ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"Sistema de Renta de Videos"

TESIS DE GRADO

Previa a la obtención del título de:

LICENCIADO EN SISTEMAS DE INFORMACIÓN

Presentada por:

Anl. Enrique Briones Cornejo
Anl. Wilson Iñiguez Beltrán
Anl. José Merchán Guaranda

Director de Tesis:

Ing. Carlos Martín Barreiro, MSIG

GUAYAQUIL – ECUADOR
AÑO 2002

AGRADECIMIENTO

A Dios ... por su amor infinito,

A nuestros Padres ... por su amor incondicional,

A nuestras familias ... por su soporte y sacrificio,

A la ESPOL ... por todo lo que representa en nuestra formación,

A nuestros amigos ... por su apoyo en el desarrollo de este proyecto....

¡Muchas Gracias!

DEDICATORIA

Este trabajo está dedicado a nuestros padres, como una pequeña retribución a su enorme sacrificio desinteresado y como un justo premio a sus más preciados sueños y anhelos.

Con este trabajo rendimos homenaje al esfuerzo del profesional ecuatoriano que pese a todas las dificultades que encuentra en su labor, puede llevar a cabo los proyectos más complicados en base al talento, esfuerzo y la perseverancia.

TRIBUNAL DE GRADUACIÓN

Ing. Mónica Villavicencio
Coordinadora

Ing. Carlos Martín, MSIG
Director de Tesis

MIEMBROS PRINCIPALES

Ing. Néstor Arreaga

Lic. Jorge Olaya, MBA

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de esta Tesis de Grado, nos corresponden exclusivamente, y el patrimonio intelectual de la misma a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL"

Anl. Enrique Briones

Anl. Wilson Iñiguez

Anl. José Merchán

RESUMEN EJECUTIVO

El presente trabajo desarrolla el análisis, diseño e implementación de un Sistema de Renta de Videos. Las dos principales características tecnológicas de este proyecto son de que, en primer lugar aprovecha la tecnología de Internet para ofrecer un sitio web en donde los clientes pueden acceder al servicio de alquiler de videos. En segundo lugar, se escogió el modelo cliente/servidor de 3 capas por sus múltiples ventajas. Este documento describe detalladamente todo el proceso de implementación del sistema.

En la primera parte se revisan los objetivos del sistema, las especificaciones funcionales y no funcionales, el alcance y las restricciones y limitaciones del mismo.

Luego se analiza el negocio, su situación en el mercado, los competidores, productos sustitutos, y las oportunidades que brinda el mercado virtual.

A continuación se analiza el negocio virtual y los procesos principales del negocio, como son el de entrega y el de cobros. Aquí se analizan los riesgos que tienen los negocios por Internet y cómo se minimizan por medio de políticas y procedimientos eficaces.

Luego viene una descripción de todo el software empleado, así como las tecnologías que se usaron. Además se hacen las recomendaciones en cuanto al hardware y software necesarios para la puesta en producción del sistema.

A continuación se describe la base de datos de la aplicación, desde su origen conceptual hasta la implementación final.

Posteriormente se describe todos los objetos de la capa de negocios. Se detallan los componentes de acceso a datos y de negocios y se da una breve explicación del funcionamiento general de los componentes. También se incluye el código fuente de las interfaces usadas en el desarrollo de los componentes de negocios y de acceso a datos.

Luego se describen los programas que interactúan con el usuario final, tanto los usuarios del sitio web como los de la aplicación de uso interno.

En la parte final del documento se encuentran unos anexos de lectura recomendada, que explican algunas de las tecnologías que se usaron en el desarrollo de este proyecto, tales como, conceptos de cliente/servidor, infraestructura COM/COM+ y programación en la web.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE CUADROS

INTRODUCCIÓN

Ecuacinema es una empresa innovadora en la actividad de renta de videos, es la única que ofrece el servicio de entrega a domicilio, gracias a que cuenta con un eficiente servicio de entrega. Apoyados en esta diferencia competitiva, es que desea ampliar su cobertura a más sectores de la ciudad, y basados en el alto incremento del uso del Internet en el hogar, comercio e industria; nosotros hemos innovado en el país al ofrecer este servicio de renta de videos mediante un sitio Web.

Ofreciendo a nuestros clientes un sitio en el cual se sienta seguro y a gusto del servicio de renta que esta adquiriendo, ahorrándole tiempo y dinero.

1

ESPECIFICACIONES DEL PROYECTO

1. ESPECIFICACIONES DEL PROYECTO

1.1. Justificación

La necesidad de ampliar la cobertura geográfica del servicio de renta de videos, sin ser necesario abrir nuevos locales, es el reto que nos propuso la empresa Ecuacinema. Esta solicitud nosotros la resolvimos implementando un sitio Web llamado www.ecuacinema.com.

Entre los servicios que ofrece nuestro sitio tenemos:

- Consultar la más amplia videoteca de películas de actualidad, como también aquellas que nunca pasan de moda.
- Adquirir talonarios con un cupo determinado de películas a alquilar.
- Alquilar las películas y solicitar el servicio de entrega y retiro a domicilio.
- Ofrecer un servicio de calidad enmarcado en el respeto a la autoría cinematográfica.

1.2. Objetivos Generales

Nuestros objetivos generales son:

- Incrementar el número de clientes, aprovechando la globalización del uso del Internet en los hogares.

- Satisfacer a los clientes en su derecho de haber pagado sólo por las películas que vaya a rentar y no atarlo a mensualidades fijas.

- Asegurar a nuestros clientes que los medios que reciben pasan por un estricto control de calidad, evitando así momentos desagradables al observar una película deteriorada.

1.3. Especificaciones funcionales

El servicio que ofrece Ecuacinema, es la renta de video y para ello debemos tener la facilidad de registrar cierta información que a continuación nombramos:

- **Clientes.-** Tipo de identificación (cédula ó RUC), número de identificación, nombres, apellidos, sector de la ciudad donde reside, dirección domiciliaria, teléfono, e-mail, forma de pago (efectivo ó tarjeta de crédito), si eligió tarjeta de crédito debe incluir el emisor de la tarjeta, número y fecha de vencimiento. Adicionalmente se le permite seleccionar los servicios de entrega y retiro de películas a domicilio, como también el login de acceso al sitio web de renta de videos.

- **Sectores.-** Nos debe permitir registrar los sectores urbanos de la ciudad (norte, centro, sur, etc), que nos permite identificar el sector domiciliario del cliente.

- **Tarjetas de crédito.-** Debe permitirnos registrar los diferentes emisores de tarjetas de crédito que la empresa acepte a sus clientes, por ejemplo, Diners, MasterCard, American Express, etc.

- **Títulos de películas.-** Nombre original de la película, nombre de la película en español, sinopsis de la película, productor, director, reparto, censura, tipos de géneros (máximo hasta tres), identificar si la película ha sido nominada y registrar la referencia de esa nominación, identificar si la película ha sido premiada y registrar la referencia de esa premiación, identificar si ha ganado el Oscar, registrar su tiempo de duración (hh:mm), el año de producción de la película y por último la ruta en donde almacenaremos un archivo de imagen de la película.

- **Ítems.-** Los ítems van a representar los diferentes medios físicos en donde se encuentra grabada la película, y entre los datos que debe contener su registro tenemos: Un código único, título de la película, el tipo de medio (VHS ó DVD), idioma en que está editada la película, estado de renta y estado del registro (activo, inactivo). Básicamente en el estado de renta, vamos a manejar dos estados que son:

Disponible, que indica que la película puede ser alquilada; Alquilada, cuando la película ya ha sido rentada por el cliente. Este registro único de ítem nos permite cumplir con el requisito de tener más de un medio con el mismo título.

- **Idiomas.-** Que nos permita registrar todos los idiomas en que vienen editadas las películas.

- **Tipos de talonarios.-** Debido a que la empresa utiliza un medio de prepago basado en la adquisición de un talonario, se debe permitir registrar los tipos de talonarios, que deben registrar información como: Código del tipo de talonario, descripción, cupo de películas que ofrece y el precio.

- **Segmentos de información para publicar en el sitio web.**- Para mantener una información actualizada en línea con el sitio Web que se va a desarrollar se necesita registrar dos grupos de información. La primera que indica las categorías como: Estrenos, Noticias, Cine Latino. Y el segundo grupo, que nos permita registrar el detalle de esas categorías, como por ejemplo, si estamos ingresando un detalle de noticias ésta se asocie a la categoría de Noticias y por ende salga publicado en el sitio Web en el segmento asignado a las Noticias.

- **Pre-Registro de clientes en el sitio web.-** Como vamos a tener un sitio web, este debe permitir registrar potenciales clientes, en donde

se le debe registrar información como: Tipo de identificación (cédula ó RUC), número de identificación, nombres, apellidos, sector de la ciudad donde reside, dirección domiciliaria, teléfono, e-mail, forma de pago (efectivo ó tarjeta de crédito), si eligió tarjeta de crédito debe incluir el emisor de la tarjeta, nombre del titular, numero y fecha de vencimiento.

- **Evaluación de solicitudes.**- El sistema nos debe presentar una pantalla en la cual, se apruebe o rechace el pre-registro de clientes. Si la solicitud es aceptada, este pre-registro debe pasar automáticamente a formar parte de nuestra base de datos de clientes.

- **Compra de talonarios.-** Se debe permitir a los clientes realizar la compra de talonarios, en donde se debe mostrar los distintos tipo de talonarios que tenemos registrados, con su cupo y valor. El cliente debe seleccionar el que desee y al registrar esta compra, se le asigna un código único de talonario. El talonario que adquiere el cliente, es el documento con el cual tendrá posibilidad de rentar las películas que desee hasta agotar el cupo asignado.

- **Alquiler de películas.-** El sitio web, debe contener un segmento en el cual se permita al cliente realizar el alquiler de películas siempre y cuando tenga disponible películas en el(los) talonario(s) que haya

adquirido. Cada vez que se alquile una película, se debe disminuir un cupo en el talonario del cliente. Además, se debe ofrecer al cliente los servicios de entrega y/o retiro a domicilio de las películas seleccionadas.

- **Entrega de películas.**- El sistema debe permitir emitir un reporte el mismo que indique todos los clientes que han deseado que se les entregue las películas a domicilio. Y una vez que nuestra empresa de servicio haya realizado la entrega, el sistema debe también registrar la notificación que ha sido entregada la película.

- **Recibo de películas.**- El sistema debe permitir emitir un reporte el mismo que indique todos los clientes que han deseado que se les retire las películas a domicilio. Y una vez que nuestra empresa de servicio haya realizado el retiro, el sistema debe también registrar la notificación que ha sido recibida la película. Esta confirmación de recibir la película nos permite poner el ítem de la película nuevamente con estado de disponible.

1.4. Especificaciones no funcionales

Apoyados en la necesidad de la empresa Ecucinema, que desea captar a los usuarios cotidianos del Internet, y estando conscientes del crecimiento de la información que esto involucra, establecemos la necesidad de implementar una aplicación de tres capas.

La primera capa, es la capa de presentación que va ser expuesta en dos tipos de aplicaciones de usuario, que son:

- **Aplicación Cliente / Servidor**.- Que va ser dirigida al personal operativo de Ecuacinema, en donde vamos a permitir registrar información como: Clientes, Sectores, Tarjetas de Crédito, Títulos de películas, Ítems, Idiomas, Tipos de Talón, Segmentos para publicar en el sitio Web, Evaluación de solicitudes, Confirmación de Entrega y Recibo de Películas.

- **Aplicación Web**.- Que va ser dirigida a los usuarios del Internet. En esta aplicación vamos a permitir realizar las siguientes operaciones: Pre-Registro de posibles clientes, Consulta de Películas, Registro de clientes aprobados (con su login y password), Compra de Talonarios con su respectivo pago con tarjeta de crédito, Alquiler de Películas.

En la segunda capa, vamos implementar el uso de la tecnología COM en el desarrollo de los componentes de negocios y de datos. Aquí aplicaremos la definición de Interfaces utilizando el lenguaje IDL, con lo que obtendremos sus "Type Libraries", que serán referenciadas por la implementación de los componentes de datos.

Una vez obtenidos los componentes de datos, de estos tomaremos sus "Type Libraries", que serán referenciadas por la implementación de los componentes de negocios.

La tercera capa, estará representada por el proceso servidor de base de datos.

1.5. Limitaciones y Restricciones

1.5.1. Limitaciones.

Entre las limitaciones que el sistema establece, podemos citar las siguientes:

- Va a tener solamente dos formas de pagos, que son: Efectivo y Tarjeta de Crédito
- En el tipo de identificación de los clientes, se dispone de dos tipos que son C = Cédula y R = RUC.
- Se va a administrar dos tipos de medio de grabación, que son VHS y DVD.
- En la aplicación Web, solo se permite realizar compra de talonarios y alquiler de películas, a los clientes ya aprobados. Aquellos que han enviado su solicitud (Pre-Registro) y que están en estado de ingresada y más aún que tenga estado de rechazada no se les permite realizar dichas operaciones.
- El cliente no podrá alquilar películas si no posee un talonario con cupo disponible.

1.5.2. Restricciones.

Entre las restricciones que el sistema establece, podemos citar las siguientes:

- Que si después del análisis de la solicitud del Pre-Registro, se determina que el candidato a cliente tiene inconsistencias como por ejemplo, nombres no válidos, direcciones no existentes, etc. Se procederá a rechazar su solicitud.

- Que se emita un decreto gubernamental que indique que ya no se pueda realizar transacciones monetarias en el Internet, esto ocasionaría que desactivemos nuestra venta de talonarios en el sitio Web de Ecuacinema.

1.6. Alcance.

El Sistema de Renta de Videos, será desarrollado en dos tipos de aplicaciones que son:

1.6.1. Aplicación Cliente / Servidor.

Esta aplicación será dirigida a los usuarios operativos de Ecuacinema. Y para una mayor comprensión se la ha dividido en tres menús con las siguientes opciones.

- **Menú de Registro de Datos.-** Aquí se registrará información de: Clientes, Títulos de Películas, Ítems, Sectores, Tarjetas de Crédito,

Idiomas, Categorías y Detalles para publicación en el sitio Web, Tipos de Talón, Parámetros de la empresa y Secuenciales de registros.

- **Menú de Transacciones.-** Aquí se registrarán las transacciones de: Entrega de Películas, Recibo de Películas y Evaluación de Solicitudes.

- **Menú de Consultas / Reportes.-** Este menú nos permite emitir reportes: De Clientes, como Solicitudes Nuevas, Clientes Nuevos e Información detallada del cliente. De Transacciones, como: Videos por Entregar, Videos por Recoger, Videos Pendientes de Devolución y Videos Rentados.

1.6.2. Aplicación Web.

Esta aplicación está dirigida a los usuarios de Internet, que ya sean nuestros clientes, así como también para los que deseen serlo. Además no es necesario que sea cliente para poder consultar las películas que forman parte de la gran videoteca de Ecuacinema, es decir, este sitio web será nuestra vitrina virtual.

Entre las operaciones que se debe permitir en la aplicación Web, mencionamos las siguientes:

- **Búsqueda de Películas.-** En esta operación se le debe permitir buscar películas con los siguientes criterios de búsqueda, como: tipo de medio (VHS ó DVD), título de la película, nombre del director,

nombres de los actores que conforman el reparto, para ello se le debe permitir ingresar un argumento del cual se basará para realizar la búsqueda.

- **Login de acceso de clientes.-** Una vez que el cliente es aceptado, este podrá hacer uso de su login y password, los mismos que le servirán para poder realizar las operaciones de comprar talonarios y alquilar películas.

- **Informativo del cupo y posesión de películas por parte del cliente.-** Cada vez que el cliente ingrese con su login y password, la aplicación debe mostrarle en el segmento de Login su cupo de películas disponibles de acuerdo a los talonarios que haya adquirido, como también debe informarle al usuario las películas que tiene en su poder.

- **Compra de talonarios.-** Se le debe permitir a los usuarios clientes de la empresa realizar la operación de compra de talonarios, para tal efecto se le debe mostrar los tipos de talón que ofrece la empresa, indicando explícitamente el cupo de películas y el precio. Además en esta operación, el pago del talonario es con tarjeta de crédito. Cada vez que el cliente adquiere un talonario, éste ya tiene cupo de películas para poder realizar el alquiler de películas.

- **Alquiler de películas.-** Se le debe permitir a los usuarios clientes de la empresa realizar la operación de alquilar películas, para ello se le permitir realizar la búsqueda de las películas que desee el cliente y una vez que se las muestra debe haber una forma en donde las pueda seleccionar y quitar la selección, antes de enviar el requerimiento de alquiler. Debemos acotar que si cliente ya envió su requerimiento de alquiler, éste queda procesado, es decir, de su talonario se rebaja el número de ítems ó películas alquiladas.

2

ANÁLISIS DE LA INDUSTRIA Y EL PRODUCTO

2. ANÁLISIS DE LA EMPRESA Y EL PRODUCTO.

2.1. Definición de la Empresa y el Producto.

Ecuacinema, es una empresa que se encuentra en el mercado de alquiler de películas desde 1997. La empresa se encuentra localizada en la ciudadela Entre Ríos, siendo su tipo de clientes de la clase media-alta.

Podemos citar que su videoteca almacena alrededor de 5.000 ítems, con lo que se le asegura al cliente una variedad de títulos a su disposición.

La empresa cuenta con una servicio de entrega que es llevada a cabo por la empresa Servientrega, de esta manera se convierte en un socio estratégico de nuestro negocio.

El servicio de alquiler de películas, es otorgado a sus clientes una vez que él adquiera un talonario de películas, el mismo que estará asociado a un cupo de películas determinado. El cliente podrá seguir adquiriendo los talonarios en el local y con la nueva implementación lo podrá realizar en el sitio Web.

2.2. Clientes

En la actualidad nuestros clientes se acercan al local ó nos llaman por teléfono para solicitar nuestro servicio de alquiler. Si los graficamos

geográficamente nuestros clientes se encuentran localizados en un radio de 15 cuadras alrededor del local.

Es esta limitación que deseamos romper y llegar a más clientes, ayudándonos con esa vitrina virtual que es el Internet al presentar nuestro sitio, el cliente sólo navegaría por nuestra videoteca y seleccionará la película que desee.

2.2.1. Comportamiento del consumidor ante el mercado convencional

En el mercado convencional, existe un alto porcentaje de locales que están al margen de la ley, debido a que ellas no poseen autorizaciones expresas de las autorías de las películas que ofrecen en alquiler. Esta falta de legalidad, es cubierta por los bajos precios de alquiler por película que ellos ofrecen, y es ése detalle que atrae a ciertas personas.

Algo que podemos resaltar es que las autoridades de control, ya están combatiendo este tipo de negocios, lo que nos lleva a pensar que dentro de poco tiempo los precios se normalizarán en el mercado.

