Escuela Superior Politécnica del Litoral
Facultad de Ingeniería en Electricidad y
Computación

TOPICO DE GRADUACION

"Análisis y Diseño de Aplicaciones Orientadas a Objetos"

Previo a la obtención del Título de:

INGENIERO EN COMPUTACION

Presentado por:
Otilia María Alejandro Molina
Luis Gustavo Avellán Roca
Mirian Katiuska Estrada Morales
Christian Roberto Ordóñez Orellana
Daniel Darío Párraga Briones
Hugo Omar Pérez Rivas

GUAYAQUIL - ECUADOR
Diciembre 11, 1997

AGRADECIMIENTO

A Dios por ser la luz que guía nuestro camino.

A nuestros padres por ser las bases que nos han impulado a crecer y superarnos. Porque con amor nos ayudaron a levantarnos ante las derrotas y nos enseñaron a ser humildes frente a nuestros triunfos. Gracias a ellos, por el apoyo incondicional que siempre nos brindan.

A nuestros hermanos por estar junto a nosotros.

A nuestros maestros por la enseñanza impartida.

A nuestros amigos, compañeros y a todas aquellas personas que topamos en el camino y que desinteresadamente nos brindaron su apoyo.

A todos ellos Gracias.

DEDICATORIA

Dedicado a todos aquellos que vendrán después, tras nuestros mismos pasos, y que creen en el Gran Tabú del Lenguaje C y los Mitos de la Tecnología Microsoft.

TRIBUNAL

ING. ARMANDO ALTAMIRANO
Presidente

ING. JOHNNY MACIAS
Director del Tópico

PH.D. ENRIQUE PELAEZ
Miembro Principal

ING. GUIDO CAICEDO
Miembro Principal

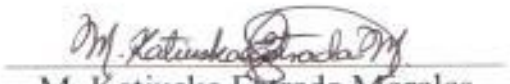
DECLARACION EXPRESA

"La responsabilidad por los hechos, ideas y doctrinas expuestos en esta tesis, nos corresponden exclusivamente; y, el patrimonio intelectual de la misma, a la ESCUELA SUPERIOR POLITECNICA DEL LITORAL".

(Reglamento de Exámenes y Títulos profesionales de la ESPOL)

Otilia Maria Alejandro Molina

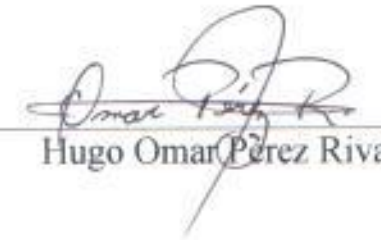
Luis Gustavo Avellán Roca



M. Katiuska Estrada Morales

Daniel Dario Párraga Briones

Christian R. Ordoñez Orellana



Hugo Omar Perez Rivas

RESUMEN

Video Shopping Center es un Sistema Multimedia que permite realizar consultas de temas musicales, de peliculas, de Juegos de Video, etc. Generación automática de facturas, y la compra o alquiler de los diferentes productos en el inventario.

Video Shopping Center hace que nuestros clientes vean los productos existentes dentro del local, con el atractivo de escucharlos y verlos mientras realizan la búsqueda. Es un método interesante e innovador de marketing, ya que así como tenemos clientes que van a una tienda a comprar algo específico, también existen clientes que no saben qué comprar. Básicamente este sistema permitirá que las personas puedan ver lo bueno y lo malo de los videos, oir los distintos estilos de música así como determinar que juegos son los mejores tanto para los niños, como para sus padres.

Video Shopping Center tiene una interfaz de fácil y rápida comprensión, típica del software multimedia; presenta una interface, botones grandes, sonidos, música, textos hablados, ayudas de dos tipos, etc. La cual brinda una interacción muy amigable así como cálida para que los usuarios se sientan como en casa.

Video Shopping Center es capaz de almacenar datos de inventario de una empresa que venda discos Compacto de Música, Peliculas, Videos y Juegos.

Almacena también fotos escaneadas referentes al producto, y la música que contienen los videos y peliculas, todo esto se almacena en la Base de Datos con referencias a sus respectivos archivos binarios.

Este Sistema permite búsquedas por diferentes criterios: por temas, por tipo de productos o por autor, mostrándose en pantalla todo lo referente al producto, así como un acceso

rápido al contenido del mismo. Como último punto, el sistema permite realizar el pedido mediante la generación de una factura para la venta del producto deseado.

El Sistema está desarrollado completamente en Visual C++, tanto el Front End, así como todo el ambiente de funciones y procedimientos.

El Front End se comunica con toda la aplicación (funciones y procedimientos) pudiendo accesar a la base de datos utilizando la interface ODBC de Windows, la cual utiliza el Driver Manager para administrar cada uno de los drivers utilizados por las Bases de Datos.

El Servidor de Base de Datos que utilizamos como Back End para este proyecto es SQL Server, aunque podría también ser ORACLE, o cualquier sistema de Base de Datos abierta, porque la conectividad a bases de datos por medio de ODBC hace independiente la aplicación Cliente de la Base de Datos.

Es un sistema de fácil manejo que tiene un bajo costo y que le generará altos ingresos. Le invitamos a probar esta interesante aplicación que le abrirá las puertas al mundo MULTIMEDIA!

INDICE GENERAL

INTRODUCCION

Existen en el mercado tanto local como internacional muchos Sistemas para manejo de Base de Datos (RDBMS), que nos permiten manejar la información que almacenan de una manera eficiente y muy segura. Estos RDBMS manejan diferentes tipos datos, ya sean numéricos, texto y en formato fecha. Pero son muy contados los RDBMS que manejan tipos de datos que almacenen objetos como parte de la información de la Base; un video, una foto, una pieza musical o multimedia en general, no son objetos que fácilmente se pueden almacenar en la Base de Datos.

Un Sistema que lleve el control de las ventas en una empresa y que además controle el inventario en una bodega, hasta hoy en día trabaja con Bases de Datos puramente transaccionales donde la Interacción de sonido, imagen y video nunca fue tomada en cuenta, llegando a tener presentaciones rígidas dirigidas al usuario final en un ambiente de sólo texto.

En la actualidad existen Bases de Datos así como Herramientas que nos permiten interactuar con sonido e imagen, pudiéndose lograr Sistemas que mezclan ambos elementos y que nos presentan información mucho más detallada.

La Multimedia hoy en día es una de las formas más fácil de presentar cualquier tipo de información. Por medio del video y el sonido podemos llegar a tener presentaciones de muy alta calidad, con imágenes reales de personas o cosas, videos, animaciones de objetos, simulaciones, etc., El sonido es un complemento que no puede faltar dentro de este paquete, nos permite escuchar cosas reales, voces, ruidos, música, etc.; puede hacer que una aplicación aburrida, se torne agradable a la vista y al oído.

En el mercado, al menos localmente, no existen sistemas con características multimedias. Sistemas que presenten información de una manera gráfica, muy amigable, en la cual se presenten videos que nos pueden explicar tanto el funcionamiento del mismo como una lección de matemáticas; sonidos de la naturaleza, voces que le dicen a la persona que hacer en un momento determinado.

Un sistema multimedia para venta de música, ya sea en CD's, o cualquier otro tipo de presentación. Un sistema multimedia donde se puedan ver los últimos videos musicales que estén en el mercado. Un sistema multimedia donde presenten los trailers de las peliculas más taquilleras, con sonidos en estéreo y con imágenes de alta calidad gráfica. Un sistema multimedia donde los juegos de video se los presente como imágenes reales en mundo de realidad virtual. Un sistema multimedia que le permita al cliente escoger, de una manera muy sencilla y sin ayuda de nadie, su video musical preferido, ver y escuchar en vivo a su grupo musical favorito. Un sistema multimedia donde se pueda accesar de una manera muy rápida y fácil a ver las peliculas más taquilleras de la semana y tener información al dia de su actores favoritos. Un sistema multimedia donde el cliente pueda realizar la compra del CD de música preferido, sin necesidad de asistencia de algún empleado de la casa disquera. Un sistema multimedia de esta naturaleza no existe en el mercado local.

Aqui en el país, en la ciudad de Quito, vimos en un almacén de venta de CD's de música, un equipo especial para escuchar CD's, el cual el cliente lo podia manipular solo, sin ayuda de nadie, pero tenia sus limitantes. Se trataba de un equipo electrónico con capacidad para almacenar hasta 10 CD's, y 20 canciones. El equipo nos permitia elegir entre las 20 canciones, tenia unas carpetas en el frente donde se encontraban los nombres de las canciones y el artista

correspondiente. Tenía un control para el volúmen, otro para el tono y para poder oír se necesitaba ponerse unos audifonos que el mismo equipo incluía.

DESCRIPCION DEL PROBLEMA

La falta de información completa y actualizada en los locales comerciales y la falta de un sistema de información adecuado a la tecnología actual generan pérdida de tiempo tanto para el cliente como para el almacén, el servicio es deplorable, atención ineficiente y el cliente la mayoría de las veces no sale satisfecho del almacén porque no encuentra lo que el quiere.

OBJETIVOS

- Crear una aplicación Multimedia Comercial Cliente-Servidor que le permita a un cliente accesar a una Base de Datos y obtener información completa tanto textual como multimedia.

- Aplicar los conocimientos adquiridos en el Tópico I y II desarrollando una aplicación orientada a objeto utilizando como herramienta de programación el lenguaje Visual C++.

DEFINICION DE LA EMPRESA

La empresa a la cual está orientado este sistema es una empresa puramente comercial, que le interese vender sus productos mediante un servicio personalizado y de autoservicio para cada cliente.

La empresa en si es un almacén de venta de música, películas y juegos de video.

Posee una red de computadoras, donde tiene interconectadas la máquina del servidor con la de la caja y las pantallas touch screen clientes.

La empresa está organizada de tal manera que el cliente no tiene un contacto directo con el vendedor, sino que es el mismo cliente el que se preocupa de buscar lo que él necesita mediante el uso del sistema, el cual estará distribuido dentro del local de tal manera que esté al alcance de cualquiera en el momento que lo necesite.

El Cliente buscará en el sistema su musical favorito o su película predilecta y tendrá la opción de hacer su pedido. Luego de esto puede acercarse a la caja a cancelar su pedido y llevarse sus productos.

CAPITULO 1

GENERALIDADES

CAPITULO 1.

GENERALIDADES

1.1 ANALISIS Y DISEÑO ORIENTADO A OBJETOS

El analista orientado a objetos ve el mundo como objetos (con estructuras de datos y métodos) y eventos que activan operaciones, las cuales modifican el estado de los objetos. Las operaciones aparecen como objetos que hacen solicitudes a otros objetos. El analista crea diagramas de la estructura de los objetos y de los eventos que los modifican. El modelo del diseñador es similar al modelo del analista, pero se toma con el detalle suficiente como para crear el código.

El análisis y diseño orientado a objetos intenta lograr la reutilización masiva de las clases de objetos.

El análisis y diseño orientado a objetos modela el mundo en términos de objetos que tienen propiedades y comportamiento, y eventos que activan operaciones que modifican el estado de los objetos. Los objetos interactúan de manera formal con otros objetos.

En el análisis orientado a objetos se obtiene el modelo de la empresa y el análisis de las áreas de la empresa. Se lo realiza tomando en cuenta dos enfóques, la estructura y el comportamiento de los objetos. En el análisis de la estructura de objetos normalmente se realiza un diagrama llamado esquema de objetos. En el análisis del comportamiento de objetos se debe ilustrar el diagrama de flujo de objetos y el esquema de eventos.

El diseño orientado a objetos también tiene dos enfóques, el diseño de la estructura de objetos donde se identifican las clases, herencias y estructuras de datos, y el diseño del

comportamiento de objetos donde se identifica la operación y los métodos. El diseño orientado a objetos abarca en si todo el diseño del sistema.

1.2 PROGRAMACION ORIENTADA A OBJETOS

La programación orientada a objetos (POO) fue desarrollada, esencialmente, por las limitaciones que otras técnicas de programación tenían y tienen, como es el caso de la programación estructurada.

La programación estructurada lo que hacía era romper un programa en unidades más pequeñas, denominadas funciones, procedimientos o subrutinas. Cada una de las funciones tiene un propósito claramente definido, así como una interfaz a las otras funciones del programa.

La POO enfatiza en los datos al contrario de la programación estructurada que enfatiza en los algoritmos. En POO carros, árboles, departamentos, publicaciones, se conocen en general como objetos. Un objeto contiene en una sola entidad y con único nombre las estructuras de datos y las acciones (procedimientos y funciones) que actúan sobre esos datos.

Una clase (objeto en general) es una plantilla o modelo que define los datos y las funciones que actúan sobre esos datos (llamados métodos). Cuando un elemento dato es miembro de una clase, se llama objeto. Se puede asemejar las clases a los tipos de datos definidos por el usuario en un lenguaje de programación tradicional, ya que los tipos definidos por las clases se comportan como los tipos incorporados a un lenguaje de programación. Una clase encapsula (encierra) en si misma datos y métodos (funciones) que manipulan los datos de los objetos de la clase. Si una clase es como un tipo de dato, un objeto es como una variable. Con frecuencia, a los objetos se les llama como instancias, casos, modelos o ejemplos (instancia) de una clase, y también variables instancia. Las funciones llamadas métodos se suelen denominar funciones miembro. Las funciones miembro definidas en una clase se pueden invocar por objetos de esa clase; esta acción se conoce como envío de un mensaje al objeto.

1.3 HERRAMIENTAS CASE

Todo el mundo conoce la historia de los hijos del zapatero, *el zapatero está tan ocupado haciendo zapato para otros que sus hijos andan descalzos*, durante los últimos 20 años muchos ingenieros de software han sido los hijos del zapatero, aunque estos ingenieros han automatizado el trabajo de otros, ellos mismos no han aolicado estas técnicas para su trabajo, es más, hasta hace poco la ingeniería de software era fundamentalmente una actividad manual en la que las herramientas se utilizaban únicamente en las etapas finales.

EL TALLER DE INGENIERIA DE SOFTWARE

Los talleres de software tienen 3 características:

- Un conjunto de herramientas útiles que ayudan en cada paso de la construcción de un producto.
- Un panel organizado que permita encontrar las herramientas rápidamente.
- Una persona cualificada que sabe como utilizar de forma efectiva tales herramientas.

El taller de ingeniería de software se denomina ***entorno de soporte de proyectos integrados*** y el conjunto de herramientas que llena este taller se denomina *CASE*.

Hoy en día las herramientas CASE se añaden a la caja de herramientas de la ingeniería de software. El CASE proporciona al ingeniero la capacidad de automatizar las actividades manuales y de mejorar su enfoque de trabajo. El CASE debe hacer mucho más, debe construir las piezas que construyan un taller para el desarrollo de proyectos de software.

El objetivo más importante del CASE a largo plazo, es conseguir la generación automática de programas desde una especificación a nivel de análisis y diseño. Mucha gente cree que el análisis y el diseño no son suficientes para el CASE sino que deben conducir a la

generación directa del producto final, este hecho hace que el reto sea más difícil, pero el resultado final, de conseguirse, será mucho más potente.

BLOQUES QUE COMPONEN EL CASE

La arquitectura de entorno está compuesta por la plataforma hardware y por el soporte del sistema operativo (incluida la red y la gestión de la base de datos), este bloque constituye la base del CASE, pero el entorno CASE en si mismo necesita de otros componentes. Un conjunto de *servicios de portabilidad* que constituyen un puente entre las herramientas CASE y su *marco de integración* con la *arquitectura de entorno*. El marco de integración es un conjunto de programas especializados que permiten a cada herramienta del CASE comunicarse con las demás, para crear una base de datos de proyectos y mostrar una apariencia homogénea al usuario final (el ingeniero de software), los servicios de portabilidad permiten que las herramientas CASE y su marco de integración puedan migrar a través de diferentes plataformas, hardware y sistemas operativos sin grandes esfuerzos de adaptación.

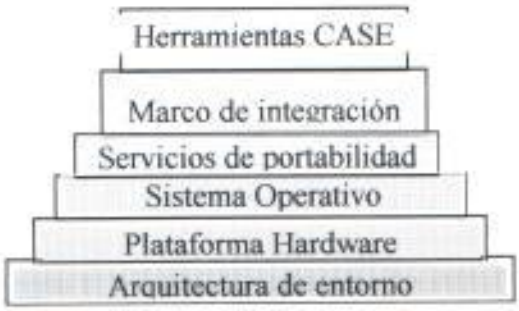


Figura 1.1 Herramientas CASE

En la figura se muestran los niveles de integración del CASE. En el nivel más bajo del espectro de integración está la herramienta individual, cuando las herramientas proporcionan

facilidades para el intercambio de datos (la mayoria lo hacen) el nivel de integración aumenta ligeramente.

Estas herramientas generan una salida en un formato estándar compatible con los de las otras herramientas que puedan leer ese formato.

CLASIFICACION DE LAS HERRAMIENTAS CASE

Las herramientas CASE se pueden clasificar por su función, por su papel como instrumentos para el personal técnico y los directivos, por la arquitectura del entorno, hardware, software que las soporta e incluso por su origen y costo. La que utilizaremos será por su funcionalidad como criterio principal.

Herramientas de planificación de sistemas de gestión.
Herramientas de gestión de proyectos, la que se subdivide en:
- Herramientas de planificación de proyectos.
- Herramientas de seguimiento de requisitos.
- Herramientas de gestión y medida.

Herramientas de soporte, la que se subdivide en:
- Herramientas de documentación.
- Herramientas para software de sistemas.
- Herramientas de control de calidad.

Herramientas de análisis y diseño, que se subdividen en:
- Herramientas de análisis y diseño de estructuras.
- Herramientas PROSIM (creación de prototipos y simulaciones).
- Herramientas para el diseño y desarrollo de interfases (interacción hombre-máquina).
- Máquinas de análisis y diseño.

Herramientas de programación, que se subdividen en:
- Herramientas de codificación convencionales (compiladores,editores y depuradores)
- Herramientas de codificación de 4ª generación (lenguajes de especificación procedimental LEP,
  ejm: sistemas de consulta a bases de datos).
- Herramientas de programación orientada a los objetos (C++, Eiffel, Objective C)

Herramientas de integración y pruebas.
- Herramientas de análisis estático.
- Herramientas de análisis dinámico.
- Herramientas de gestión de pruebas.

Herramientas de creación de prototipos.
Herramientas de mantenimiento de software.
Herramientas de estructura.

1.4 MULTIMEDIA

Multimedia es cualquier combinación de texto, arte gráfico, sonido, animación y video que nos llega a través de una computadora u otros medios electrónicos. Cuando se permite que el usuario final -el observador de un proyecto de multimeedia- pueda controlar ciertos elementos y cuando deben presentarse se le llama ***multimedia interactiva***; cuando se proporciona una serie de elementos ligados a través de los cuales el usuario puede navegar, entonces multimedia interactiva se convierte en ***Hipermedia***.

ARQUITECTURA MULTIMEDIA DE WINDOWS

Windows cuenta con una interface de control de medios (*Media Control Interface,**MCI***) la cual se encarga de recibir un método de software unificado, manejado por órdenes para comunicarse con dispositivos periféricos de multimedia. Utilizando los controladores apropiados (normalmente suministrados por el fabricante del dispositivo), puede manejar el dispositivo con cadenas simples de órdenes o códigos enviados al MCI.

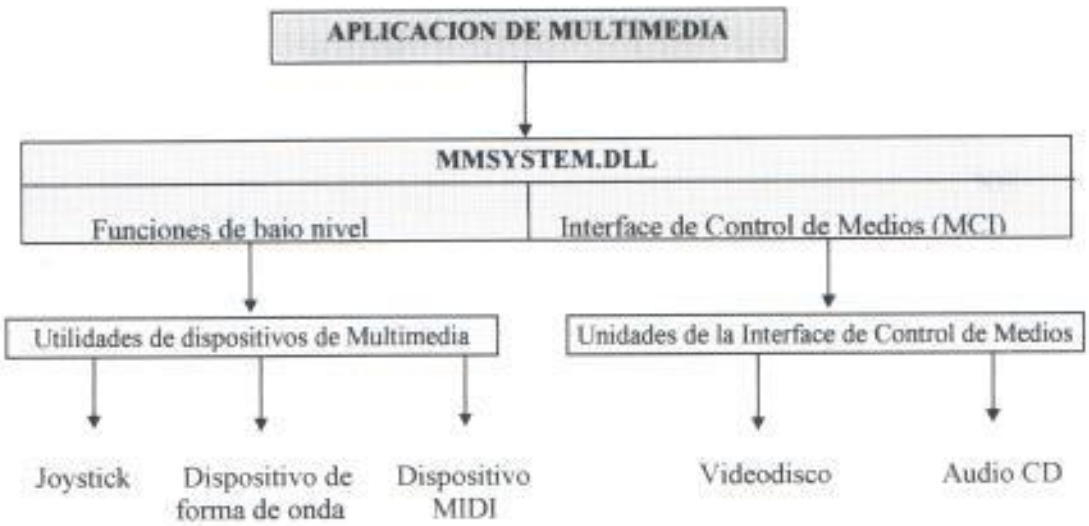


Figura 1.2 Aplicación Multimedia

Los dispositivos multimedia y controladores son manejados por el archivo SYSTEM.INI de Windows, en las secciones [mci] y [drivers] de ese archivo. Leyendo el archivo de texto SYSTEM.INI al empezar. Windows sabe cuales dispositivos de multimedia están presentes en su sistema; esta información es crítica. Cuando se instalan programas de multimedia en Windows, el programa de instalación escribe los renglones apropiados de datos en el archivo SYSTEM.INI. Las entradas tipicas de multimedia en el archivo SYSTEM.INI pueden verse como estas:

```
[mci]
CDAudio=mcicda.drv  ;para reproducir CD-Audio
AVIVideo=mciavi.drv  ;para Audio Video Interfoliado
```

ANIMACION, VIDEO Y PELICULAS DIGITALES

Las animaciones y las peliculas de video digital son secuencia de escenas de gráficos de mapas de bits (cuadros) reproducidas con gran rapidez. Pero las animaciones pueden hacerse también con el sistema de desarrollo cambiando rápidamente la localización de *objetos o duendes (sprites)* para generar apariencia de movimiento. La mayoría de las herramientas de desarrollo adoptan un enfoque por cuadro o una orientación a objetos para la animación, pero rara vez ambos.

Las herramientas para hacer cine aprovechan las tecnologías de QuickTime (Macintosh) y Microsoft Video para Windows (Audio Video Interleaved, AVI), y le permiten crear, editar y presentar segmentos de video digitalizado en movimiento, en general en una pequeña ventana en su proyecto. Para hacer cine a partir de video necesita equipo especial para convertir la señal de

video analógica en datos digitales. Las herramientas para hacer cine, como Premiere le permiten editar y ensamblar secuencias de video capturadas desde la cámara, cinta, otros segmentos de cine digitalizados, animaciones, imágenes digitalizadas y de audio digitalizado o archivos MIDI. La secuencia terminada, que a menudo incluye transiciones y efectos especiales, puede entonces reproducirse -ya sea en forma independiente o en una ventana dentro de su proyecto.

1.5 ARQUITECTURA CLIENTE-SERVIDOR

A pesar de ser una de las tecnologías más utilizadas actualmente en la industria, no se ha llegado a un concenso en el real significado de estos términos. Usando nuestras propias palabras podríamos dar la siguiente definición: ***clientes y servidores son entidades lógicas separadas que trabajan en conjunto a través de una red para poder llevar a cabo una tarea***.

Habria que establecer las verdaderas diferencias entre cliente/servidor y tantas otras tecnologías de software distribuidas. Todos los sistemas cliente/servidor tienen las siguientes características:

- **Servicio:** cliente/servidor es principalmente una relación entre procesos corriendo en máquinas diferentes. El proceso servidor es un proveedor de servicios. El cliente es un consumidor de dichos servicios. En esencia cliente/servidor provee una separación de funciones basándose en la idea de servicio.
- **Recursos compartidos:** un servidor puede servir al mismo tiempo a muchos clientes y además regular su acceso a los recursos compartidos.
- **Protocolos asimétricos:** existe una relación muchos-a-uno entre clientes y servidores. Los clientes siempre inician el diálogo cuando requieren un servicio, mientras que los servidores esperan pacientemente por la solicitud de algún servicio.
- **Localización transparente:** el servidor es un proceso que puede residir ya sea en la misma máquina que el cliente o en otra máquina a través de una red. El software cliente/servidor usualmente encapsula la localización del servidor a los clientes mediante el

redireccionamiento de las llamadas de servicio cuando se lo necesita. Un programa puede ser cliente, servidor o ambos.

- **Mezcla y apareo:** el software cliente/servidor ideal es independiente del hardware o de las plataformas del software del sistema operativo. Uno deberia ser capaz de mezclar y aparear al cliente y las plataformas del servidor.
- **Intercambios basados en mensajes:** clientes y servidores son sistemas acoplados de manera no rigida los cuales interactúan a través de un mecanismo pasa-mensajes. El mensaje es el mecanismo de entrega de solicitudes de servicio y de respuestas a estas solicitudes.
- **Encapsulación de servicios:** el servidor es un "especialista". El mensaje le dice al servidor que servicio es requerido; luego este es "subido" al servidor para determinar como realizar el trabajo. Los servidores pueden ser actualizados sin afectar a los clientes a menos que la interface de mensajes sea cambiada.
- **Escalabilidad:** los sistemas cliente/servidor pueden escalar ya sea horizontal o verticalmente. Horizontal se refiere a añadir o remover estaciones de trabajo clientes con un leve impacto en el rendimiento. Vertical se refiere a migrar a una máquina servidora más grande o más rápida o hacia multiservidores.
- **Integridad:** los datos y el código servidor se mantienen centralmente, lo que significa un mantenimiento más barato y una integridad de datos compartidos más confiable. Al mismo tiempo los clientes permanecen personales e independientes.

¿CUAL ES LA VERDADERA TECNOLOGIA CLIENTE/SERVIDOR?

