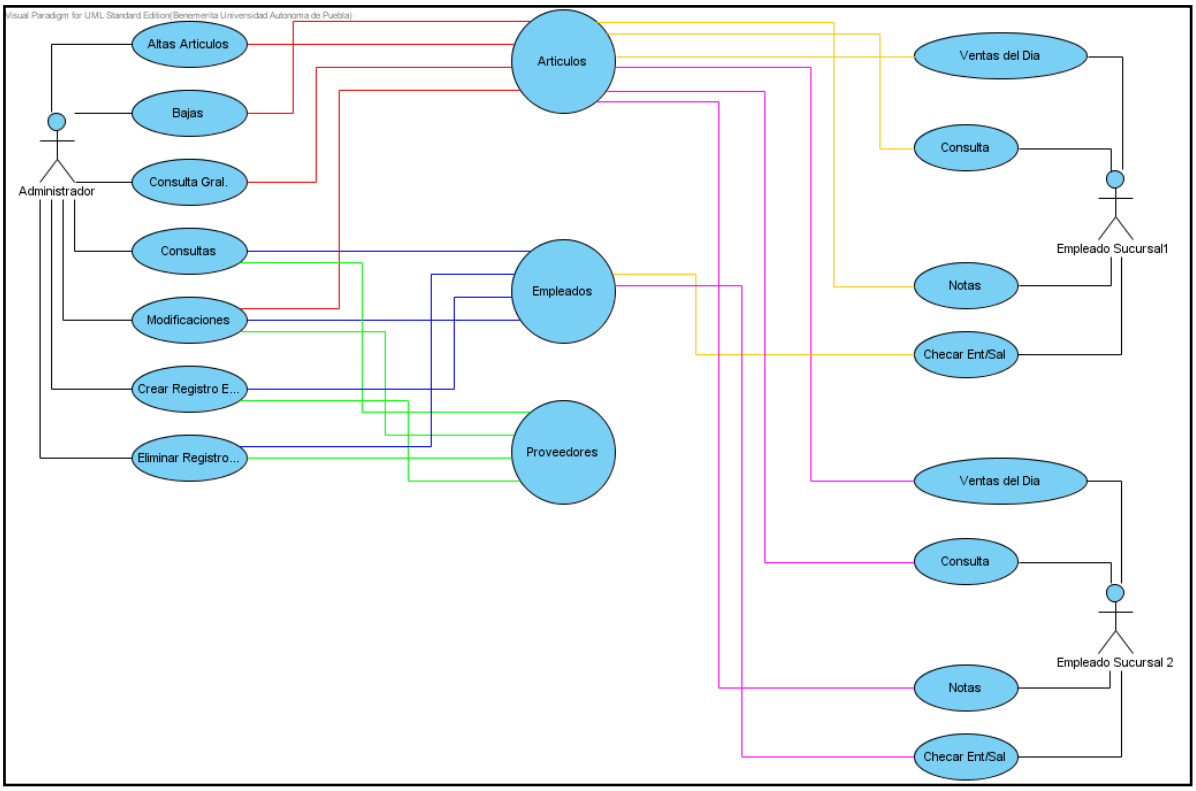

Figura 3.7.4. Diagrama de Casos de uso

En la figura 3.7.4. El diagrama de casos de uso con un actor (administrador) y siete casos de uso, en caso del (empleado sucursal1 y empleado sucursal2) tienen cinco

casos de uso cada uno, los casos de uso están relacionados con las tablas artículos, empleados y proveedores tanto el administrador como los empleados. [6]

3.8. Diagrama de Casos de uso Mediante una colaboración (modelo análisis)

En la Figura 3.8.5. Escribimos cómo se lleva a cabo el caso de uso Altas Artículos mediante una colaboración (es decir, una realización de caso de uso) con una dependencia de "traza" (una traza es un estereotipo de dependencia que se indica mediante las comillas << y>>) entre ellos, y cómo tres clases participan y desempeñan roles en la realización del caso de uso. Como puede verse en la figura, la notación para una realización de caso de uso o colaboración es una elipse con una línea de trazo discontinuo. [6]