2.2.2. Reacción del consumidor ante el mercado virtual

El hecho de que cada día se incorpore nuevos usuarios al mundo del Internet ha sido un punto decisivo en la implementación de nuestro sistema. Ya que es un campo, que recién lo están explotando otras actividades, como son, la banca que por ejemplo ofrecen el pago de los

servicios básicos desde sus páginas web, evitando que el cliente se acerque a las recaudaras de esos servicios.

Esto nos ha ayudado a explorar ese potencial número de clientes que acceden al Internet, y ofrecer desde nuestra página el servicio de alquiler de películas, más los servicios de entrega y retiro a domicilio de las películas.

2.2.3. Mercado virtual vs. Mercado convencional.

2.2.3.1. Ventajas.

Mercado Virtual	Mercado Convencional
Mayor cobertura del mercado.	Precios más bajos.
Los clientes no necesitan acercarse, existe una vitrina virtual, y se ofrece servicios de entrega y retiro a domicilio.	Existen varios locales.
Poseen un control de la calidad de las videos, se separan aquellos medios con problemas.	Menos selectivo en su cartera de clientes.
Se cuenta con películas de actualidad.	
Se permite consultar el cupo de películas en cualquier computador con acceso al Internet.	
Pueden incorporar cualquier nuevo formato de medio que salga en el mercado.	

2.2.3.2. Desventajas.

Mercado Virtual	Mercado Convencional
Precio acorde a los servicios que ofrece, se paga un poco más por un mejor servicio.	Su cobertura esta limitada al sector donde se encuentra el local.
Solo se cuenta con una bodega central.	En la gran mayoría no tienen control de calidad de los medios.
La cobertura del local.	Si en el mercado aparece otro formato, son pocos los locales que lo innovarían.
	Aquellos que alquilan copias ilegales se exponen a ser sancionados.

2.3. Competidores

Existen dos sectores de competidores:

El primero, que se trata de locales pequeños localizados en los barrios ó ciudadelas.

El segundo, está representado por locales con una mayor variedad de películas. Estos locales son franquicias internacionales, casi siempre localizados cerca de los grandes centros comerciales.

2.3.1. Funcionalidad de los competidores convencionales existentes.

El primer sector de competidores, locales pequeños al captar un nuevo cliente, le solicitan una cantidad de dinero en garantía, no se recaba mayor información del cliente.

Si ya se trata de un cliente habitual, no se aplica el valor de garantía y el pago es contra entrega de la película en el local.

El cliente debe devolver la película a mas tardar dentro de 24 horas.

El segundo sector, es más formal, sí se recaba información general del cliente. Aquí se estipula una mensualidad fija que el cliente debe pagar, alquile o no la película en un mes. El cliente debe retirar y devolver la película.

2.3.2. Desafíos encarados por competidores convencionales.

Los competidores del primer sector darían una escasa competencia al nuevo esquema, ya que si ellos ofrecen mas servicios y descuentos especiales, esos servicios les obligaría a incrementar el valor del alquiler.

En cambio, los competidores del segundo sector, tendrán que arriesgar su ingreso mensual fijo para contrarrestar el mecanismo de talonario que implementamos en nuestro esquema virtual, en donde el cliente cancela su cupo de películas que va a alquilar.

Los dos sectores tendrían que implementar el servicio a domicilio para contrarrestar en algo una de las cualidades que diferencian el servicio virtual.

2.4. Proveedores

Las relaciones con nuestros proveedores se lleva en los mejores términos, gracias a la honestidad y el respeto a los derechos de autoría que ellos tienen sobre las películas.

Lo anterior expuesto, no lo tiene el sector de locales pequeños de nuestros competidores, ya que un gran porcentaje de este sector no respetan estos derechos.

Los competidores que poseen franquicias si mantienen buenas relaciones comerciales con sus proveedores.

2.5. Servicios sustitutos.

Cabe citar que para hablar de servicios sustitutos, debemos conocer a nuestro cliente objetivo. Ya que cuando el cliente alquila, busca su comodidad y ahorro.

Un servicio con similar objetivo, es la televisión por cable. Un factor que desanima a que las personas adquieran este servicio, es su tarifa fija

mensual, y que aún a pesar de estar la economía dolarizada, ha sufrido incrementos continuos.

2.6. Análisis FODA del Sistema

2.6.1. Factores internos: Fortalezas.

Permite a los clientes adquirir talonarios para el cupo de películas que él estime que solicitará, esto ayuda al cliente a no comprometer su dinero en mensualidades fijas.

Los clientes pueden alquilar sus películas desde cualquier computador conectado al Internet, con ello se asegura su reservación.

Además si eligió los servicios de entrega y/o retiro, nuestra flota de repartidores le hará llegar a domicilio su pedido.

Gracias a la implementación de los talonarios, la empresa se asegura un control de los alquileres de películas hacia los clientes.

2.6.2. Factores internos: Debilidades.

Una debilidad es el hecho de no contar con mas locales en la ciudad. Cabe indicar que está debilidad, le retribuye en ahorros operativos al no contar con más locales.

2.6.3. Factores externos: Oportunidades.

De que si la empresa apertura otro local en otra ciudad, la implementación sería muy similar a la actual.

2.6.4. Factores externos: Amenazas.

Si las autoridades de control no continúan con los operativos contra la piratería en defensa de los derechos de autor, la competencia atraería a los clientes por sus precios más bajos.

Que el sector formal de renta de videos incursione en el mercadeo vía web.

3

MODELO DE NEGOCIOS

3. MODELO DE NEGOCIOS.

3.1. Definición del modelo de negocios.

La empresa Ecuacinema con la innovación de captar nuevos clientes que son habituales usuarios del Internet, estará implementando el modelo de negocio B2C, que involucra hacer negocios con el consumidor final directamente, nuestro producto es el alquiler de películas que ofrecemos a nuestros clientes.

3.2. Funcionalidad general.

Para garantizar la funcionalidad de la parte operativa del negocio de Ecuacinema, se debe cumplir los siguientes objetivos:

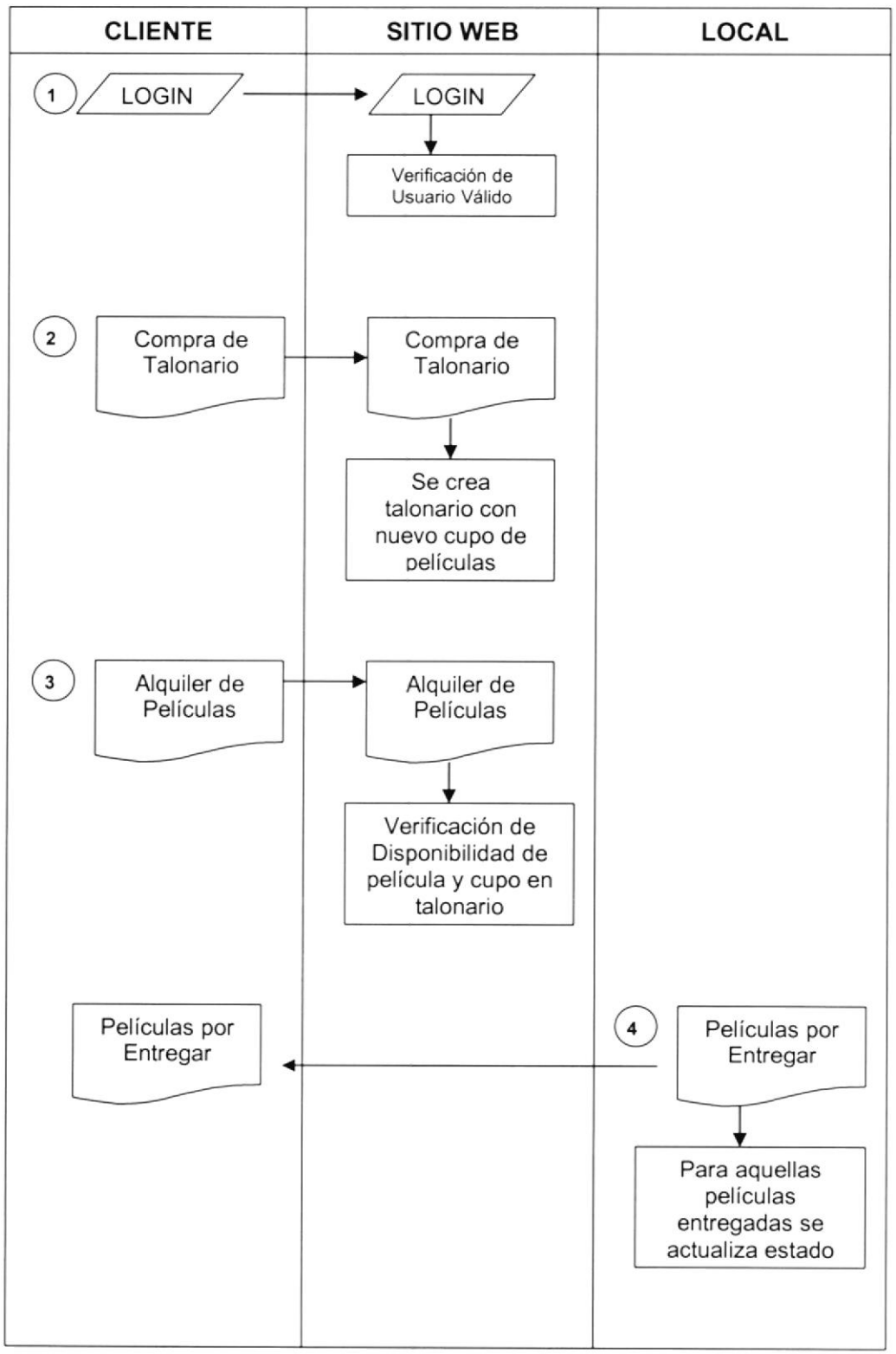
- Permitir registrar información específica de los clientes.
- Permitir registrar información de los títulos, tipo de medio de grabación, idioma de la películas y establecer un registro de ítems con lo cual se garantice tener más de un ítem con un mismo título.
- Permitir registrar información relevante como: Sectores de la ciudad, emisoras de tarjetas de crédito, segmentos para publicar en el sitio web.
- Permitir realizar la compra de talonarios, con lo que el cliente adquiere un cupo de películas por alquilar. De igual forma de haber

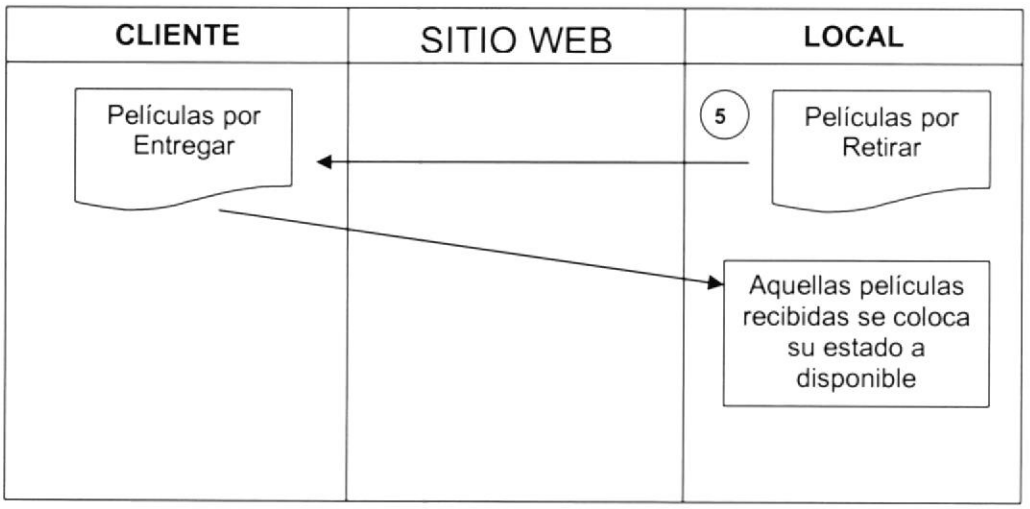
un mecanismo para registrar los diferentes Tipos de Talón que la empresa innove.

- Que al realizar la búsqueda de las películas que se desee alquilar, se implemente la búsqueda por los criterios de: título de la películas, nombre del director, nombres en el reparto de actores.

- Permitir realizar la operación de alquiler de películas. Y que por cada alquiler se actualice el cupo disponible del talonario del cliente.

3.2.1. Diagrama de flujo del proceso.





Narrativa.-

1. El usuario para realizar la renta de videos, debe haber sido aceptado como un cliente de la empresa y tener un login y password asociado.

2. Antes de proceder a alquilar películas, el cliente debe adquirir un talonario con un cupo de películas por alquilar.

3. El cliente realiza su solicitud de películas y si los ítems están disponibles y si tiene cupo en sus talonarios, se acepta su solicitud de alquiler.

4. Si en la solicitud de alquiler el cliente expresa su deseo de obtener el servicio de entrega, se procede a enviar las películas a su domicilio.

5. Si en la solicitud de alquiler el cliente expresa su deseo de obtener el servicio de retiro, se procede a retirar las películas del domicilio del

cliente, y después se activan los ítems de películas recibidas al estado de disponible para un próximo alquiler.

3.3. Sistema de entrega.

El servicio de entrega y retiro de nuestro producto "Películas alquiladas" no tiene ningún costo para el cliente. Y es está diferencia competitiva que ha mantenido en el mercado a Ecuacinema como una de las pocas empresas que ofrece un "Servicio al Cliente" de calidad.

Este servicio Ecuacinema lo tiene contratado con la empresa Servientrega que es un socio estratégico para el buen desempeño de las operaciones de alquiler.

3.4. Forma de pago.

Ecuacinema ofrece dos formas de pagos a sus clientes.

La primera que es en efectivo y la otra con tarjeta de crédito. El pago con tarjeta de crédito involucra que el cliente en la aplicación Web debe registrar el emisor de la tarjeta, el nombre del titular, su número y fecha de vencimiento.

3.5. Seguridad.

La seguridad de la información el Sistema de Renta de Videos, la administra con los dos tipos de aplicación que propone.

En la aplicación Cliente / Servidor, sólo se permite el acceso al personal de la empresa. Aquí se permite el acceso al sistema mediante el uso de un login y password, con que se garantiza de que el usuario es parte del sistema.

En la aplicación Web, se permite navegar en la consulta (búsqueda) de películas a todos los usuarios del Internet que estén navegando en el sitio Web de Ecuacinema. Pero si el usuario de Internet desea proceder al alquiler de películas, éste primero debe ser un usuario que ya la empresa Ecuacinema lo haya incorporado en su base de datos de clientes.

4

ARQUITECTURA DEL SISTEMA

4. ARQUITECTURA DEL SISTEMA

4.1. Justificación de la selección del modelo.

Para el desarrollo del Sistema de Renta de Videos se escogió usar el modelo de aplicaciones de 3 capas, es decir, la capa de presentación, la capa de negocios y la capa de datos.

Para la capa de presentación se usaron clientes "standalone" así como también clientes tipo browser para que la aplicación tenga acceso desde Internet.

En la capa de negocios se decidió crear componentes de negocios en donde residen las reglas del negocio y componentes de datos en donde se encuentran las llamadas a los stored procedures de la base de datos.

En la capa de datos, el servidor de base de datos mediante la ejecución de stored procedures realiza las diferentes operaciones sobre la base de datos.

Las razones que justificaron nuestra decisión fueron:

- Los clientes tipo browser nos permiten alcanzar una gran cantidad de clientes potenciales a través de nuestras opciones disponibles en nuestro sistema web.

- Para evitar la duplicación de código se utilizaron los mismos componentes de datos y negocios para los dos diferentes tipos de clientes.

- Para ser capaces de reaccionar rápidamente a los cambios en las políticas empresariales. La mayoría de los cambios que se soliciten no interrumpirán la operación del negocio.

- Para facilitar el mantenimiento de la aplicación. Debido a la separación clara de los tipos de lógica es fácil determinar en donde se deben realizar los cambios a la aplicación.

- El modelo de 3 capas nos da mejores posibilidades de escalabilidad a la aplicación, en caso de que el número de usuarios de Internet exceda las expectativas.

- Por la naturalidad con la que se puede distribuir el desarrollo del proyecto en un grupo de desarrolladores con habilidades particulares.

4.2. Tipos de Procesos Clientes.

El Sistema de Renta de Videos tiene dos tipos de clientes, el cliente browser que accede al sitio web desde Internet y el cliente standalone que funciona en el local de la empresa.

4.2.1. Cliente Browser (Dependiente)

Para nuestros clientes de browser, se desarrolló un sitio web con páginas ASP y HTML. Nuestros clientes pueden accesar el sitio usando el browser Internet Explorer, aunque en realidad pueden usar cualquier browser que soporte Javascript, como el Netscape.

4.2.2. Cliente Standalone (Independiente).

Los usuarios de la empresa utilizan el módulo de Renta de Videos, que se trata de una aplicación Visual Basic en la cual tienen las opciones que son propias de la administración del negocio y que por lo tanto no tienen que estar disponibles a clientes desde Internet.

4.3. Tipos de Procesos Servidores.

Para la implementación del sistema se necesitó disponer de los siguientes procesos servidores:

4.3.1. Servidor de Componentes y de Transacciones.

Para poder implementar la capa 2 de negocios y de acceso a datos, se necesitaba disponer de un servidor de objetos con soporte a transacciones. El software debía funcionar en un ambiente distribuído en una red Windows.

Todos los servicios mencionados los ofrece COM+, que viene incluído en Windows 2000. Además de ello, COM+ nos da servicios de

administración distribuída, seguridad mediante la implementación de roles de usuario y definición de transacciones que abarcan varios componentes de código.

Los componentes de negocios y datos se implementaron como procesos in-process, es decir de tipo DLL dentro de la aplicación contenedora COM+.

4.3.2. Servidor Web.

Para atender los requerimientos de los clientes de browser se necesitaba un servidor web que tenga las siguientes características:

- Que sea un producto estable y de amplia aceptación en el mercado.
- Que ofrezca un buen rendimiento.
- Que de soporte a la programación de scripts del lado del servidor, como ASP.
- Que pueda hacer uso de los componentes de negocios y datos que iban a residir en la capa 2.

Por estas razones, se decidió escoger al Internet Information Server de Microsoft que es el producto que mejor soporta ASP en el mercado y a la vez es muy conocido.