La idea de repartir una aplicación a través de las líneas cliente/servidor ha sido usada en los últimos diez años para crear varias formas de soluciones de software para redes de área local. Estas soluciones típicamente se venden como pequeños paquetes de software y muchos de ellos por más de un vendedor.

Sin embargo, cada una de estas soluciones se distingue por la naturaleza del servicio que proporcionan a sus clientes.

Existen servidores de Archivos, Bases de Datos, Groupwares, de Objetos; haremos una breve explicación de las tecnologías actualmente vigentes y las más relevantes para el proyecto:

SERVIDORES DE ARCHIVOS

Con un servidor de archivos, el cliente (típicamente un PC) envía requerimientos de registros de archivos al servidor, a través de una red. Este una forma muy primitiva de servicio de datos que requiere de muchos intercambios de mensajes en la red para poder hallar el dato requerido. Los servidores de archivos son útiles para compartir archivos en la red. Son indispensables para crear repositorios compartidos de documentos, imágenes, dibujos de ingeniería y otros grandes objetos de datos.

Figura 1.3 Servidor de Archivos

SERVIDORES DE BASE DE DATOS

Cuando se tiene un servidor de base de datos, el cliente envía requerimientos SQL como mensajes al servidor. Los resultados de cada comando SQL son devueltos en la red. El código que procesa el requerimiento SQL y los datos residen en la misma máquina. El servidor usa su propio poder de procesamiento para hallar los datos que son requeridos en lugar de enviar todos

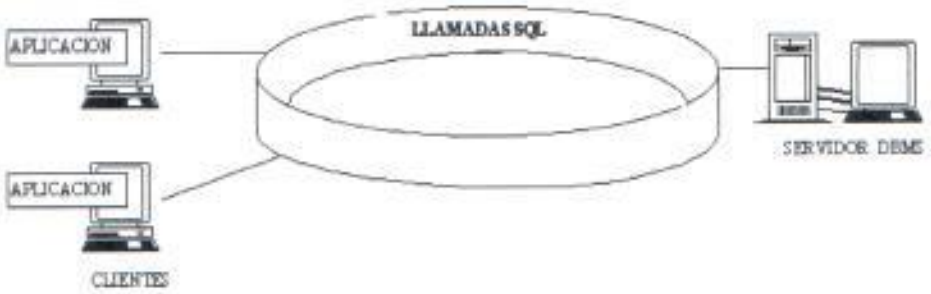


Figura 1.4 Servidor de Base de Datos

los registros al cliente para que el se encargue de hallar sus propios datos (*como es el caso del servidor de archivos*). El resultado es una utilización mucho más eficiente del poder de procesamiento distribuido a pesar de que a menudo se necesita codificar un poco más para la aplicación cliente. Los servidores de base de datos proveen el soporte para sistemas de toma de decisiones que requieren consultas dinámicas y reportes flexibles. Podrían diferenciarse de los servidores de transacciones por que para este tipo de servidores se debe crear la aplicación Cliente / Servidor escribiendo el código para ambos componentes, los del Cliente y los del Servidor, mientras que en Bases de Datos una de las aplicaciones ya se encuentran creadas (O.D.B.C.).

SERVIDOR DE OBJETOS

Con un servidor de objetos, se escribe la aplicación cliente/servidor como un conjunto de objetos que se comunican entre sí. Los objetos cliente se comunican con los objetos del servidor usando un Corredor de Requerimientos de Objetos (***ORB***). El cliente invoca un método el cual es soportado por una clase del servidor del objeto. El ORB ubica una instancia de esa clase del servidor del objeto, invoca el método requerido y retorna el resultado al objeto cliente. Los objetos del servidor deben ser capaces de proveer soporte de concurrencia y de compartición. El ORB los trae todos juntos.

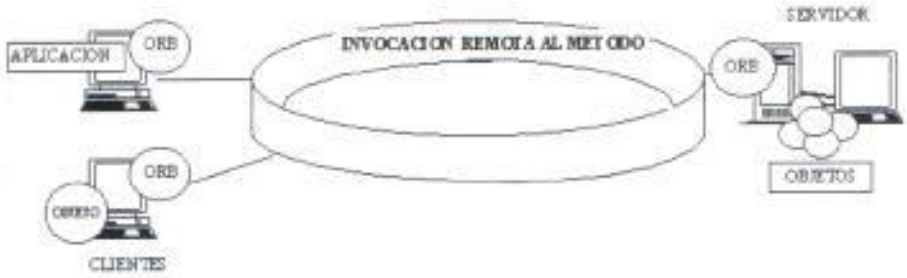


Figura 1.5 Servidor de Objetos

1.6 BASES DE DATOS

La disponibilidad de almacenamiento secundario de acceso directo y gran capacidad a bajo costo ha provocado una tremenda actividad de investigación y desarrollo en el área de los sistemas de bases de datos. Una base de datos es un conjunto integrado de datos controlado centralmente; un sistema de base de datos incluye:

- Los datos mismos.
- El equipo en el cual residen los datos.
- Los programas (llamados sistema de administración de Base de Datos o DBMS) que controlan el almacenamiento y obtención de los datos.
- Los usuarios del sistema.

VENTAJAS

Dentro de las ventajas más importantes se encuentran las siguientes:

- Disminución de la redundancia.
- Eliminación de inconsistencias.
- Compartición de los datos.
- Imposición de normas.
- Restricciones de seguridad.
- Mantenimiento de integridad.
- Equilibrio de requerimientos en conflicto.

En los sistemas convencionales, no de bases de datos, cada aplicación individual mantiene sus propios archivos, a menudo con bastante redundancia y con diversos formatos físicos. En los sistemas de bases de datos se reduce la redundancia mediante la integración de los archivos individuales. La posibilidad de compartir datos es uno de los beneficios más importantes de los sistemas de bases de datos, las aplicaciones ya existentes pueden hacer referencia a los mismos datos mientras que las aplicaciones nuevas pueden hacer referencia a los datos ya existentes.

El control centralizado hace posible lograr un cumplimiento rígido de las normas, lo que tiene especial importancia en las redes de computadores en las que hay *migración de datos* de un sistema a otro.

La seguridad es un aspecto curioso en los sistemas de bases de datos: es posible que los datos estén en mayor peligro al reunirse y conservarse en una localidad central en vez de distribuirse en archivos separados físicamente en muchas localidades; para contrarrestar esto, es preciso diseñar los sistemas de bases de datos con controles complejos.

INDEPENDENCIA DE LOS DATOS

En estos sistemas las aplicaciones no necesitan preocuparse de la forma como se almacenan físicamente los datos o se obtiene acceso a ellos; se dice que una aplicación es dependiente de los datos si no es posible modificar la estructura de almacenamiento y la estrategia de acceso sin afectar de manera significativa la aplicación.

Por la independencia de los datos, resulta conveniente que diversas aplicaciones tengan vistas diferentes de los mismos datos. Desde el punto de vista del sistema, la independencia de los datos hace posible modificar la estructura de almacenamiento y la estrategia de acceso en

respuesta a cambios en los requerimientos de instalación, pero sin tener que modificar las aplicaciones en funcionamiento.

LENGUAJES DE BASES DE DATOS

Los usuarios obtienen acceso a una base de datos mediante proposiciones en alguna forma de *lenguaje de bases de datos*. Los programas de aplicaciones pueden emplear un lenguaje convencional de alto nivel orientado hacia los procedimientos como PL-SQL, Pascal, Cobol; el usuario de una terminal puede emplear un lenguaje de consulta de diseño especial (SQL), el cual facilita la expresión de solicitudes en el contexto de una aplicación particular.

Estos lenguajes se los denomina por lo general *lenguajes anfitriones*. Cada lenguaje anfitrión incluye casi siempre un sublenguaje de datos (data sublanguage, DSL) que se ocupa de los aspectos específicos de los objetos y operaciones de las bases de datos. Cada sublenguaje de datos es por lo general una combinación de dos lenguajes: un lenguaje para definición de datos (data definition language, DDL) que ofrece recursos para definir los objetos de la base de datos y un lenguaje para manipulación de datos (data manipulation language, DML) el cual ofrece recursos para especificar el procesamiento por realizar con los objetos de la base de datos.

BASE DE DATOS RELACIONAL

El modelo relacional es una estructura lógica más que física. Los principios de la administración de la base de datos relacional pueden considerarse de manera bastante apropiada sin necesidad de preocuparse por la puesta en práctica física de las estructuras de datos.

Una base de datos relacional se compone de *relaciones*.

Relación: EMPLEADO

Una tupla

NUMERO	*NOMBRE*	*DEPARTAMENTO*	*SALARIO*	*CIUDAD*
23603	DELGADO A.	413	3'700.000	QUITO
24568	ASCENSIO N.	413	3'720.000	QUITO
23003	GAVICA J.	657	4'650.000	LOJA
32567	DE AVILA A.	611	5'478.500	LOJA
15432	CARABALI H.	657	4'700.000	QUITO

Clave primaria

Un dominio

Tabla 1. Base de Datos Relacional

La figura anterior ilustra un ejemplo de relación que podría emplearse en un sistema de control de personal, el nombre de la relación es EMPLEADO y su propósito primordial es demostrar las interrelaciones de diversos atributos de cada empleado con ese empleado. Cualquier fila específica de la relación se llama *tupla*, esta relación consta de cinco tuplas. El primer campo de cada tupla, el número de empleado, se usa como clave primaria para hacer referencia a los datos de la relación. Las tuplas de una relación están ordenadas por clave primaria.

Cada columna de la relación representa un *dominio* diferente. Las tuplas deben ser únicas por (clave primaria) dentro de una relación, pero los valores específicos de un dominio se

pueden repetir entre tuplas, por ejemplo dos tuplas distintas en el ejemplo contienen el número de departamento 413. El número de dominios en una relación indica el *grado de la relación*. Las relaciones de grado 2 son relaciones binarias, las relaciones de grado 3 son relaciones ternarias y las de grado n son relaciones n-arias.

Diferentes usuarios de una base de datos estarán interesados en elementos de información diferentes y diferentes interrrelaciones entre los datos. Algunos usuarios requerirán sólo ciertos subconjuntos de las columnas de las tablas, otros requerirán tablas más pequeñas para formar unas más grandes y así producir relaciones más complejas; a la operación de subconjuntos se la conoce como *proyección* y a la de combinación como *reunión (join)*.

- La organización relacional de las base de datos presenta las siguientes ventajas:
- La representación tabular empleada en el esquema relacional es entendida con facilidad por los usuarios, y también es fácil de llevar a la práctica en el sistema físico de bases de datos.
- Es relativamente fácil convertir cualquier otro tipo de estructura de bases de datos al esquema relacional. Así pues el esquema puede considerarse como una forma universal de representación.
- Las operaciones de proyección y reunión son fáciles de llevar a la práctica y facilitan la creación de nuevas relaciones requeridas para aplicaciones específicas.
- La puesta en práctica del control de acceso a datos delicados es sencilla. Basta con colocar estos datos en relaciones separadas y controlar el acceso a estas relaciones mediante algún tipo de esquema de autorización o acceso.
- Las búsquedas pueden ser más rápidas que en esquemas en los cuales se siguen cadenas de apuntadores.

- La claridad y visibilidad de los datos mejora con la estructura relacional. Es mucho más fácil examinar datos tabulares que desenmarañar interconecciones de complejidad posiblemente arbitraria entre elementos de información con un mecanismo de apuntadores.

EL MODELO ERA

El modelo de Análisis Extendido Relacional (Extended Reletional Analysis,*ERA*) explica los conceptos de modelamiento de datos (entidades, relaciones y atributos) envueltos en la produccion de una base de datos totalmente normalizada.

ENTIDADES

Una entidad puede ser cualquier objeto de interés acerca del cual se pueden coleccionar datos. Este objeto se vuelve identificable de los otros objetos. Por ejemplo, personas, lugares y objetos describen a las entidades. Las entidades son modeladas en la base de datos como tablas. Si tuviéramos una tabla "cliente", un cliente que se encuentre en esa tabla seria diferente de todos los otros clientes y podría ser tratado como una sola unidad.

RELACIONES

Entidades separadas pueden estar relacionadas con cada una de las otras. Las relaciones se definen por medio de las llaves primarias y secundarias y su misión es conservar la integridad referencial. Las relaciones pueden ser almacenadas en columnas dentro de la tabla o, dependiendo del tipo de relación , en sus propias tablas.

RELACIONES	*MODELADAS COMO:*	*EJEMPLOS*
Uno-a-una	columna de llave foránea en cada tabla.	empleado enviando información por correo a empleado que la agrupa.
Uno-a-muchas	columna de llave foránea en la tabla de 'muchos'.	miembro-a-numero de libros
Muchos-a-muchos	tabla de "enlace" separada con columnas de llave foránea hacia ambas tablas.	estuadiantes-a-cursos

Tabla II. Relaciones

ATRIBUTOS

Los atributos son cualidades o identificadores adicionales de una entidad. Los atributos se almacenan como columnas o campos de las tablas. Estos son modelados como columnas dentro de la tabla. Si buscáramos, por ejemplo, que información describe a un cliente (la entidad), los atributos que quisiéramos en la base de datos pueden incluir el nombre del cliente y su número telefónico.

	ENTIDADES	RELACIONES	ATRIBUTOS
Existe como	Tablas	Columnas o Tablas	Columnas (o Campos) de una Tabla
Ejemplos	Información del cliente	Clientes a Facturas	Nombre del cliente, dirección, número telef.

Tabla III. Atributos

1.7 ODBC

La mayoria de Aplicaciones Windows de Microsoft ofrecen una propiedad y facilidad para la conectividad con Bases de Datos Abiertas, como ejemplos de estas aplicaciones tenemos a Microsoft Excel, Microsoft Visual Basic, Microsoft Visual C++ , Microsoft Report Writer y muchas más. Esta herramienta que provee Microsoft se la conoce como O.D.B.C., Las Aplicaciones Windows generalmente necesitan de los datos almacenados en una base de datos, para hacer diferentes presentaciones de estos datos en diferentes formatos. La interface Microsoft® Open Database Connectivity (**ODBC**), es una interface programada en lenguaje C que sirve para implementar la conectividad con Bases de Datos. Para poder acceder a los datos en la D.B. desde la Aplicación Windows es necesario instalar y configurar el Administrador O.D.B.C. El Administrador O.D.B.C. es un objeto con varias DLL (Librería de Enlace Dinámico) que se lo encuentra en el Panel de Control de Windows 3.1 o Windows 95, existe el controlador O.D.B.C. para 16 y 32 bits. El Administrador de O.D.B.C. es el encargado de las funciones de interface entre la Aplicación Cliente Windows y el Driver controlador de O.D.B.C. para el RDBMS. El Administrador de O.D.B.C. puede manejar a la vez varios Sistemas Administradores de Bases de Datos, por Ejemplo: driver O.D.B.C. para SQL Server 6.5 y/o a la vez el driver de O.D.B.C. para Bases de Datos Access 2.0 y/o el driver O.D.B.C. para RDBMS de Oracle ORACLE Server 7.0, etc.

TEORIA DE OPERACION

La interface ODBC permite que las aplicaciones accesen a los datos almacenados en sistemas administradores de bases de datos (**DBMS**) mediante el uso de Lenguaje de Consultas Estructurado (**SQL**) como estándar para el acceso a los datos. Lo que se logra con esto es lo que

se conoce como *interoperabilidad* , ya que una sola aplicación puede accesar diferentes DBMS's, lo que permite al desarrollador de la aplicación desarrollar, compilar y ejecutar la aplicación sin necesidad de apuntar a un DBMS específico. La versatilidad de este esquema es que los usuarios pueden después añadir módulos llamados drivers de base de datos, los mismos que enlazarán la aplicación con el DBMS que requieran.

En el mundo tradicional de bases de datos, las aplicaciones se desarrollaban con lo que se conocía como '*embedded SQL*', lo que conllevaba a la necesidad de readecuar el código y compilar para cada uno de los diferentes ambientes en los que operaban las aplicaciones ej: DB2, Oracle.

ODBC ofrece una nueva perspectiva: proveer un programa separado para extraer la información de la base de datos y luego tener una vía para que las aplicaciones puedan importar los datos. Debido ha que existen y probablemente siempre existirán diferentes métodos de comunicación, protocolos de datos y capacidades de los DBMS, la solución que da ODBC es permitir utilizar diferentes tecnologías mediante el uso de una interface estándar.

Esta solución es la que conlleva a la implementación de los drivers de bases de datos que no son más que librerías de enlace dinámico (**dll**) que pueden ser invocadas por la aplicación cuando se quiera tener acceso a una fuente particular de datos a través de un método particular de comunicaciones. ***ODBC provee la interface estándar que permite tanto a los programadores de aplicaciones como a los proveedores de estas librerías, disparar los datos entre aplicaciones y fuentes de datos.***

COMPONENTES DE ODBC

La arquitectura de ODBC consiste de cuatro componentes:

- Aplicación, es la que lleva a cabo el procesamiento y llama a las funciones ODBC para enviar sentencias SQL y recibir los resultados.
- Driver Manager, carga los drivers necesarios para la aplicación.
- Driver, procesa las llamadas a funciones ODBC, envía los requerimientos SQL a fuentes de datos específicas y devuelve los resultados a la aplicación. De ser necesario, el driver modifica el requerimiento de la aplicación para que este se apegue a la sintáxis que soporta el DBMS asociado.
- Fuente de Datos, consiste en los datos a los que el usuario quiere accesar y su sistema operativo asociado, el DBMS y la plataforma de red (de existir) usada para accesar al DBMS.

La arquitectura aplicación/controlador es la siguiente:

Figura 1.6 Arquitectura ODBC

1.8 DEFINICION DE UNA RED DE COMPUTADORAS

Una red de computadoras está formada básicamente por dos computadoras que se comunican entre sí.

La mayoría de las redes constan de dos o más computadoras. No obstante, los principios de comunicación son los mismos para dos, tres o mil computadoras.

En general, las redes caen en uno de los dos siguientes grupos: redes de área local y redes de área amplia. Una red de área local (LAN, Local Area Network), conecta computadoras cercanas una de la otra. En algunos casos, "local" significa dentro de la misma habitación o edificio, en otros se refiere a computadoras ubicadas a varias millas de distancia. En contraste, las redes de área amplia (WAN, Wide Area Network) constan de computadoras que se encuentran en diferentes ciudades, estados e incluso países.

COMO TRANSFIEREN DATOS LAS COMPUTADORAS

Cuando las computadoras se comunican con otros dispositivos, transfieren los datos en formato paralelo o serial. La mayoría de las PC emplean comunicación paralela para transferir datos a sus impresoras. La comunicación paralela significa que, de modo simultáneo, se transfiere información a través de múltiples líneas o cables. Para enviar un byte (ocho bits) de información a un cable paralelo, la computadora envía al mismo tiempo los ocho bits a ocho cables. La comunicación paralela utiliza líneas conectadas en paralelo. En contraste, la transferencia serial usa un solo cable, a razón de un bit a la vez. Normalmente, las redes utilizan comunicación serial para transferir datos de una computadora a otra. La comunicación serial requiere que los datos de bits se alineen uno detrás del otro.

Las computadoras utilizan diferentes métodos para transferir datos por cables. Los términos más comunes para definir dichos métodos son: simplex, half-duplex y full-duplex. La comunicación simplex, sucede cuando los datos viajan en una sola dirección; la half-duplex permite que los datos viajen en dos direcciones, una a la vez, y la full-duplex permite a los datos viajar simultáneamente en ambas direcciones.

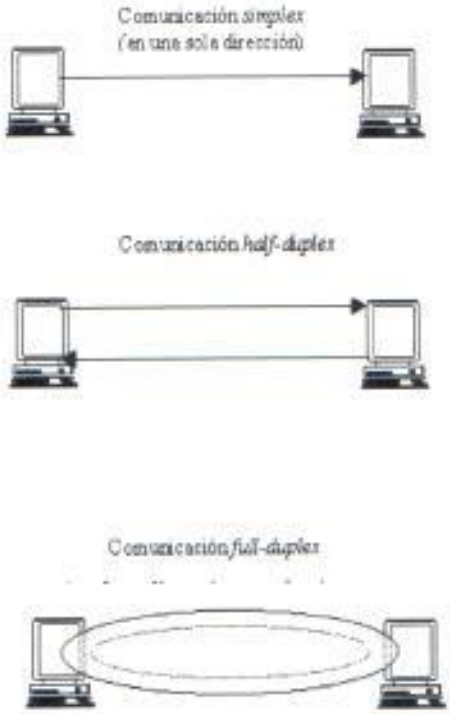


Figura 1.7 Modos de Comunicación

TOPOLOGIA DE REDES

Como se puede imaginar, existe un número ilimitado de formas de conectar computadoras. Cada conexión crea una nueva ruta para que viaje la información. La *topología de red* especifica la forma o arreglo físico de las computadoras y proporciona un método para comparar y clasificar redes. Las tres topologías más comunes son:

TOPOLOGIA EN ESTRELLA

En una topología en estrella todas las computadoras (nodos) se conectan a una computadora central (hub). Aquí no puede existir la comunicación directa entre dos computadoras (a menos que sea la central).

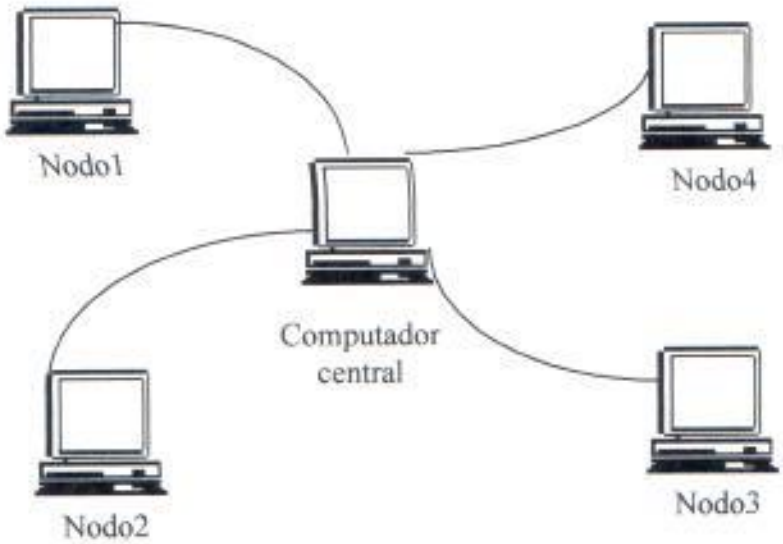


Figura 1.8 Topología Estrella

Los paquetes de datos en una topología en estrella viajan a un computador central que las vuelve a transmitir hacia su destino, que pueden ser cualquiera de las otras computadoras.

La mayor ventaja de esta tecnología es que si llega a haber una interrupción en la comunicación entre cualquier computadora y la central, no afecta a ninguna otra computadora en la red. La mayor desventaja de la topología en estrella es que si la central se descompone, toda la red se vendría abajo.

TOPOLOGIA EN BUS

Una *topología en bus* utiliza un solo medio de transmisión llamado *bus* (cable). Todas las computadoras en una red similar se conectan directamente al bus. Casi siempre el medio de transmisión es un cable coaxial.

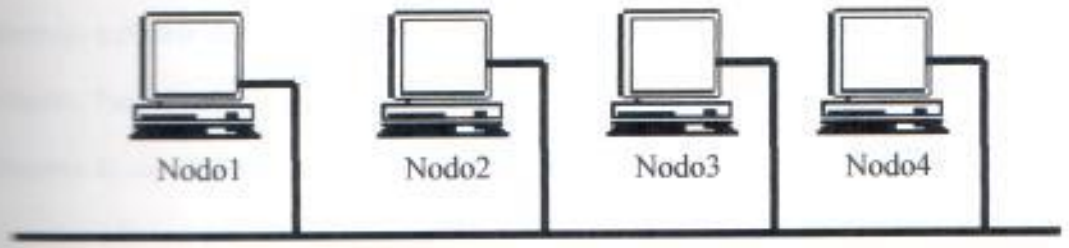


Figura 1.9 Topología Bus

En una topología en bus la información puede viajar en ambas direcciones. Esta topología requiere conexiones terminales (o terminadores) especiales en ambos extremos del bus. Al igual que una red en anillo, una interrupción física en cualquier parte del bus ocasiona una falla en todas las comunicaciones de la red. En las topologías en bus y en anillo la seguridad es débil por las mismas razones: la información pasa por cada una de las computadoras.

TOPOLOGIAS Y TECNOLOGIAS DE RED

Mucha gente utiliza el término *topología* cuando en realidad desea referirse a una tecnología específica. Por ejemplo:

Ethernet es una tecnología desarrollada en 1973 por un equipo de investigadores del Centro de Investigaciones de Xerox en Palo Alto. Las redes Ethernet pueden configurarse en estrella o bus. En general, si se utiliza cable coaxial como medio de transmisión, se configurará la red en bus; si se utiliza cable par trenzado, la Ethernet se configurará en estrella.

ARQUITECTURA DE REDES

Para crear software modular reutilizable, los programadores de aplicaciones emplean diversos métodos de diseño, como la programación estructurada o la programación orientada a objetos. Para alcanzar objetivos similares, los diseñadores de redes usan un método de diseño llamado de *capas*. En pocas palabras, estos diseñadores dan una función específica a cada capa. La capa inferior, por ejemplo, incluye la función de la comunicación básica. Los diseñadores ponen una segunda capa sobre la de comunicación básica que agrega detección de errores. Al asignar una tarea a cada capa se crean redes de capas funcionales que son más fáciles de comprender. El modelo de referencia ISO/OSI representa una red como capas funcionales.

CAPAS

Figura 1.10 Modelo OSI

El modelo ISO/OSI utiliza capas para organizar una red en módulos funcionales bien definidos. Los diseñadores emplean las descripciones de las capas del modelo para construir redes reales pero, de acuerdo al propósito de la red, el diseñador puede modificar el número, nombre y función de las capas. Como resultado, una red construida a partir del modelo ISO/OSI puede tener variaciones significativas comparada con él o con otras redes construidas con base en él.