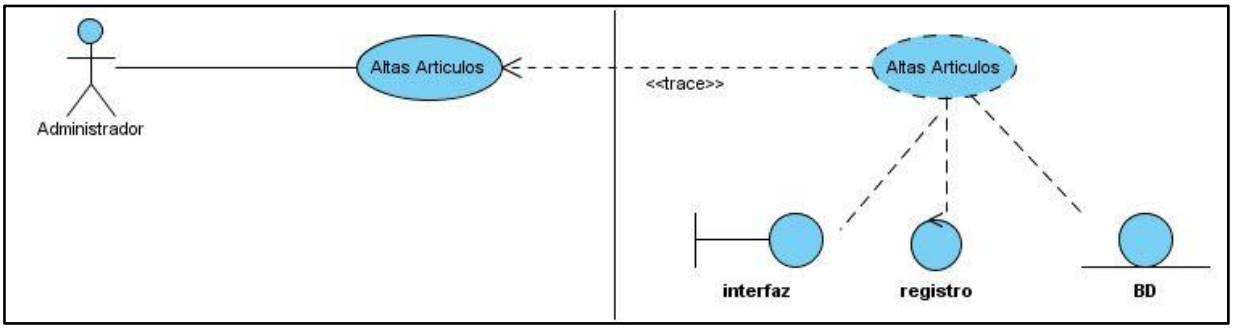

Figura 3.8.5. Modelo Análisis.

En la Figura 3.8.5. Las clases de análisis que participan en la realización de Altas Artículos. Interfaz son clases de interfaz, registro es una clase de control, y bd es una clase de entidad.

3.9. Uso de Diagrama de colaboración para describir una realización de caso de uso (modelo análisis)

En la figura 3.9.6. Utilizamos un diagrama de colaboración para describir como una sociedad de objetos de análisis lleva a cabo la realización del caso de uso Altas artículos.

El diagrama muestra cómo el control pasa de un objeto a otro a medida que se lleva a cabo el caso de uso, y los mensajes que se envían entre los objetos. Un mensaje de un objeto dispara al objeto receptor para que tome el control y lleve a cabo una de las responsabilidades de la clase.

El nombre de un mensaje indica el motivo del objeto que realiza la llamada en su interacción con el objeto invocado. [6]

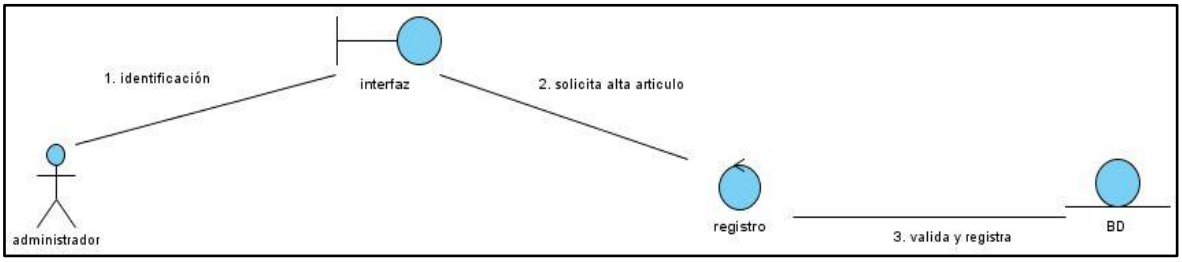

Figura 3.9.6. Modelo de Análisis.

La Figura 3.9.6. Un diagrama de colaboración para la realización del caso de uso Altas Artículos en el modelo de análisis.

3.10. Creación del modelo de diseño a partir del modelo de análisis

El modelo de diseño se crea tomando el modelo de análisis como entrada principal, pero se adapta al contorno de implementación elegido. Por tanto, mientras en el modelo de análisis sirve como una primera aproximación del modelo de diseño, el modelo de diseño funciona como esquema para la implementación.

De igual forma que el modelo de análisis, el modelo de diseño también define clasificadores (clases, subsistemas, e interfaces), relaciones entre esos clasificadores, y colaboraciones que llevan a cabo los casos de uso (las realizaciones de casos de uso). Sin

embargo, los elementos definidos en el modelo de diseño son las "contrapartida de diseño" de los elementos, más conceptuales, definidos en el modelos de análisis, en el sentido de que los primeros (diseño) se adaptan al entorno de la implementación mientras que los últimos (análisis) no. Dicho de otra forma, el modelo de diseño es mas "físico" por naturaleza, mientras que el modelo de análisis es más "conceptual".

Puede hacerse la traza de una realización de caso de uso en el modelo de análisis a partir de una realización de caso de uso en el modelo de diseño.