4.3.3. Servidor de Base de Datos.

Para garantizar la persistencia e integridad de los datos de la aplicación, se necesitaba de un software manejador de bases de datos que tuviera las siguientes características:

- Que permita la implementación de bases de datos relacionales que soporten bien aplicaciones tipo OLTP o de procesamiento típicamente transaccional.
- Debe ser un producto estable y de tecnología madura.
- Que soporte cargas de trabajo exigentes y variables.
- Que disponga la tecnología de stored procedures para descargar ciertos procesos de negocio en el servidor de la base de datos y de esta manera mejorar el rendimiento de la aplicación.
- Ser altamente compatible con el servidor de objetos COM+.
- Proveer facilidades para la programación desde clientes Visual Basic y desde scripts ASP.
- Ser un estándar de la industria.

Por estas razones se escogió usar el software Microsoft SQL Server, ya que es un producto con años en el mercado, que soporta stored-procedures usando el lenguaje Transact-SQL, naturalmente ligado al

servidor transaccional COM+ y al ambiente de programación de Visual Basic.

4.4. Tipos de Middleware.

Para lograr la comunicación entre el cliente browser y nuestro servidor web, nos valemos del protocolo HTTP (HyperText Transfer Protocol). Este recibe los requerimientos de nuestros clientes browser y los envía al servidor web, luego de que el servidor web procesa el requerimiento envía los resultados de vuelta al cliente browser.

Para la comunicación entre clientes standalone y los componentes que se encuentran en el servidor de componentes, nos valemos del DCOM (Distributed COM) y su arquitectura de comunicación entre procesos. También usamos DCOM entre los componentes de negocios y datos y entre los scripts ASP y los componentes de negocios.

Para la comunicación entre los componentes de acceso a datos y nuestro servidor de base de datos utilizamos la interfaz de alto nivel ADO (ActiveX Data Objects). Esta interfaz hace uso a su vez del API de bajo nivel OLEDB. Ambos son estándares de la tecnología de acceso a datos de Microsoft.

4.5. Tecnologías y herramientas escogidas para la implementación.

Las principales tecnologías que hemos usado en el desarrollo del proyecto son:

Nombre	Significado	Uso
COM	Component Object Model	En la infraestructura de desarrollo de aplicaciones basada en componentes.
DCOM	Distributed COM	Para acceder a los objetos que residen en computadoras remotas.
COM+	DCOM + MTS	Al crear y usar las aplicaciones de servidor en Windows 2000. Para el soporte transaccional.
ASP	Active Server Pages	En el desarrollo de los scripts en el servidor web.
HTTP	HyperText Transfer Protocol	En la interacción entre el browser y el servidor web.
ADO	ActiveX Data Objects	Interfaz de alto nivel para el acceso a datos.
OLEDB	Object Linking and Embedding DataBase	API de bases de datos
HTML	HyperText Markup Language	En la creación de páginas web.
DHTML	Dynamic HTML	Para facilitar las validaciones en las páginas HTML.

Cuadro 1. Tecnologías usadas en la aplicación.

Entre las principales herramientas que se utilizaron para el desarrollo de la aplicación están:

Nombre	Uso
Visual Studio 6.0 Professional SP 5 (Visual Basic y Visual Interdev, GuidGen, Compilador MIDL)	Visual Basic para el desarrollo de componentes de negocios, de datos y programas clientes. Visual Interdev para el desarrollo de las páginas ASP y la administración del sitio web. GUIDGen para generar los GUIDs usados en las interfaces y MIDL para convertir las interfaces IDL en type libraries.
Administrador de Componentes COM+	Para instalar y administrar los componentes COM.
SQL Server 7.0 SP 3	Para que sea el repositorio de los datos de la aplicación.
MacroMedia DreamWeaver 4.0	Para la definición de formatos o plantillas del sitio.
MacroMedia Flash 5.0	Para la creación de animaciones para el sitio web.
MacroMedia FireWorks 4.0	Para la edición de imágenes para el sitio web.
Adobe Photoshop	Para la edición de imágenes usadas en la web.
Adobe Illustrator	Para edición de imágenes tipo vector.
Swift 3D	Para crear animaciones de 3 dimensiones.
Microsoft Internet Explorer 6.0	Para probar la aplicación web.

Nombre	Uso
Farpoint Spread Control 3.0	Para el desarrollo de pantallas que utilizan controles tipo grid en el cliente standalone.
Seagate Crystal Reports Professional 8.0	Para el desarrollo de los reportes del módulo standalone.

Cuadro 2. Herramientas usadas en el desarrollo de la aplicación

4.6. Requerimientos de Hardware y Software para la puesta en Producción.

Los requerimientos mínimos de hardware y software del Sistema de Renta de Videos son:

4.6.1. Requerimientos del servidor de Base de Datos.

Los requerimientos mínimos de hardware y software del servidor de base de datos se destacan en el cuadro inferior:

Hardware	Computadora Servidor de Base de Datos (Intel) PC Pentium IV con 256MB RAM, 30 GB de almacenamiento secundario, dispositivo de respaldo, tarjeta de red Fast-Ethernet de 100Mbps
Software	Windows 2000 Advanced Server, Microsoft SQL Server 7.0 SP3

Cuadro 3. Requerimientos del Servidor de Base de Datos

Se recomienda que el servidor implemente algún tipo de esquema RAID en los medios de almacenamiento para asegurar la disponibilidad del sistema.

4.6.2. Requerimientos del servidor Web.

Los requerimientos mínimos de hardware y software del servidor web se destacan en el cuadro inferior:

Hardware	Computadora Servidor Web (Intel) PC Pentium IV con 256MB RAM, 30 GB de almacenamiento secundario, dispositivo de respaldo, tarjeta de red Fast-Ethernet de 100Mbps
Software	Windows 2000 Advanced Server, Internet Information Server 5.0 (viene integrado al Windows 2000).

Cuadro 4. Requerimientos del Servidor Web

4.6.3. Requerimientos del servidor Transaccional.

Los requerimientos mínimos de hardware y software del servidor transaccional se destacan en el cuadro inferior:

Hardware	Computadora Servidor Web (Intel) PC Pentium IV con 256MB RAM, 30 GB de almacenamiento secundario, dispositivo de respaldo, tarjeta de red Fast-Ethernet de 100Mbps

Software	Windows 2000

Cuadro 5. Requerimientos del Servidor Transaccional

4.6.4. Requerimientos del cliente standalone.

Los requerimientos mínimos de hardware y software del cliente standalone se destacan en el cuadro inferior:

Hardware	Computadora Personal (Intel) PC Pentium con 64MB RAM, 10 GB de almacenamiento secundario, monitor de 15" con resolución de 1024 x 768 pixels, tarjeta de red Fast-Ethernet de 100Mbps
Software	Windows 9X, Windows NT ó Windows 2000

Cuadro 6. Requerimientos del Cliente Standalone

4.6.5. Requerimientos del cliente browser.

Los requerimientos mínimos de hardware y software del cliente browser se destacan en el cuadro inferior:

Hardware	Computadora Personal (Intel) PC Pentium con 64MB RAM, 10 GB de almacenamiento secundario, monitor de 15" con resolución de 1024 x 768 pixels, modem para conexión a Internet o equivalente.

Software	Windows 98 ó Windows 2000 Professional, Internet Explorer 6.0

Cuadro 7. Requerimientos del Cliente Browser

4.6.6. Consideraciones finales.

Además del software mencionado, es necesario la instalación de un paquete anti-virus para evitar que se infecte algún computador y se pierda información.

Cabe resaltar el hecho de que el sistema puede ser implementado sin ningún inconveniente en un sólo servidor que contenga la base de datos, servidor web y servidor transaccional.

Posteriormente se puede escalar a la solución de múltiples servidores sin alterar una sola línea de código y para mejorar el rendimiento de la misma.

5

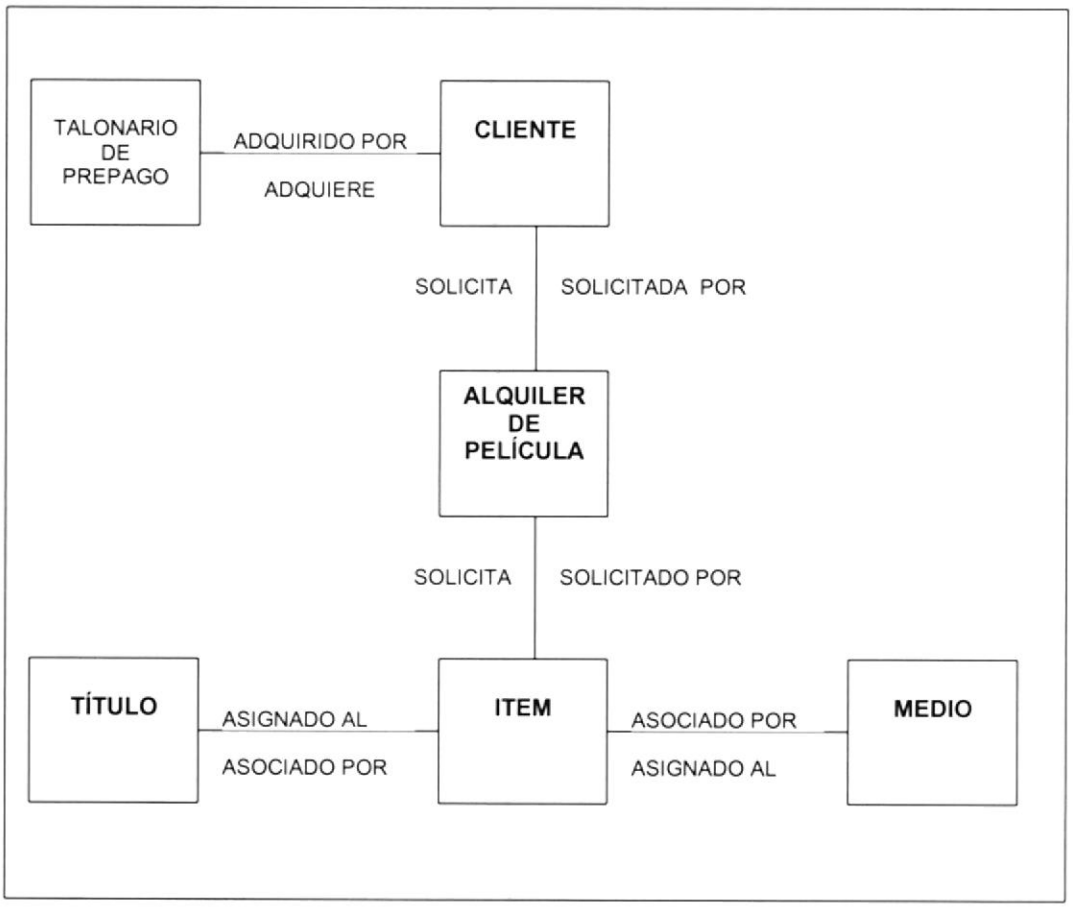
IMPLEMENTACIÓN DE LA CAPA DE DATOS

5. IMPLEMENTACIÓN DE LA CAPA DE DATOS.

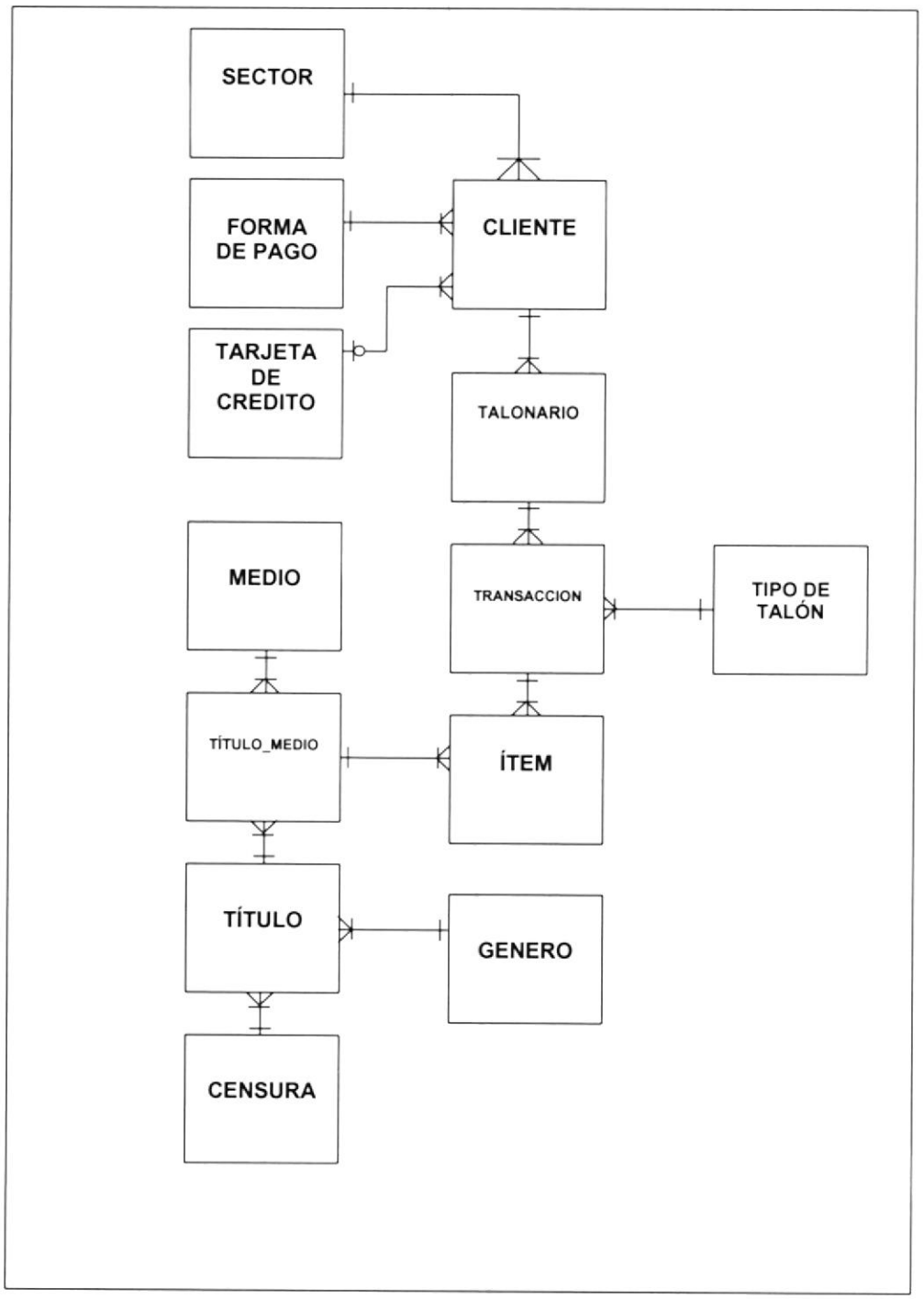
Debemos informar que la implementación de la capa de datos la realizaremos en SQL Server 7.0 y la base de datos del Sistema de Renta de Videos se llamará DB_VIDEO.

A continuación mostraremos el modelo lógico del Sistema, el modelo Entidad-Relación y el modelo Físico de nuestra base de datos.

5.1. Modelo Lógico de Datos.

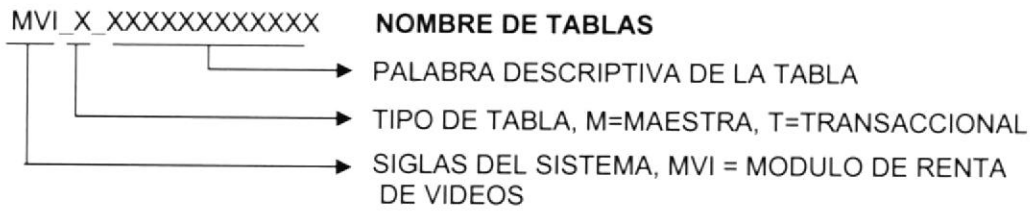


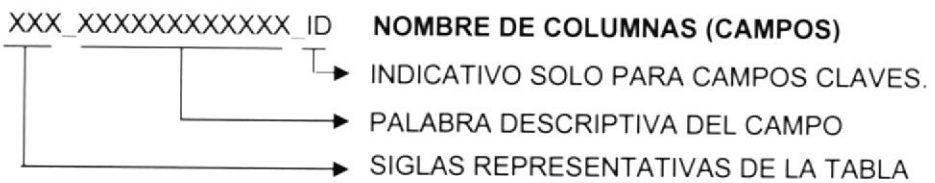
5.2. Diagrama de Entidad-Relación.



5.3. Modelo Físico de Datos.

Para la nomenclatura de los nombres de las tablas y columnas (campos) hemos utilizada la siguiente convención:





A continuación detallaremos las tablas que utilizará el Sistema de Renta de Videos.

5.3.1. Tablas Maestras.

DISEÑO DE LA TABLA						
TABLA: **mvi_m_categoriaweb**			BASE DE DATOS: **DB_VIDEO**			
DESCRIPCIÓN: Contendrá los tipos de categorías que se publicarán en la pagina principal del sitio Web.						
No.	**Columna**	**Tipo**	**Long.**	**Prc.**	**Scl.**	**Contenido**
1	cat_categoria_id	Int	4	10	0	Código de categoría.
2	cat_descripcion	char	25	25	0	Descripción de la categoría
TIPO DE INDICE:		**COLUMNA(S):**				
Primary Key		cat_categoria_id				

DISEÑO DE LA TABLA						
TABLA: **mvi_m_censura**			BASE DE DATOS: **DB_VIDEO**			
DESCRIPCIÓN: Contendrá los tipos de censuras que serán asociados a los títulos de las películas.						
No.	**Columna**	**Tipo**	**Long.**	**Prc.**	**Scl.**	**Contenido**
1	cen_censura_id	char	1	1	0	Código de censura.
2	cen_descripcion	char	60	60	0	Descripción de la censura.
TIPO DE INDICE:		**COLUMNA(S):**				
Primary Key		cen_censura_id				

DISEÑO DE LA TABLA

TABLA: **mvi_m_cliente**	BASE DE DATOS: **DB_VIDEO**

DESCRIPCIÓN: Contendrá la información específica de los clientes de la empresa.

No.	Columna	Tipo	Long.	Prc.	Scl.	Contenido
1	cli_cliente_id	decimal	5	7	0	Código del cliente.
2	cli_nombre	varchar	25	25	0	Nombres.
3	cli_apellido	varchar	25	25	0	Apellidos.
4	cli_tipo_identificacion	char	1	1	0	Tipo de Identificación (C=Cédula, R=RUC)
5	cli_num_identificacion	Char	15	15	0	Número de identificación.
6	cli_direccion	varchar	50	50	0	Domicilio del cliente.
7	sec_sector_id	smallint	2	5	0	Código del sector de la ciudad.
8	cli_telefono	varchar	15	15	0	Teléfono.
9	cli_email	varchar	30	30	0	Dirección e-mail.
10	fpg_forma_pago_id	smallint	2	5	0	Código de la forma de pago.
11	tar_tarjeta_id	smallint	2	5	0	Código de la tarjeta de crédito.
12	cli_num_tarjeta	char	16	16	0	Número de la tarjeta.
13	cli_tar_fecha_vencim	datetime	8	0	0	Fecha de Vencimiento de la tarjeta.