En una red de capas, cada módulo o capa proporciona funcionalidad especifica o servicios a sus capas adyacentes. Además, cada capa protege a las capas uperiores de los detalles de la implementación de las inferiores. Dicho de otra forma, cada capa se preocupa por su interface con la siguiente capa, es importante anotar esta característica de

Modelo OSI porque permitirá que el proyecto transfiera datos a nivel de Red utilizando el Protocolo de Transmisión así como su capa de Aplicación.

1.9. VISUAL C++

Microsoft Visual C++™ representa un paso revolucionario en el progreso de herramientas de desarrollo de aplicaciones de alto-rendimiento para sistemas operativos Microsoft Windows®. Todo esto es posible gracias a dos características importantes: el uso de herramientas de diseño totalmente integradas con Windows y la adaptación del popular paradigma de la interface-orientada-al-usuario al proceso tradicional de desarrollo C/C++.

Microsoft Visual Workbench es la piedra angular de la plataforma desarrollo Visual C++, el cual es un poderoso ambiente de desarrollo en sí mismo, contiene varias herramientas integradas, incluyendo un editor, debugger y un browser gráfico. Pero más aún, ya que Microsoft Visual Workbench también se ajusta dentro de la vasta estrategia de desarrollo actuando como el punto central desde el cual se llevan a cabo todas las otras actividades de las herramientas. Desde de los menues de Microsoft Visual Workbench se puede:

- Invocar las herramientas de construcción de Visual C++, tales como el compilador y el enlazador.
- Ejecutar el editor de recursos de App Studio para desarrollar los componentes de la interface de usuario.
- Ejecutar los wizards de AppWizard y ClassWizard para obtener ayuda en el desarrollo de aplicaciones de la librería de Microsoft Foundation Class.
- Ejecutar sus propias herramientas, las cuales pueden ser instaladas en el menú herramientas.

¿QUE ES UNA APLICACION VISUAL C++?

Una aplicación Visual C++ es una aplicación para Windows que se diseña y desarrolla usando la Librería de Microsoft Foundation Class, las herramientas de construcción de Microsoft Visual C++, el Visual Workbeanch y las herramientas App Studio basadas en ventanas.

Usando un ambiente totalmente integrado, se logra un aproximación a la programación de la aplicación de la misma manera que el usuario lo harían utilizando el programa-desde los elementos de las interfaces visuales. Visual C++ llama a estos elementos *'objetos de interface-de-usuario'*. Primero se diseñan los objetos de interface-de-usuario y luego se utiliza las herramientas de Visual C++ para crear y manejar el código que las soportará. Las herramientas de Visual C++ ayudan a automatizar el proceso tedioso y con riesgo de errores de creación de clases derivadas, miembros de funciones y del mapeo de estos a los mensajes. Esta automatización permite que los programadores se concentren en el diseño de los recursos para la aplicación y en la escritura del código funcional que manejará los mensajes.

También se pueden usar las herramientas Visual C++ para desarrollar aplicaciones de Windows SDK estándar in C o C++, debido a que Visual C++ incluye un editor de texto, manejador de proyectos, utilidad de construcción, browser, debugger y editor de recursos.

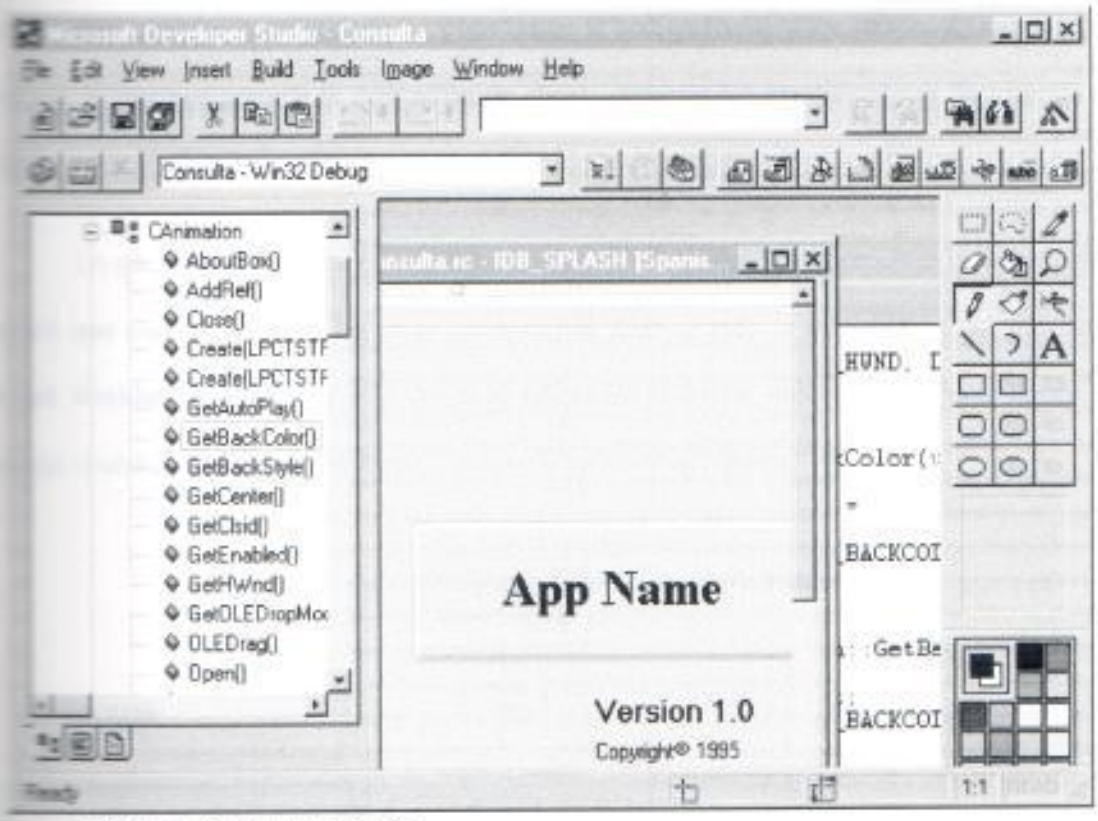


Figura 1.11 **Microsoft Developer Studio**

EL PROCESO ENTERO

Supongamos que se ha desarrollado una aplicación para Windows antes de hacer una para Visual C++, se podria ver el proceso de desarrollo estándar moviendo mentalmente el AppWizard y el ClassWizard que se incluyen en la figura -en la cual el Visual Workbeanch representa el editor fuente y el App Studio representa el editor de recursos que se pudo haber usado para crear código fuente y archivos de recursos. Este modelo de desarrollo ha sido ampliamente utilizado y aún es soportado por Visual C++.

El AppWizard simplemente nos ayuda en la creación de archivos de arranque personalizados que residen donde prefiera el desarrollador. ClassWizard añade otra dimensión a

este modelo estándar, manteniéndo la "pista" entre el código fuente y los objetos de interface de usuario, lo que nos permite la derivación de clases, conectar los identificadores de recursos al código y editar el código fuente desde un simple punto de ventaja.

Lo que no se sobresalta en este diagrama es que el Visual Workbeanch es la herramienta central que coordina a todas las otras herramientas manteniendo la información del proyecto. Visual Workbeanch no es tan solo donde se editan los archivos, también es el lugar donde se maneja el código fuente, se construye y se pone a punto la aplicación.

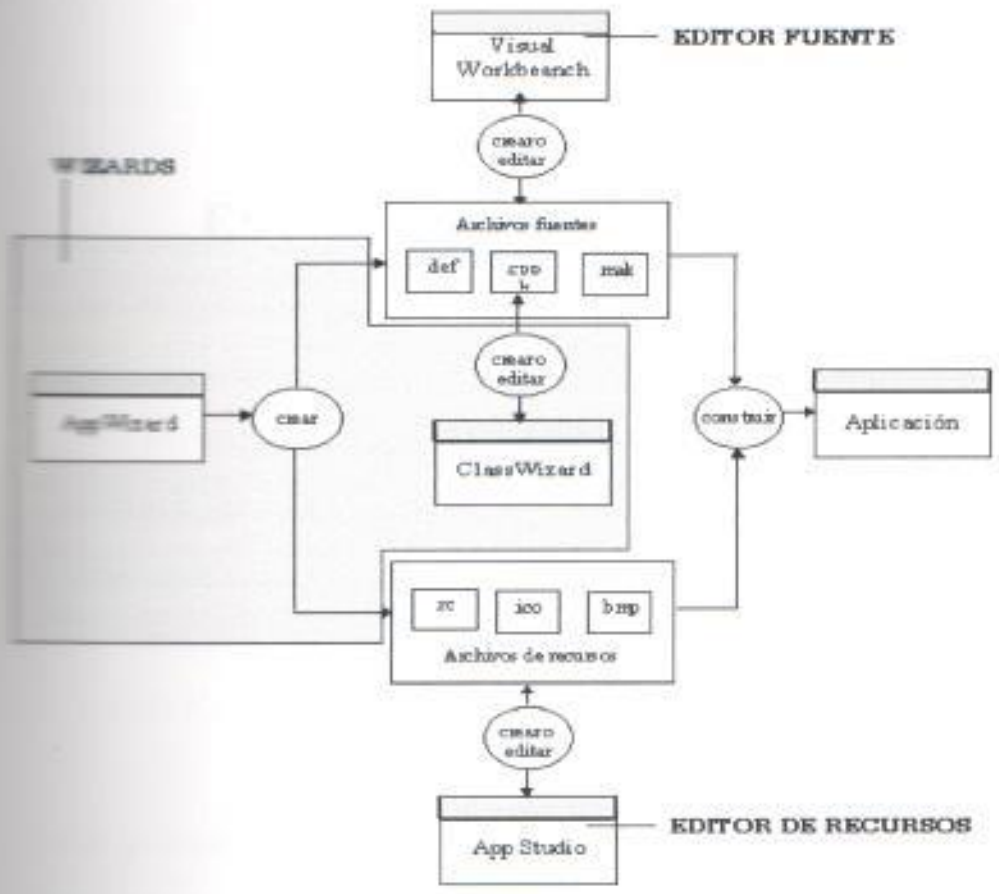


Figura 1.12 Esquema funcional del Developer Strudio

CAPITULO 2

ESTRATEGIA DE SOLUCION

CAPITULO 2

ESTRATEGIA DE SOLUCION

Este capítulo contiene el enfoque que se utilizó para desarrollar la estrategia de solución, también las razones del porqué se escogieron las herramientas utilizadas y de qué manera fueron empleadas para lograr nuestro objetivo.

Nuestro desafío era que el Kiosko Multimedio cumpla con las expectativas y objetivos trazados para lo cual fue necesario investigar como satisfacer los requerimientos que el sistema exigía. Entre los requerimientos que el sistema exigía podemos anotar que :

- Debería estar basado en la Arquitectura Cliente - Servidor.
- La aplicación Cliente (Front - End) debería ser analizada y diseñada orientada a objetos y desarrollada en Visual C++.
- Que sea independiente del RDBMS, es decir que la Base de Datos pueda ser cualquiera de las que se encuentran en el Mercado.
- Los recursos o archivos multimedios deberían estar centralizados y relacionados a la Base de Datos, para que desde cualquier Workstation estén disponibles y así no desperdiciar recursos en cada Cliente.
- La Aplicación Cliente o Front - End debería ser capaz de reproducir cualquier recurso o archivo multimedio.

Para una estrategia de solución adecuada que satisfaga los requerimientos antes mencionados se debió contemplar los siguientes puntos:

- Redes. (Topología de Red, Protocolos, Configuración).
- Sistemas Operativos. (S.O. en los Clientes y S.O. en el Servidor.)
- O.D.B.C. (Conectividad a Bases de Batos Abiertas), con lo cual nuestro sistema se convertía en Cliente - Servidor.
- SQL Server 6.5 como elección del RDBMS que se utilizó.
- E-R win, como herramienta CASE para el diseño estructural y referencial de la Base de Datos.
- Herramientas Shareware para la captura, edición y depuración de los archivos multimedios.
- Visual C++ 4.0, como herramienta de desarrollo de la aplicación Cliente.
- O.L.E., como método para la reproducción de los multimedios desde la aplicación Cliente.

En los gráficos se muestra la manera en que interactúan las herramientas utilizadas.

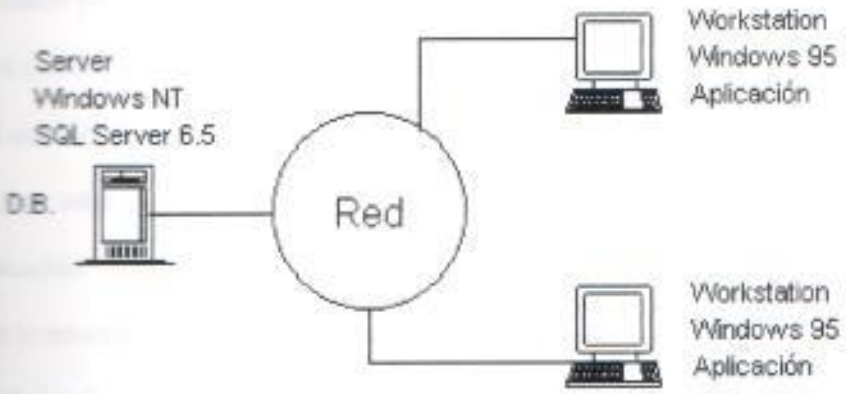


Figura 2.1 La Red

Figura 2.2 ODBC

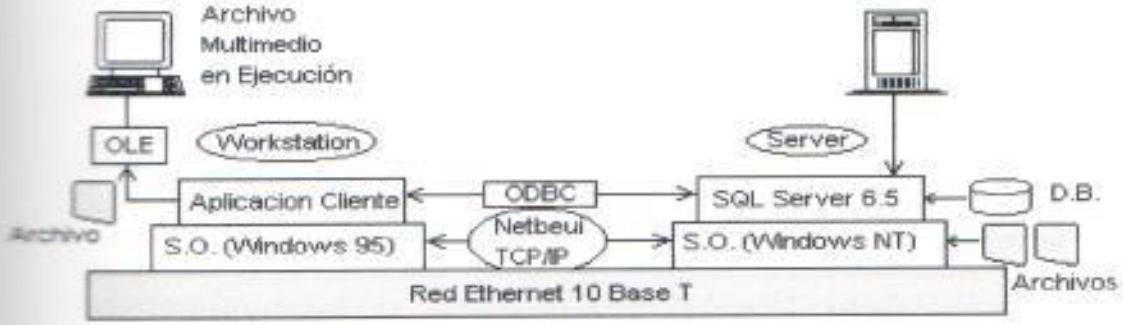


Figura 2.3 Comunicación entre capas

Nuestro proyecto fue desarrollado utilizando varias computadoras, las cuales comparten recursos (archivos, impresoras, dispositivos, etc.) por medio de una red Ethernet 10 Base T.

El servidor de esta red utiliza el S.O. Windows NT y las estaciones clientes Windows 95.

Sobre estas plataformas corren diferentes servicios y aplicaciones que ayudan en la comunicación.

En la estación cliente se ejecuta nuestro sistema, el cual fue implementado en Visual C++ siguiendo los lineamientos de las aplicaciones de Windows. Esta envía sus requerimientos al servidor para hacer consultas a la base de datos, utilizando como interfase de comunicación la tecnología ODBC, la cual nos permite transformar nuestra aplicación en una aplicación

Cliente-Servidor, conectándonos así con una base de datos implementada en SQL Server residente en el servidor.

Además, nuestra aplicación Cliente realiza una llamada a los servicios de red para poder transferir los archivos multimedia almacenados en el servidor hacia nuestra aplicación y ejecutarlos por medio de OLE.

2.1 LA RED

Hay diferentes tipos de Redes: LAN, WAN, MAN, TAN; y diferentes topologías: Bus (Ethernet) o Anillo (Token Ring). El criterio por el cual se decidió qué tipo y topología de Red usar, se debe básicamente a factores que insiden directamente en el rendimiento del Sistema, tales como:

- Espacio Físico y Distancia.- Como este Kiosko Multimedia, es un sistema integrado que está orientado a ser utilizado por una casa disquera para que se puedan hacer consultas multimedios, reportes de inventarios de productos y también de facturación y recaudación, pensamos que el espacio físico no podría ser mayor al de un establecimiento o local comercial, por lo tanto escogimos una Red LAN.
- Tiempo de Respuesta.- Como los recursos o archivos multimedios se encuentran centralizados en el Servidor, al momento de realizar una consulta de algún producto, la aplicación Cliente debe recuperar la referencia al archivo multimedio (referencia que se encuentra almacenada en la base de datos), con la cual ésta puede accesar a los directorios del servidor para transferir los archivos multimedios por medio de la red. Cuando se tratan de operaciones o transacciones en sentencias SQL a la Base de Datos la transferencia en bytes es mínima, dependiendo del estado de la base, pero si se tratan de transferencias de archivos multimedios que oscilan entre 2 y 10 Mbytes, por ejemplo archivo *.avi de tamaño 44Mb. tendrá un tiempo de respuesta de 35 seg en una Red 10 Mbps siempre y cuando sea este usuario el único connectado al servidor. El tiempo de respuesta se ve afectado si el ancho de banda en la Red no es lo suficientemente amplia además de la cantidad de Clientes que aceesen al servidor.

- Experiencia en administración y configuración de Redes.- Otro factor muy importante en la elección del tipo de topología y protocolos de Red fue la experiencia y el tiempo que teníamos trabajando con cada una de ellas.

Por todos los factores antes mencionados elegimos una **Red LAN Ethernet 10 Base T**, siempre y cuando el (cálculo pertinente del) tráfico en la Red no se exceda en la demanda. Esto depende principalmente del número de estaciones que cumplen las funciones de las consultas a los archivos multimedios, según las pruebas hechas por nuestro grupo de desarrollo creemos que más de 4 estaciones trabajando al mismo tiempo podría afectar en el tiempo de respuesta del Sistema, por lo que se recomendaría escoger una Red LAN que ofrezca mayor capacidad de transferencia de Datos.

Esquema de Configuración de La Red LAN Ethernet.

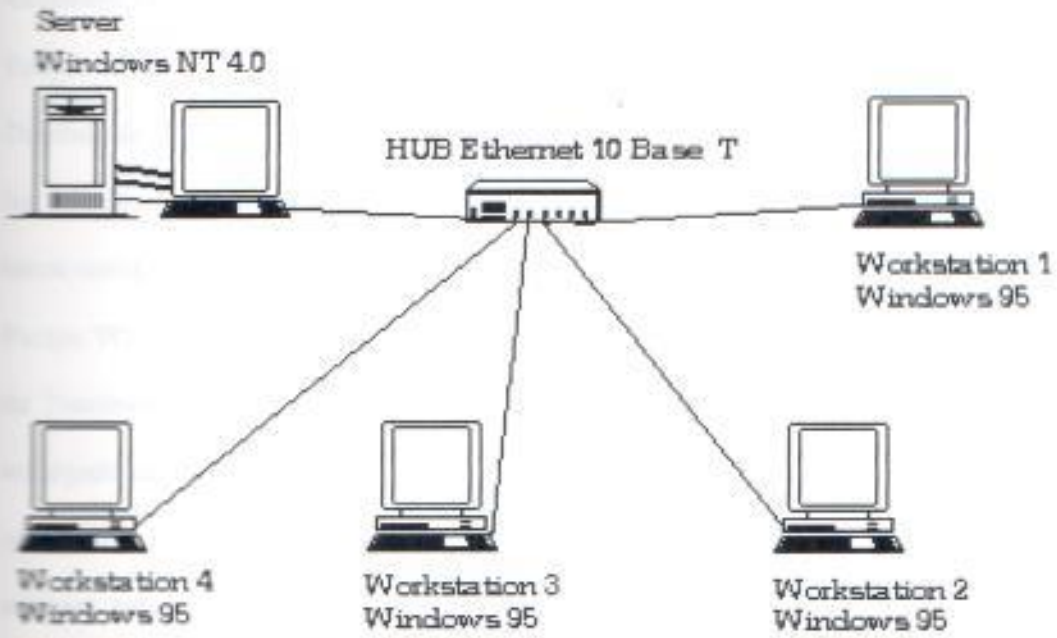


Figura 2.4 Esquema de configuración de una red LAN Ethernet

Protocolos Utilizados.

Para este proyecto utilizamos 2 tipos de protocolos que nos ofrece Microsoft Windows 95 en el paquete para la Estación de Trabajo (Cliente), y asimismo en el paquete de Windows NT para el Servidor. Estos protocolos fueron: El Netbeui (Protocolo Estándar para Redes Microsoft) y el protocolo TCP/IP de Microsoft.

Las razones por las cuales utilizamos estos protocolos son las siguientes:

- NETBEUI es un protocolo estándar para Redes Microsoft, que permite compartir los recursos por todas las computadoras en la LAN que lo tenga definido. Cada PC en la LAN con este protocolo definido puede compartir recursos como: discos duros, directorios,

impresoras, etc. Con diferentes privilegios y hace tan sencillas las operaciones como transferencias de archivos, impresiones a un puerto compartido, etc.

- Porque NETBEUI ofrece de una manera sencilla y visual el inicio de sesión en un Dominio de Windows NT que es controlado por un Servidor con Windows NT y recupera los permisos, privilegios y demás perfiles sobre los recursos del servidor al usuario que inicia sesión en el Sistema Operativo de Red.
- Porque TCP / IP, con uno de sus 4 conjuntos de protocolos, el TCP (Protocolo de Control de Transmisión) proporciona una entrega fiable de los paquetes, y este protocolo es el encargado de retransmitir el paquete TCP si es que el paquete se daña o se pierde, esto convierte a TCP / IP, en el protocolo más fiable y adecuado para transmitir datos por sesiones, para aplicaciones Cliente - Servidor y Transferencia de Archivos.
- Por el control en el enrutamiento hacia la dirección IP destino de TCP / IP que hace más rápido y directa la transferencia de archivos.

  Es necesario acotar que para la configuración de esta tabla de enrutamiento fue necesario en la instalación del S.O. del Servidor especificar que éste se encargue de la resolución de la tabla de enrutamiento.
- Por la experiencia en estos dos protocolos, Por ser de fácil instalación y configuración.
- Por los accesos que hace la Aplicación Cliente a Directorios del Servidor para la transferencia de los archivos multimedios, el protocolo NETBEUI ofrece una transferencia simple y rápida de estos archivos.

La forma en que se manejo la Red Windows fue simple, cada una de las computadoras se les definió dentro de un mismo Dominio NT. Cada estación Cliente solicita el usuario en el

Dominio NT y su contraseña, para de esta manera iniciar sesión en la Red NT (previamente se deben crear los usuarios y sus cuentas dentro del Sistema Operativo NT).

Desde ese momento el usuario que hizo el logon recupera sus conexiones de Red, impresoras y permisos sobre los recursos del servidor.

Entre una de estas conexiones deberá estar especificada una conexión a un directorio del servidor donde se encuentran los archivos de la aplicación del Kiosko Multimedio, en este caso, los archivos multimedios, donde cada usuario tendrá permisos de lectura, modificación y eliminación para el grupo de usuarios administradores del sistema.

2.2 SISTEMAS OPERATIVOS

La elección de los sistemas operativos se debió así mismo a diferentes factores. Para hacer una buena elección se debe considerar el tipo de computadora que se tiene, el uso que se le va a dar, etc. Por esto se seleccionó un S.O. para el Cliente y otro para el Servidor.

El Sistema Operativo utilizado en el Cliente es Microsoft Windows 95, y las razones por las cuales lo elegimos son:

- Por ser el Sistema Operativo más difundido en el mundo para PCs. De características como ser multitareas y full direccionamiento 32 bits. Y el más utilizado como Workstation.
- G.U.I., por su interface gráfica y visual orientada a objetos.
- Por ser el sistema operativo nativo de 32 bits en el que se pueden desarrollar aplicaciones basadas en Windows
- Por la gran cantidad de controladores para los diferentes periféricos y su propiedad de configuración automática para periféricos plug and play.
- Por ser el sistema operativo de Microsoft para Clientes o Workstations que permite el manejo del Driver o controlador O.D.B.C. también de Microsoft y la utilización del API de los objetos para manejar D.D.E., OLE, etc.

En el Servidor el Sistema Operativo elegido fue Windows NT Server 4.0 y a continuación citaremos las razones para seleccionarlo.

- Gracias a su G.U.I., interface gráfica y visual muy parecida a Microsoft Windows 95, hace que las tareas de configuración y administración del Servidor sean mucho más sencillas que cualquier otro Sistema Operativo de Red Multiusuarios ya que no es necesario utilizar ningún comando.
- Por ser un sistema operativo nativo multiusuario que permite la configuración de la Red LAN o Redes Remotas de una manera simple, aparte de que ofrece muchas ventajas por que el mismo servidor puede configurar y Administra la Red utilizando protocolos NETBEUI, TCP / IP.
- Por la gran cantidad de controladores para los diferentes periféricos y su propiedad de configuración automática para periféricos plug and play.
- Por ser el sistema operativo de Microsoft que trabaja conjuntamente con Microsoft SQL Server para atender los requerimientos de los PCs Workstation que accedan a las Bases de Datos de SQL Server por medio del controlador o driver de O.D.B.C. el estándar de Microsoft.
- Por la arquitectura paralela con la que fue diseñado Windows NT Server y SQL Server para atender los requerimientos Clientes (Front - End).
- Por sus múltiples ventajas en el manejo de los recursos del servidor Administración de la Memoria, Seguridades a los Directorios, archivos y recursos del Servidor a los usuarios, su nuevo y mejorado Sistema de archivos, etc. y las herramientas administrativas para la creación de cuentas para los usuarios y designación de privilegios sobre los recursos de éste para los usuarios o grupos de trabajo.
- Por la factibilidad del producto, ya que este producto está al alcance de todos economicamente y es de fácil soporte, configuración y administración.

2.3 ¿QUE ES SQL-SERVER 6.5?

Microsoft SQL Server 6.5 es un sistema manejador de base de datos (DBMS) escalable y de alto rendimiento, el cual ha sido diseñado para satisfacer los requerimientos de los ambientes distribuidos cliente/servidor.