Recuérdese de la sección 3.8.5. (Figura 3.8.5.) Que las clases de análisis interfaz, registro, bd participan en la realización del caso de uso altas artículos en el modelo de análisis. Sin embargo, cuando se diseñan esas clases de análisis, todas ellas especifican y hacen surgir clases de diseño más refinadas que se adaptan al entorno de implementación, cómo se ejemplifica en la Figura 3.10.7. [6]

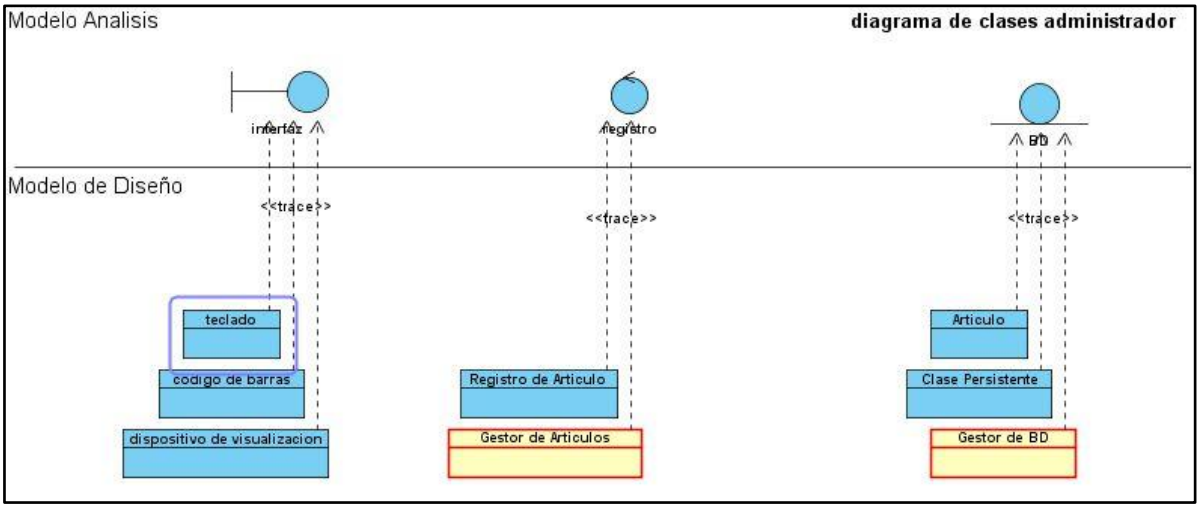

Figura 3.10.7. Clases de diseño del modelo de análisis con trazas hacia clases del modelo de análisis.

Por ejemplo, la clase de análisis llamada interfaz se diseña mediante tres clases de diseño: teclado, código de barras y dispositivos de visualización.

Obsérvese en la figura 3.10.7. Que la mayoría de las clases de diseño normalmente tienen una sola traza a una clase de análisis. Esto es habitual para las clases de diseño

que son especificas de la aplicación, diseñadas para dar soporte a una aplicación o a un conjunto de ellas. Por lo tanto, la estructura del sistema definida por el modelo de análisis se conserva de forma natural durante el diseño.

Como consecuencia, la realización del caso de uso altas artículos en el modelo de diseño debe describir como se realiza el caso de uso en términos de clases de diseño correspondientes. La figura 3.10.8. Muestra un diagrama de clases que es parte de la realización del caso de uso.

Es evidente que este diagrama de clases incluye más detalles que el diagrama de clases del modelo de análisis (Figura 3.9.6.), debemos identificar la interacción detallada entre los objetos de diseño que tiene lugar cuando se lleva a cabo el caso de uso en el modelo de diseño. Utilizamos principalmente diagramas de secuencia para modelar las interacciones entre objetos del diseño, como se muestra en la Figura 3.10.9.

El diagrama de secuencia muestra cómo el control que comienza en la esquina superior izquierda pasa de un objeto a otro a medida que se ejecuta el caso de uso y a medida que se envían mensajes entre objetos. Un mensaje enviado por un objeto dispara la toma del control en el objeto receptor y la realización de las operaciones de su clase. [6]