DISEÑO DE LA TABLA

TABLA: **mvi_m_cliente**	BASE DE DATOS: **DB_VIDEO**

DESCRIPCIÓN: Contendrá la información específica de los clientes de la empresa.

No.	Columna	Tipo	Long.	Prc.	Scl.	Contenido
14	cli_servicio_entrega	char	1	1	0	Indicador si desea entrega a domicilio (S=Si, N=No).
15	cli_servicio_retiro	char	1	1	0	Indicador si desea retiro a domicilio (S=Si, N=No).
16	cli_login	varchar	8	8	0	Login del cliente.
17	cli_clave	varchar	8	8	0	Clave del cliente.
18	cli_estado_reg	char	1	1	0	Estado del registro.
19	cli_fecha_registro	datetime	8	0	0	Fecha de registro del cliente.

TIPO DE INDICE:	**COLUMNA(S):**
Primary Key	cli_cliente_id
Foreign key	fpg_forma_pago_id referencia a mvi_m_forma_pago
Foreign key	tar_tarjeta_id referencia a mvi_m_tarjeta
Foreign key	sec_sector_id referencia a mvi_m_sector

DISEÑO DE LA TABLA

TABLA: **mvi_m_forma_pago**	BASE DE DATOS: **DB_VIDEO**

DESCRIPCIÓN: Contendrá la información de las formas de pago que manejará el sistema.

No.	Columna	Tipo	Long.	Prc.	Scl.	Contenido
1	fpg_forma_pago_id	smallint	2	5	0	Código de la forma de pago.
2	fpg_descripción	char	60	60	0	Descripción de la forma de pago.
3	fpg_validez_local	char	1	1	0	Indicador si la forma de pago se aplica en el local.
4	fpg_validez_internet	char	1	1	0	Indicador si la forma de pago se aplica en la aplicación Web.

TIPO DE INDICE:	**COLUMNA(S):**
Primary Key	fpg_forma_pago_id

DISEÑO DE LA TABLA

TABLA: **mvi_m_genero**	BASE DE DATOS: **DB_VIDEO**

DESCRIPCIÓN: Contendrá la información de los tipos de géneros que serán asociados a las películas dependiendo el contenido de ellas.

No.	Columna	Tipo	Long.	Prc.	Scl.	Contenido
1	gen_genero_id	char	3	3	0	Código del género.
2	gen_descripcion	char	60	60	0	Descripción del género.

TIPO DE INDICE:	**COLUMNA(S):**
Primary Key	gen_genero_id

DISEÑO DE LA TABLA

TABLA: **mvi_m_idioma**	BASE DE DATOS: **DB_VIDEO**

DESCRIPCIÓN: Contendrá la información de los tipos de idiomas que serán asociados a cada ítem de películas que se cree..

No.	Columna	Tipo	Long.	Prc.	Scl.	Contenido
1	idm_idioma_id	smallint	2	5	0	Código del idioma.
2	idm_descripcion	char	25	25	0	Descripción del idioma.

TIPO DE INDICE:	**COLUMNA(S):**
Primary Key	idm_idioma_id

DISEÑO DE LA TABLA

TABLA: **mvi_m_item**	BASE DE DATOS: **DB_VIDEO**

DESCRIPCIÓN: Contendrá la información de los tipos de idiomas que serán asociados a cada ítem de películas que se cree..

No.	Columna	Tipo	Long.	Prc.	Scl.	Contenido
1	itm_item_id	decimal	5	7	0	Código del ítem.
2	tit_titulo_id	decimal	5	7	0	Código del título.
3	med_medio_id	Int	4	10	0	Código del medio.
4	idm_idioma_id	smallint	2	5	0	Código del idioma.
5	itm_estado_reg	char	1	1	0	Estado del registro (A =Activo, I =Inactivo).
6	itm_estado_renta	char	1	1	0	Estado de Renta (D =Disponible, A =Alquilado)

TIPO DE INDICE:	**COLUMNA(S):**
Primary Key	itm_item_id
Foreign Key	tit_titulo_id referencia a mvi_m_titulo
Foreign Key	med_medio_id referencia a mvi_m_medio
Foreign Key	idm_idioma_id referencia a mvi_m_idioma

DISEÑO DE LA TABLA

TABLA: **mvi_m_medio**	BASE DE DATOS: **DB_VIDEO**

DESCRIPCIÓN: Contendrá la información de los tipos de medios que va manejar el sistema, que son VHS y DVD.

No.	Columna	Tipo	Long.	Prc.	Scl.	Contenido
1	med_medio_id	int	4	10	0	Código del medio.
2	med_descripcion	char	25	25	0	Descripción del medio.
3	med_costo_reposicion	decimal	5	7	2	Costo de reposición del medio.
4	med_logo_medio	varchar	50	50	0	Ruta del archivo de imagen que representa el medio.

TIPO DE INDICE:	**COLUMNA(S):**
Primary Key	med_medio_id

DISEÑO DE LA TABLA

TABLA: **mvi_m_parámetro**	BASE DE DATOS: **DB_VIDEO**

DESCRIPCIÓN: Contendrá los parámetros que define la empresa.

No.	Columna	Tipo	Long.	Prc.	Scl.	Contenido
1	par_param_id	varchar	20	20	0	Código del parámetro.
2	par_descripcion	varchar	50	50	0	Descripción del parámetro.
3	par_valor	decimal	5	9	2	Valor del parámetro.
4	par_texto	varchar	60	60	0	Contenido del parámetro, tipo texto.

TIPO DE INDICE:	**COLUMNA(S):**
Primary Key	par_param_id

DISEÑO DE LA TABLA

TABLA: **mvi_m_sector**	BASE DE DATOS: **DB_VIDEO**

DESCRIPCIÓN: Contendrá la información de los sectores de cómo se encuentra dividida la ciudad.

No.	Columna	Tipo	Long.	Prc.	Scl.	Contenido
1	sec_sector_id	smallint	2	5	0	Código del sector.
2	sec_descripcion	char	25	25	0	Descripción del sector.

TIPO DE INDICE:	COLUMNA(S):
Primary Key	sec_sector_id

DISEÑO DE LA TABLA

TABLA: **mvi_m_secuencial**	BASE DE DATOS: **DB_VIDEO**

DESCRIPCIÓN: Contendrá la información de los secuenciales de los códigos únicos de las tablas del Sistema.

No.	Columna	Tipo	Long.	Prc.	Scl.	Contenido
1	sec_secuencial_id	smallint	2	5	0	Código del secuencial.
2	sec_descripcion	varchar	50	50	0	Descripción del secuencial, esto nos indicará la tabla a la que pertenece.
3	sec_valor	decimal	5	7	0	Valor del último correlativo asignado.

TIPO DE INDICE:	**COLUMNA(S):**
Primary Key	sec_secuencial_id

DISEÑO DE LA TABLA

TABLA: **mvi_m_talontipo**	BASE DE DATOS: **DB_VIDEO**

DESCRIPCIÓN: Contendrá la información de los tipos de talón que comercializará la empresa para el alquiler de las películas.

No.	Columna	Tipo	Long.	Prc.	Scl.	Contenido
1	tti_talontipo_id	int	4	10	0	Código del tipo de talón.
2	tti_descripción	varchar	80	80	0	Descripción del tipo de talón.
3	tti_numero_tickets	int	4	10	0	Cupo de películas.
4	tti_precio	decimal	5	5	2	Precio del talón.

TIPO DE INDICE:	**COLUMNA(S):**
Primary Key	tti_talontipo_id

DISEÑO DE LA TABLA

TABLA: **mvi_m_tarjeta**	BASE DE DATOS: **DB_VIDEO**

DESCRIPCIÓN: Contendrá la información de los diferentes emisores de tarjetas de crédito.

No.	Columna	Tipo	Long.	Prc.	Scl.	Contenido
1	tar_tarjeta_id	smallint	2	5	0	Código de la tarjeta.
2	tar_descripcion	varchar	25	25	0	Descripción de la tarjeta (emisor).

TIPO DE INDICE:	**COLUMNA(S):**
Primary Key	tar_tarjeta_id

DISEÑO DE LA TABLA

TABLA: **mvi_m_tipotrans**	BASE DE DATOS: **DB_VIDEO**

DESCRIPCIÓN: Contendrá la información de los dos tipos de transacciones que vamos a manejar en la tabla de transacciones. (1 =Compra Talonario, 2 =Alquiler).

No.	Columna	Tipo	Long.	Prc.	Scl.	Contenido
1	ttr_tipotrans_id	smallint	2	5	0	Código del tipo de transacción.
2	ttr_descripción	varchar	50	50	0	Descripción del tipo de transacción.
3	ttr_signo	char	1	1	0	Signo de la transacción.

TIPO DE INDICE:	**COLUMNA(S):**
Primary Key	ttr_tipotrans_id

DISEÑO DE LA TABLA

TABLA: **mvi_m_titulo**	BASE DE DATOS: **DB_VIDEO**

DESCRIPCIÓN: Contendrá la información de los títulos de las películas.

No.	Columna	Tipo	Long.	Prc.	Scl.	Contenido
1	tit_titulo_id	Decimal	5	7	0	Código del título.
2	tit_nombre_orig	Varchar	100	100	0	Título original de la película.
3	tit_nombre	varchar	100	100	0	Título en español.
4	tit_sinopsis	varchar	500	500	0	Sinopsis de la película.
5	tit_productor	varchar	50	50	0	Nombre del productor.
6	tit_director	varchar	60	60	0	Nombre del director.
7	tit_reparto	varchar	100	100	0	Nombres de los actores del reparto.
8	cen_censura_id	char	1	1	0	Código de la censura.
9	gen_genero_1	char	3	3	0	Código del género (obligatorio).
10	gen_genero_2	char	3	3	0	Código del género (opcional).
11	gen_genero_3	char	3	3	0	Código del género (opcional).
12	tit_nominacion_sn	char	1	1	0	Indicador que si la película ha sido nominada al Oscar (S =Si, N = No).

DISEÑO DE LA TABLA						
TABLA: **mvi_m_titulo**			BASE DE DATOS: **DB_VIDEO**			
No.	**Columna**	**Tipo**	**Long.**	**Prc.**	**Scl.**	**Contenido**
13	tit_nominaciones	varchar	100	100	0	Referencia de las nominaciones que haya alcanzado.
14	tit_premios_sn	char	1	1	0	Indicador que si la película ha ganado algún premio (S =Si, N = No).
15	tit_premios	varchar	100	100	0	Referencia de los premios que haya alcanzado.
16	tit_oscares_sn	char	1	1	0	Indicador que si la película ha ganado el Oscar (S =Si, N = No).
17	tit_duracion	char	5	5	0	Tiempo de duración de la película.
18	tit_produccion	int	4	10	0	Año de producción de la película.
19	tit_logo_pelicula	varchar	50	50	0	Ruta del archivo de imagen que representa el titulo de la película.
TIPO DE INDICE:		**COLUMNA(S):**				
Primary Key		tit_titulo_id				

DISEÑO DE LA TABLA

TABLA: **mvi_m_titulo_medio**	BASE DE DATOS: **DB_VIDEO**

DESCRIPCIÓN: Contendrá un registro por cada titulo y tipo de medio diferentes.

No.	Columna	Tipo	Long.	Prc.	Scl.	Contenido
1	tit_titulo_id	decimal	5	7	0	Código del Título de la película.
2	med_medio_id	Int	4	10	0	Código del Título de la película.

TIPO DE INDICE:	COLUMNA(S):
Primary Key	tit_titulo_id, med_medio_id

DISEÑO DE LA TABLA

TABLA: **msg_m_usuario**	BASE DE DATOS: **DB_VIDEO**

DESCRIPCIÓN: Contendrá los usuarios de la aplicación cliente / servidor que se operará en el local.

No.	Columna	Tipo	Long.	Prc.	Scl.	Contenido
1	usu_login_id	varchar	20	20	0	Login del usuario.
2	usu_nombre	varchar	50	50	0	Nombre del usuario.
3	usu_password	varchar	50	50	0	Clave
4	usu_fecha_reg	datetime	8	0	0	Fecha de registro del usuario
5	usu_estado_reg	char	1	1	0	Estado del registro.

TIPO DE INDICE:	**COLUMNA(S):**
Primary Key	usu_login_id

5.3.2. Tablas Transaccionales.

DISEÑO DE LA TABLA						
TABLA: **mvi_t_preregistro**			BASE DE DATOS: **DB_VIDEO**			
DESCRIPCIÓN: Contendrá la información de las solicitudes realizadas en el Pre-Registro que llenen los usuarios en la aplicación Web.						
No.	**Columna**	**Tipo**	**Long.**	**Prc.**	**Scl.**	**Contenido**
1	prg_tipo_identificacion	char	1	1	0	Tipo de Identificación.
2	prg_num_identificacion	char	15	15	0	Número de identificación.
3	prg_login	char	8	8	0	Login de acceso.
4	prg_clave	char	8	8	0	Clave de acceso.
5	prg_nombre	varchar	50	50	0	Nombres.
6	prg_apellido	varchar	50	50	0	Apellidos.
7	sec_sector_id	smallint	2	5	0	Código del sector de la ciudad.
8	prg_direccion	varchar	70	70	0	Domicilio.
9	prg_telefono	varchar	15	15	0	Teléfono.
10	prg_email	varchar	40	40	0	Dirección e-mail.
11	fpg_forma_pago_id	smallint	2	5	0	Código de la forma de pago.
12	tar_tarjeta_id	smallint	2	5	0	Código de la tarjeta de crédito.
13	prg_num_tarjeta	char	16	16	0	Número de la tarjeta.
14	prg_venc_tarjeta	char	6	6	0	Vencimiento de la tarjeta.
15	prg_titular_tarjeta	varchar	70	70	0	Nombre del titular

DISEÑO DE LA TABLA

TABLA: **mvi_t_preregistro**			BASE DE DATOS: **DB_VIDEO**			
No.	**Columna**	**Tipo**	**Long.**	**Prc.**	**Scl.**	**Contenido**
16	prg_servicio_entrega	char	1	1	0	Indicador si desea entrega a domicilio.
17	prg_servicio_retiro	char	1	1	0	Indicador si desea retiro a domicilio.
18	prg_fecha_registro	datetime	8	0	0	Fecha de registro de la solicitud
19	prg_eval_fecha	datetime	8	0	0	Fecha de la evaluación.
20	prg_eval_calif	char	1	1	0	Calificación de la evaluación.
21	prg_eval_texto	varchar	80	80	0	Comentarios de la evaluación.
22	usu_eval_id	varchar	20	20	0	Login del usuario administrativo que evaluó.
23	cli_cliente_id	decimal	5	7	0	Código de cliente, si es aprobado.
24	prg_estado_reg	char	1	1	0	Estado del Registro

TIPO DE INDICE:	**COLUMNA(S):**
Primary Key	prg_tipo_identificación, prg_num_identificación

DISEÑO DE LA TABLA

TABLA: **mvi_t_publiweb**	BASE DE DATOS: **DB_VIDEO**

DESCRIPCIÓN: Contendrá la información de los detalles de las categorías que se publican en el sitio Web.

No.	Columna	Tipo	Long.	Prc.	Scl.	Contenido
1	pub_codigo_id	int	4	10	0	Código del detalle.
2	cat_categoria_id	int	4	10	0	Código de la categoría.
3	pub_titulo	varchar	50	50	0	Nombre del título que se va a redactar en detalle.
4	pub_descripcion	varchar	500	500	0	Detalle de la redacción del título.
5	pub_ruta_foto	varchar	50	50	0	Ruta del archivo de imagen que representa el titulo que se desea redactar.
6	pub_estado_reg	char	1	1	0	Estado del registro.
7	tit_titulo_id	decimal	5	7	0	Código del Título de la película.

TIPO DE INDICE:	**COLUMNA(S):**
Primary Key	pub_codigo_id
Foreign Key	cat_categoria_id referencia a mvi_m_categoriaweb

DISEÑO DE LA TABLA

TABLA: **mvi_t_talonario**	BASE DE DATOS: **DB_VIDEO**

DESCRIPCIÓN: Contendrá la información de los talonarios que adquieran los clientes.

No.	Columna	Tipo	Long.	Prc.	Scl.	Contenido
1	tal_talon_id	decimal	5	7	0	Código del talonario.
2	cli_cliente_id	decimal	5	7	0	Código del cliente.
3	tal_fecha_compra	datetime	8	0	0	Fecha de compra.
4	tti_talontipo_id	int	4	10	0	Código del tipo de talón.
5	tal_numero_tickets	int	4	10	0	Cupo de películas
6	tal_saldo_tickets	int	4	10	0	Saldo películas por alquilar.
7	tal_valor	decimal	5	5	2	Valor del talonario
8	trn_trans_id	decimal	5	7	0	Código de Transacción, que se genero cuando realizó la compra.
9	fpg_forma_pago_id	smallint	2	5	0	Código de forma de pago.
10	tar_tarjeta_id	smallint	2	5	0	Código de tarjeta de crédito.
11	tar_num_tarjeta	varchar	16	16	0	Número de la tarjeta.
12	tar_fecha_vencim	char	6	6	0	Vencimiento de la tarjeta.

DISEÑO DE LA TABLA

TABLA: **mvi_t_talonario**	BASE DE DATOS: **DB_VIDEO**

DESCRIPCIÓN: Contendrá la información de los talonarios que adquieran los clientes.

No.	Columna	Tipo	Long.	Prc.	Scl.	Contenido
13	tar_titular	varchar	60	60	0	Nombre del titular de la tarjeta.

TIPO DE INDICE:	COLUMNA(S):
Primary Key	tal_talon_id
Foreign Key	tti_talontipo_id referencia a mvi_m_talontipo
Foreign Key	cli_cliente_id referencia a mvi_m_cliente
Foreign Key	fpg_forma_pago_id referencia a mvi_m_forma_pago
Foreign Key	trn_trans_id referencia a mvi_t_transaccion

DISEÑO DE LA TABLA

TABLA: **mvi_t_transacción**	BASE DE DATOS: **DB_VIDEO**

DESCRIPCIÓN: Contendrá todos los registros que por concepto de compra de talonario y alquiler de películas se genere en e Sistema.