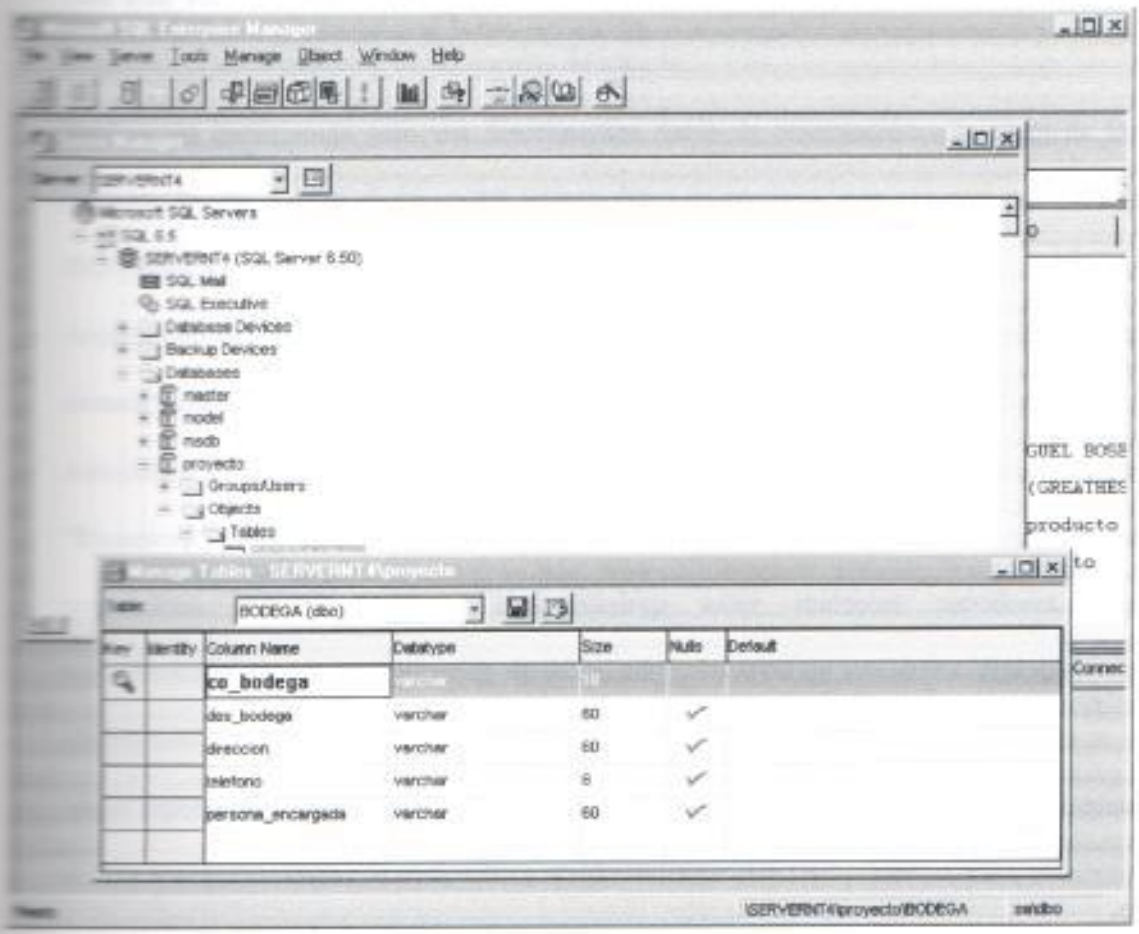


Figura 2.5 SQL Server 6.5

Microsoft SQL Server proporciona:

- Integración con el "threading" y administrador de servicios de Microsoft Windows NT®, monitor de rendimiento, visor de eventos.

  Una sola ventana de logon de Windows NT tanto para la red como para SQL Server, simplifica la administración de las cuentas de usuario.
- Replicación incorporada para una diseminación fiable de información a lo largo de una empresa, lo que reduce el riesgo del tiempo fuera de servicio, a la vez que posibilita la disponibilidad de información exacta cerca de la gente que la necesita.
- Arquitectura paralela. Se incrementa dramáticamente el rendimiento y la escalabilidad del sistema mediante la ejecución en paralelo de funciones internas de la base de datos.
- Administración centralizada de servidores a lo largo de la empresa mediante el uso del "framework" comprensivo distribuido. Una interface de administración basada en ventanas proporciona controles visuales drag-and-drop sobre múltiples servidores, para administración remota de replicación de datos, administración de servidores, diagnósticos y afinamiento.
- Mejor soporte para bases de datos muy grandes gracias a su arquitectura en paralelo. Reduce el I/O para muchas tareas de desarrollo y mantenimiento.

Las razones por las cuales escogimos el SQL Server 6.5 como RDBMS, son las siguientes:

- SQL Server permite el desarrollo de aplicaciones Cliente - Servidor, no solo utilizando las herramientas de desarrollo de Microsoft, como Visual Basic o Visual C++, sino con otros tipos de herramientas para desarrollo Front - End. Por medio de la conectividad a bases de datos abiertas estándar de Microsoft O.D.B.C. o por medio de librerías de conectividad que brinda el mismo SQL Server.
- Por ser uno de los RDBMS que trabaja alcanzando un mejor rendimiento con el Sistema Operativo escogido.
- Por experiencias previas, ya que algunos integrantes del grupo ya han trabajado con este administrador de bases de datos y conocen el tipo de programación y administración del Sistema Administrador de las bases de datos.
- Por su disponibilidad y costos.

2.4 O.D.B.C. OPEN BATABASE CONNECTIVITY (CONECTIVIDAD A BASES DE DATOS ABIERTAS)

INTRODUCCION A LA ARQUITECTURA ODBC

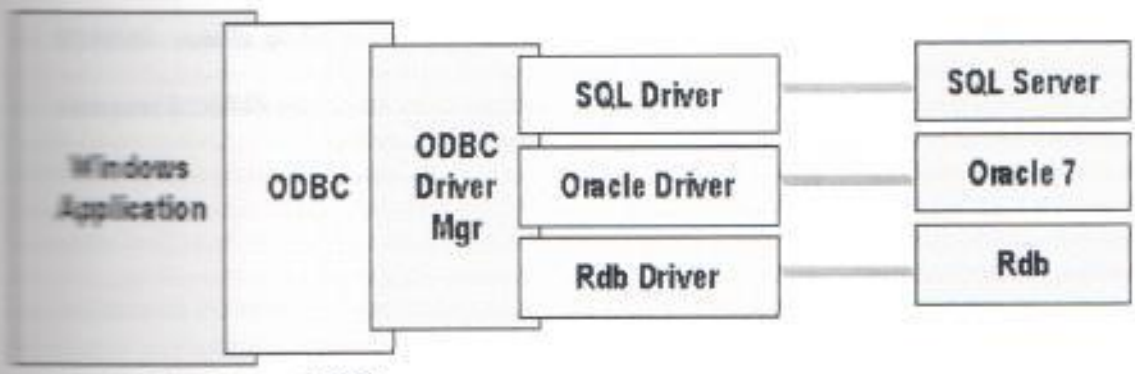


Figura 2.6 Arquitectura ODBC

Requerimientos:

El controlador SQL Server requiere el siguiente software:

- Microsoft Windows versión 3.0, o una versión posterior
- Administrador de controladores O.D.B.C. versión 1.0, o una versión posterior (ODBC.DLL)

Las siguientes son las razones por las cuales elegimos utilizar O.D.B.C. para la conectividad con la base de datos:

- La configuración del Administrador O.D.B.C., es muy sencilla.
- Hace transparente para el programador la manera en que el O.D.B.C. para el RDBMS maneja y utiliza los protocolos de Red, maneja la conectividad al RDBMS y la forma en que este procesa el query y retorna los datos o resultados

- Por ser un estándar para las aplicaciones de Microsoft además las herramientas de desarrollo como Visual Basic y Visual C++ permiten la utilización de O.D.B.C. y ofrecen gran cantidad de información sobre las funciones y rutinas del API de los Controladores O.D.B.C.
- Por que permiten la conectividad con casi todos los paquetes RDBMS en el mercado, y de esta manera, el sistema no dependería de la Base de Datos y se podría migrar de uno a otro RDBMS con la ayuda de un software CASE que ayuda a la creación de la D.B. para cualquier RDBMS comercial en el mercado.

2.5. ER-WIN

ERwin es una herramienta de modelamiento de datos que pertenece a la gran familia de Logic Works®. Una de las principales características de ERwin es la de poder conectarse a un repositorio común de datos compartidos.

Las razones por las cuales escogimos E-Rwin como herramienta de desarrollo para base de datos son las siguientes:

- Por facilitar las operaciones de la creación de la base de datos, con una herramienta CASE se reduce el tiempo de creación de la estructura e integridad referencial de la base de datos.
- Por ser una de las herramientas CASE más utilizadas para el desarrollo de sistemas Cliente / Servidor.
- Por la gran cantidad de RDBMS que se pueden elegir para el diseño del esquema de la base de datos.
- Por que presenta de una manera gráfica el modelo Entidad - Relación haciendo más comprensivo el esquema y tiene todas las propiedades de cualquier aplicación Windows (drag - drop, etc.)

2.6 HERRAMIENTAS PARA LA CAPTURA, EDICIÓN Y DEPURACIÓN DE LOS ARCHIVOS MULTIMEDIOS.

Los formatos de los archivos con los que trabajamos son (*.avi , *.wav , *.mid) y para manejar las imágenes el formato escogido fue el de mapas de bits (*.bmp), una razón para esto es que el control MCI (Media Control Interface) sólo soporta este tipo de archivos. Por esta razón fue necesario utilizar herramientas como: PBTV4, Adobe Premier V1.1, Multimedia Component System MCS para Windows 95, La Grabadora de sonidos de Windows 95, Paint Shop Pro V3.1 , Graphic WorkShop V1.1. Cada una de estas herramientas nos sirvió para capturar, editar y modificar los archivos de recursos multimedios que utilizamos.

Las aplicaciones PBTV4 y el Adobe Premier V1.1 nos sirvieron para poder capturar archivos de video en formato *.avi y para cambiar de formato algunos recursos de videos que se encontraban en otros formatos como *.mov. El PBTV4 trabaja directamente con una tarjeta capturadora de video que se instala conjuntamente con la tarjeta de video. Con la instalación de la aplicación PBTV4 se instalan en el sistema los drivers y controladores para este periférico y en la misma instalación se configura el dispositivo, esta aplicación puede tener varias entradas de video como: televisión, video cintas, video grabadoras, etc. y podemos escoger el formato del archivo final en nuestro caso *.avi.

El Adobe Premier nos sirvió para convertir y editar los archivos de video, como algunos archivos de video los conseguimos por medio del Internet y la gran mayoria de ellos con formato *.mov fue necesario editarlos y convertirlos a formato *.avi.

El Multimedia Component System MCS para Windows 95 y La Grabadora de sonidos de Windows nos sirvió para grabar las canciones y demás archivos de sonidos en formato *.wav. Con el Multimedia Component System MCS reproducimos las canciones de un disco compacto y

Luego con la misma aplicación grabamos en archivos con formato *.wav partes de las canciones, hay que recalcar que estos archivos tienen una gran capacidad por que la calidad de la grabación es muy buena lo que hace que el archivo de onda sonora sea de gran tamaño, lo mismo sucede con los archivos de video. La Grabadora de sonidos nos sirvió para editar los archivos y en algunos casos aumentarles algunos efectos.

Paint Shop Pro, Graphic WorkShop, son herramientas para manejar archivos de imágenes que usamos para editar los archivos gráficos, las imágenes utilizadas fueron las portadas de los discos compactos, que fueron digitalizadas con la ayuda de un escáner y un software para capturarlas, luego de tener las imágenes digitalizadas las estandarizamos a una misma dimensión y en formato *.bmp.

Para efectuar estas operaciones se necesita tener una computadora con suficiente espacio disponible en disco, un buen procesador rápido y que soporte grandes exigencias. Y tener en consideración que los archivos para recursos multimedios son de gran tamaño.

Control Multimedia MCI

El Control de Interface Multimedia MCI es una interface de alto nivel a dispositivos multimedios y a archivos de recursos multimedios. MCI provee varios usos con capacidades a dispositivos independientes para controlar audio y periféricos visuales. Su aplicación Visual Basic Visual C++ puede usar el control MCI para cualquier soporte a dispositivos Multimedia, incluso aparatos o dispositivos de onda de audio, secuenciadores MIDI, aparatos de audio CD, y dispositivos de video digital. MCI provee comandos estándar para reproducir los recursos a dispositivos multimedios y graba archivos del recurso multimedio. Estos comandos son una interface virtual genérica a cada dispositivo Multimedia.

El control Multimedia MCI es programable por medio de diferentes vias:

- El control puede ser visible o invisible en tiempo de ejecución.
- Se puede aumentar o redefinir completamente la funcionalidad de los botones en el control.
- Se puede utilizar múltiples controles de dispositivos en una ventana.

Los eventos (o definiciones de los botones) del control Multimedia MCI son programables por medio del API que ofrece.

En el proyecto Visual C++ los botones están desabilitados, dejamos el control invisible.

El Control Multimedia MCI maneja la grabación y reproducción de archivos Multimedia sobre la Interface Multimedia del Control (MCI) en los dispositivos. Conceptualmente, este control es un conjunto de botones presionables que emiten comandos MCI para los dispositivos como a tarjetas de audio, secuenciadores MIDI, controladores o drivers CD-ROM, reproductores de audio en CD, reproductores de video - disco, reproductores y grabadores de video - cintas, etc. El control MCI también puede soportar la reproducción de Video para Windows en archivos con formato (*.AVI).

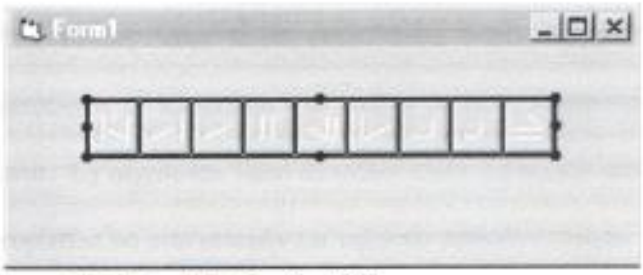


Figura 2.7 Control Multimedia MCI

Las propiedades pueden ser configuradas en tiempo de ejecución o en tiempo de diseño.Los botones están definidos como: Prev, Next, Play, Pause, Back, Step, Stop, Record, and Eject, respectivamente y también pueden ser seteados para que estén visibles o no.

Nombres de Archivos:

MCI16.OCX (para 16 bits), MCI32.OCX (para 32 bits)

Nombre de Clase

MMControl

La propiedad Comando

Desde el código fuente se pueden especificar los comandos para ejecutar y reproducir los archivos multimedios. Esta propiedad no está disponible en tiempo de diseño. Pero si dejamos habilitados los botones y el usuario los presiona, éstos emitirán los comandos necesarios automáticamente para ejecutar y reproducir los archivos multimedia.

Sintaxis

MMControl.Command =cmdstring$

Comentarios

El argumento cmdstring$ da el nombre del comando MCI para ejecutar: Open, Close, Play, Pause, Stop, Back, Step, Prev, Next, Record, Eject, Sound o Save. El comando es ejecutando de inmediato y el código de error es almacenado en la propiedad del control llamada Error. La siguiente lista describe cada comando and lista las propiedades que usa. Si una propiedad no está seteada, un valor de default es usado.

Comando	Descripción/Propiedades usadas
Open	Abre un dispositivo usando el comando MCI_OPEN.
Close	Cierra un dispositivo usando el comando MCI_CLOSE.
Play	Reproduce un dispositivo usando el comando MCI_PLAY.
Pause	Pone en pausa la reproducción o grabación usando el comando MCI_PLAY. Si es ejecutado mientras el dispositivo es pausado, intenta enviar un resumen de la reproducción o de grabación usando el comando MCI_RESUME.
Stop	Para la reproducción o grabación usando el comando MCI_STOP.
Back	Retrocede paso a paso usando el comando MCI_STEP.
Step	Paso a paso hacia adelante usando el comando MCI_STEP.
Prev	Va al track previo o canción anterior. Usa el comando Seek.
Next	Va al siguiente track (si estamos en el último track, va al track inicial) usando el comando Seek.
Seek	Si no está reproduciéndose, busca una posición usando el comando MCI_SEEK. Si reproduce, continua reproduciendo desde la posición que le da usando el Comando MCI_PLAY.
Eject	Arroja el medio usando el comando MCI_SET.
Sound	Reproduce el sonido usando el comando MCI_SOUND.
Save	Guarda un archivo abierto usando el comando MCI_SAVE.

CAPITULO 3

ANALISIS Y DISEÑO

ORIENTADO A OBJETOS

CAPITULO 3.

ANALISIS Y DISEÑO ORIENTADO A OBJETOS

Este capítulo contiene la parte medular del sistema. Muestra toda la etapa del análisis que se realizó previo a la implementación del sistema.

El Análisis es la etapa de investigación donde se pone en claro la situación del sistema y junto con el diseño se modela el sistema que al final se implementará.

El análisis realizado por nosotros tiene dos partes, el análisis de la estructura de los objetos y el análisis del comportamiento de los objetos.

El diseño de la aplicación también se divide en dos partes, el diseño de la estructura de los objetos y el diseño del comportamiento de los objetos.

3.1. ANALISIS DE LA ESTRUCTURA DE OBJETOS

Entre el análisis de la estructura de objetos podremos ver los tipos de objetos, asociaciones de un objeto, generalizaciones y la composición de objetos.

A continuación se muestran los diagramas de esquemas de objetos de una manera general y luego de manera más detallada, donde se destacan las clases más significativas del sistema.

3.1.1. ESQUEMA GENERAL DE OBJETOS

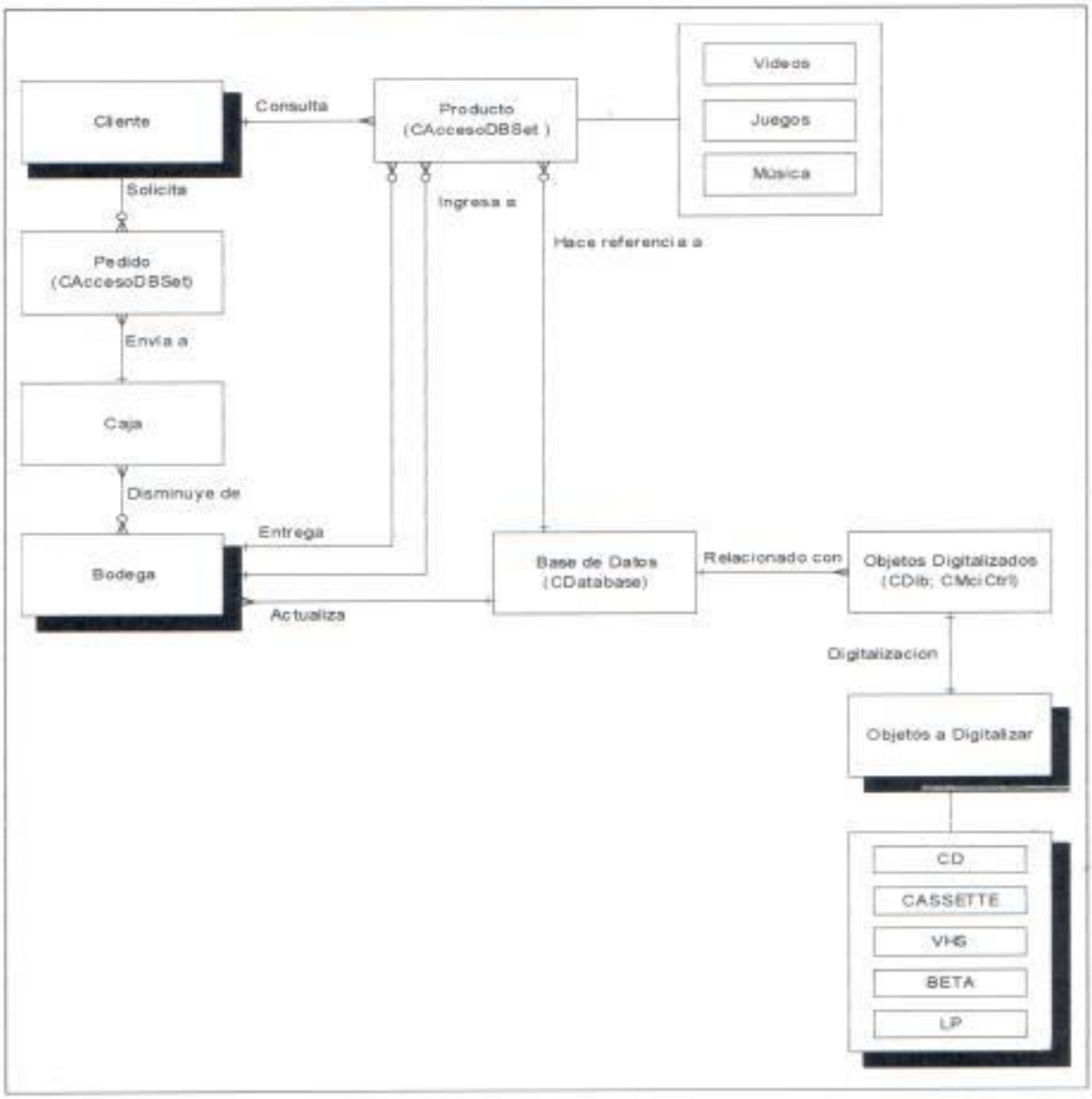


Figura 3.1 Esquema General de Objetos del Sistema

3.1.2. ESQUEMA DETALLADO DE OBJETOS

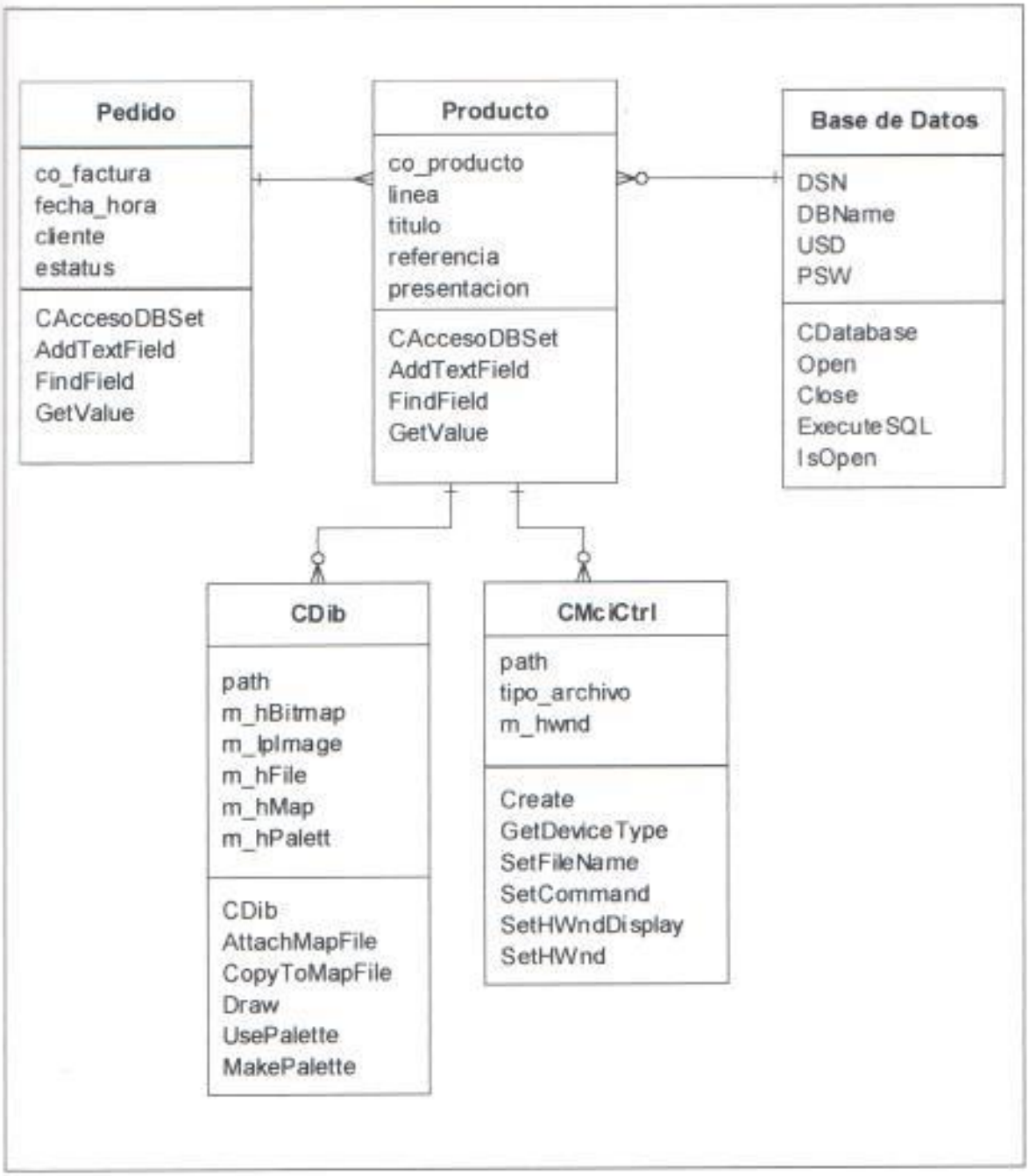


Figura 3.2 Esquema Detallado de Objetos

3.1.3. MODELO ENTIDAD RELACION DE LA BASE DE DATOS

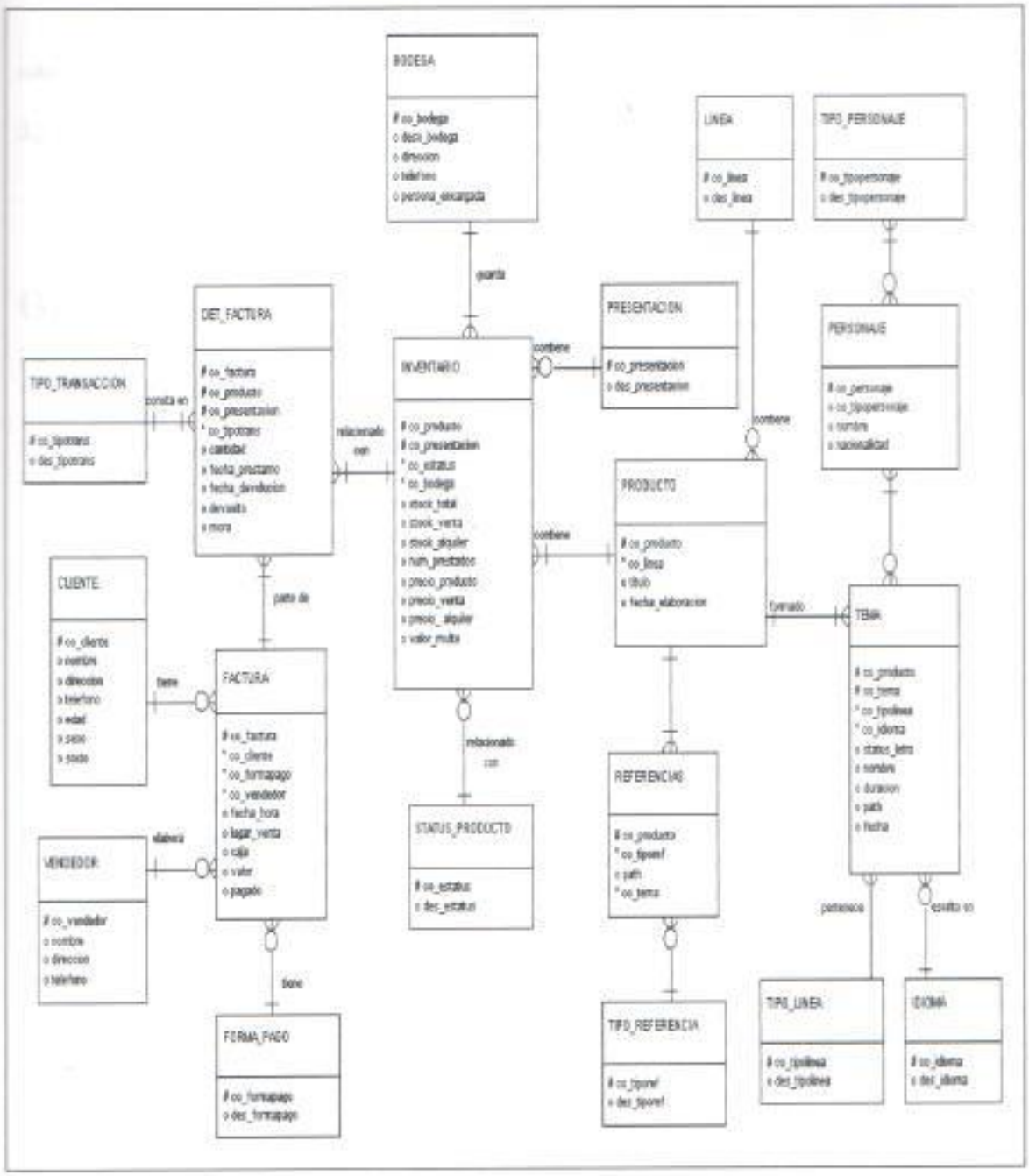


Figura 3.3 Modelo Entidad Relación de la Base de Datos

3.2. ANALISIS DEL COMPORTAMIENTO DE OBJETOS

Dentro del análisis del comportamiento de objetos encontraremos los tipos de objetos, estados, reglas de activación, condiciones de control y funciones.