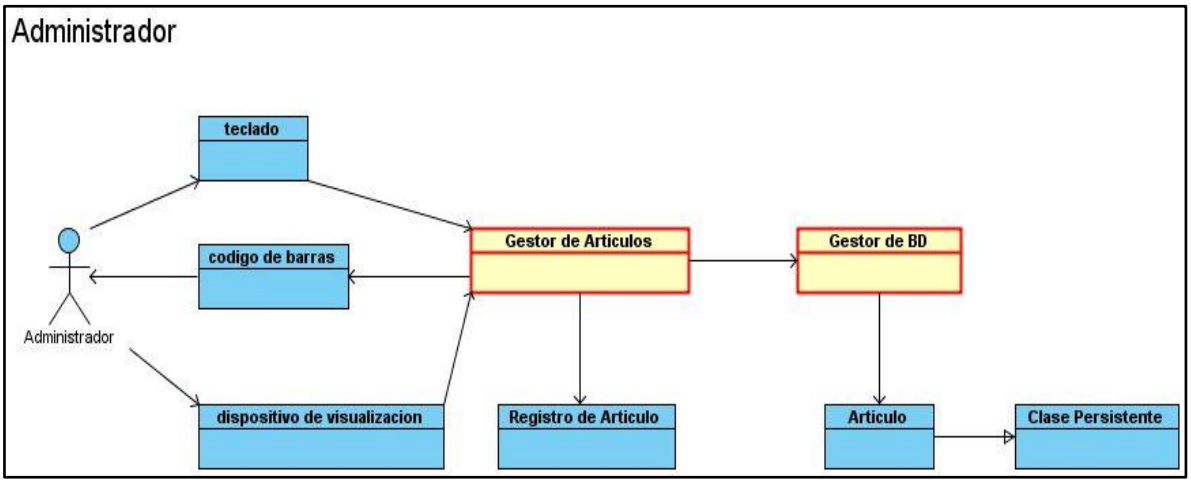

Figura 3.10.8. Un diagrama de clases que es parte de la realización del caso de uso altas artículos en el modelo de diseño. Cada clase de diseño participa y asume roles en la realización del caso de uso.

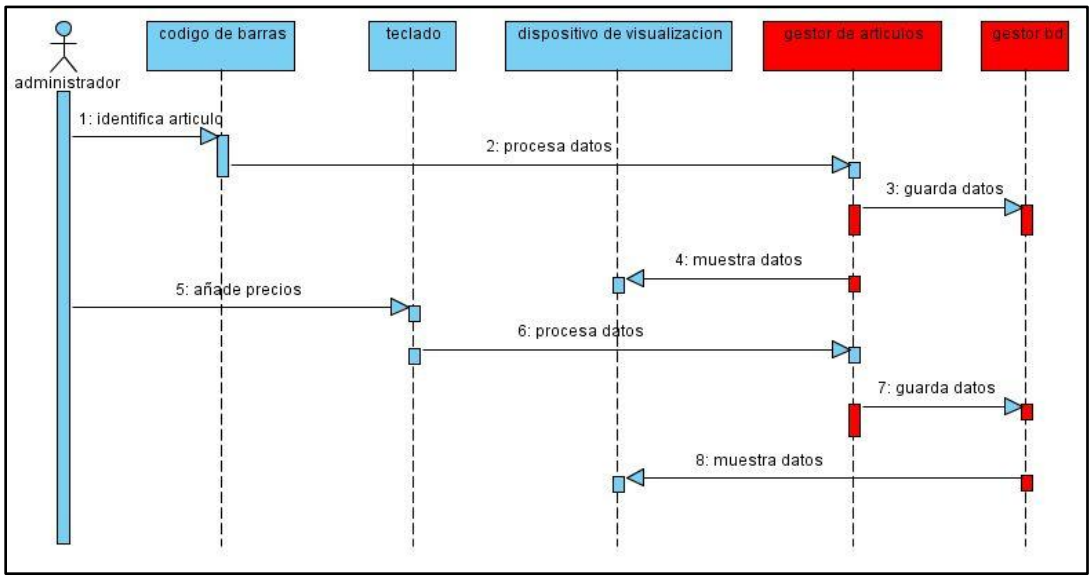

Figura 3.10.9. Un diagrama de secuencia que es parte de la realización del caso de uso altas artículos en el modelo de diseño.

Capítulo IV

Implementación.

4.1. Introducción.

Tengamos en cuenta que en este capítulo se mostrara parte del código utilizado en la elaboración de la aplicación. No cabe mencionar que para la elaboración de la aplicación se utilizo el Lenguaje de Programación PHP (Hypertext Pre-processor) y el servidor de bases de datos MYSQL.

PHP es un lenguaje interpretado de propósito general ampliamente usado y que está diseñado especialmente para desarrollo web y puede ser incrustado dentro de código HTML. Generalmente se ejecuta en un servidor web, tomando el código en PHP como su entrada y creando páginas web como salida. Puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas sin costo alguno.

4.2. Aplicación Completa

La aplicación estará formada por dos sesiones son los siguientes:

Sesión Administrador.	La que tendrá todos los privilegios de la base de datos y la del sistema en general.
Sesión Usuarios.	Como su nombre lo dice solo será usuario y tendrá aque llos privilegios que le asigne el administrador.