No.	Columna	Tipo	Long.	Prc.	Scl.	Contenido
1	trn_trans_id	decimal	5	7	0	Código de la transacción.
2	cli_cliente_id	decimal	5	7	0	Código del cliente.
3	trn_fecha_trans	datetime	8	0	0	Fecha de transacción.
4	ttr_tipotrans_id	smallint	2	5	0	Tipo de transacción (1 = Compra, 2 = Alquiler)
5	tal_talon_id	decimal	5	7	0	Código de Talonario referenciado.
6	trn_cantidad	int	4	10	0	Número de ítems de películas que afecta la transacción.
7	itm_item_id	decimal	5	7	0	Código del ítem de la película.
8	tit_titulo_id	decimal	5	7	0	Código del título.
9	med_medio_id	int	4	10	0	Código del medio.
10	trn_serv_entrega	char	1	1	0	Indicador si desea entrega a domicilio.
11	trn_serv_retiro	char	1	1	0	Indicador si desea retiro a domicilio.

DISEÑO DE LA TABLA

TABLA: **mvi_t_transacción**	BASE DE DATOS: **DB_VIDEO**

DESCRIPCIÓN: Contendrá todos los registros que por concepto de compra de talonario y alquiler de películas se genere en e Sistema.

No.	Columna	Tipo	Long.	Prc.	Scl.	Contenido
12	trn_fecha_entrega	datetime	8	0	0	Fecha de entrega de la película al cliente.
13	trn_tipo_entrega	char	1	1	0	Lugar de entrega (L =Local, D =Domicilio).
14	trn_fecha_estim_dev	datetime	8	0	0	Fecha estimada de devolución de la película.
15	trn_fecha_dev	datetime	8	0	0	Fecha que devuelve la película el cliente.
16	trn_status	char	1	1	0	Estado de la transacción.

TIPO DE INDICE:	**COLUMNA(S):**
Primary Key	Trn_trans_id
Foreign Key	cli_cliente_id referencia a mvi_m_cliente
Foreign Key	ttr_tipotrans_id referencia a mvi_m_tipotrans
Foreign Key	itm_item_id referencia a mvi_m_item

5.4. Procedimientos almacenados.

5.4.1. Procedimientos almacenados de inserción de registros.

PROCEDIMIENTO ALMACENADO					
NOMBRE: **sp_add_CategoriaWeb**		BASE DE DATOS: **DB_VIDEO**			
DESCRIPCIÓN: Permite insertar un registro en la tabla mvi_m_categoriaweb.					
No.	**Parámetro**	**Tipo**	**Long.**	**Prc.**	**Scl.**
1	@i_cat_id	int	4	10	0
2	@i_cat_descripción	varchar	25	25	0
COMPONENTE DE DATO QUE LE REFERENCIA: DB_CATEGORIAWEBC					

PROCEDIMIENTO ALMACENADO					
NOMBRE: **sp_add_cliente**		BASE DE DATOS: **DB_VIDEO**			
DESCRIPCIÓN: Permite insertar un registro en la tabla mvi_m_cliente.					
No.	**Parámetro**	**Tipo**	**Long.**	**Prc.**	**Scl.**
1	@i_cli_cliente_id	decimal	5	7	0
2	@i_cli_nombre	varchar	50	50	0
3	@i_cli_apellido	varchar	50	50	0
4	@i_cli_tipo_identificacio	char	1	1	0

PROCEDIMIENTO ALMACENADO

NOMBRE: **sp_add_cliente**	BASE DE DATOS: **DB_VIDEO**

DESCRIPCIÓN: Permite insertar un registro en la tabla mvi_m_cliente.

No.	Parámetro	Tipo	Long.	Prc.	Scl.
5	@i_cli_num_identificacion	char	15	15	0
6	@i_cli_direccion	varchar	50	50	0
7	@i_sec_sector_id	smallint	2	5	0
8	@i_cli_telefono	varchar	15	15	0
9	@i_cli_email	varchar	30	30	0
10	@i_fpg_forma_pago_id	smallint	2	5	0
11	@i_bco_banco_id	smallint	2	5	0
12	@i_cli_num_cta_bco	char	15	15	0
13	@i_tar_tarjeta_id	smallint	2	5	0
14	@i_cli_num_tarjeta	char	16	16	0
15	@i_cli_tar_fecha_vencim	datetime	8	0	0
16	@i_cli_servicio_entrega	char	1	1	0
17	@i_cli_servicio_retiro	char	1	1	0
18	@i_cli_login	char	8	8	0
19	@i_cli_clave	char	8	8	0
20	@i_cli_estado_reg	char	1	1	0

COMPONENTE DE DATO QUE LE REFERENCIA: DB_CLIENTEC

PROCEDIMIENTO ALMACENADO

NOMBRE: **sp_add_Item**	BASE DE DATOS: **DB_VIDEO**

DESCRIPCIÓN: Permite insertar un registro en la tabla mvi_m_item.

No.	Parámetro	Tipo	Long.	Prc.	Scl.
1	@i_itm_item_id	decimal	5	7	0
2	@i_tit_titulo_id	decimal	5	7	0
3	@i_med_medio_id	int	4	10	0
4	@i_idm_idioma_id	smallint	2	5	0
5	@i_itm_estado_reg	char	1	1	0
6	@i_itm_estado_renta	char	1	1	0

COMPONENTE DE DATO QUE LE REFERENCIA: DB_ITEMC

PROCEDIMIENTO ALMACENADO					
NOMBRE: **sp_add_medio**		BASE DE DATOS: **DB_VIDEO**			
DESCRIPCIÓN: Permite insertar un registro en la tabla mvi_m_medio.					
No.	**Parámetro**	**Tipo**	**Long.**	**Prc.**	**Scl.**
1	@i_medio_id	int	4	10	0
2	@i_descripcion	char	25	25	0
COMPONENTE DE DATO QUE LE REFERENCIA: DB_MEDIOC					

PROCEDIMIENTO ALMACENADO					
NOMBRE: **sp_add_PreRegistro**		BASE DE DATOS: **DB_VIDEO**			
DESCRIPCIÓN: Permite insertar un registro en la tabla mvi_t_preregistro.					
No.	**Parámetro**	**Tipo**	**Long.**	**Prc.**	**Scl.**
1	@i_prg_tipo_identif	char	1	1	0
2	@i_prg_num_identif	char	15	15	0
3	@i_prg_login	char	8	8	0
4	@i_prg_clave	char	8	8	0
5	@i_prg_nombre	varchar	50	50	0
6	@i_prg_apellido	varchar	50	50	0

PROCEDIMIENTO ALMACENADO

NOMBRE: **sp_add_PreRegistro**	BASE DE DATOS: **DB_VIDEO**

DESCRIPCIÓN: Permite insertar un registro en la tabla mvi_t_preregistro.

No.	Parámetro	Tipo	Long.	Prc.	Scl.
7	@i_sec_sector_id	smallint	2	5	0
8	@i_prg_direccion	varchar	70	70	0
9	@i_prg_telefono	varchar	15	15	0
10	@i_prg_email	varchar	40	40	0
11	@i_fpg_forma_pago_id	smallint	2	5	0
12	@i_tar_tarjeta_id	smallint	2	5	0
13	@i_prg_num_tarjeta	char	16	16	0
14	@i_prg_venc_tarjeta	char	6	6	0
15	@i_prg_titular_tarjeta	varchar	70	70	0
16	@i_prg_servicio_entrega	char	1	1	0
17	@i_prg_servicio_retiro	char	1	1	0
18	@i_prg_estado_reg	char	1	1	0

COMPONENTE DE DATO QUE LE REFERENCIA: DB_PREREGISTROC

PROCEDIMIENTO ALMACENADO

NOMBRE: **sp_add_PubliWeb**	BASE DE DATOS: **DB_VIDEO**

DESCRIPCIÓN: Permite insertar un registro en la tabla mvi_t_publiweb.

No.	Parámetro	Tipo	Long.	Prc.	Scl.
1	@i_pub_codigo_id	int	4	10	0
2	@i_pub_categoria_id	int	4	10	0
3	@i_pub_titulo	varchar	50	50	0
4	@i_pub_descripcion	varchar	500	500	0
5	@i_pub_ruta_foto	varchar	50	50	0
6	@i_pub_estado_reg	char	1	1	0
7	@i_pub_titulo_id	decimal	5	7	0

COMPONENTE DE DATO QUE LE REFERENCIA: DB_PUBLIWEBC

PROCEDIMIENTO ALMACENADO					
NOMBRE: **sp_add_secuencial**		BASE DE DATOS: **DB_VIDEO**			
DESCRIPCIÓN: Permite insertar un registro en la tabla mvi_m_secuencial.					
No.	**Parámetro**	**Tipo**	**Long.**	**Prc.**	**Scl.**
1	@i_sec_id	smallint	2	5	0
2	@i_descripcion	char	50	50	0
3	@i_sec_valor	decimal	5	5	0
COMPONENTE DE DATO QUE LE REFERENCIA: DB_SECUENCIALC					

PROCEDIMIENTO ALMACENADO					
NOMBRE: **sp_add_TalonTipo**		BASE DE DATOS: **DB_VIDEO**			
DESCRIPCIÓN: Permite insertar un registro en la tabla mvi_m_talontipo.					
No.	**Parámetro**	**Tipo**	**Long.**	**Prc.**	**Scl.**
1	@i_tti_talontipo_id	int	4	10	0
2	@i_tti_descripcion	varchar	80	80	0
3	@i_tti_num_tickets	int	4	10	0
4	@i_tti_precio	decimal	5	5	2
COMPONENTE DE DATO QUE LE REFERENCIA: DB_TALONTIPOC					

PROCEDIMIENTO ALMACENADO

NOMBRE: **sp_add_Talonario**	BASE DE DATOS: **DB_VIDEO**

DESCRIPCIÓN: Permite insertar un registro en la tabla mvi_t_talonario.

No.	Parámetro	Tipo	Long.	Prc.	Scl.
1	@i_tal_talon_id	decimal	5	7	0
2	@i_cli_cliente_id	decimal	5	7	0
3	@i_tti_talontipo_id	int	4	10	0
4	@i_tal_num_tickets	int	4	10	0
5	@i_tal_valor	decimal	5	5	2
6	@i_fpg_formapago_id	smallint	2	5	0
7	@i_tar_tarjeta_id	smallint	2	5	0
8	@i_tar_numtarjeta	varchar	16	16	0
9	@i_tar_fecvencim	char	6	6	0
10	@i_tar_titular	varchar	60	60	0

COMPONENTE DE DATO QUE LE REFERENCIA: DB_TALONARIOC

PROCEDIMIENTO ALMACENADO					
NOMBRE: **sp_add_Tarjeta**		BASE DE DATOS: **DB_VIDEO**			
DESCRIPCIÓN: Permite insertar un registro en la tabla mvi_m_tarjeta.					
No.	**Parámetro**	**Tipo**	**Long.**	**Prc.**	**Scl.**
1	@i_tar_id	int	4	10	0
2	@i_tar_descripcion	char	25	25	0
COMPONENTE DE DATO QUE LE REFERENCIA: DB_TARJETAC					

PROCEDIMIENTO ALMACENADO					
NOMBRE: **sp_add_Título**		BASE DE DATOS: **DB_VIDEO**			
DESCRIPCIÓN: Permite insertar un registro en la tabla mvi_m_título.					
No.	**Parámetro**	**Tipo**	**Long.**	**Prc.**	**Scl.**
1	@i_tit_titulo_id	decimal	5	7	0
2	@i_tit_nombre	varchar	100	100	0
3	@i_tit_nombre_orig	varchar	100	100	0
4	@i_tit_sinopsis	varchar	500	500	0
5	@i_tit_productor	varchar	50	50	0
6	@i_tit_director	varchar	60	60	0
7	@i_tit_reparto	varchar	100	100	0
8	@i_cen_censura_id	char	1	1	0

PROCEDIMIENTO ALMACENADO

NOMBRE: **sp_add_Título**	BASE DE DATOS: **DB_VIDEO**

DESCRIPCIÓN: Permite insertar un registro en la tabla mvi_m_título.

No.	Parámetro	Tipo	Long.	Prc.	Scl.
9	@i_gen_genero_1	char	3	3	0
10	@i_gen_genero_2	char	3	3	0
11	@i_gen_genero_3	char	3	3	0
12	@i_tit_nominacion_sn	char	1	1	0
13	@i_tit_nominaciones	varchar	100	100	0
14	@i_tit_premios_sn	char	1	1	0
15	@i_tit_premios	varchar	100	100	0
16	@i_tit_oscares_sn	char	1	1	0
17	@i_tit_duracion	char	5	5	0
18	@i_tit_produccion	int	4	10	0
19	@i_tit_logo_pelicula	varchar	50	50	0

COMPONENTE DE DATO QUE LE REFERENCIA: DB_TITULOC

5.4.2. Procedimientos almacenados de consulta individual del registro.

PROCEDIMIENTO ALMACENADO					
NOMBRE: **sp_cons_ById_CategoriaWeb**			BASE DE DATOS: **DB_VIDEO**		
DESCRIPCIÓN: Permite consultar un registro en la tabla mvi_m_categoriaweb.					
No.	**Parámetro**	**Tipo**	**Long.**	**Prc.**	**Scl.**
1	@i_cat_id	int	4	10	0
COMPONENTE DE DATO QUE LE REFERENCIA: DB_CATEGORIAWEBC					

PROCEDIMIENTO ALMACENADO					
NOMBRE: **sp_cons_ById_Censura**			BASE DE DATOS: **DB_VIDEO**		
DESCRIPCIÓN: Permite consultar un registro en la tabla mvi_m_censura.					
No.	**Parámetro**	**Tipo**	**Long.**	**Prc.**	**Scl.**
1	@i_cen_id	char	3	3	0
COMPONENTE DE DATO QUE LE REFERENCIA: DB_CENSURAC					

PROCEDIMIENTO ALMACENADO					
NOMBRE: **sp_cons_ById_Cliente**		BASE DE DATOS: **DB_VIDEO**			
DESCRIPCIÓN: Permite consultar un registro en la tabla mvi_m_cliente.					
No.	**Parámetro**	**Tipo**	**Long.**	**Prc.**	**Scl.**
1	@i_cli_id	decimal	5	7	0
COMPONENTE DE DATO QUE LE REFERENCIA: DB_CLIENTEC					

PROCEDIMIENTO ALMACENADO					
NOMBRE: **sp_cons_ById_FormaPago**		BASE DE DATOS: **DB_VIDEO**			
DESCRIPCIÓN: Permite consultar un registro en la tabla mvi_m_forma_pago.					
No.	**Parámetro**	**Tipo**	**Long.**	**Prc.**	**Scl.**
1	@i_fpg_id	smallint	2	5	0
COMPONENTE DE DATO QUE LE REFERENCIA: DB_FORMAPAGOC					

PROCEDIMIENTO ALMACENADO

NOMBRE: **sp_cons_ById_Genero**	BASE DE DATOS: **DB_VIDEO**

DESCRIPCIÓN: Permite consultar un registro en la tabla mvi_m_genero.

No.	Parámetro	Tipo	Long.	Prc.	Scl.
1	@i_gen_id	char	3	3	0

COMPONENTE DE DATO QUE LE REFERENCIA: DB_GENEROC

PROCEDIMIENTO ALMACENADO

NOMBRE: **sp_cons_ById_Idioma**	BASE DE DATOS: **DB_VIDEO**

DESCRIPCIÓN: Permite consultar un registro en la tabla mvi_m_idioma.

No.	Parámetro	Tipo	Long.	Prc.	Scl.
1	@i_idm_idioma_id	smallint	2	5	0

COMPONENTE DE DATO QUE LE REFERENCIA: DB_IDIOMAC

PROCEDIMIENTO ALMACENADO					
NOMBRE: **sp_cons_ById_Ítem**		BASE DE DATOS: **DB_VIDEO**			
DESCRIPCIÓN: Permite consultar un registro en la tabla mvi_m_ítem.					
No.	**Parámetro**	**Tipo**	**Long.**	**Prc.**	**Scl.**
1	@i_itm_item_id	decimal	5	7	0
COMPONENTE DE DATO QUE LE REFERENCIA: DB_ÍTEMC					

PROCEDIMIENTO ALMACENADO					
NOMBRE: **sp_cons_ById_Medio**		BASE DE DATOS: **DB_VIDEO**			
DESCRIPCIÓN: Permite consultar un registro en la tabla mvi_m_medio.					
No.	**Parámetro**	**Tipo**	**Long.**	**Prc.**	**Scl.**
1	@i_medio_id	int	4	10	0
COMPONENTE DE DATO QUE LE REFERENCIA: DB_MEDIOC					

PROCEDIMIENTO ALMACENADO	
NOMBRE: **sp_cons_ById_Parámetro**	BASE DE DATOS: **DB_VIDEO**
DESCRIPCIÓN: Permite consultar un registro en la tabla mvi_m_parámetro.	

No.	Parámetro	Tipo	Long.	Prc.	Scl.
1	@i_param_id	varchar	20	20	0

COMPONENTE DE DATO QUE LE REFERENCIA: DB_PARÁMETROC

PROCEDIMIENTO ALMACENADO	
NOMBRE: **sp_cons_ById_PreRegistro**	BASE DE DATOS: **DB_VIDEO**
DESCRIPCIÓN: Permite consultar un registro en la tabla mvi_t_preregistro.	

No.	Parámetro	Tipo	Long.	Prc.	Scl.
1	@i_prg_tipo_identif	char	1	1	0
2	@i_prg_num_identif	char	15	15	0

COMPONENTE DE DATO QUE LE REFERENCIA: DB_PREREGISTROC

PROCEDIMIENTO ALMACENADO					
NOMBRE: **sp_cons_ById_PubliWeb**		BASE DE DATOS: **DB_VIDEO**			
DESCRIPCIÓN: Permite consultar un registro en la tabla mvi_t_publiweb.					
No.	**Parámetro**	**Tipo**	**Long.**	**Prc.**	**Scl.**
1	@i_pub_codigo_id	int	4	10	0
COMPONENTE DE DATO QUE LE REFERENCIA: DB_PUBLIWEBC					

PROCEDIMIENTO ALMACENADO					
NOMBRE: **sp_cons_ById_Sector**		BASE DE DATOS: **DB_VIDEO**			
DESCRIPCIÓN: Permite consultar un registro en la tabla mvi_m_sector.					
No.	**Parámetro**	**Tipo**	**Long.**	**Prc.**	**Scl.**
1	@i_sct_id	smallint	2	5	0
COMPONENTE DE DATO QUE LE REFERENCIA: DB_SECTORC					

PROCEDIMIENTO ALMACENADO	
NOMBRE: **sp_cons_ById_Secuencial**	BASE DE DATOS: **DB_VIDEO**
DESCRIPCIÓN: Permite consultar un registro en la tabla mvi_m_secuencial.	

No.	Parámetro	Tipo	Long.	Prc.	Scl.
1	@i_sec_id	smallint	2	5	0

COMPONENTE DE DATO QUE LE REFERENCIA: DB_SECUENCIALC

PROCEDIMIENTO ALMACENADO	
NOMBRE: **sp_cons_ById_TalonTipo**	BASE DE DATOS: **DB_VIDEO**
DESCRIPCIÓN: Permite consultar un registro en la tabla mvi_m_talontipo.	