3.2.1. ESQUEMAS DE EVENTOS

Consulta de Productos

Opción : *Videos*

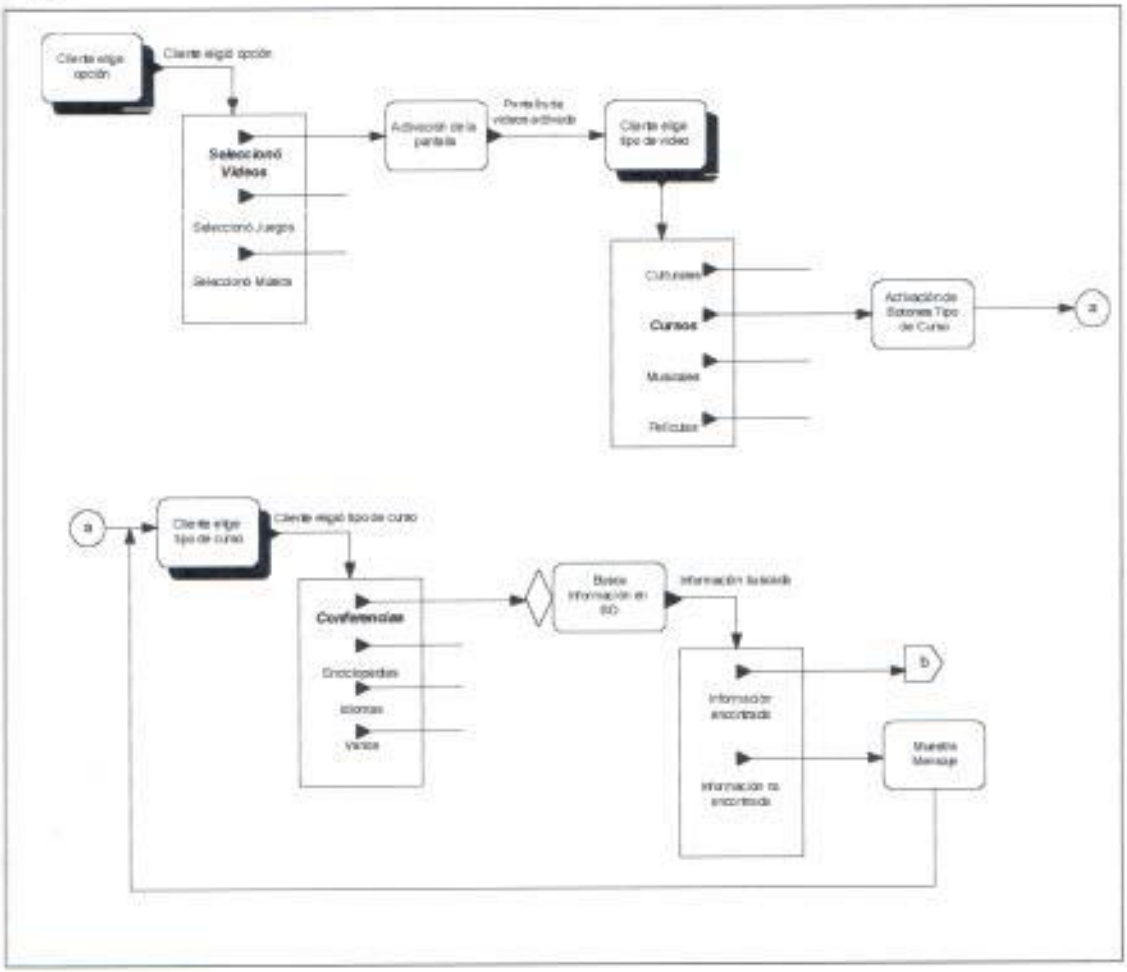


Figura 3.4 Esquema de Eventos – opción Videos-Cursos

Refierase a la sección 3.3 para las analogías

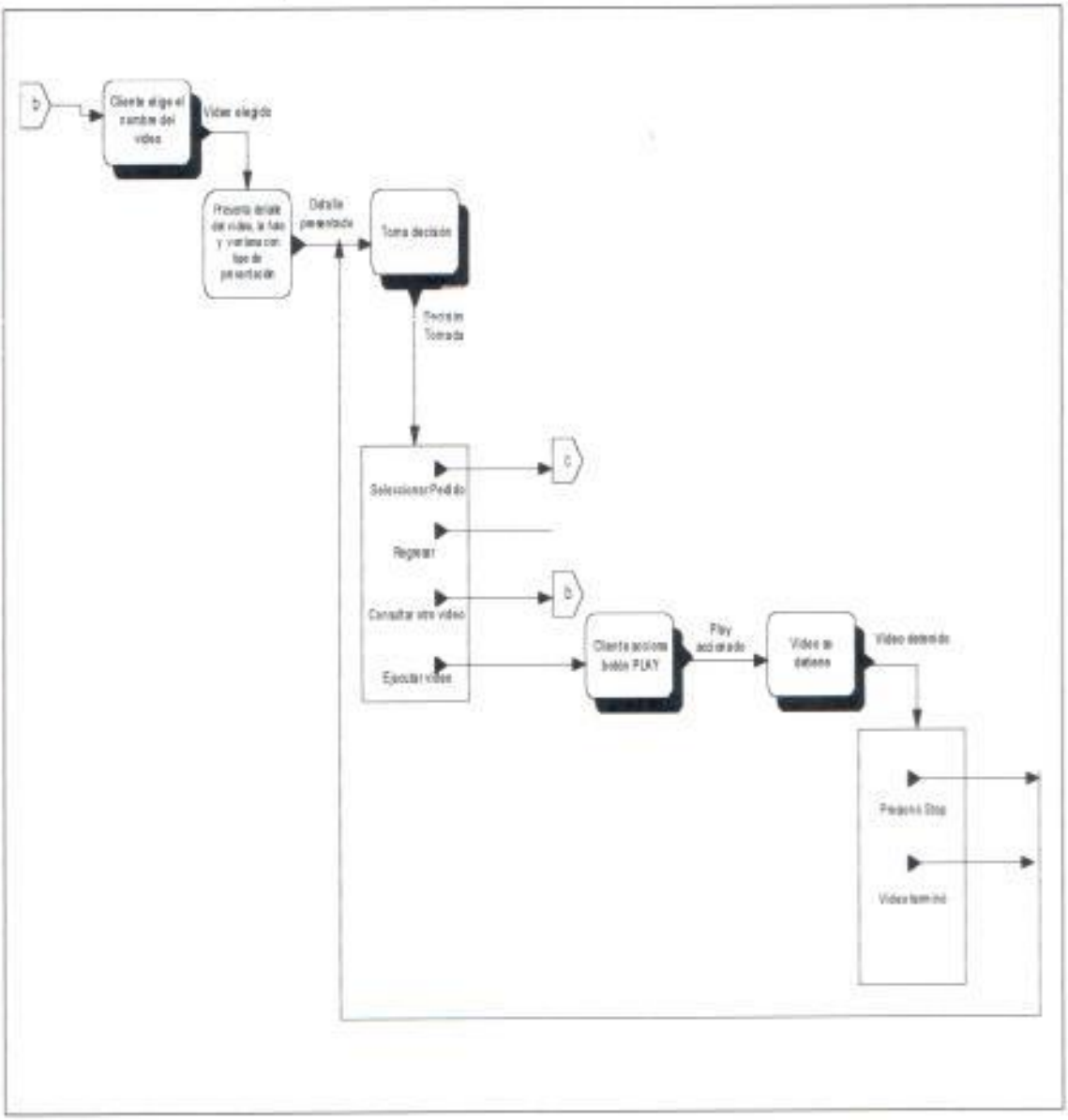


Figura 3.5 Esquema de Eventos – Consulta de Videos

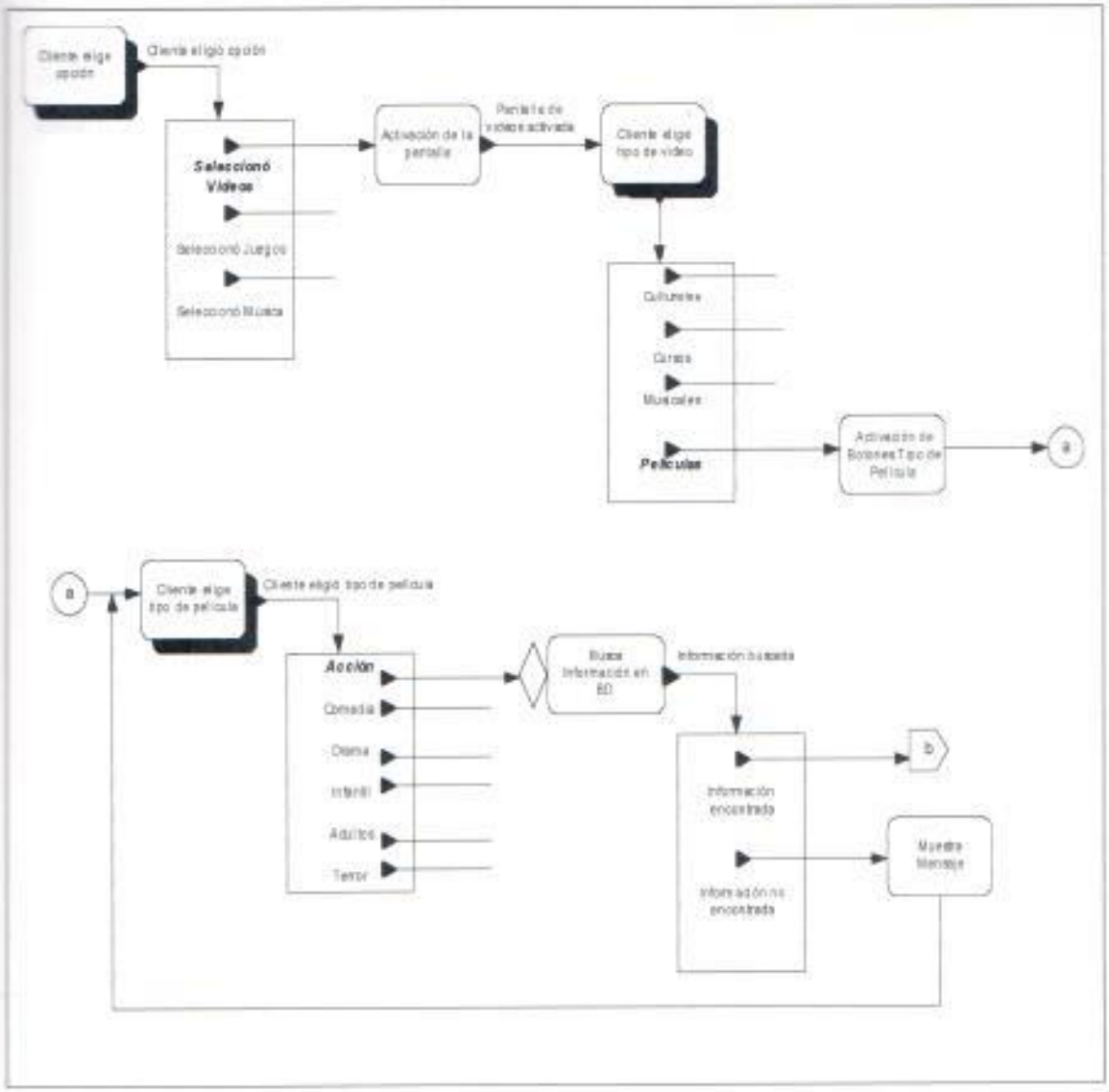


Figura 3.6 Esquema de Eventos – opción Videos-Películas

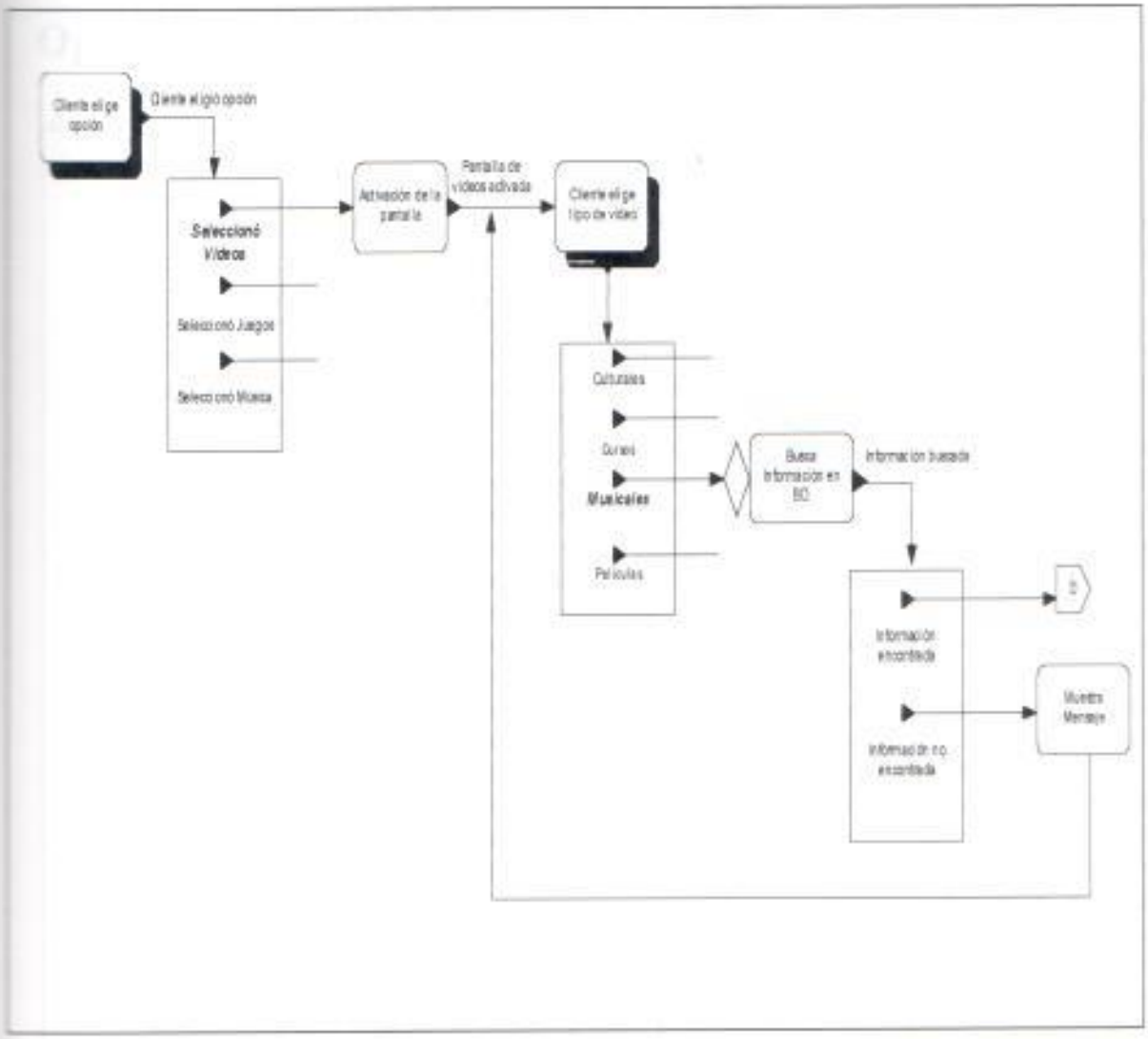


Figura 3.7 Esquema de Eventos – opción Videos-Musicales

Opción : *Juegos*

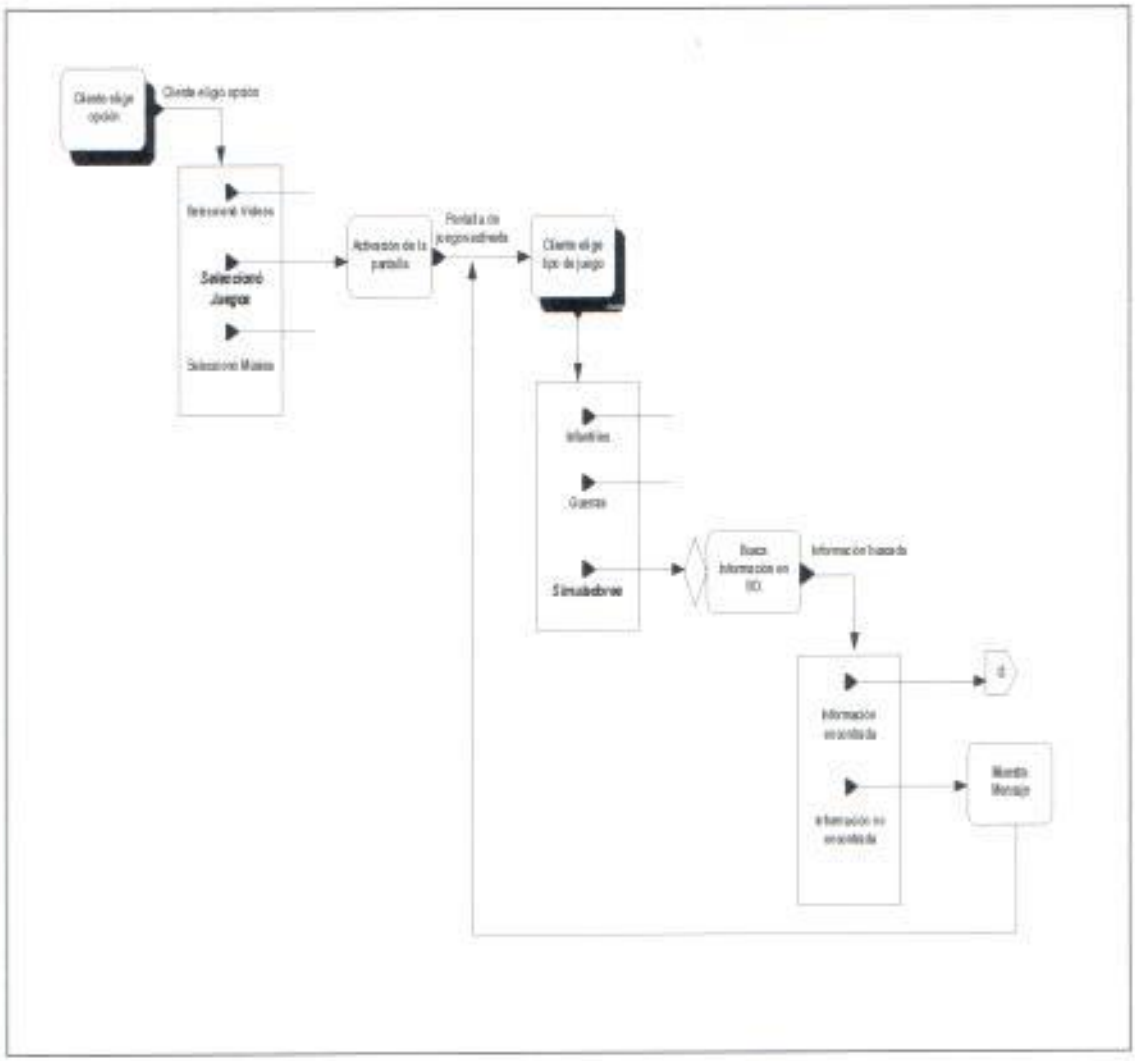


Figura 3.8 Esquema de Eventos – opción Juegos

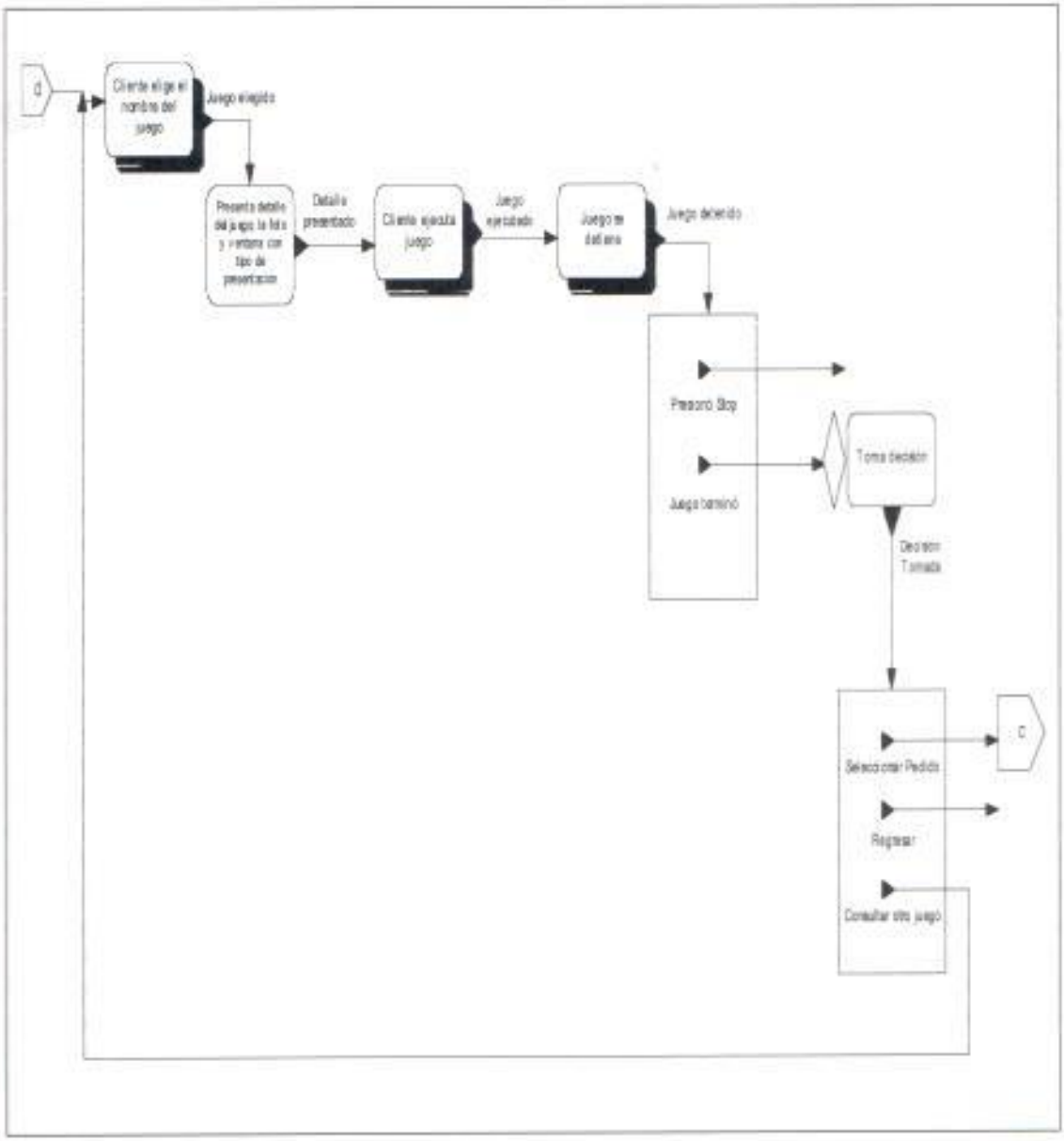


Figura 3.9 Esquema de Eventos – Consulta de Juegos

Opción : *Música*

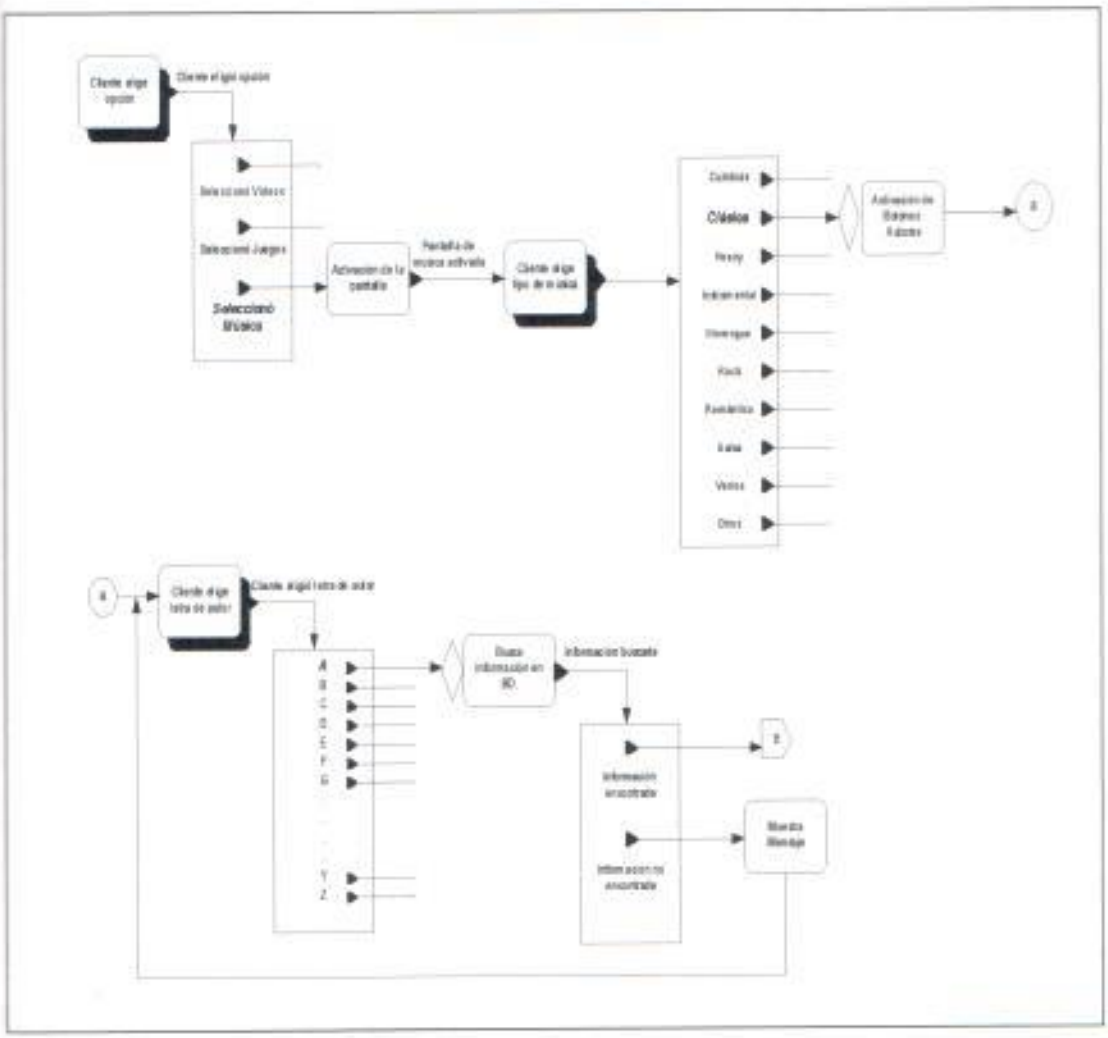


Figura 3.10 Esquema de Eventos – opción Música

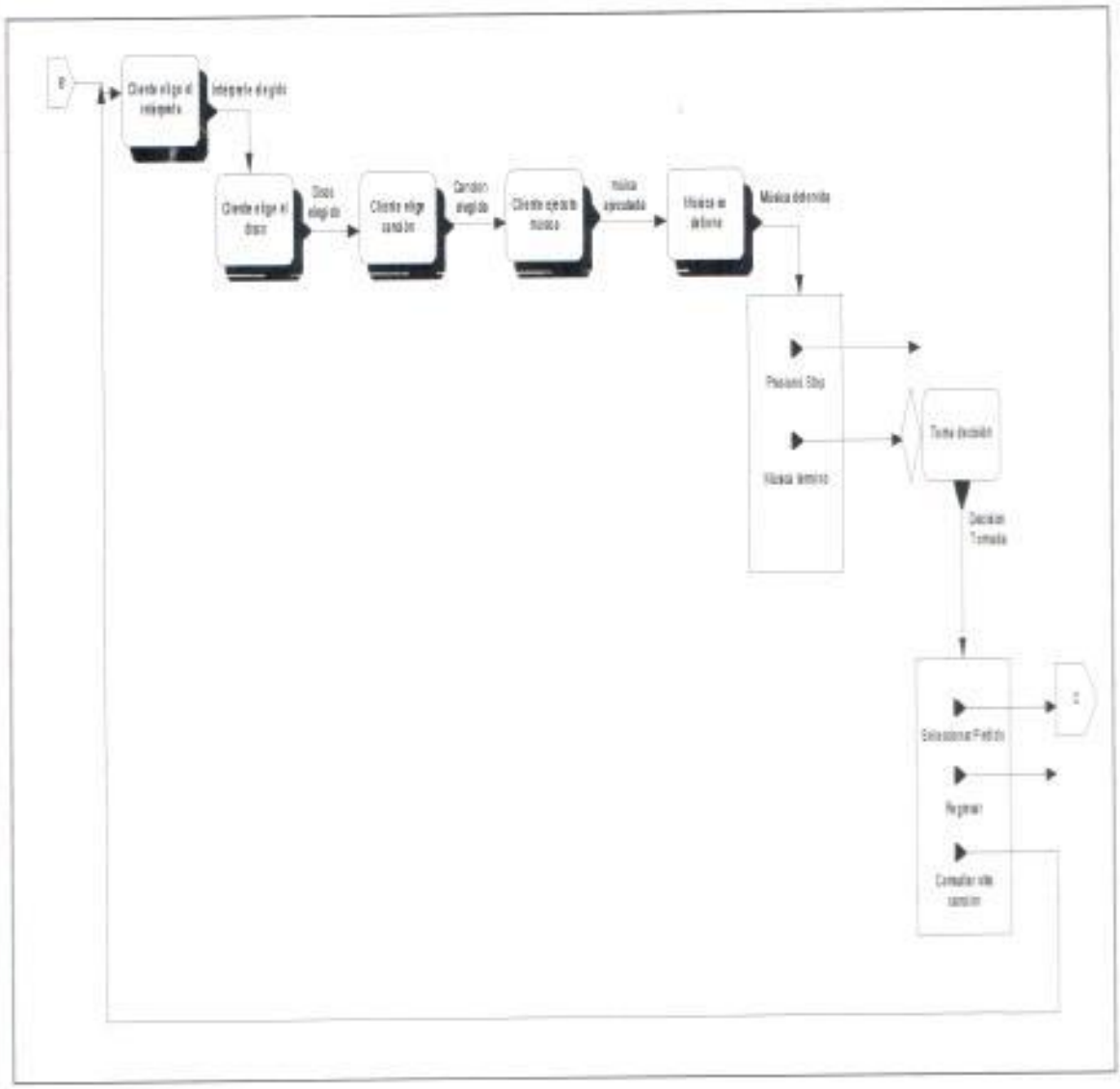


Figura 3.11 Esquema de Eventos – Consulta de Música

Seleccionar Producto para llevar

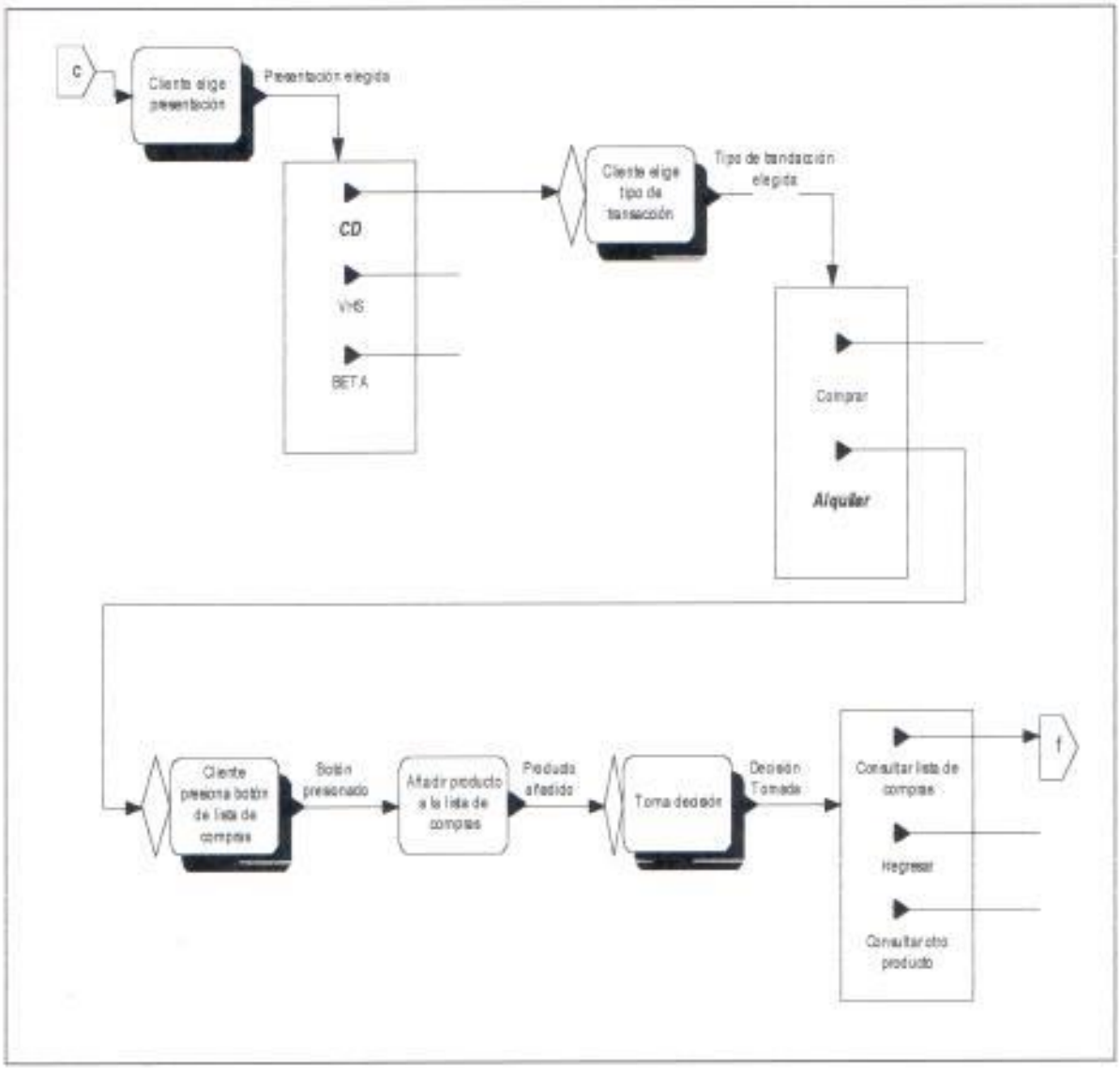


Figura 3.12 Esquema de Eventos – Selección de productos para comprar

Consulta del Listado de Compras

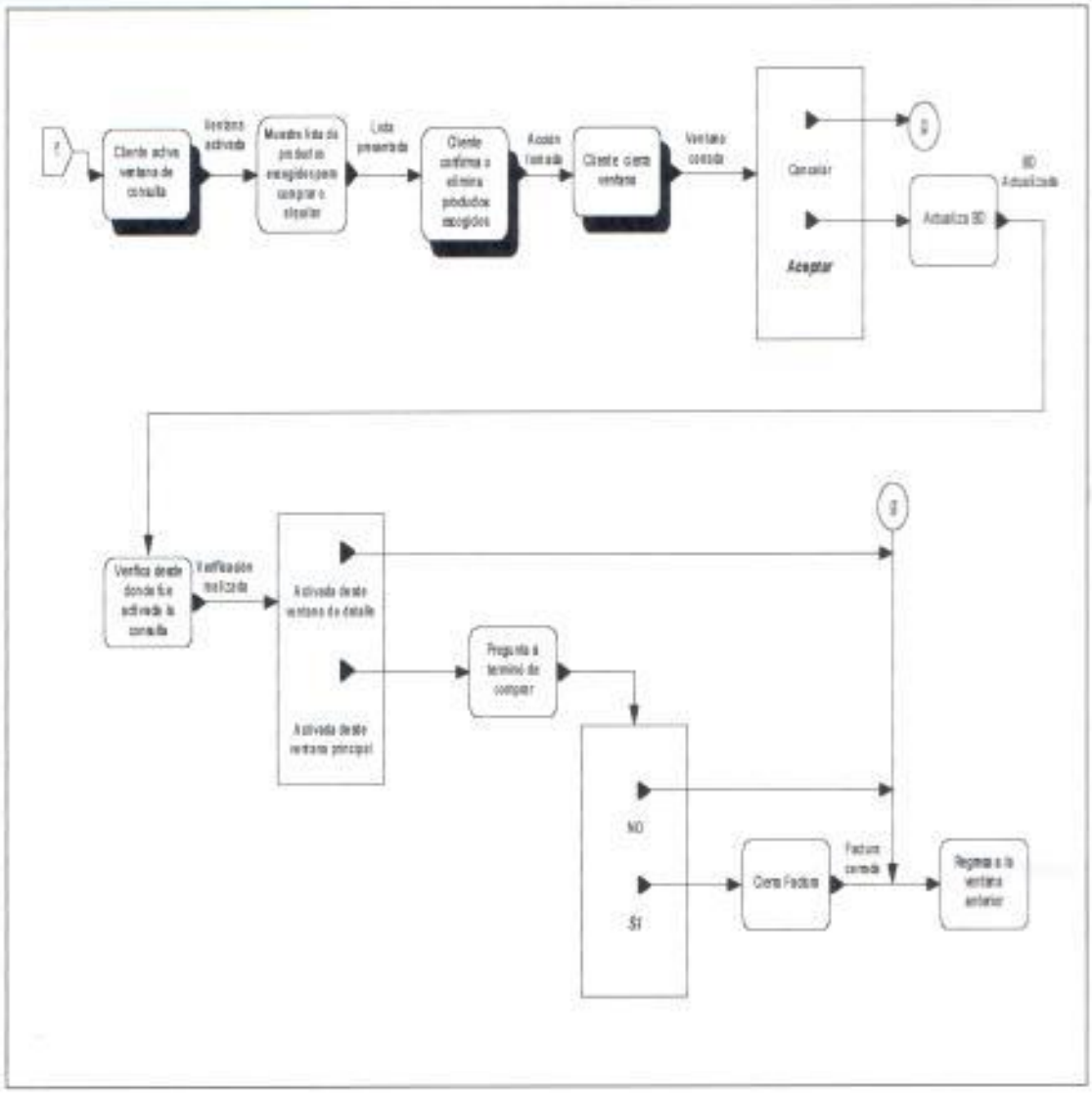


Figura 3.13 Esquema de Eventos – Lista de compras

3.2.2. DIAGRAMA DE FLUJO DE OBJETOS

A continuación los diagramas más relevantes.

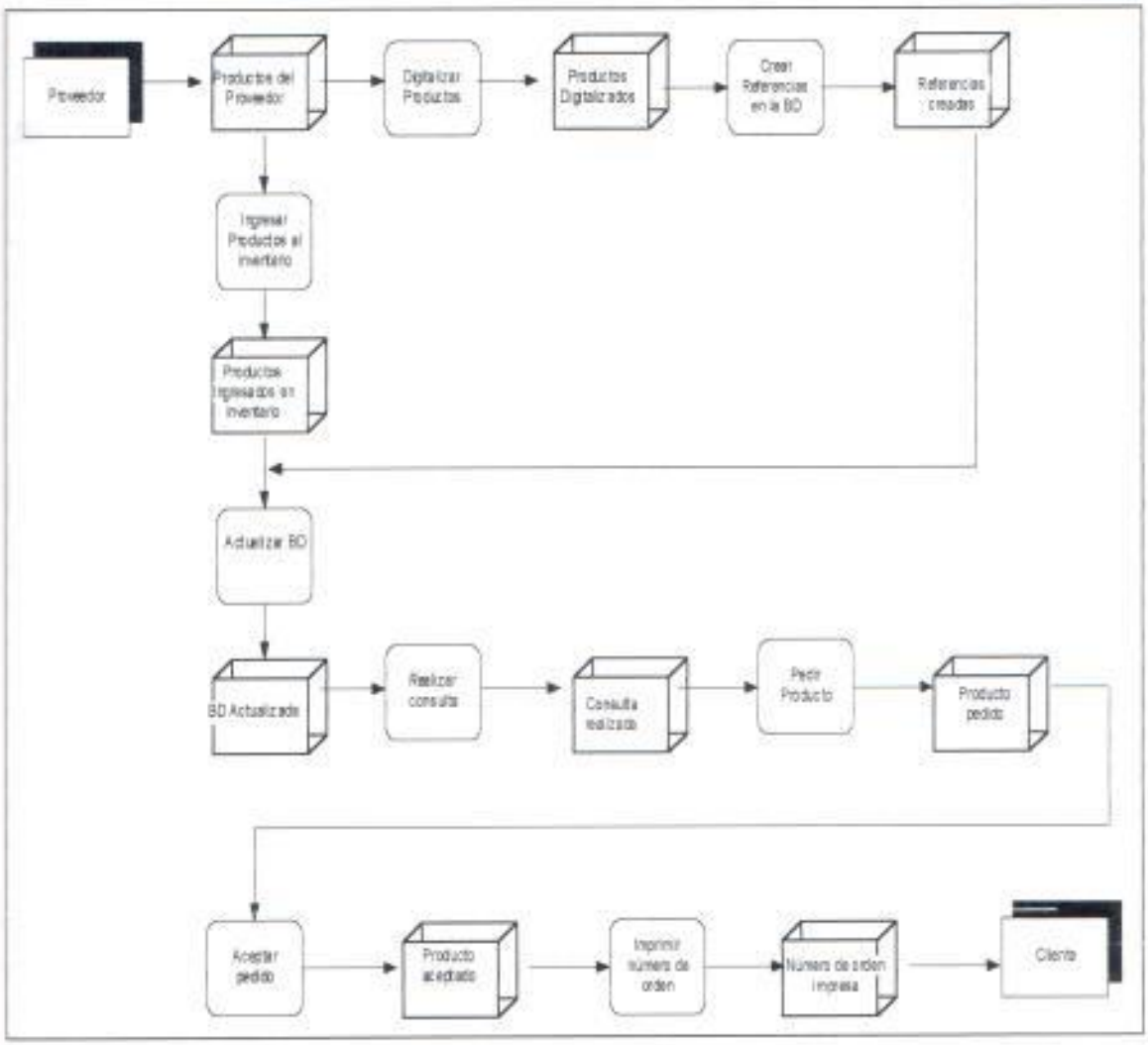


Figura 3.14 Diagrama de Flujo de Objetos – Ingreso de Productos y Pedido

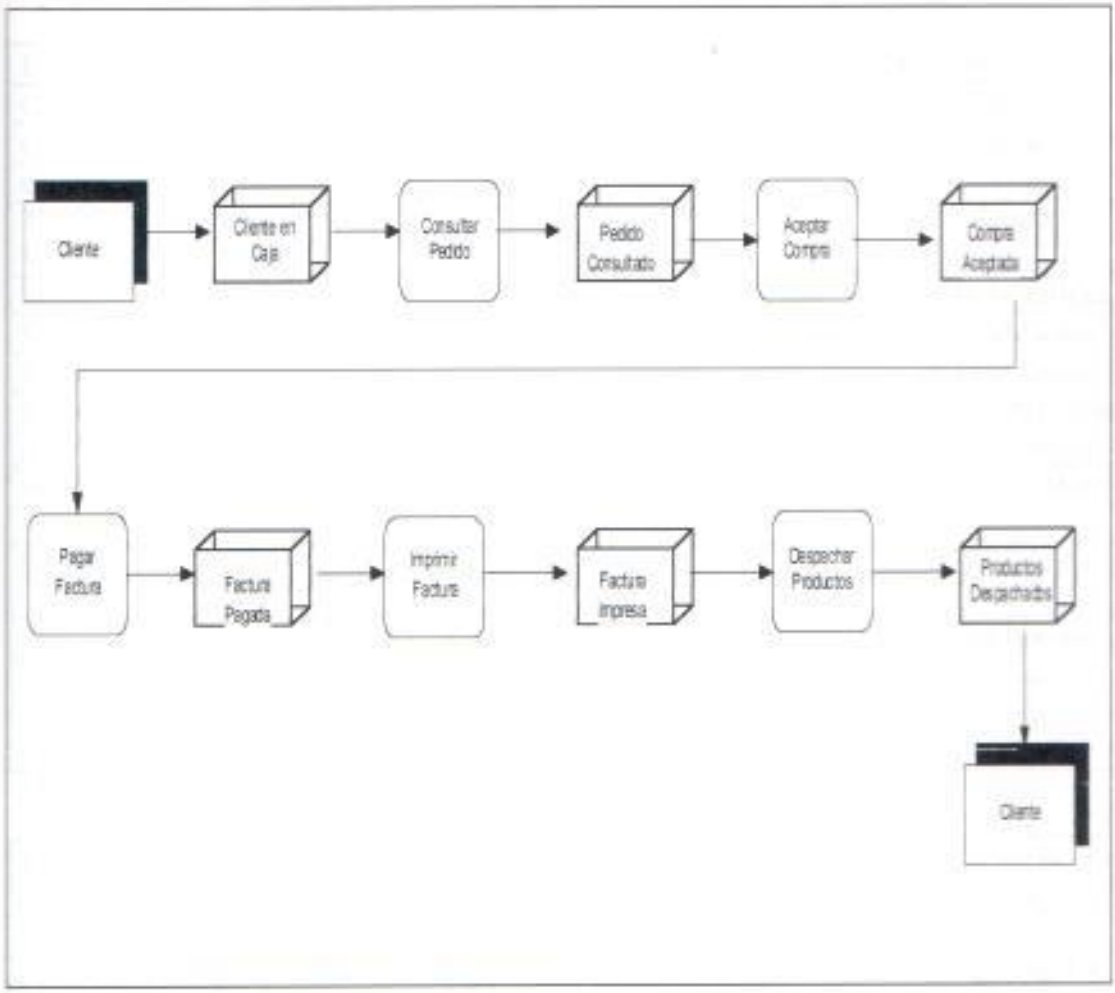


Figura 3.15 Diagrama de Flujo de Objetos – Pagar y despachar

3.3 ANALOGIAS ENTRE ANALISIS Y DISEÑO CON IMPLEMENTACION

Evento	Clase	Control	Evento de la clase
Cliente Selecionó Videos	Principal	Botón Videos	OnClickVideos
Cliente Selecionó Juegos	Principal	Botón Juegos	OnClickJuegos
Cliente Selecionó Música	Principal	Botón Musica	OnClickMusica
Activación de la Ventana de Videos	videos		OnInitDialog
Selecciona Video Cultural	videos	Botón Cultural	OnClickCultural
Selecciona Video Musical	videos	Botón Musical	OnClickMusical
Selecciona Video Cursos	videos	Botón Cursos	OnClickCursos
Selecciona Videos de Peliculas	videos	Botón Peliculas	OnClickPeliculas
Activación de Botones	videos		MostrarBotones
Selecciona Conferencias	videos	Botón Conferencia	OnClickConferencias