Bueno para la programación de la aplicación se utilizaron las cuatro operaciones básicas de una base de datos las cuales son:

- Consulta (Select)
- Inserción (Insert)
- Borrar (Delete)
- Modificar (Update)

1) Comando Consulta (Select)

El lenguaje de consultas SQL es a la vez sintético y muy potente. Cuando queremos realizar consultas de selección (esto es recuperar registros de una o más tablas) podemos recuperar millones de registros con tan solo una línea de código. A continuación, hallaremos las distintas posibilidades que existen para extraer datos y luego trabajar con ellos.

a) Selección Simple

Con el término **selección simple** hacemos referencia a consultas a una sola tabla por vez. Ese tipo de consultas no son frecuentemente utilizadas en aquellos sistemas que poseen un diseño complejo.

```
SELECT * FROM TABLA1;
```

b) Consultas Multitabla

Consiste en consultas que implican varias tablas a la vez, esto significa que recuperan datos de varias fuentes.

```
SELECT * FROM TABLA1, TABLA2, WHERE TABLA1.A = TABLA2.B;
```

En la elaboración de la aplicación se ocupo tanto la ***selección simple como la multitabla*** por ejemplo:

En este pedazo de código lo que se hace es hacer una simple consulta, utilizando consultas multitabla. Y lo como resultado de lo que se obtiene se ve en la figura 4.2.1.

```
$sql  = "select * from articulos_suc1, articulos_en_existencia_suc1, pre
                                                        cios_articulos_suc1";
$sql .= " where articulos_suc1.id_articulo =
                                        articulos_en_existencia_suc1.id_articulo";
$sql .= " and articulos_suc1.id_articulo = precios_articulos_suc1.id_articulo";
$sql .= " order by nombre asc";
```

Articulo	Marca	Modelo	Proveedor	Codigo	No Sucursal	Talla	Color	Existencias	Precio Compra	Precio Mayoreo	Precio menudeo
balon_basquetball_no.5_junior	strack	junior	Strack	bb5j	1	5	varios	0	40.00	55.00	60.00
balon_futbol_no.3_economico			Ramiro	bf3e	1	3	varios	1	37.00	43.00	50.00
balon_futbol_no.4_nike	nike	90	Campeón	bf4n	1	4	varios	11	350.00	420.00	490.00
guante_portero_replica_rinat	rinat	calero	Aurelio	gprr	1	4-12	varios	0	110.00	140.00	165.00
venda_5_cm			Farmacia	v5c	1		blanco	0	1.20	4.00	7.00

Figura 4.2.1 Consulta de la tablas artículos, artículos en existencia y precios artículos.

c) Subconsultas

Las subconsultas se implementaron en MySql a partir de la versión 4.1.

SELECT * FROM TABLA1 WHERE TABLA1.A = (SELECT B FROM TABLA WHERE C>3);

2) Comando Inserción (Insert)

Podemos realizar ingresos en una tabla a través de las funciones mysql_query o mysql_db_query, pero PHP nos brinda una función que puede ser muy útil cuando se trabaja con campos de auto incremento.

a) Campos Autoincrementables

MySql nos permite insertar registros en una tabla evitando de forma automática que un campo toma valores duplicados. Estos campos se llaman autoincrementables y se definen al momento de crear una tabla. [9]

Por ejemplo:

```
CREATE TABLE `articulos_suc1` (`id_articulo` int(6) unsigned NOT NULL auto_increment, `nombre` varchar(50) default NULL, `marca` varchar(25) default NULL, `modelo` varchar(25) default NULL, `proveedor` varchar(25) default NULL, `código` varchar(25) default NULL, `imagen` blob NOT NULL, PRIMARY KEY  (`id_articulo`) ) TYPE=MyISAM;

$sql  = "insert into articulos_suc1 values (";

$sql .= $max.", ";

$sql .= "'".$_POST[nombre]."', ";

$sql .= "'".$_POST[marca]."', ";

$sql .= "'".$_POST[modelo]."', ";

$sql .= "'".$_POST[proveedor]."', ";

$sql .= "'".$_POST[código]."', ";
```

```
        $sql .= "'".$_POST[imagen]."') ";

        $res =mysql_query ($sql);

        $res =mysql_query ('COMMIT');
$sql  = "Insert into articulos_en_existencia_suc1 valúes (";

        $sql .= $max.", ";

        $sql .= "'".$_POST[no_sucursal]."', ";

        $sql .= "'".$_POST[talla]."', ";

        $sql .= "'".$_POST[color]."', ";

        $sql .= "'".$_POST[existencias]."') ";

        $res =mysql_query($sql);

        $res =mysql_query('COMMIT');


        $sql  = "insert into precios_articulos_suc1 values (";

        $sql .= $max.", ";

        $sql .= "'".$_POST[precio_compra]."', ";

        $sql .= "'".$_POST[precio_mayoreo]."', ";

        $sql .= "'".$_POST[precio_menudeo]."') ";

        $res =mysql_query($sql);

        $res =mysql_query('COMMIT');
```

Como consecuencia de este pedazo de código el resultado arrojado, vea en la figura 4.2.2.