No.	Parámetro	Tipo	Long.	Prc.	Scl.
1	@i_tti_talontipo_id	int	4	10	0

COMPONENTE DE DATO QUE LE REFERENCIA: DB_TALONTIPOC

PROCEDIMIENTO ALMACENADO					
NOMBRE: **sp_cons_ById_Tarjeta**		BASE DE DATOS: **DB_VIDEO**			
DESCRIPCIÓN: Permite consultar un registro en la tabla mvi_m_tarjeta.					
No.	**Parámetro**	**Tipo**	**Long.**	**Prc.**	**Scl.**
1	@i_trj_id	smallint	2	5	0
COMPONENTE DE DATO QUE LE REFERENCIA: DB_TARJETAC					

PROCEDIMIENTO ALMACENADO					
NOMBRE: **sp_cons_ById_Titulo**		BASE DE DATOS: **DB_VIDEO**			
DESCRIPCIÓN: Permite consultar un registro en la tabla mvi_m_titulo.					
No.	**Parámetro**	**Tipo**	**Long.**	**Prc.**	**Scl.**
1	@i_titulo_id	decimal	5	7	0
COMPONENTE DE DATO QUE LE REFERENCIA: DB_TITULOC					

5.4.3. Procedimientos almacenados de eliminación.

PROCEDIMIENTO ALMACENADO					
NOMBRE: **sp_del_CategoriaWeb**		BASE DE DATOS: **DB_VIDEO**			
DESCRIPCIÓN: Permite eliminar un registro en la tabla mvi_m_categoriaweb.					
No.	**Parámetro**	**Tipo**	**Long.**	**Prc.**	**Scl.**
1	@i_cat_id	int	4	10	0
COMPONENTE DE DATO QUE LE REFERENCIA: DB_CATEGORIAWEBC					

PROCEDIMIENTO ALMACENADO					
NOMBRE: **sp_del_Medio**		BASE DE DATOS: **DB_VIDEO**			
DESCRIPCIÓN: Permite eliminar un registro en la tabla mvi_m_medio.					
No.	**Parámetro**	**Tipo**	**Long.**	**Prc.**	**Scl.**
1	@i_medio_id	int	4	10	0
COMPONENTE DE DATO QUE LE REFERENCIA: DB_MEDIOC					

PROCEDIMIENTO ALMACENADO					
NOMBRE: **sp_del_PubliWeb**		BASE DE DATOS: **DB_VIDEO**			
DESCRIPCIÓN: Permite eliminar un registro en la tabla mvi_t_publiweb.					
No.	**Parámetro**	**Tipo**	**Long.**	**Prc.**	**Scl.**
1	@i_pub_codigo_id	int	4	10	0
COMPONENTE DE DATO QUE LE REFERENCIA: DB_PUBLIWEBC					

PROCEDIMIENTO ALMACENADO					
NOMBRE: **sp_del_TalonTipo**		BASE DE DATOS: **DB_VIDEO**			
DESCRIPCIÓN: Permite eliminar un registro en la tabla mvi_m_talontipo.					
No.	**Parámetro**	**Tipo**	**Long.**	**Prc.**	**Scl.**
1	@i_tti_talontipo_id	int	4	10	0
COMPONENTE DE DATO QUE LE REFERENCIA: DB_TALONTIPOC					

PROCEDIMIENTO ALMACENADO					
NOMBRE: **sp_del_Tarjeta**		BASE DE DATOS: **DB_VIDEO**			
DESCRIPCIÓN: Permite eliminar un registro en la tabla mvi_m_tarjeta.					
No.	**Parámetro**	**Tipo**	**Long.**	**Prc.**	**Scl.**
1	@i_trj_id	smallint	2	5	0
COMPONENTE DE DATO QUE LE REFERENCIA: DB_TARJETAC					

5.4.4. Procedimientos almacenados de consulta de todos los registros de una tabla.

PROCEDIMIENTO ALMACENADO	
NOMBRE: **sp_ListAll_CategoriaWeb**	BASE DE DATOS: **DB_VIDEO**
DESCRIPCIÓN: Permite consultar todos los registros de la tabla mvi_m_categoriaweb.	
COMPONENTE DE DATO QUE LE REFERENCIA: DB_CATEGORIAWEBC	

PROCEDIMIENTO ALMACENADO	
NOMBRE: **sp_ListAll_Censura**	BASE DE DATOS: **DB_VIDEO**
DESCRIPCIÓN: Permite consultar todos los registros de la tabla mvi_m_censura.	
COMPONENTE DE DATO QUE LE REFERENCIA: DB_CENSURAC	

PROCEDIMIENTO ALMACENADO	
NOMBRE: **sp_ListAll_Cliente**	BASE DE DATOS: **DB_VIDEO**
DESCRIPCIÓN: Permite consultar todos los registros de la tabla mvi_m_cliente.	
COMPONENTE DE DATO QUE LE REFERENCIA: DB_CLIENTEC	

PROCEDIMIENTO ALMACENADO	
NOMBRE: **sp_ListAll_FormaPago**	BASE DE DATOS: **DB_VIDEO**
DESCRIPCIÓN: Permite consultar todos los registros de la tabla mvi_m_forma_pago.	
COMPONENTE DE DATO QUE LE REFERENCIA: DB_FORMAPAGOC	

PROCEDIMIENTO ALMACENADO	
NOMBRE: **sp_ListAll_Genero**	BASE DE DATOS: **DB_VIDEO**
DESCRIPCIÓN: Permite consultar todos los registros de la tabla mvi_m_genero.	
COMPONENTE DE DATO QUE LE REFERENCIA: DB_GENEROC	

PROCEDIMIENTO ALMACENADO	
NOMBRE: **sp_ListAll_Idioma**	BASE DE DATOS: **DB_VIDEO**
DESCRIPCIÓN: Permite consultar todos los registros de la tabla mvi_m_idioma.	
COMPONENTE DE DATO QUE LE REFERENCIA: DB_IDIOMAC	

PROCEDIMIENTO ALMACENADO	
NOMBRE: **sp_ListAll_Ítem**	BASE DE DATOS: **DB_VIDEO**
DESCRIPCIÓN: Permite consultar todos los registros de la tabla mvi_m_ítem.	
COMPONENTE DE DATO QUE LE REFERENCIA: DB_ÍTEMC	

PROCEDIMIENTO ALMACENADO	
NOMBRE: **sp_ListAll_Medio**	BASE DE DATOS: **DB_VIDEO**
DESCRIPCIÓN: Permite consultar todos los registros de la tabla mvi_m_medio.	
COMPONENTE DE DATO QUE LE REFERENCIA: DB_MEDIOC	

PROCEDIMIENTO ALMACENADO	
NOMBRE: **sp_ListAll_Parámetro**	BASE DE DATOS: **DB_VIDEO**
DESCRIPCIÓN: Permite consultar todos los registros de la tabla mvi_m_parámetro.	
COMPONENTE DE DATO QUE LE REFERENCIA: DB_PARÁMETROC	

PROCEDIMIENTO ALMACENADO	
NOMBRE: **sp_ListAll_Secuencial**	BASE DE DATOS: **DB_VIDEO**
DESCRIPCIÓN: Permite consultar todos los registros de la tabla mvi_m_secuencial.	
COMPONENTE DE DATO QUE LE REFERENCIA: DB_SECUENCIALC	

PROCEDIMIENTO ALMACENADO	
NOMBRE: **sp_ListAll_TalonTipo**	BASE DE DATOS: **DB_VIDEO**
DESCRIPCIÓN: Permite consultar todos los registros de la tabla mvi_m_talontipo.	
COMPONENTE DE DATO QUE LE REFERENCIA: DB_TALONTIPOC	

PROCEDIMIENTO ALMACENADO	
NOMBRE: **sp_ListAll_Titulo**	BASE DE DATOS: **DB_VIDEO**
DESCRIPCIÓN: Permite consultar todos los registros de la tabla mvi_m_titulo.	
COMPONENTE DE DATO QUE LE REFERENCIA: DB_TITULOC	

5.4.5. Procedimientos almacenados de consulta con parámetros.

PROCEDIMIENTO ALMACENADO					
NOMBRE: **sp_login**		BASE DE DATOS: **DB_VIDEO**			
DESCRIPCIÓN: Permite determinar si el login y clave ingresados están autorizados para ingresar al sistema.					
No.	**Parámetro**	**Tipo**	**Long.**	**Prc.**	**Scl.**
1	@i_cli_login	varchar	8	8	0
2	@i_cli_clave	varchar	8	8	0
COMPONENTE DE DATO QUE LE REFERENCIA: DB_CLIENTEC					

PROCEDIMIENTO ALMACENADO					
NOMBRE: **sp_cons_InfoCuenta**		BASE DE DATOS: **DB_VIDEO**			
DESCRIPCIÓN: Nos devuelve el cupo de películas disponibles para rentar, como también el numero de películas que el cliente tiene en su poder.					
No.	**Parámetro**	**Tipo**	**Long.**	**Prc.**	**Scl.**
1	@i_cli_id	decimal	5	7	0
COMPONENTE DE DATO QUE LE REFERENCIA: DB_CLIENTEC					

PROCEDIMIENTO ALMACENADO

NOMBRE: **sp_list_ByLocation_FormaPago**	BASE DE DATOS: **DB_VIDEO**

DESCRIPCIÓN: Nos devuelve las formas de pago, que se pueden recibir dependiendo la localización (por ejemplo, en el sitio Web sólo se permite tarjeta de crédito).

No.	Parámetro	Tipo	Long.	Prc.	Scl.
1	@i_fpg_location	char	1	1	0

COMPONENTE DE DATO QUE LE REFERENCIA: DB_FORMAPAGOC

PROCEDIMIENTO ALMACENADO

NOMBRE: **sp_cons_GetDisponible_Item**	BASE DE DATOS: **DB_VIDEO**

DESCRIPCIÓN: Nos devuelve el código del ítem que coincida el código del título y el código del medio solicitado por el cliente, caso contrariom le devuelve 0.

No.	Parámetro	Tipo	Long.	Prc.	Scl.
1	@i_titulo_id	decimal	5	7	0
2	@i_medio_id	int	4	10	0

COMPONENTE DE DATO QUE LE REFERENCIA: DB_ITEMC

PROCEDIMIENTO ALMACENADO	
NOMBRE: **sp_list_ByEstado_PreRegistro**	BASE DE DATOS: **DB_VIDEO**
DESCRIPCIÓN: Nos devuelve los registros de las solicitudes ingresadas en el sitio Web, se puede consultar por su estado, que puede ser: ingresadas, aprobadas, rechazadas.	

No.	Parámetro	Tipo	Long.	Prc.	Scl.
1	@i_prg_estado_reg	char	1	1	0

COMPONENTE DE DATO QUE LE REFERENCIA: DB_PREREGISTROC

PROCEDIMIENTO ALMACENADO	
NOMBRE: **sp_list_bycat_PubliWeb**	BASE DE DATOS: **DB_VIDEO**
DESCRIPCIÓN: Nos devuelve los registros de detalle de las categorías que se publicarán en el sitio Web.	

No.	Parámetro	Tipo	Long.	Prc.	Scl.
1	@i_cat_categoria_id	int	4	10	0

COMPONENTE DE DATO QUE LE REFERENCIA: DB_PUBLIWEBC

PROCEDIMIENTO ALMACENADO

NOMBRE: **sp_cons_ById_TituloxMedio**	BASE DE DATOS: **DB_VIDEO**

DESCRIPCIÓN: Nos permite determinar si un determinado título, existe en el medio consultado.

No.	Parámetro	Tipo	Long.	Prc.	Scl.
1	@i_titulo_id	decimal	5	7	0
2	@i_medio_id	int	4	10	0

COMPONENTE DE DATO QUE LE REFERENCIA: DB_TITULOC

PROCEDIMIENTO ALMACENADO

NOMBRE: **sp_cons_ByParam_TituloxMedio**	BASE DE DATOS: **DB_VIDEO**

DESCRIPCIÓN: Nos ayuda para realizar la búsqueda de películas, basados en el medio, y un argumento a buscar, sea éste: título de la película, nombre del director, nombres de actores en el reparto.

No.	Parámetro	Tipo	Long.	Prc.	Scl.
1	@i_medio_id	int	4	10	0
2	@i_busquedad_id	smallint	2	5	0
3	@i_argumento	varchar	50	50	0

COMPONENTE DE DATO QUE LE REFERENCIA: DB_TITULOC

PROCEDIMIENTO ALMACENADO					
NOMBRE: **sp_List_ByEstado_Transacción**			BASE DE DATOS: **DB_VIDEO**		
DESCRIPCIÓN: Nos devuelve todos los registros referente al alquiler, películas alquiladas, entregadas, recibidas. También lo podemos filtrar alternativamente por el código del cliente.					
No.	**Parámetro**	**Tipo**	**Long.**	**Prc.**	**Scl.**
1	@i_trn_estado	char	1	1	0
2	@i_cli_cliente_id	decimal	5	7	0
COMPONENTE DE DATO QUE LE REFERENCIA: DB_TRANSACCIONC					

5.4.6. Procedimientos almacenados de actualización.

PROCEDIMIENTO ALMACENADO					
NOMBRE: **sp_upd_CategoriaWeb**			BASE DE DATOS: **DB_VIDEO**		
DESCRIPCIÓN: Permite actualizar un registro en la tabla mvi_m_categoriaweb.					
No.	**Parámetro**	**Tipo**	**Long.**	**Prc.**	**Scl.**
1	@i_cat_id	int	4	10	0
2	@i_cat_descripción	varchar	25	25	0
COMPONENTE DE DATO QUE LE REFERENCIA: DB_CATEGORIAWEBC					

PROCEDIMIENTO ALMACENADO

NOMBRE: **sp_upd_cliente**	BASE DE DATOS: **DB_VIDEO**

DESCRIPCIÓN: Permite actualizar un registro en la tabla mvi_m_cliente.

No.	Parámetro	Tipo	Long.	Prc.	Scl.
1	@i_cli_cliente_id	decimal	5	7	0
2	@i_cli_nombre	varchar	50	50	0
3	@i_cli_apellido	varchar	50	50	0
4	@i_cli_tipo_identificacio	char	1	1	0
5	@i_cli_num_identificacion	char	15	15	0
6	@i_cli_ciudad	char	20	20	0
7	@i_sec_sector_id	smallint	2	5	0
8	@i_cli_telefono	varchar	15	15	0
9	@i_cli_email	varchar	30	30	0
10	@i_fpg_forma_pago_id	smallint	2	5	0
11	@i_tar_tarjeta_id	smallint	2	5	0
12	@i_cli_num_tarjeta	char	16	16	0
13	@i_cli_tar_fecha_vencim	datetime	8	0	0
14	@i_cli_servicio_entrega	char	1	1	0
15	@i_cli_servicio_retiro	char	1	1	0
16	@i_cli_login	char	8	8	0
17	@i_cli_clave	char	8	8	0
18	@i_cli_estado_reg	char	1	1	0

COMPONENTE DE DATO QUE LE REFERENCIA: DB_CLIENTEC

PROCEDIMIENTO ALMACENADO					
NOMBRE: **sp_upd_Item**		BASE DE DATOS: **DB_VIDEO**			
DESCRIPCIÓN: Permite actualizar un registro en la tabla mvi_m_item.					
No.	**Parámetro**	**Tipo**	**Long.**	**Prc.**	**Scl.**
1	@i_itm_item_id	decimal	5	7	0
2	@i_tit_titulo_id	decimal	5	7	0
3	@i_med_medio_id	int	4	10	0
4	@i_idm_idioma_id	smallint	2	5	0
5	@i_itm_estado_reg	char	1	1	0
6	@i_itm_estado_renta	char	1	1	0
COMPONENTE DE DATO QUE LE REFERENCIA: DB_ITEMC					

PROCEDIMIENTO ALMACENADO					
NOMBRE: **sp_upd_medio**		BASE DE DATOS: **DB_VIDEO**			
DESCRIPCIÓN: Permite actualizar un registro en la tabla mvi_m_medio.					
No.	**Parámetro**	**Tipo**	**Long.**	**Prc.**	**Scl.**
1	@i_medio_id	int	4	10	0
2	@i_descripcion	char	25	25	0
COMPONENTE DE DATO QUE LE REFERENCIA: DB_MEDIOC					

PROCEDIMIENTO ALMACENADO					
NOMBRE: **sp_upd_PreRegistro**		BASE DE DATOS: **DB_VIDEO**			
DESCRIPCIÓN: Permite actualizar un registro en la tabla mvi_t_preregistro.					
No.	**Parámetro**	**Tipo**	**Long.**	**Prc.**	**Scl.**
1	@i_prg_tipo_identif	char	1	1	0
2	@i_prg_num_identif	char	15	15	0
3	@i_prg_eval_fecha	datetime	8	0	0
4	@i_prg_eval_calif	char	1	1	0
5	@i_prg_eval_texto	varchar	80	80	0
6	@i_prg_eval_usu_id	varchar	20	20	0

PROCEDIMIENTO ALMACENADO					
NOMBRE: **sp_upd_PreRegistro**		BASE DE DATOS: **DB_VIDEO**			
DESCRIPCIÓN: Permite actualizar un registro en la tabla mvi_t_preregistro.					
No.	**Parámetro**	**Tipo**	**Long.**	**Prc.**	**Scl.**
7	@i_prg_eval_cli_id	decimal	5	7	0
8	@i_prg_estado_reg	char	1	1	0
COMPONENTE DE DATO QUE LE REFERENCIA: DB_PREREGISTROC					

PROCEDIMIENTO ALMACENADO					
NOMBRE: **sp_upd_PubliWeb**		BASE DE DATOS: **DB_VIDEO**			
DESCRIPCIÓN: Permite actualizar un registro en la tabla mvi_t_publiweb.					
No.	**Parámetro**	**Tipo**	**Long.**	**Prc.**	**Scl.**
1	@i_pub_codigo_id	int	4	10	0
2	@i_pub_categoria_id	int	4	10	0
3	@i_pub_titulo	varchar	50	50	0
4	@i_pub_descripcion	varchar	500	500	0
5	@i_pub_ruta_foto	varchar	50	50	0
6	@i_pub_estado_reg	char	1	1	0
7	@i_pub_titulo_id	decimal	5	7	0
COMPONENTE DE DATO QUE LE REFERENCIA: DB_PUBLIWEBC					

PROCEDIMIENTO ALMACENADO	
NOMBRE: **sp_upd_secuencial**	BASE DE DATOS: **DB_VIDEO**
DESCRIPCIÓN: Permite actualizar un registro en la tabla mvi_m_secuencial, es de éste procedimiento que vamos actualizando los correlativos de las tablas maestras y transaccionales.	