Selecciona Enciclopedias	videos	Botón Enciclopedias	OnClickEnciclopedias
Selecciona Idiomas	videos	Botón Idiomas	OnClickIdiomas
Selecciona Varios	videos	Botón Varios	OnClickVarios
Activación de Ventana de Detalles	detvideos		OnInitDialog

Tabla IV - Mapeo de Eventos Fig 3.4

Evento	Clase	Control	Evento de la clase
Cliente Selecionó Video	detvideos	ListBox Videos	OnSelChangeVideo
Presenta Detalle Video	detvideos	ListBox detVideos	MostrarDetalles
Presenta Foto del Video	detvideos	Frame Foto	OnPaint
Seleccionar Video	detvideos	Botón Carrito	OnClickCarrito
Consultar Otro Video	detvideos	ListBox Videos	OnSelChangeVideo
Ejecutar Video	detvideos	Botón Play	OnPlay
Presionó Stop	detvideos	Botón Stop	OnStop
Video Terminó	detvideos		OnDone
Regresar	detvideos	Botón Regresar	OnClickRegresar

Tabla V - Mapeo de Eventos Fig 3.5

Evento	Clase	Control	Evento de la clase
Selecciono Presentración CD	detvideos	Checkbox CD	OnCD
Selecciono Presentración Beta	detvideos	Checkbox Beta	OnBeta
Selecciono Presentración VHS	detvideos	Checkbox VHS	OnVHS
Seleciona Comprar	detvideos	Botón Comprar	OnClickComprar
Seleciona Alquilar	detvideos	Botón Alquilar	OnClickAlquilar
Seleciona Boton de Compras	detvideos	Botón Carrito	OnClickCarrito
Añadir Producto a la Lista de Compras	detvideos		AgregarItem
Consultar lista de Compras	carrito		OnIntitDialog
Consultar Otro Producto	detvideos	ListBox Videos	OnSelChangeVideo
Regresar	detvideos	Botón Regresar	OnClickRegresar

Tabla VI - Mapeo de Eventos Fig 3.12

Evento	Clase	Control	Evento de la clase
Activación de ventana de Consultas	carrito		OnInitCarrito
Muestra Productos Selecionados	carrito		OnInitDialog
Cliente Confirma o Elimina Productos	carrito	Spread Detalles	OnClickDetalles
Cliente Selecciona Aceptar	carrito	Botón Aceptar	OnClickAceptar
Cliente Selecciona Cancelar	carrito	Botón Cancelar	OnClickCancelar
Actualiza Base de Datos	carrito		ActualizarBase
Pregunta si Termino de Comprar	carrito		Preguntar
Cierra Factura	carrito		OnCancelar

Tabla VII - Mapeo de Eventos Fig 3.13

3.4. DISEÑO DE LA ESTRUCTURA Y COMPORTAMIENTO DE OBJETOS

3.4.1. DESCRIPCION DE OBJETOS

A continuación detallaremos las principales clases utilizadas en este sistema.