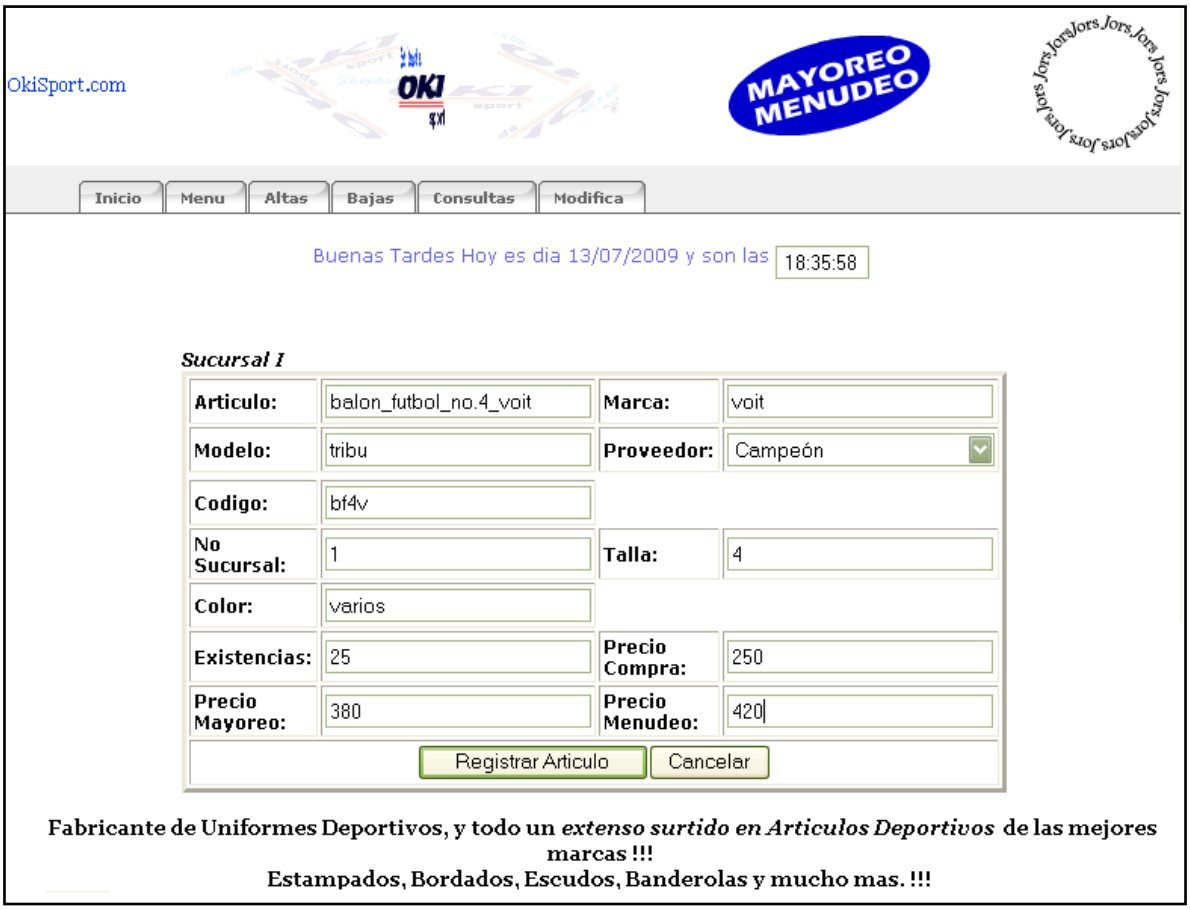

Figura 4.2.2. Inserción de datos en las tablas artículos, artículos en existencia y precios artículos.

3) Comando Borrar (Delete)

No existen funciones específicas para borrar tablas o registros de una tabla o columnas de una tabla; todo esto se hace mediante la función mysql_query o la mysql_db_query. [9]

Por ejemplo:

```
$sql  = "delete from articulos_suc1 where id_articulo = ".$_POST["$temp"];

$sql  = "delete from articulos_en_existencia_suc1 where id_articulo =
                                                    ".$_POST["$temp"];
```

```
$sql  = "delete from precios_articulos_suc1 where id_articulo =
                                                    ".$_POST["$temp"];
```

En este código lo único que hace es borrar algún artículo de las tablas artículos, artículos en existencia y precios artículos vea en la figura 4.2.3 el resultado arrojado por este pequeño código.