No.	Parámetro	Tipo	Long.	Prc.	Scl.
1	@i_sec_id	smallint	2	5	0
2	@i_descripcion	char	50	50	0
3	@i_sec_valor	decimal	5	5	0

COMPONENTE DE DATO QUE LE REFERENCIA: DB_SECUENCIALC

PROCEDIMIENTO ALMACENADO	
NOMBRE: **sp_upd_TalonTipo**	BASE DE DATOS: **DB_VIDEO**
DESCRIPCIÓN: Permite actualizar un registro en la tabla mvi_m_talontipo.	

No.	Parámetro	Tipo	Long.	Prc.	Scl.
1	@i_tti_talontipo_id	int	4	10	0
2	@i_tti_descripcion	varchar	80	80	0
3	@i_tti_num_tickets	int	4	10	0
4	@i_tti_precio	decimal	5	5	2

COMPONENTE DE DATO QUE LE REFERENCIA: DB_TALONTIPOC

PROCEDIMIENTO ALMACENADO	
NOMBRE: **sp_upd_Tarjeta**	BASE DE DATOS: **DB_VIDEO**
DESCRIPCIÓN: Permite actualizar un registro en la tabla mvi_m_tarjeta.	

No.	Parámetro	Tipo	Long.	Prc.	Scl.
1	@i_tar_id	int	4	10	0
2	@i_tar_descripcion	char	25	25	0

COMPONENTE DE DATO QUE LE REFERENCIA: DB_TARJETAC

PROCEDIMIENTO ALMACENADO	
NOMBRE: **sp_upd_Título**	BASE DE DATOS: **DB_VIDEO**
DESCRIPCIÓN: Permite actualizar un registro en la tabla mvi_m_título.	

No.	Parámetro	Tipo	Long.	Prc.	Scl.
1	@i_tit_titulo_id	decimal	5	7	0
2	@i_tit_nombre	varchar	100	100	0
3	@i_tit_nombre_orig	varchar	100	100	0
4	@i_tit_sinopsis	varchar	500	500	0
5	@i_tit_productor	varchar	50	50	0
6	@i_tit_director	varchar	60	60	0
7	@i_tit_reparto	varchar	100	100	0
8	@i_cen_censura_id	char	1	1	0

PROCEDIMIENTO ALMACENADO					
NOMBRE: **sp_upd_Título**		BASE DE DATOS: **DB_VIDEO**			
DESCRIPCIÓN: Permite actualizar un registro en la tabla mvi_m_título.					
No.	**Parámetro**	**Tipo**	**Long.**	**Prc.**	**Scl.**
9	@i_gen_genero_1	char	3	3	0
10	@i_gen_genero_2	char	3	3	0
11	@i_gen_genero_3	char	3	3	0
12	@i_tit_nominacion_sn	char	1	1	0
13	@i_tit_nominaciones	varchar	100	100	0
14	@i_tit_premios_sn	char	1	1	0
15	@i_tit_premios	varchar	100	100	0
16	@i_tit_oscares_sn	char	1	1	0
17	@i_tit_duracion	char	5	5	0
18	@i_tit_produccion	int	4	10	0
19	@i_tit_logo_pelicula	varchar	50	50	0
COMPONENTE DE DATO QUE LE REFERENCIA: DB_TITULOC					

5.4.7. Procedimientos almacenados de las transacciones principales de la renta de videos.

PROCEDIMIENTO ALMACENADO					
NOMBRE: **sp_Trans_AlquilarItem**		BASE DE DATOS: **DB_VIDEO**			
DESCRIPCIÓN: Este procedimiento es el que nos permite registrar el alquiler de un ítem de película, se encarga de actualizar el estado del ítem a alquilado, rebajar el cupo del talonario del cliente y generar una transacción por el alquiler efectuado.					
No.	**Parámetro**	**Tipo**	**Long.**	**Prc.**	**Scl.**
1	@i_trn_transaccion_id	decimal	5	7	0
2	@i_cli_cliente_id	decimal	5	7	0
3	@i_tit_titulo_id	decimal	5	7	0
4	@i_med_medio_id	int	4	10	0
5	@i_trn_serv_entrega	char	1	1	0
6	@i_trn_serv_retiro	char	1	1	0
COMPONENTE DE DATO QUE LE REFERENCIA: DB_TRANSACCIONC					

PROCEDIMIENTO ALMACENADO	
NOMBRE: **sp_Trans_EntregarItem**	BASE DE DATOS: **DB_VIDEO**
DESCRIPCIÓN: Este procedimiento es el que nos permite a actualizar el estado del ítem alquilado en estado de película entregada. Aquí registramos si entregamos el ítem en el local ó en el domicilio del cliente.	

No.	Parámetro	Tipo	Long.	Prc.	Scl.
1	@i_trn_transaccion_id	decimal	5	7	0
2	@i_fec_entrega	datetime	8	0	0
3	@i_tipo_entrega	char	1	1	0

COMPONENTE DE DATO QUE LE REFERENCIA: DB_TRANSACCIONC

PROCEDIMIENTO ALMACENADO	
NOMBRE: **sp_Trans_RecibirItem**	BASE DE DATOS: **DB_VIDEO**
DESCRIPCIÓN: Este procedimiento es el que nos permite a actualizar el estado del ítem entregado en estado de película recibida. Aquí registramos la fecha de recepción. Esto nos ayuda para colocar el ítem de la película en estado de disponibilidad nuevamente.	

No.	Parámetro	Tipo	Long.	Prc.	Scl.
1	@i_trn_transaccion_id	decimal	5	7	0
2	@i_fec_recepcion	datetime	8	0	0

COMPONENTE DE DATO QUE LE REFERENCIA: DB_TRANSACCIONC

5.5. Proceso Servidor de Base de Datos.

El proceso servidor de Base de Datos fue implementado en el entorno SQL Server 7.0.

En él creamos la base de datos DB_VIDEO, que contiene todas las tablas, los índices y los procedimientos almacenados que nos ayudarán en la administración de la información del Sistema de Renta de Videos.

El servidor de base de datos atiende los requerimientos de los componentes de acceso a datos.

6

IMPLEMENTACIÓN DE LA CAPA DE NEGOCIOS

6. IMPLEMENTACIÓN DE LA CAPA DE NEGOCIOS.

6.1. Estructura de los componentes de acceso a datos.

Los componentes de acceso a datos siguen las especificaciones recomendadas por Microsoft para el desarrollo de componentes de este tipo.

Están implementados en bibliotecas DLL que generalmente están asociadas a una tabla de la base de datos. El nombre del componente empieza con el prefijo **db_** y finaliza con la letra **C** para indicar el hecho de que es un componente. Ejemplos:

db_ClienteC.dll	Para las operaciones sobre la tabla de Clientes.
db_ItemC.dll	Para las operaciones sobre la tabla de Items.

Todos los componentes son de tipo in-process, es decir, son bibliotecas dll que se ejecutan dentro de la aplicación COM+, y fueron desarrollados en Visual Basic.

El primer paso fue definir las interfaces IDL para cada uno de los componentes de acceso a datos. Estas siguieron la siguiente nomenclatura:

```
I + db + <tabla de la base de datos> + .idl
```

Por ejemplo:

IdbTitulo.idl Indica la interface de acceso a datos de la tabla Titulo

En las interfaces se definieron los métodos que iban a ser implementados en los componentes. Típicamente los métodos que existen en la mayoría de los objetos son:

Método	Objetivo
Add	Para agregar un registro
Update	Para actualizar un registro
Delete	Para eliminar un registro
GetById	Para retornar información de un registro
ListAll	Para retornar todos los registros

Las interfaces se compilaron usando el MIDL (Microsoft IDL compiler) y se generaron las "type libraries" apropiadas. Para el ejemplo anterior se generó un archivo llamado IdbTitulo.tlb.

En el desarrollo del componente se hizo referencia a las "type libraries" de acceso a datos que se iban a usar en el componente.

En los componentes se crearon clases con el mismo nombre que el de la tabla a la cual se iba a dar mantenimiento con el componente. Para el ejemplo que revisamos la clase se denominó Titulo.

Para la conexión a la base de datos usamos un archivo externo UDL (Universal Data Link) llamado `RentaVideo.udl`. En este archivo especificamos el proveedor OLEDB, en nuestro caso usamos el "OLEDB

Provider for Microsoft SQL Server". También indicamos el nombre del servidor SQL Server, el usuario y contraseña con el cual se accede a la conexión y el nombre de la base de datos. Este archivo debe residir en un subdirectorio UDL bajo el directorio SYSTEM32 del servidor COM+ de transacciones. A continuación mostramos el contenido del archivo `RentaVideo.udl`:

```
[oledb]
Provider=SQLOLEDB.1;Persist Security Info=False;User
ID=sa;Initial Catalog=DB_VIDEO;Data
Source=EB1;Application Name=RentaVideo
```

En el desarrollo del componente como tal, se implementó la type library de cada componente utilizando la sentencia `Implements` de Visual Basic. Por ejemplo para el componente `db_TituloC`:

```
Implements IdbTituloLib.IdbTitulo
```

La lógica de cada método siguió la siguiente estructura:

1. Se activó la detección de errores.

2. Se armó la sentencia SQL que se iba a ejecutar o se construyó la llamada al stored procedure con los parámetros necesarios.

3. Se conectó a la base de datos usando el archivo `RentaVideo.udl`.

4. Se ejecutó la instrucción SQL.

5. Se retornó el recordset, el status de éxito de la llamada o se generó la excepción COM para que la detecte el programa llamador.

6. Se cerró la conexión y se indicó al servidor transaccional COM+ que la transacción fue exitosa mediante la instrucción `GetObjectContext.SetComplete` o si la transacción fue fallida con el `GetObjectContext.SetAbort`.

6.2. Objetos distribuídos de acceso a datos.

A continuación detallamos los componentes de acceso a datos que se han creado y los objetos que ofrecen:

Componente	Clase / Objeto	Tabla relacionada
db_BancoC	Banco	mvi_m_banco
db_CategoriaWebC	CategoriaWeb	mvi_m_categoriaweb
db_CensuraC	Censura	mvi_m_censura
db_ClienteC	Cliente	mvi_m_cliente
db_FormaPagoC	FormaPago	mvi_m_forma_pago
db_GeneroC	Genero	mvi_m_genero
db_IdiomaC	Idioma	mvi_m_idioma
db_ItemC	Item	mvi_m_item
db_MedioC	Medio	mvi_m_medio
db_ParametroC	Parametro	mvi_m_parametro
db_PreRegistroC	PreRegistro	mvi_t_preregistro
db_PubliWebC	PubliWeb	mvi_t_publiweb

db_Sector	Sector	mvi_m_sector
db_SecuencialC	Secuencial	mvi_m_secuencial
db_TablasNumC	TablasNum	Accesa una tabla diferente de acuerdo a un parámetro
db_TalonarioC	Talonario	mvi_t_talonario
db_TalonTipoC	TalonTipo	mvi_m_talontipo
db_TarjetaC	Tarjeta	mvi_m_tarjeta
db_TituloC	Titulo	mvi_m_titulo
db_TransaccionC	Transaccion	mvi_t_transaccion

6.3. Comunicación con el proceso servidor de base de datos.

A continuación se listan los componentes de negocios y los componentes de datos que utilizan cada uno de ellos:

Componente de Negocios	Objetos de datos que utilizan
bus_BancoC	db_BancoC.Banco db_SecuencialC.Secuencial
bus_CategoriaWebC	db_CategoriaWebC.CategoriaWeb db_SecuencialC.Secuencial
bus_CensuraC	db_CensuraC.Censura
bus_ClienteC	db_ClienteC.Cliente db_SecuencialC.Secuencial
bus_FormaPagoC	db_FormaPagoC.FormaPago db_SecuencialC.Secuencial
bus_GeneroC	db_GeneroC.Genero

bus_IdiomaC	db_IdiomaC.Idioma db_SecuencialC.Secuencial
bus_ItemC	db_ItemC.Item
bus_MedioC	db_MedioC.Medio
bus_ParametroC	db_ParametroC.Parametro
bus_PreRegistroC	db_PreRegistroC.PreRegistro
bus_PubliWebC	db_PubliWebC.PubliWeb db_SecuencialC.Secuencial
bus_SectorC	db_SectorC.Sector db_SecuencialC.Secuencial
bus_SecuencialC	db_SecuencialC.Secuencial
bus_TablasNumC	db_TablasNumC.TablasNum db_SecuencialC.Secuencial
bus_TalonarioC	db_TalonarioC.Talonario db_SecuencialC.Secuencial
bus_TalonTipoC	db_TalonTipoC.TalonTipo db_SecuencialC.Secuencial
bus_TarjetaC	db_TarjetaC.Tarjeta db_SecuencialC.Secuencial
bus_TituloC	db_TituloC.Titulo db_SecuencialC.Secuencial
bus_TransaccionC	db_TransaccionC.Transaccion db_TalonarioC.Talonario db_SecuencialC.Secuencial

A continuación se detalla las llamadas que reciben cada uno de los componentes de acceso a datos:

Objeto de Datos	Usado en Componentes de Negocios
db_BancoC.Banco	bus_BancoC
db_CategoriaWebC.CategoriaWeb	bus_CategoriaWebC
db_CensuraC.Censura	bus_CensuraC
db_ClienteC.Cliente	bus_ClienteC
db_FormaPagoC.FormaPago	bus_FormaPagoC
db_GeneroC.Genero	bus_GeneroC
db_IdiomaC.Idioma	bus_IdiomaC
db_ItemC.Item	bus_ItemC
db_MedioC.Medio	bus_MedioC
db_ParametroC.Parametro	bus_ParametroC
db_PreRegistroC.PreRegistro	bus_PreRegistroC
db_PubliWebC.PubliWeb	bus_PubliWebC
db_SectorC.Sector	bus_SectorC
db_SecuencialC.Secuencial	bus_BancoC bus_CategoriaWebC bus_ClienteC bus_FormaPagoC bus_IdiomaC bus_PubliWebC

	bus_SectorC bus_SecuencialC bus_TablasNumC bus_TalonarioC bus_TalonTipoC bus_TarjetaC bus_TituloC bus_TransaccionC
db_TablasNumC.TablasNum	bus_TablasNumC
db_TalonarioC.Talonario	bus_TalonarioC bus_TransaccionC
db_TalonTipoC.TalonTipo	bus_TalonTipoC
db_TarjetaC.Tarjeta	bus_TarjetaC
db_TituloC.Titulo	bus_TituloC
db_TransaccionC.Transaccion	bus_TransaccionC

6.4. Estructura de los componentes de negocios.

Los componentes de negocios se implementaron como bibliotecas DLL y corresponden a los procesos del negocio. El nombre del componente empieza con el prefijo **bus_** y finaliza con la letra **C** para indicar el hecho de que es un componente. Ejemplos:

`bus_ClienteC.dll` Para las reglas de negocio sobre Clientes.

`bus_TransaccionC.dll` Para las reglas de negocios sobre Transacciones de Renta y Compra de Talonarios.

Todos los componentes son de tipo in-process, se ejecutan dentro de una aplicación COM+, y fueron desarrollados en Visual Basic.

En primer lugar se definieron las interfaces IDL para cada uno de los componentes de negocios. Estas siguieron la siguiente nomenclatura:

```
I + bus + <entidad> + .idl
```

Por ejemplo:

`IbusTransaccion.idl` Indica la interface de negocios de Transacciones

En las interfaces se definieron los métodos que iban a ser implementados en los componentes. Algunos de los métodos que existen en la mayoría de los componentes son:

Método	**Objetivo**
Agregar	Para agregar un registro
Actualizar	Para actualizar un registro
Eliminar	Para eliminar un registro
ObtenerRegistro	Para retornar información de un registro
ObtenerTodos	Para retornar todos los registros

Las interfaces se compilaron usando el MIDL (Microsoft IDL compiler) y se generaron las "type libraries" apropiadas. Para el ejemplo anterior se generó un archivo llamado `IbusTransaccion.tlb`.

En el desarrollo del componente se hizo referencia a las "type libraries" de negocios y de acceso a datos que se iban a usar en el componente.

En el componente se crearon clases con el mismo nombre que el del proceso asociado al componente. Para el ejemplo que revisamos la clase se denominó `Transaccion`.

En el desarrollo del componente como tal, se implementó la type library de cada componente utilizando la sentencia `Implements` de Visual Basic. Por ejemplo para el componente `bus_TransaccionC`:

```
Implements IbusTransaccionLib.IbusTransaccion
```

La lógica de cada método siguió la siguiente estructura:

1. Se activó la detección de errores.
2. Se instanciaron los objetos que se iban a usar en la lógica del método. Se usó la instrucción `CreateInstance`.
3. Se implementó la lógica propia de cada método.
4. Se indicó al servidor transaccional COM+ que la transacción fue exitosa mediante la instrucción `GetObjectContext.SetComplete` o si la transacción fue fallida con el `GetObjectContext.SetAbort`.

6.5. Objetos distribuídos de negocios.

6.5.1. Interfaces de objetos de acceso a datos.

Interface de db_BancoC.Banco

```
// Proyecto: Renta de Videos
// Typelib: IdbBanco.tlb
// Descripcion: Componente de Datos de Bancos

[
  uuid(28E3D7AA-275E-4666-B7D9-F931AD1C5822),
  version(1.0),
  helpstring("IdbBanco 1.0 Type Library")
]

library IdbBancoLib
{

    // TypeLib: OLE Automation
    importlib("stdole2.tlb");

    // TypeLib: Microsoft ActiveX Data Objects 2.5 Library
    importlib("..\msado15.dll");

    // Forward declare all types defined in this typelib


    interface IdbBanco ;
    [
      odl,
      uuid(B6694868-A909-432c-AC65-CA0E96F26974),
      version(1.0),
      helpstring("IdbBanco Interface"),
      nonextensible,
      oleautomation
    ]


    interface IdbBanco : IUnknown {
        [id(0), helpstring("method Add")]
        HRESULT Add(
            [in] short BcoId,
            [in] BSTR  BcoNombre
                    );

        [id(1), helpstring("method Delete")]
        HRESULT Delete(
            [in] short BcoId
                    );

        [id(2), helpstring("method Update")]
        HRESULT Update(
            [in] short BcoId,
            [in] BSTR  BcoNombre
                    );

        [id(3), helpstring("method GetById")]
        HRESULT GetById(
            [in] long BcoId,
            [out, retval] _Recordset**
                    );
```

```
        [id(4), helpstring("method ListAll")]
        HRESULT ListAll([out, retval] _Recordset**
                        );
    };
};
```

Esta interface define los métodos para las operaciones de Add (Agregar), Delete (Eliminar), Update (Actualizar), GetById (Para recuperar un registro de la tabla de bancos) y ListAll (Para obtener todos los registros de la tabla de bancos en forma de un recordset).