Clase CDatabase (derivada de CObject)

La clase **CDatabase** representa una conexión a una fuente de datos, permitiéndole operarla recuperando y almacenando información. Una fuente de datos es una instancia especifica provista por algún Sistema Manejador de Base de Datos (DBMS), que podría ser Microsoft SQL Server, Microsoft Access, Borland® dBASE®, and xBASE. Se puede tener uno o más objetos **CDatabase** activos al mismo tiempo en la aplicación. Las principales funciones utilizadas en el sistema son:

CDatabase().- Constructor de la clase, sólo crea el objeto, pero no establece ninguna conexión.

virtual BOOL Open(LPCSTR lpszDSN,	Nombre del origen de datos.
BOOL bExclusive = FALSE,	Indica si la base se abrirá en forma exclusiva o no.
BOOL bReadOnly = FALSE,	Indica si la base se abrirá sólo para lectura.
LPCSTR lpszConnect = "ODBC;",	Parámetros de conexión, serán utilizados por el ODBC.
BOOL bUseCursorLib = TRUE);	

virtual void Close().- Cierra la conexión con la base de datos, luego el mismo objeto puede utilizarse para abrir una nueva conexión incluso con otra fuente de datos.

void ExecuteSQL(LPCSTR lpszSQL).- Ejecuta en la base la sentencia SQL indicada en *lpszSQL*, esta puede ser un requerimiento directo o una llamada a un procedimiento almacenado.

Clase CAccesoDBSet (derivada de CRecordset)

Un objeto **CRecordset** representa un conjunto de registros seleccionados desde una fuente de datos (data source) físicamente almacena los datos en un archivo temporal. La clase CAccesoDBSet dinámicamente modifica su estructura permitiendo agregar o eliminar columnas de los registros, es utilizada para realizar las consultas a la base. Además de las funciones heredadas de CRecordset se utilizaron las siguientes:

CAccesoDBSet (CDatabase pDatabase = NULL)*.- Constructor de la clase, recibe *pDatabase* que es un puntero a una base de datos la cual previamente debió ser abierta.

CAccesoDBSet::~CAccesoDBSet().- Destructor, no retorna valores.

void AddTextField(CString& strName).- Agrega una columna cuyo nombre es *strName* a los registros.

*BOOL FindField(const char *szName)*.- Busca una columna llamada *szName* en el objeto, retorna VERDADERO si la encuentra, de otra manera FALSO.

CString CAccesoDBSet::GetValue(const char szName)*.- Obtiene el valor de el registro *szName*, retorna un objeto CString.

Clase CDib (derivada de CObject)

Esta clase es la que nos permite mostrar todas las imágenes en pantalla. Carga un archivo BMP o DIB en memoria y luego lo muestra según los parámetros proporcionados, las funciones utilizadas en el sistema fueron:

CDib().- Es el constructor estándar de la clase.

~CDib().- Destructor de la clase.

BOOL AttachMapFile(const char strPathname, BOOL bShare = FALSE)*.- Carga un archivo BMP en el objeto, el archivo se encuentra en la ruta *strPathname*, si el gráfico pudo ser cargado retorna VERDADERO, de lo contrario FALSO.

BOOL Draw(CDC pDC, CPoint origin, CSize size)*.- Pinta el gráfico en la pantalla, lo hace únicamente en el evento *OnPaint* o *OnDraw* de las ventanas, utiliza el dispositivo propio de la ventana, el gráfico se ubica en el punto *origin* con las dimensiones de *size*.

UINT UsePalette(CDC pDC, BOOL bBackground = FALSE)*.- Habilita el objeto para tomar la paleta de colores de la ventana o dispositivo donde será mostrado, si no se ejecuta antes de *DRAW* el gráfico no podrá ser mostrado correctamente.

Clase CMciCtrl (derivada de CWnd)

Esta clase representa a un control multimedia que permite grabar y reproducir este tipo de archivos en *"Media Control Interface (MCI) devices"*. Conceptualmente, este control es un conjunto de botones que usan comandos MCI. Esta clase suporta la reproduccion de archivos WAV, AVI, MID entre otros.

VENTANAS DE DIALOGOS

La clase **CDialog** es una clase básica usada para mostrar ventanas de diálogos en la pantalla. Existen dos tipos de ventanas: *modal* y *nomodal*. Una ventana *modal* debe ser cerrada por el usuario para continuar con la aplicación. Una *nomodal* permite mostrar el dialogo y continuar con otra tarea sin necesidad de cerrar o cancelar el diálogo.

Clase autores (derivada de Cdialog)

Es utilizada para mostrar los diferentes intérpretes, sus discos y las canciones de estos discos, para ello en su constructor recibe el tipo de búsqueda que debe realizar:

autores(CWnd pParent = NULL,*	Es la referencia a la ventana padre.
CString tipo="10001",	Indica el tipo de música seleccionado, se utiliza al llenar la lista de autores.
CString CADENA = "ODBC;",	Son los parámetros de conexión que utilizara el ODBC.
CString factura="0000",	Número de factura en la cual se registrarán los pedidos.
CString Title = "MUSICA")	Indica el titulo que debe mostrar la ventana de diálogo.

Al seleccionar un intérprete se muestran los productos de él y los temas. Permite agregar los discos seleccionados a la lista de compras.

Clase carrito (derivada de CDialog)

Muestra la lista de compras hecha por el usuario, el usuario puede indicar los productos que no desea, si sale aceptando los cambios, el objeto registra los cambios en la base, dando la opción de continuar agregando items o terminar la lista. El constructor de la clase recibe los parámetros de conexión y la factura

carrito(CWnd pParent = NULL,*	Es la referencia a la ventana padre.
CString CADENA = "ODBC;",	Son los parámetros de conexión que utilizara el ODBC
CString factura="0000",	Código de la factura cuyos artículos va a mostrar
BOOL Bandera = FALSE)	Bandera que indica si debe o no dar la opción de terminar con la lista de compras

Clase detjuegos (derivada de CDialog)

Es utilizada para mostrar los diferentes juegos disponibles, según el tipo seleccionado, para ello en su constructor recibe el tipo de búsqueda que debe realizar:

detjuegos(CWnd pParent = NULL,*	Es la referencia a la ventana padre.
CString tipo="10001",	Indica el tipo de juego.
CString CADENA = "ODBC;",	Son los parámetros de conexión que utilizará el ODBC.
CString factura="0000",	Indica el número de factura.
CString Title = "JUEGOS");	Es el título que deberá presentar la ventana.

Permite agregar juegos a la lista de compras.

Clase detvideo (derivada de CDialog)

Es utilizada para mostrar los diferentes videos disponibles, según el tipo seleccionado, para ello en su constructor recibe el tipo de búsqueda que debe realizar:

detvideo(CWnd pParent = NULL*	Es la referencia a la ventana padre.
CString tipo="10001",	Indica el tipo de video o pelicula a mostrar.
CString CADENA = "ODBC;",	Son los parámetros de conexión que utilizará el ODBC.
CString factura="0000",	Indica el número de factura.
CString Title = "VIDEOS");	Es el titulo que deberá presentar la ventana.

Permite agregar videos a la lista de compras.

Clase juegos (derivada de CDialog)

Brinda las opciones para seleccionar un determinado tipo de juego, según la decisión del usuario llama a un objeto tipo *detjuegos* para que nuestre los videos existentes.

juegos(CWnd pParent = NULL,*	Es la referencia a la ventana padre.
CString CADENA ="ODBC;",	Son los parámetros de conexión que utilizara el ODBC.
CString factura = "0000")	Indica el número de factura.

Clase musica (derivada de CDialog)

Es la que permite elegir el tipo de música, y los intérpretes deseados, según la elección hecha por el usuario llama a un objeto tipo *autores* para que muestre los intérpretes y sus canciones.

musica(CWnd pParent = NULL,*	Es la referencia a la ventana padre.
CString CADENA = "ODBC;",	Son los parámetros de conexión que utilizará el ODBC.
CString factura = "0000")	Indica el número de factura.

Clase Principal (derivada de CDialog **)**

Es la ventana inicial del sistema muestra las tres principales opciones: Videos, Juegos y Música, al momento de seleccionar cualquiera de ellas; si no hay factura, crea una y la transmite a los demás diálogos para que la utilicen. En esta también se puede dar por terminada la lista de compras.

Principal(CWnd pParent = NULL)*

Clase videos (derivada de CDialog **)**

Es la que permite elegir el tipo de video o película que se desea ver, según la elección hecha por el usuario llama a un objeto tipo *detvideos* para que muestre los productos correspondientes.

videos(CWnd pParent = NULL,*	Es la referencia a la ventana padre.
CString CADENA ="ODBC;",	Son los parámetros de conexión que utilizará el ODBC.
CString factura = "0000")	Indica el número de factura.

3.4.2. HERENCIAS DE ESTRUCTURAS Y METODOS

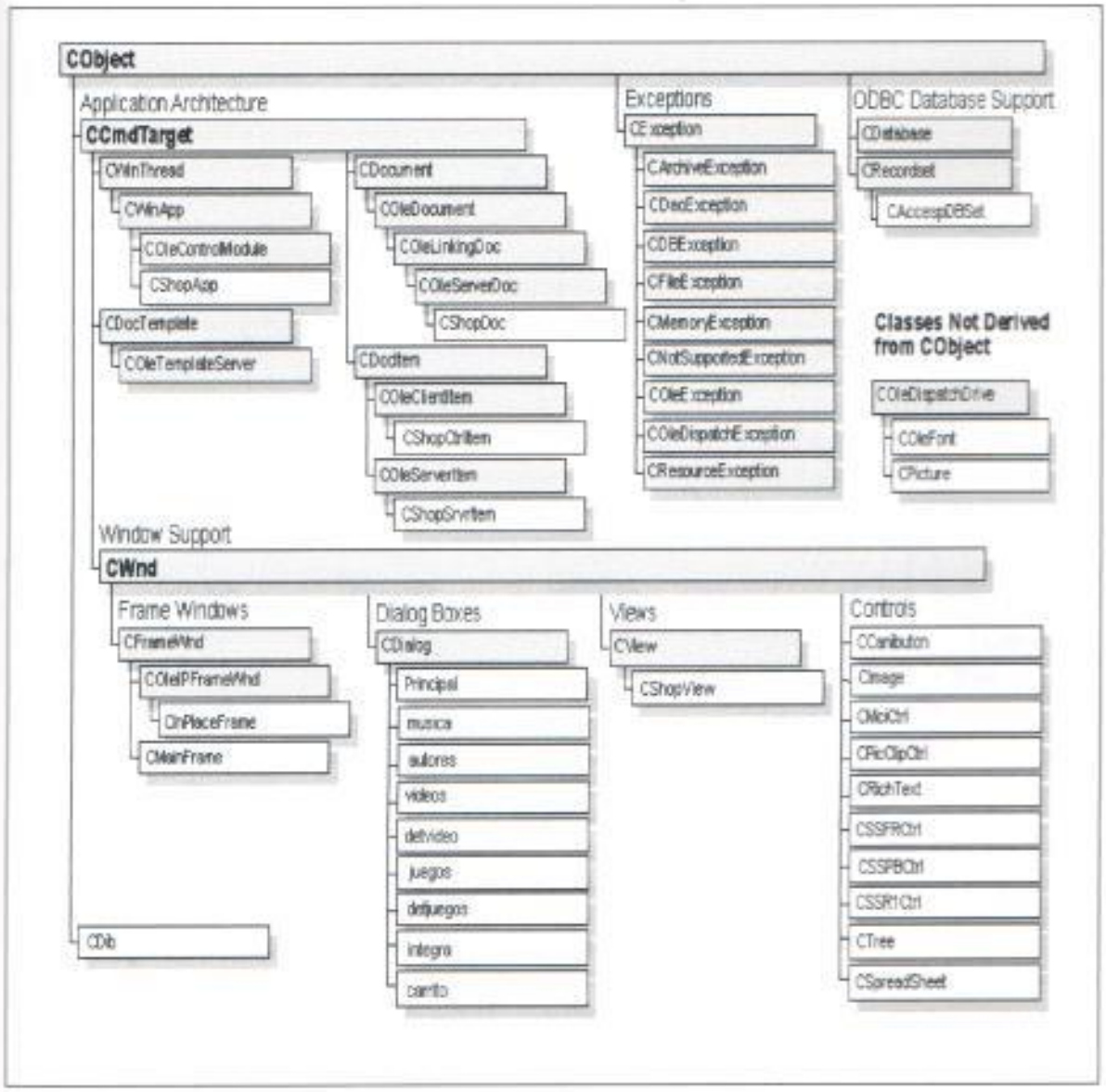


Figura 3.16 Herencias de Estructuras y Métodos de las Clases

3.5. DIAGRAMA DE JERARQUIA DEL SISTEMA

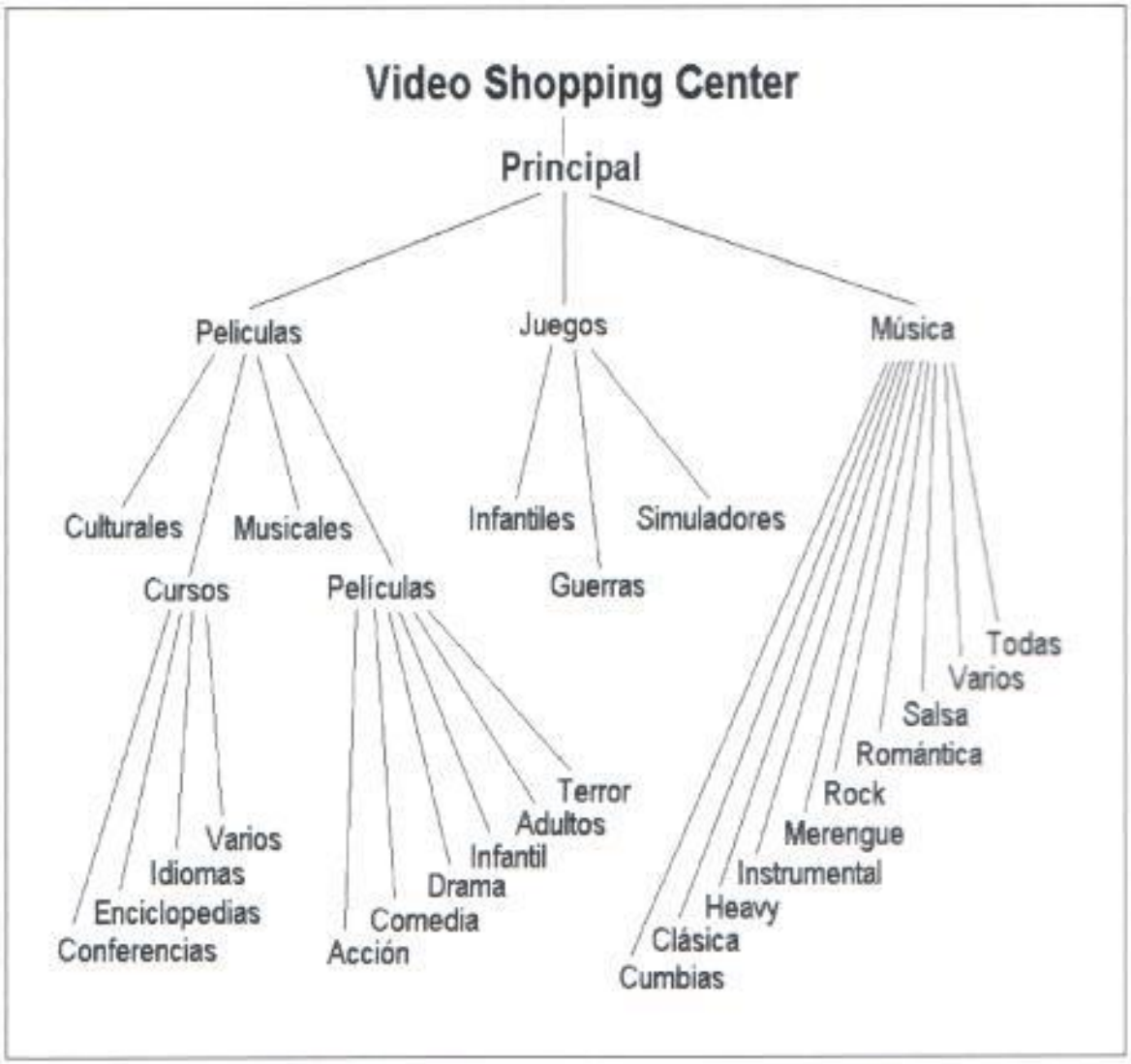


Figura 3.17 Diagrama de Jerarquía del Sistema

3.6. MODELO DE LA BASE DE DATOS

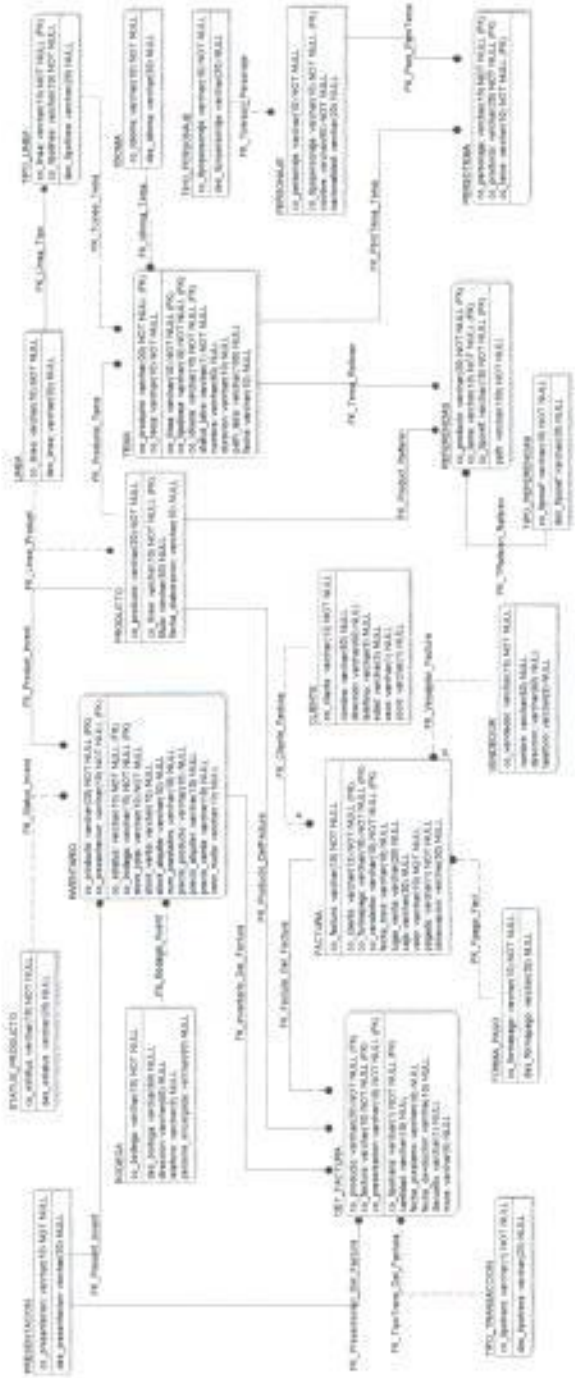


Figura 3.18 Modelo de la Base de Datos

3.7. DESCRIPCION DE OBJETOS DE LA BASE DE DATOS

3.7.1. DESCRIPCION DE TABLAS

CAMPO	TIPO DE DATO	TAMAÑO	NULL	DESCRIPCION
co_bodega	int	4		código único de la bodega
des_bodega	varchar	60	✓	nombre de la bodega
Direccion	varchar	60	✓	dirección de la bodega
Telefono	varchar	9	✓	número telefónico de bodega
Persona_encargada	varchar	60	✓	responsable

TABLA VIII - *BODEGA*

CAMPO	TIPO DE DATO	TAMAÑO	NULL	DESCRIPCION
co_cliente	int	4		código único del cliente
Nombre	varchar	60	✓	nombres y apellidos del cliente
Direccion	varchar	60	✓	dirección domiciliaria
Telefono	varchar	9	✓	teléfono de domicilio
Edad	varchar	3	✓	edad del cliente
Sexo	varchar	1	✓	sexo del cliente
Socio	varchar	1	✓	identificador de socio

TABLA IX - *CLIENTE*

CAMPO	TIPO DE DATO	TAMAÑO	NULL	DESCRIPCION
co_producto	int	4		código único de producto que se va a comprar o alquilar
co_factura	int	4		código único de factura
co_presentacion	int	4		código único de presentación del producto
co_tipotrans	int	4	✓	identificador de compra o alquiler
Cantidad	varchar	5	✓	cantidad de producto en factura
Fecha_prestamo	date		✓	fecha de préstamo (alquiler)
Fecha_devolucion	date		✓	fecha de devolución (alquiler)

Devuelto	varchar	1	✓	identificador de producto devuelto
Mora	number	2	✓	interés por mora en la devolución

TABLA X - *DET_FACTURA*

CAMPO	TIPO DE DATO	TAMAÑO	NULL	DESCRIPCION
co_factura	int	4		código único de la factura
co_cliente	int	4		código del cliente
co_formapago	int	4		código único de forma de pago
co_vendedor	int	4		código único del vendedor
Fecha_hora	date		✓	fecha y hora de emisión de factura
Lugar_venta	varchar	20	✓	almacén donde se emite factura
Caja	varchar	10	✓	caja donde se cobra la factura
Valor	varchar	10		monto de la factura
Pagado	varchar	1		status de cancelado o no
Observacion	varchar	30	✓	observaciones

TABLA XI - *FACTURA*

CAMPO	TIPO DE DATO	TAMAÑO	NULL	DESCRIPCION
co_formapago	int	4		código único de forma de pago
des_formapago	varchar	20	✓	descripción de forma de pago

TABLA XII - *FORMA_PAGO*

CAMPO	TIPO DE DATO	TAMAÑO	NULL	DESCRIPCION
co_idioma	int	4		código único de idioma
des_idioma	varchar	20	✓	descripción de idioma

TABLA XIII - *IDIOMA*

CAMPO	TIPO DE DATO	TAMAÑO	NULL	DESCRIPCION
co_producto	int	4		código único del producto
co_presentación	int	4		código único de presentación
co_estatus	int	4		código único de status del producto
co_bodega	int	4		código único de bodega
Stock_total	varchar	4		cantidad total existente del producto
Stock_venta	varchar	4	✓	cantidad de producto disponible para la venta
Stock_alquiler	varchar	4	✓	cantidad de producto disponible para alquiler
num_prestados	varchar	4	✓	cantidad de producto alquilado
Precio_producto	varchar	7	✓	precio de compra del producto
Precio_alquiler	varchar	7	✓	precio de alquiler del producto
Precio_venta	varchar	7	✓	precio de venta del producto
Valor_multa	varchar	4	✓	multa por no devolución

TABLA XIV - *INVENTARIO*

CAMPO	TIPO DE DATO	TAMAÑO	NULL	DESCRIPCION
co_linea	int	4		código único de línea
des_linea	varchar	20	✓	descripción de línea

TABLA XV - *LINEA*

CAMPO	TIPO DE DATO	TAMAÑO	NULL	DESCRIPCION
co_personaje	int	4		código único de personaje
co_tipopersonaje	int	4		código unico de tipo de personaje
nombre	varchar	60		nombre del personaje
nacionalidad	varchar	20	✓	nacionalidad

TABLA XVI - *PERSONAJE*

CAMPO	TIPO DE DATO	TAMAÑO	NULL	DESCRIPCION
co_personaje	int	4		código único de personaje
co_producto	int	4		código único de producto
co_tema	int	4		código único de tema

TABLA XVII *PERSOTEMA*

CAMPO	TIPO DE DATO	TAMAÑO	NULL	DESCRIPCION
co_presentacion	int	4		código único de presentación
des_presentacion	varchar	20	✓	descripcion de presentación

TABLA:XVIII.- *PRESENTACION*

CAMPO	TIPO DE DATO	TAMAÑO	NULL	DESCRIPCION
co_producto	int	4		código único del producto
co_linea	int	4		código único de línea
Titulo	varchar	50	✓	título del producto
Fecha_elaboración	date		✓	fecha de elaboración del producto

TABLA XIX - *PRODUCTO*

CAMPO	TIPO DE DATO	TAMAÑO	NULL	DESCRIPCION
co_producto	int	4		código único del producto
co_tema	int	4		código del tema
co_tiporef	int	4		código de tipo referencia
Path	varchar	100		ruta completa del archivo multimedia

TABLA XX - *REFERENCIAS*

CAMPO	TIPO DE DATO	TAMAÑO	NULL	DESCRIPCION
co_estatus	int	4		código único de status del producto
des_estatus	varchar	20	✓	descripción de status del producto