Figura 4.2.3 Borrado de los datos de las tablas artículos, artículos en existencia y precios artículos.

Existe la función mysql_drop_db para borrar bases de datos completas. Por ejemplo:

$bbas = mysql_drop_db (“mibase”) or DIE (“no se puede eliminar la base”);

Echo “
base de datos eliminada”;

4) Comando Modificar (Update)

Al igual que para borrar datos o estructuras, no existen funciones especificas para modificar o actualizar estructuras de tablas o registros: todo esto se hace mediante la función mysql_query o la mysql_db_query. [9]

Por ejemplo:

```
$sql =  " update articulos_suc1 set ";

$sql .= " nombre = '".$_POST[nombre]."', ";

$sql .= " marca = '".$_POST[marca]."', ";

$sql .= " modelo = '".$_POST[modelo]."', ";

$sql .= " proveedor = '".$_POST[proveedor]."', ";

$sql .= " código = '".$_POST[código]."' ";

$sql .= " where id_articulo = ".$_POST[id_articulo];


$sql =  " update articulos_en_existencia_suc1 set ";

$sql .= " id_articulo = '".$_POST[id_articulo]."', ";

$sql .= " no_sucursal = '".$_POST[no_sucursal]."', ";

$sql .= " talla = '".$_POST[talla]."', ";

$sql .= " color = '".$_POST[color]."', ";

$sql .= " existencias = '".$_POST[existencias]."' ";

$sql .= " where id_articulo = ".$_POST[id_articulo];

mysql_db_query("sucursal1",$db);
```

```
$sql =  “update precios_articulos_suc1 set ";

$sql .= " id_articulo = '".$_POST[id_articulo]."', ";

$sql .= " precio_compra = '".$_POST[precio_compra]."', ";

$sql .= " precio_mayoreo = '".$_POST[precio_mayoreo]."', ";

$sql .= " precio_menudeo = '".$_POST[precio_menudeo]."' ";

$sql .= " where id_articulo = ".$_POST[id_articulo];
```

El resultado de este código vea en la figura 4.2.4.

Figura 4.2.4. Modifica las tablas artículos, artículos en existencia y precios artículos.

Capítulo V

Resultados.

5.1 Introducción.

En el presente capítulo veremos algunas de las tantas ventanas que contiene la Aplicación Cliente/servidor.

5.2 Sesión Usuario

En la figura 5.2.1. Se muestra la pantalla de la ventana principal de la sesión de los usuarios, esta tiene varias pestañas, siendo cada una un medio, la cual el usuario va a poder utilizar. Más adelante en el punto 5.3. se muestra un pequeño mapa de cada una de las partes que conforma esta sesión.

Figura 5.2.1 Ventana principal de la sesión de los usuarios.

5.3 Mapa Sesión Usuario

- R.F.C. Clientes.
  - Inicio.
  - Altas R.F.C. Clientes.
  - Bajas R.F.C. Clientes.
  - Consultas R.F.C. Clientes.
  - Modifica R.F.C. Clientes.
- Días Laborados.
  - Inicio.
  - Entrada/Salida.
  - Consultas.
- Consulta Artículo.
  - Inicio.
  - Consulta Palabra Clave.
  - Consulta General Artículo.
- Carrito de Compras.
  - Inicio

5.4 Sesión Administrador

De igual manera que en la sesión del usuario se mostro la ventana principal, así también se muestra en la figura5.4.2 la ventana principal de la sesión del administrador. No cabe mencionar que también tiene varias pestañas, de las cuales con estas podrá el administrador dar de altas, bajas consultas y modificaciones a cada una de las tiendas, con el cual el podrá hacer las acciones necesarias dependiendo el caso o necesidad que se aparezca.

Figura 5.4.2 Ventana principal de la sesión del administrador.

5.5 Mapa Sesión Administrador

- Artículos.
  - Inicio.
  - Bodega.
    - Inicio.
    - Menú.
    - Altas.
    - Bajas.
    - Consultas.
    - Modifica.
  - Sucursal1.
    - Inicio.

        - Menú.
        - Altas.
        - Bajas.
        - Consultas.
        - Modifica.
- Empleados.
    - Inicio.
    - Altas Empleados.
    - Bajas Empleados.
    - Consultas Empleados.
    - Modifica Empleados.
    - Días Laborados.
        - Inicio.
        - Menú.
        - Bajas Días Laborados.
        - Consultas Días Laborados.
        - Modifica Días Laborados.
- Proveedores.
    - Inicio.
    - Altas Proveedor.
    - Bajas Proveedor.
    - Consultas Proveedor.
    - Modifica Proveedor.
- Nuevo Usuario.
    - Inicio.
    - Altas Usuario.
    - Bajas Usuario.
    - Consultas Usuario.
    - Modifica Usuario.

Capítulo VI

Conclusión.

En el mundo del comercio en especial de una cadena de tiendas de deporte, existe una gran variedad de artículos, ya sea futbol, basquetbol, voleibol, beisbol, Karate Do etc. Aquí es donde el empresario o dueño de este tipo de negocios tiene una gran incógnita, la de cómo saber cuántos productos tiene. Suponiendo que se las ingenia de una u otra manera el ya sabe cuánta mercancía tiene, pero el problema sigue ahí y lo más grave es saber que hasta cuando hará lo mismo para saber cuánta mercancía tiene. Es por eso que se creó una aplicación cliente servidor para una cadena de tiendas.

Analizando con cuidado el problema se comenzó por hacer una investigación y aplicando cierta tecnología, en este caso la de una base de datos, se llego como conclusión la de crear una aplicación que se encargue de llevar el control de todos los artículos que existen en este medio. La aplicación ha reducido tiempo en cuanto al conteo de la mercancía vendida y en existencia además esto ayuda a llevar el control exacto de compra y venta de los artículos; además al empresario le ha ayudado mucho a llevar el control de entrada y salida de cada uno de los empleados, Hoy en día la aplicación está en función y el problema que había, ahora ya no existe.

Haber hecho este tipo de trabajo fue una gran experiencia ya que pude aprender exactamente cómo es que funciona una aplicación cliente/servidor, la de saber cuáles son sus ventajas y desventajas, ya que si no se hace una muy buena investigación respecto a este, se puede confundir uno, con lo que es una aplicación distribuida, en cuestión del tiempo en la que se realizo la aplicación fue un poco largo, creo que la de haber puesto en prueba la aplicación fue una de las principales cuestiones por la que el tiempo se fue volando, pues en ese punto hubieron varias modificaciones. Respecto a lo económico, no se le invirtió mucho pues se contaba con la mayoría de los equipos que se ocuparían.

BIBLIOGRAFIA

LIBROS

[1]Abraham Sánchez López.: "Bases de Datos Distribuidas", Diplomado en computación, FCC-BUAP, 2000.

[2]Adoración de Miguel, Paloma Martínez, Elena Castro, José Ma. Cavero, dolores Cuadra, Ana Ma. Iglesias, Carlos Nieto, "Diseño de Bases de Datos: Problemas Resueltos", Ra-ma, 2001.

[3]Abraham Silbershatz, Henry F. Korth y S. Sudarshan. "Fundamentos de Bases de Datos". Tercera Edición, Editorial Mc Graw Hill, 1999.

[4]Annette Scott. "Oracle Data Modeling and Relational Database Design", Student Guide. Edición 1.0.

[5]Date. "Introducción a los Sistemas de Bases de Datos", Prentice hall, Séptima. Edición, 2001.

[6]Francisco José Minera, "Php y MySql" 1ª edición MP Ediciones, 2005.

[7]Ivar Jacobson, Grady Booch, James Rumbaugh "El proceso unificado de desarrollo de software", Addison Wesley, 2000.

[8]Jacobo Pavón Puertas, "Navegar en Internet. Creación de un Portal con Php y MySql." 3ª Edición, Alfaomega Grupo Editor, S.A. de C.V., Mexico.

[9]Peralta Verónica, "Taller de Sistemas de Información," http://proa.ei.uvigo.es

Otras referencias:

References: Date, C.J. An Introduction to Database Systems, Vol. 1 Addison-Wesley, 1981. Hawryszkiewycz, I.T.; Database Analysis and Design, SRA Inc; 1984. Kroenke, D.M.; Database Processing: Fundamentals, design, Implementation. SRA Inc.; 1983. "Diseño e Implementación se bases de Datos", Capitulo 10. Diseño de Bases de Datos: formas normales de relaciones.

Sitio oficial de PHP, http://www.php.net

Museo de PHP, http://museum.php.net

Sitio oficial de MySql, http://www.mysql.com/,

http://www.mysql.com/documentation

Sitio oficial UML, http://www.omg.org/uml/