TABLA XXI - *STATUS_PRODUCTO*

CAMPO	TIPO DE DATO	TAMAÑO	NULL	DESCRIPCION
co_producto	int	4		código único del producto
co_tema	int	4		código único del tema
co_linea	int	4		código único de línea del producto
co_tipolinea	int	4		código único de tipolinea del producto
co_idioma	int			código único de idioma del tema
Status_letra	varchar	1		status de letra disponible o no
Nombre	varchar	80	✓	título del tema
Duración	varchar	10	✓	tiempo de duración en horas, minutos y segundos
path_letra	varchar	100	✓	ruta completa del archivo texto donde está la letra del tema
Fecha	date		✓	fecha de ingreso del tema

TABLA XXII - *TEMA*

CAMPO	TIPO DE DATO	TAMAÑO	NULL	DESCRIPCION
co_linea	int	4		código único de línea del producto
co_tipolinea	int	4		código de tipo de linea del producto
des_tipolinea	varchar	20	✓	descripción de las diferentes combinaciones de tipolinea

TABLA XXIII *TIPO_LINEA*

CAMPO	TIPO DE DATO	TAMAÑO	NULL	DESCRIPCION
co_tipopersonaje	int	4		código único del personaje
des_tipopersonaje	varchar	20	✓	descripción de tipos de personajes

TABLAXXIV - *TIPO_PERSONAJE*

CAMPO	TIPO DE DATO	TAMAÑO	NULL	DESCRIPCION
co_tiporef	int	4		código único de tipos de referencias al producto
des_tiporef	varchar	30	✓	descripción de las diferentes extensiones de los archivos que se referencian con el producto

TABLA XXV - *TIPO_REFERENCIAS*

CAMPO	TIPO DE DATO	TAMAÑO	NULL	DESCRIPCION
co_tipotrans	int	4		código único de tipo de transacción
des_tipotrans	varchar	20	✓	descripción de tipo de transacción

TABLA XXVI - *TIPO_TRANSACCION*

CAMPO	TIPO DE DATO	TAMAÑO	NULL	DESCRIPCION
co_vendedor	int	4		código del vendedor
Nombre	varchar	60	✓	nombre y apellidos del vendedor
Direccion	varchar	60	✓	dirección domiciliaria del vendedor
Telefono	varchar	9	✓	teléfono del domicilio del vendedor

TABLA XXVII - *VENDEDOR*

3.7.2. DESCRIPCION DE LLAVES PRIMARIAS Y FORANEAS

TABLA	NOMBRE DEL INDICE	CAMPOS	ATRIBUTOS
BODEGA	PK_BODEGA	co_bodega	unique key clustered
CLIENTE	PK_CLIENTE	co_cliente	unique key clustered
DET_FACTURA	PK_DETFACTURA	co_factura co_producto co_presentacion	unique key clustered
FACTURA	PK_FACTURA	co_factura	unique key clustered
FORMA_PAGO	PK_FORMAPAGO	co_formapago	unique key clustered
IDIOMA	PK_IDIOMA	co_idioma	unique key clustered
INVENTARIO	PK_INVENTARIO	co_producto co_presentacion	unique key clustered
LINEA	PK_LINEA	co_linea	unique key clustered
PERSONAJE	PK_PERSONAJE	co_personaje	unique key clustered
PERSOTEMA	PK_PERSOTEMA	co_personaje co_producto co_tema	unique key clustered
PRESENTACION	PK_PRESENTACION	co_presentacion	unique key clustered
PRODUCTO	PK_PRODUCTO	co_producto	unique key clustered
REFERENCIAS	PK_REFERENCIAS	co_producto co_tema co_tiporef	unique key clustered
STATUS_PRODUCTO	PK_STATUSPRODUC	co_estatus	unique key clustered
TEMA	PK_TEMA	co_producto co_tema	unique key clustered
TIPO_LINEA	PK_TIPOLINEA	co_tipolinea co_linea	unique key clustered

TIPO_PERSONAJE	PK_TIPOPERSONAJE	co_tipopersonaje	unique key clustered
TIPO_REFERENCIAS	PK_TIPOREF	co_tiporef	unique key clustered
TIPO_TRANSACCION	PK_TIPOTRANSAC	co_tipotrans	unique key clustered
VENDEDOR	PK_VENDEDOR	co_vendedor	unique key clustered

TABLA XXVIII - *ESTRUCTURAS*

TABLA	LLAVE FORANEA	CAMPOS	TABLA REFERENCIADA	CAMPOS REFERENCIADOS
DET_FACTURA	*FK_DETFACTURA_PRODUCTO*	co_producto	**PRODUCTO**	co_producto
	FK_DETFACTURA_PRESENTACION	co_presentacion	**PRESENTACION**	co_presentacion
	FK_DETFACTURA_TIPOTRANS	co_tipotrans	**TIPO_TRANSACCION**	co_tipotrans
FACTURA	*FK_FACTURA_FORMAPAGO*	co_formapago	**FORMA_PAGO**	co_formapago
INVENTARIO	*FK_INVENTARIO_BODEGA*	co_bodega	**BODEGA**	co_bodega
	FK_INVENTARIO_PRESENTACION	co_presentacion	**PRESENTACION**	co_presentacion
	FK_INVENTARIO_PRODUCTO	co_producto	**PRODUCTO**	co_producto
	FK_INVENTARIO_STATUSPRODUCTO	co_estatus	**STATUS_PRODUCTO**	co_estatus
PERSONAJE	*FK_PERSONAJE_TIPOPERSONAJE*	co_tipopersonaje	**TIPO_PERSONAJE**	co_tipopersonaje
PERSOTEMA	*FK_PERSOTEMA_TEMA*	co_producto co_tema	**TEMA**	co_producto co_tema
	FK_PERSOTEMA_PERSONAJE	co_personaje	**PERSONAJE**	co_personaje
PRODUCTO	*FK_PRODUCTO_LINEA*	co_linea	**LINEA**	co_linea
REFERENCIAS	*FK_REFERENCIAS_PRODUCTO*	co_producto	**PRODUCTO**	co_producto
	FK_REFERENCIAS_TIPOREFERENCIAS	co_tiporef	**TIPO_REFERENCIAS**	co_tiporef
	FK_REFERENCIAS_TEMA	co_producto co_tema	**TEMA**	co_producto co_tema
TEMA	*FK_TEMA_TIPOLINEA*	co_linea co_tipolinea	**TIPO_LINEA**	co_linea co_tipolinea
	FK_TEMA_IDIOMA	co_idioma	**IDIOMA**	co_idioma
	FK_TEMA_PRODUCTO	co_producto	**PRODUCTO**	co_producto
TIPO_LINEA	*FK_TIPOLINEA_LINEA*	co_linea	**LINEA**	co_linea

TABLA XXIX – *LLAVES FORANEAS*

TABLA	TRIGGER	ACCION
BODEGA	tD_BODEGA	No permite ***borrar*** un registro en la tabla BODEGA si es que existe un registro relacionado en la tabla INVENTARIO.
	tU_BODEGA	No permite ***actualizar co_bodega*** en la tabla BODEGA si es que existe un registro relacionado en la tabla INVENTARIO.
CLIENTE	tD_CLIENTE	No permite **borrar** un registro en la tabla CLIENTE si es que existe un registro relacionado en la tabla FACTURA.
	tU_CLIENTE	No permite ***actualizar co_cliente*** en la tabla CLIENTE si es que existe un registro relacionado en la tabla FACTURA.
FACTURA	tI_FACTURA	No permite ***insertar*** un registro en la tabla FACTURA si es que no existe un cliente relacionado en la tabla CLIENTE.
	tU_FACTURA	No permite ***actualizar co_cliente*** en ningún registro de la tabla FACTURA si es que no existe un cliente relacionado en la tabla CLIENTE.
FORMA_PAGO	tD_FORMA_PAGO	No permite ***borrar*** un registro en la tabla FORMA_PAGO si es que existe un registro relacionado en la tabla FACTURA.
	tU_FORMA_PAGO	No permite ***actualizar co_formapago*** en ningún registro de la tabla FORMA_PAGO si es que existe un registro relacionado en la tabla FACTURA.
IDIOMA	tD_IDIOMA	No permite ***borrar*** un registro en la tabla IDIOMA si es que existe un registro relacionado en la tabla TEMA.
	tU_IDIOMA	No permite ***actualizar co_idioma*** en ningún registro de la tabla IDIOMA si es que existe un registro relacionado en la tabla TEMA.
INVENTARIO	tI_INVENTARIO	No permite ***insertar*** un registro en la tabla INVENTARIO si es que no existe un status relacionado en la tabla STATUS_PRODUCTO.
	tU_INVENTARIO	No permite ***actualizar co_producto, co_presentacion, co_estatus, co_bodega*** en ningún registro de la tabla INVENTARIO si es que estos campos no se relacionan con las respectivas tablas.
LINEA	**tD_CLIENTE**	
PERSONAJE	tD_CLIENTE	

TABLA	TRIGGER	ACCION
PERSOTEMA	tD_CLIENTE	
PRESENTACION	tD_CLIENTE	
PRODUCTO	tD_CLIENTE	
REFERENCIAS	tD_CLIENTE	
STATUS_PRODUCTO	tD_CLIENTE	
TEMA	tD_CLIENTE	
TIPO_LINEA	tD_CLIENTE	
TIPO_PERSONAJE	tD_CLIENTE	
TIPO_REFERENCIAS	tD_CLIENTE	
TIPO_TRANSACCION	tD_CLIENTE	
VENDEDOR	tD_CLIENTE	

TABLA XXX – *TRIGGERS*

CAPITULO 4

PROBLEMAS DE IMPLEMENTACION

CAPITULO 4.

IMPLEMENTACION

En este capitulo se encuentran todos los problemas de implementación que tuvimos al programar nuestro sistema y las soluciones que aplicamos para cada caso.

Los problemas que se nos presentaron son:

1. Conectividad
2. Multimedia
3. Visual C++

4.1. Conectividad

Uno de los objetivos de nuestro proyecto era hacer una aplicación Cliente – Servidor, para lo cual se pensó en desarrollar una aplicación Cliente y una aplicación Servidora para poder accesar a los datos.

Al utilizar la tecnología ODBC para la conexión con la base de datos, ya no fue necesario desarrollar la aplicación servidora gracias a que el ODBC ya es Cliente-Servidor.

4.2. Problemas con Multimedia

El primer requerimiento de nuestro sistema es que sea una aplicación multimedia y por lo tanto que maneje videos y sonidos.

Pensamos que el manejo de la multimedia sería uno de nuestros mayores problemas, sin embargo encontramos dentro de la documentación de Visual Basic temas al respecto. Investigamos y probamos el control MCI (provisto por Microsoft), creando pequeñas

aplicaciones de prueba en Visual Basic y de esta manera apendimos a manejar dicho control para luego implementarlo en nuestra aplicación en Visual C++.

Con los controles MCI podemos manejar archivos en formato WAV, AVI, MIDI, que es lo que utilizamos para nuestra implementación.

Para alimentar la base de datos con la información necesaria para probar el sistema, recuperamos de Internet archivos de video con formato MOV, el cual no es reconocido por el control MCI.

La alternativa fue utilizar una aplicación que convierta de formato MOV a formato AVI. Esto se realizó con Adobe Premiere, pero al cambiar el formato se presentó el inconveniente de que el archivo de video aumenta su tamaño en bytes, lo que implica mayor capacidad de almacenamiento en disco duro, optando por no grabar el video completo, sino una pequeña muestra.

Debido al tamaño de los archivos multimedia decidimos tener un servidor de archivos.

Ahora el problema era que nuestro sistema pueda recuperar los archivos multimedia desde el servidor. Esto fue solucionado creando un campo de referencia en cada uno de los archivos, en el cual se graba la ruta (path).

Solucionado todos nuestros problemas, el último obstáculo con los videos fue que al presentarlo en nuestra aplicación no se posicionaba en un mismo lugar dentro de la pantalla.

La solución a esto fue presentar el video contenido en un frame, teniendo así el control de la posición de este.

4.3. Lenguaje Visual C++

La herramienta utilizada para desarrollar el sistema fue Visual C++, de la cual encontramos tres versiones diferentes: 1.52, 4.0 y 5.0.

Para poder seleccionar que versión de Visual C++ debiamos utilizar, estudiamos las caracteristicas de cada una, las mismas que presentamos a continuación:

Descripción	Ver. 1.52	Ver. 4.0	Ver 5.0
Comunicación entre aplicaciones	16 bits	32 bits	32 bits
Número de colores	16 colores	256 colores	256 colores
Tipo de Control	VBX	OCX	OCX
Navegador de Objetos	No existe	Sencillo	Complicado
Funciones de Controles MCI	Pocas	Muchas	Muchas
Tamaño del Lenguaje instalado	~ 70Mb	~ 150Mb	~ 300Mb

TABLA XXXI - Caracteristicas de las versiones de Visual C++

Considerando que nuestro sistema debía tener :

- Soporte para 32 bits.
- Mínimo 256 colores en gráficos y fotos.
- Funcionalidad en Control Multimedia.
- Herramienta que nos ayude en la programación.

La versión que mejor cumplía con nuestras necesidades fue la versión 4.0; aunque tuvimos problemas cuando sobrecargamos las pantallas con gráficos, ya que el Wizard no soportó el tamaño que se manejaba en el resource. Optamos por sacar todos los gráficos y cargarlos mediante programación.

CAPITULO 5

RECURSOS Y COSTOS

CAPITULO 5.

RECURSOS Y COSTOS

La implementación del sistema demandó la utilización de diferentes recursos tanto en software como en hardware.

A continuación mostraremos en detalle cada uno de los equipos e implementos utilizados por nuestro grupo en el desarrollo del sistema, así como también los equipos adicionales que podrían ser utilizados para mejorar tanto las presentaciones, el manejo de las herramientas y dar muchas más facilidades en el desarrollo del mismo.

5.1. Recursos utilizados para la Implementación

Los siguientes recursos fueron los utilizados tanto para el desarrollo del sistema como para las investigaciones preliminares del mismo.

• Servidor: COMPAQ PROSIGNIA 500	HD 2 Gb RAM 48 Mb CD-ROM FD 3.5" HD 3COM-Etherlink teclado mouse monitor 14" SVGA	$ 4.000
• IBM 8224	ETHERNET STACKABLE HUB 16 puertos	$ 1.000
• Cables de red UTP	5 cables	
• PC Compatible #1	PENTIUM 166 MHz MMX HD 2Gb RAM 32 Mb	$ 1.400

	Tarjeta Sonido Parlantes 40 Watt Microfono de condenso Modem/Fax 14.400 bps FD 3.5" HD 3COM-Etherlink teclado mouse monitor 14" SVGA	
• PC Compatible #2	PENTIUM 100 MHz HD 600 Mb RAM 16 Mb Tarjeta Sonido Modem/Fax 14.400 bps FD 3.5" HD 3COM-Etherlink teclado mouse monitor 14" SVGA	$ 1.000
• Tarjeta TV	Capturadora de video	$ 150
• PC Compatible #3	PENTIUM 120 MHz HD 1.2Gb RAM 24 Mb CD-ROM Tarjeta Sonido Parlantes 40 Watt Modem/Fax 33.600 bps FD 3.5" HD 3COM-Etherlink teclado mouse monitor 14" SVGA	$ 900
• PC Compatible #4	486 DX4 100 MHz HD 2.5 Gb RAM 32 Mb CD-ROM Modem/Fax/Voice 33.6Kbps	$ 1.000

	FD 3.5" HD 3COM-Etherlink teclado mouse monitor 14" SVGA	
• VHS Panasonic		$ 200
• UPS	250 KVA	$ 180
• REGULADOR	2 reguladores	$ 50x2 = 100
• SUPRESOR DE PICO	2 supresores	$ 10x2 = 20
• IMPRESORA	CANON BJC 250	$ 195
• Scanner	Pagina entera, color	$ 350
• WINDOWS NT SERVER	Versión 4.0	$ 700
• SQL SERVER	Versión 6.5	$ 900
• Visual C++	Versión 4.0 Enterprise	$ 600
• Adobe Premiere	Versión 1.1	$ 200
• QuickTime	Shareware	$ 0
• Software Capturadora		$ 180
• Internet	mensual	$ 100
• Linea Telefónica	45 horas = 2700 min	S/.59.400 ~ $ 15
• Luz Eléctrica	720 horas ~ 2.500 KVA	S/.125.000 ~ $ 30
	TOTAL	**$ 12.240**

TABLA XXXII - Recursos utilizados para la implementación

5.2. Recursos mínimos necesarios para la Implantación del Kiosko Multimedia

Para que el sistema funcione de una manera correcta y adecuada se deberán considerar los siguientes recursos.

• Servidor: COMPAQ PROSIGNIA 500	HD 2 Gb RAM 64 Mb CD-ROM 24x FD 3.5" HD 3COM-Etherlink teclado mouse monitor 14" SVGA	$ 4.000
• ARRAY de discos	5 discos de 2 Gb.Clase 'A'	$ 700x5 = 3.500
• IBM 8224	ETHERNET STACKABLE HUB 16 puertos	$ 350
• Cables de red UTP	10 Base T	
• PC Cliente (para consulta)	PENTIUM 166 MHz MMX HD 2Gb RAM 32 Mb Tarjeta Sonido Parlantes 40 Watt FD 3.5" HD 3COM-Etherlink teclado mouse	$ 900
Monitor Touch Screen	14" SVGA	$ 1.800
• PC Cliente (para la caja)	PENTIUM 166 MHz MMX HD 2Gb RAM 32 Mb FD 3.5" HD 3COM-Etherlink teclado	$ 1.300

	mouse Monitor 14" SVGA	
• UPS	1.500 KVA	$ 700
• REGULADOR	3 reguladores	$ 50x3 = 150
• SUPRESOR DE PICO	2 supresores	$ 10x2 = 20
• IMPRESORA	CANON BJC 250	$ 195
• WINDOWS NT SERVER	Versión 4.0	$ 700
• SQL SERVER	Versión 6.5	$ 900
	TOTAL	**$ 14.515**

TABLA XXXIII - Recursos mínimos necesarios para la implementación

5.3. Recursos adicionales requeridos tanto para la administración del sistema, el ingreso de datos, digitalización de productos y mantenimiento de las referencias.

Los recursos adicionales que se plantean aquí son los necesarios para poder realizar el ingreso de datos y la digitalización de los videos, música, peliculas, etc.

• Tarjeta TV	Capturadora de video	$ 150
• Scanner	Pagina entera, color	$ 350
• Mini Camara de video	Cámara conección serial	$ 150
• Adobe Premiere	Versión 1.1	$ 200
• QuickTime	Shareware	$ 0
• Software Capturadora		$ 180
• Internet	mensual	$ 100
• Software graba CD's		$ 80
	TOTAL	**$ 1.200**

TABLA XXXIV - Recursos adicionales requeridos para la administración

Al momento de digitalizar sus productos, recuerde lo siguiente:

- Grabar 30 segundos de una canción utilizando 16 bits para los datos y con sonido en estéreo ocupa aproximadamente 3 Mb de espacio físico en el disco. Si grabáramos una canción entera de 4 minutos de duración, ocuparía aproximadamente 24 Mb de espacio.

- Cada video capturado con el capturador de Videos con unos 30 segundos de duración ocupa aproximadamente 40 Mb de espacio en disco.

Estas son dos razones por las cuales se pide dentro de los recursos requeridos que se prepare un servidor con un arreglo de discos con 10 Gb de almacenamiento. Esto puede cambiar dependiendo de la forma de grabar los productos y de la cantidad de productos que se tengan o se quieran presentar.

5.4. Costo del Sistema Multimedia, Video Shopping Center

Nuestro sistema multimedia, se entregaría con el siguiente material.

• CD Instalador	- Instalador del sistema. - Instalador de los drivers necesarios para la aplicación. - Instalador de ODBC. - Demo de la aplicación. - Help para el manejo de la aplicación. - Manual para el usuario final. - Manual para el administrador del sistema. - Manual de errores. - Guía de instalación del sistema. - Guía para el ingreso de datos y la digitalización.	$ 9.500
• Manuales escritos	- Manual para el usuario final. - Manual para el administrador del sistema. - Manual de errores. - Guía de instalación del sistema. - Guía para el ingreso de datos y la digitalización	$ 300
	TOTAL	**$ 9.800**

TABLA XXXV – Costo del Sistema Video Shopping Center

5.5. Recomendaciones para la Instalación de la Red

Para la instalación de la red se recomienda los siguiente:

Si usted tiene un almacén pequeño y no desea hacer una inversión grande puede optar por poner una sola computadora cliente, Kiosco Multimedia, en la cual sus clientes podrán realizar cualquier tipo de consultas sobre los productos disponibles en el almacén utilizando tan solo la pantalla sensible del monitor. Esta computadora debe tener la capacidad de soportar multimedia, tener buen cache de video y buena resolución.

Si usted puede invertir un poco más y la clientela también lo demanda, puede optar por implantar dos o tres computadoras con pantalla sensible distribuidas en el almacén de tal manera que cubran la demanda y áreas estratégicas del almacén.

En la caja, se necesita otra computadora cliente. Esta sólo será para el cobro de los productos que se compran o alquilan. Para un almacén pequeño puede cubrirse con una sola caja, que tal vez esté cerca de la puerta de salida. Mientras que para un almacén un poco más grande puede recomendarse poner más de una caja, con lo cual también cubre la demanda.

El equipo central al cual se conectan y del cual traen información de los diferentes productos las computadoras clientes, debe estar en un cuarto aislado, debe tener buena capacidad de disco para almacenamiento y cache, tener buena velocidad de respuesta de procesador y de disco duro y un buen cache en video.

Estos equipos se conectan entre si por medio de un hub a puntos de red en cada una de las máquinas.

CONCLUSIONES

Y

RECOMENDACIONES

CONCLUSIONES

- De todas las etapas del ciclo de desarrollo del proyecto, la que requirió más esfuerzo fue la implementación en programación orientada a objetos. Porque la herramienta utilizada Visual C++ es compleja y requiere que el programador conozca la teoria de la POO y amplios conocimientos de la arquitectura de Windows para poder implementar una aplicación que interactúe con éste. La implementación hubiera sido más sencilla utilizando una herramienta RAD más completa como Visual Basic, Power Builder, entre otras.
- La POO es mucho más real que la Programación Estructurada. El mundo real se puede interpretar mejor como un grupo de objetos relacionándose entre si, que como una secuencia lógica de actividades.
- El DOO y la POO se justifican en sistemas grandes, en cosas pequeñas el diseño y la programación estructurada son suficientes.
- Para que el sistema sea comercializado deberá ir acompañado de un estudio de mercado que le permita al cliente analizar los costos y los beneficios que podria obtener con el suto del sistema.
- En nuestro pais se ha explotado muy poco los recursos multimedia y no existen sistemas con las caracteristicas del nuestro, lo cual lo hace novedoso y por lo tanto comercializable.
- El sistema es portable en la parte servidora porque el modelo de la base de datos fue creado utilizando una herramienta Case, la misma que puede generar los scipts de creación de dicha base de datos para cualquier RDBMS.

- Con el desarrollo de este sistema pudimos poner en práctica los conocimientos teóricos adquiridos de análisis y diseño orientado a objetos, y profundizar en el manejo de la herramienta Visual C++.
- Por las características de nuestro sistema los requerimientos de administración de la base de datos son mínimos.

RECOMENDACIONES

- Debido a que los archivos multimedia residen en una sola máquina (servidor), debemos transmitir muy frecuentemente archivos de tamaños considerables, por lo que hay que tener en cuenta el número de usuarios concurrentes para poder hacer un estudio del tráfico en la red y elegir los equpos y medios de transmisión adecuados.
- Cuantificar el crecimiento de la base de datos para escoger la capacidad óptima del disco duro y determinar la frecuencia con que se respaldan los datos.
- El tratamiento de imágenes, sonido y especialmente videos ocupa demasiado espacio en disco, por lo que se recomienda tener una política para la digitalización de estos objetos, almacenando sólo apenas pequeñas muestras de cada producto.

  Si se desea mantener una digitalización completa de los objetos necesitará una gran inversión en capacidad de almacenamiento.
- Un gran impacto en la reducción del costo del sistema es la elección de licencias más económicas tanto para las base de datos como para el cliente.

BIBLIOGRAFIA

1. CEVALLOS, FRANCISCO JAVIER, Enciclopedia del Lenguaje C, Addison-Wesley, Madrid, 1993.

2. FAIRLEY, RICHARD, Ingeniería de Software, McGraw-Hill, 1988.

3. GUREWICH, ORI y NATHAN, Teach Yourself Visual C++ 4 in 21 Days, SAMS Publishing, United States, 1996.

4. JAMSA, KRIS y COPE, KEN, Programación en Internet, McGraw-Hill, México, 1996.

5. JOYANES AGUILAR, LUIS, Microsoft C/C++, McGraw-Hill, Madrid, 1994.

6. KENDALL Y KENDALL, Análisis y Diseño de Sistemas, Prentice-Hall, México de 1991.

7. KELLEY/POHL, Lenguaje C Introducción a la Programación, Addison-Wesley, 1987.

8. KRUGLINSKI, DAVID J., Programación Avanzada con Visual C++, McGraw-Hill, Madrid, 1996.

9. MARTIN, JAMES y ODELL, JAMES J., Análisis y Diseño Orientado a Objetos, Prentice Hall, México, 1994.

10. MICROSOFT, Implementing a Database Design on Microsoft SQL Server 6.5, Microsoft Corporation, United States, 1996.

11. MICROSOFT, Microsoft Visual C++, Microsoft Corporation, United States, 1995, Volumen 1 al 3.

12. PETZOLD, CHARLES, Programación en Windows 95, McGraw-Hill, Madrid, 1996.

13. VAUGHAN, TAY, Todo El Poder de Multimedia, McGraw-Hill, Segunda Edición, México, 1995.

14. WHITING, BILL, MORGAN, BRYAN y PERKINS, JEFF, Teach yourself ODBC Programming in 21 days, SAMS Publishing, United States, 1996.