Interface de dbCategoriaWebC.CategoriaWeb

```
// Proyecto: Renta de Videos
// Typelib: IdbCategoriaWeb.tlb
// Descripcion: Componente de Datos de Categoria en Web
// Fecha: 2002/Jun/09

[
  uuid(41E437B0-EDAF-44dc-A39F-CC3EF221C601),
  version(1.0),
  helpstring("IdbCategoriaWeb 1.0 Type Library")
]

library IdbCategoriaWebLib
{
    // TypeLib: OLE Automation
    importlib("stdole2.tlb");

    // TypeLib: Microsoft ActiveX Data Objects 2.5 Library
    importlib("..\msado15.dll");

    // Forward declare all types defined in this typelib
    interface IdbCategoriaWeb ;

    [
      odl,
      uuid(54621D45-21B6-4b98-92A6-C39404FF4019),
      version(1.0),
      helpstring("IdbCategoriaWeb Interface"),
      nonextensible,
      oleautomation
    ]

    interface IdbCategoriaWeb : IUnknown {
        [id(0), helpstring("method Add")]
        HRESULT Add(
            [in] short CatId,
            [in] BSTR  CatDescripcion
                        );

        [id(1), helpstring("method Delete")]
        HRESULT Delete(
```

```
            [in] short CatId
                        );

        [id(2), helpstring("method Update")]
        HRESULT Update(
            [in] short CatId,
            [in] BSTR  CatDescripcion
                        );

        [id(3), helpstring("method GetById")]
        HRESULT GetById(
            [in] short CatId,
            [out, retval] _Recordset**
                        );

        [id(4), helpstring("method ListAll")]
        HRESULT ListAll(
             [out, retval] _Recordset**
                        );
    };
};
```

Esta interface define los métodos para las operaciones de Add (Agregar), Delete (Eliminar), Update (Actualizar), GetById (Para recuperar un registro de la tabla de categorías web) y ListAll (Para obtener todos los registros de la tabla de categorías web en un recordset).

Interface de db_CensuraC.Censura

```
// Proyecto: Renta de Videos
// Typelib: IdbCensura.tlb
// Descripcion: Componente de Datos de Censuras
// Fecha: 2002/Jun/09

[
  uuid(55F90FAB-3E28-4a65-A554-402B4F637B0B),
  version(1.0),
  helpstring("IdbCensura 1.0 Type Library")
]
library IdbCensuraLib
{

    // TypeLib: OLE Automation
    importlib("stdole2.tlb");

    // TypeLib: Microsoft ActiveX Data Objects 2.5 Library
    importlib("..\msado15.dll");

    // Forward declare all types defined in this typelib
    interface IdbCensura ;
    [
      odl,
      uuid(8255F447-F8C1-40f6-B262-1C35C53AB525),
      version(1.0),
```

```
      helpstring("IdbCensura Interface"),
      nonextensible,
      oleautomation
    ]

    interface IdbCensura : IUnknown {
        [id(0), helpstring("method Add")]
        HRESULT Add(
            [in] BSTR  CenId,
            [in] BSTR  CenDescripcion
                    );

        [id(1), helpstring("method Delete")]
        HRESULT Delete(
            [in] BSTR  CenId
                    );

        [id(2), helpstring("method Update")]
        HRESULT Update(
            [in] BSTR  CenId,
            [in] BSTR  CenDescripcion
                    );

        [id(3), helpstring("method GetById")]
        HRESULT GetById(
            [in] BSTR  CenId,
            [out, retval] _Recordset**
                    );

        [id(4), helpstring("method ListAll")]
        HRESULT ListAll(
             [out, retval] _Recordset**
                    );
    };
};
```

Esta interface define los métodos para las operaciones de Add (Agregar), Delete (Eliminar), Update (Actualizar), GetById (Para recuperar un registro de la tabla de Censuras) y ListAll (Para obtener todos los registros de la tabla de Censuras en un recordset).

Interface de db_ClienteC.Cliente

```
// Proyecto: Renta de Videos
// Typelib: IdbCliente.tlb
// Descripcion: Componente de Datos de Clientes
// Fecha: 2002/May/29

[
  uuid(DF093354-509D-45cb-BA9B-666617629AD8),
  version(1.0),
  helpstring("IdbCliente 1.0 Type Library")
]
```

```
library IdbClienteLib
{

    // TypeLib: OLE Automation

    importlib("stdole2.tlb");

    // TypeLib: Microsoft ActiveX Data Objects 2.5 Library

    importlib("..\msado15.dll");


    interface IdbCliente ;

    [
      odl,
      uuid(867816AB-0A4A-4d9c-9ADA-C81D81738721),
      version(1.0),
      helpstring("IdbCliente Interface"),
      nonextensible,
      oleautomation
    ]

    interface IdbCliente : IUnknown {
        [id(0), helpstring("method Add")]
        HRESULT Add(
                        [in] long  CliId,
                        [in] BSTR  CliNombre,
                        [in] BSTR  CliApellido,
                        [in] BSTR  CliTipoIdentificacion,
                        [in] BSTR  CliNumIdentificacion,
                        [in] BSTR  CliCiudad,
                        [in] BSTR  CliDireccion,
                        [in] short CliSectorId,
                        [in] BSTR  CliTelefono,
                        [in] BSTR  CliEmail,
                        [in] short CliFormaPagoId,
                        [in] short CliBcoId,
                        [in] BSTR  CliNumCtaBco,
                        [in] short CliTarjetaId,
                        [in] BSTR  CliNumTarjeta,
                        [in] DATE  CliTarjFechaVencim,
                        [in] BSTR  CliServicioEntrega,
                        [in] BSTR  CliServicioRetiro,
                        [in] BSTR  CliLogin,
                        [in] BSTR  CliClave,
                        [in] BSTR  CliEstadoReg
                        );

        [id(1), helpstring("method Update")]
        HRESULT Update(
                        [in] long  CliId,
                        [in] BSTR  CliNombre,
                        [in] BSTR  CliApellido,
                        [in] BSTR  CliTipoIdentificacion,
                        [in] BSTR  CliNumIdentificacion,
                        [in] BSTR  CliCiudad,
                        [in] BSTR  CliDireccion,
                        [in] short CliSectorId,
                        [in] BSTR  CliTelefono,
                        [in] BSTR  CliEmail,
                        [in] short CliFormaPagoId,
                        [in] short CliBcoId,
                        [in] BSTR  CliNumCtaBco,
                        [in] short CliTarjetaId,
                        [in] BSTR  CliNumTarjeta,
                        [in] DATE  CliTarjFechaVencim,
                        [in] BSTR  CliServicioEntrega,
                        [in] BSTR  CliServicioRetiro,
```

```
                    [in] BSTR  CliLogin,
                    [in] BSTR  CliClave,
                    [in] BSTR  CliEstadoReg
                    );

        [id(2), helpstring("method GetById")]
        HRESULT GetById(
            [in] long CliId,
            [out, retval] _Recordset**
                    );

        [id(3), helpstring("method GetByName")]
        HRESULT GetByName(
            [in] BSTR CliNombre,
            [out, retval] _Recordset**
                    );

        [id(4), helpstring("method ListAll")]
        HRESULT ListAll(
               [out, retval] _Recordset**
                    );

        [id(5), helpstring("method CheckLogin")]
        HRESULT CheckLogin(
            [in] BSTR CliLogin,
            [in] BSTR CliClave,
            [out, retval] long*
                    );

        [id(6), helpstring("method GetInfoCuenta")]
        HRESULT GetInfoCuenta(
            [in] long CliId,
            [out, retval] _Recordset**
                    );
    );
);
```

Esta interface define los métodos para las operaciones de Add (Agregar), Delete (Eliminar), Update (Actualizar), GetById (Para recuperar un registro de la tabla de clientes), ListAll (Para obtener todos los registros de la tabla de clientes en un recordset), GetByName (Para recuperar los clientes que coincidan con un nombre dado), CheckLogin (Para verificar la validez de una contraseña de usuario) y GetInfoCuenta (Que retorna información de la cuenta del cliente).

Interface de db_FormaPagoC.FormaPago

```
// Proyecto: Renta de Videos
// Typelib: IdbFormaPago.tlb
// Descripcion: Componente de Datos de Formas de Pago
// Fecha: 2002/Jun/08

[
  uuid(CC9B0807-B2E7-4b1b-BA21-30F58005A3FA),
  version(1.0),
  helpstring("IdbFormaPago 1.0 Type Library")
]

library IdbFormaPagoLib
{

    // TypeLib: OLE Automation

    importlib("stdole2.tlb");

    // TypeLib: Microsoft ActiveX Data Objects 2.5 Library

    importlib("..\msado15.dll");


    // Forward declare all types defined in this typelib

    interface IdbFormaPago ;

    [
      odl,
      uuid(53207E4C-1465-4754-9879-3F6988BE1620),
      version(1.0),
      helpstring("IdbFormaPago Interface"),
      nonextensible,
      oleautomation
    ]

    interface IdbFormaPago : IUnknown {
        [id(0), helpstring("method Add")]
        HRESULT Add(
                        [in] short   FpgId,
                        [in] BSTR    FpgDescripcion,
                        [in] BSTR    FpgValidezLocal,
                        [in] BSTR    FpgValidezWeb
                        );

        [id(1), helpstring("method Delete")]
        HRESULT Delete(
                        [in] short   FpgID
                        );

        [id(2), helpstring("method Update")]
        HRESULT Update(
                        [in] short    FpgId,
                        [in] BSTR     FpgDescripcion,
                        [in] BSTR     FpgValidezLocal,
                        [in] BSTR     FpgValidezWeb
                        );

        [id(3), helpstring("method GetById")]
        HRESULT GetById(
                        [in] short    FpgId,
                        [out, retval] _Recordset**
                        );
```

```
        [id(4), helpstring("method ListAll")]
        HRESULT ListAll(
                        [out, retval] _Recordset**
                        );

        [id(5), helpstring("method ListByLocation")]
        HRESULT ListByLocation(
                        [in] BSTR     FpgLocation,
                        [out, retval] _Recordset**
                        );
    };
};
```

Esta interface define los métodos para las operaciones de Add (Agregar), Delete (Eliminar), Update (Actualizar), GetById (Para recuperar un registro de la tabla de formas de pago), ListAll (Para obtener todos los registros de la tabla de formas de pago en un recordset) y ListByLocation (Para obtener todas las formas de pago que sean válidas en una ubicación dada).

Interface de db_GeneroC.Genero

```
// Proyecto: Renta de Videos
// Typelib: IdbGenero.tlb
// Descripcion: Componente de Datos de Generos de Videos
// Fecha: 2002/Jun/09

[
  uuid(918ABF2B-E748-46b0-89FE-3B33414FE094),
  version(1.0),
  helpstring("IdbGenero 1.0 Type Library")
]
library IdbGeneroLib
{

    // TypeLib: OLE Automation

    importlib("stdole2.tlb");

    // TypeLib: Microsoft ActiveX Data Objects 2.5 Library

    importlib("..\msado15.dll");


    // Forward declare all types defined in this typelib

    interface IdbGenero ;

    [
      odl,
```

```
      uuid(7CA4BCDB-C71B-4e04-8E35-E5C5999BCEC9),
      version(1.0),
      helpstring("IdbGenero Interface"),
      nonextensible,
      oleautomation
    ]

    interface IdbGenero : IUnknown {
        [id(0), helpstring("method Add")]
        HRESULT Add(
            [in] BSTR  GenId,
            [in] BSTR  GenDescripcion
                    );

        [id(1), helpstring("method Delete")]
        HRESULT Delete(
            [in] BSTR  GenId
                    );

        [id(2), helpstring("method Update")]
        HRESULT Update(
            [in] BSTR  GenId,
            [in] BSTR  GenDescripcion
                    );

        [id(3), helpstring("method GetById")]
        HRESULT GetById(
            [in] BSTR  GenId,
            [out, retval] _Recordset**
                    );

        [id(4), helpstring("method ListAll")]
        HRESULT ListAll(
            [out, retval] _Recordset**
                    );
    };
};
```

Esta interface define los métodos para las operaciones de Add (Agregar), Delete (Eliminar), Update (Actualizar), GetById (Para recuperar un registro de la tabla de géneros) y ListAll (Para obtener todos los registros de la tabla de géneros en un recordset).

Interface de db_IdiomaC.Idioma

```
// Proyecto: Renta de Videos
// Typelib: IdbIdioma.tlb
// Descripcion: Componente de Datos de Idiomas
// Fecha: 2002/Jun/10

[
  uuid(4E31FD12-AEAB-4c08-90A0-B0C03511E70B),
  version(1.0),
  helpstring("IdbIdioma 1.0 Type Library")
```

```
]
library IdbIdiomaLib
{

    // TypeLib: OLE Automation

    importlib("stdole2.tlb");

    // TypeLib: Microsoft ActiveX Data Objects 2.5 Library

    importlib("..\msado15.dll");


    // Forward declare all types defined in this typelib

    interface IdbIdioma ;

    [
      odl,
      uuid(207C30F2-ABEC-4828-ABCF-73BB05CF8A06),
      version(1.0),
      helpstring("IdbIdioma Interface"),
      nonextensible,
      oleautomation
    ]

    interface IdbIdioma : IUnknown {
        [id(0), helpstring("method Add")]
        HRESULT Add(
            [in] short IdiId,
            [in] BSTR  IdiDescripcion
                    );

        [id(1), helpstring("method Delete")]
        HRESULT Delete(
            [in] short IdiId
                    );

        [id(2), helpstring("method Update")]
        HRESULT Update(
            [in] short IdiId,
            [in] BSTR  IdiDescripcion
                    );

        [id(3), helpstring("method GetById")]
        HRESULT GetById(
            [in] short IdiId,
            [out, retval] _Recordset**
                    );

        [id(4), helpstring("method ListAll")]
        HRESULT ListAll(
             [out, retval] _Recordset**
                    );
    };
};
```

Esta interface define los métodos para las operaciones de Add (Agregar), Delete (Eliminar), Update (Actualizar), GetById (Para recuperar un

registro de la tabla de idiomas) y ListAll (Para obtener todos los registros de la tabla de idiomas en un recordset).

Interface de db_ItemC.Item

```
// Proyecto: Renta de Videos
// Typelib: IdbItem.tlb
// Descripcion: Componente de Datos de Items
// Fecha: 2002/Jun/01

[
  uuid(052995AB-616E-40b0-AA73-B977016179E9),
  version(1.0),
  helpstring("IdbItem 1.0 Type Library")
]

library IdbItemLib
{

    // TypeLib: OLE Automation

    importlib("stdole2.tlb");

    // TypeLib: Microsoft ActiveX Data Objects 2.5 Library

    importlib("..\msado15.dll");


    interface IdbItem ;

    [
      odl,
      uuid(0A87B9A5-9992-4211-BCCA-CDBE0C48D2BE),
      version(1.0),
      helpstring("IdbItem Interface"),
      nonextensible,
      oleautomation
    ]

    interface IdbItem : IUnknown {
        [id(0), helpstring("method Add")]
        HRESULT Add(
            [in] long  ItmId,
            [in] long  ItmTitId,
            [in] short ItmMedioId,
            [in] short ItmIdiomaId,
            [in] BSTR  ItmEstadoRegistro,
            [in] BSTR  ItmEstadoRenta,
            [in] BSTR  ItmMotivoIngreso
                      );

        [id(1), helpstring("method Update")]
        HRESULT Update(
            [in] long  ItmId,
            [in] long  ItmTitId,
            [in] short ItmMedioId,
            [in] short ItmIdiomaId,
            [in] BSTR  ItmEstadoRegistro,
            [in] BSTR  ItmEstadoRenta,
            [in] BSTR  ItmMotivoIngreso
                      );
```

```
        [id(2), helpstring("method GetById")]
        HRESULT GetById(
            [in] long ItmId,
            [out, retval] _Recordset**
                    );

        [id(3), helpstring("method ListAll")]
        HRESULT ListAll(
             [out, retval] _Recordset**
                    );

        [id(4), helpstring("method GetItemDisponible")]
        HRESULT GetItemDisponible(
            [in] short MedioId,
            [in] long  TituloId,
            [out, retval] long*
                    );
    };
};
```

Esta interface define los métodos para las operaciones de Add (Agregar), Delete (Eliminar), Update (Actualizar), GetById (Para recuperar un registro de la tabla de items), ListAll (Para obtener todos los registros de la tabla de items en un recordset) y GetItemDisponible (Para retornar un código de ítem que corresponda al medio y título dados).

Interface de db_MedioC.Medio

```
// Proyecto: Renta de Videos
// Typelib: IdbMedio.tlb
// Descripcion: Componente de Datos de Medios
// Fecha: 2002/May/09

[
  uuid(AC6D6F1C-C749-45ea-9BB9-8AB037E5C1C0),
  version(1.0),
  helpstring("IdbMedio 1.0 Type Library")
]
library IdbMedioLib
{
    // TypeLib: OLE Automation

    importlib("stdole2.tlb");

    // TypeLib: Microsoft ActiveX Data Objects 2.5 Library

    importlib("..\msado15.dll");


    // Forward declare all types defined in this typelib
```

```
    interface IdbMedio ;

    [
      odl,
      uuid(6770BC34-8BD1-4825-AC8E-A9FB066050B6),
      version(1.0),
      helpstring("IdbMedio Interface"),
      nonextensible,
      oleautomation
    ]

    interface IdbMedio : IUnknown {
        [id(0), helpstring("method Add")]
        HRESULT Add(
                        [in] long MedID,
                        [in] BSTR MedDesc,
                        [in] CURRENCY MedCostoRepos,
                        [in] BSTR MedLogo
                        );

        [id(1), helpstring("method Delete")]
        HRESULT Delete(
                        [in] long MedID
                        );

        [id(2), helpstring("method Update")]
        HRESULT Update(
                        [in] long MedID,
                        [in] BSTR MedDesc,
                        [in] CURRENCY MedCostoRepos,
                        [in] BSTR MedLogo
                        );

        [id(3), helpstring("method GetById")]
        HRESULT GetById(
                        [in] long MedID,
                        [out, retval] _Recordset**
                        );

        [id(4), helpstring("method ListAll")]
        HRESULT ListAll(
                        [out, retval] _Recordset**
                        );
    };
};
```

Esta interface define los métodos para las operaciones de Add (Agregar), Delete (Eliminar), Update (Actualizar), GetById (Para recuperar un registro de la tabla de medios) y ListAll (Para obtener todos los registros de la tabla de medios en un recordset).

Interface de db_ParametroC.Parametro

```
// Proyecto: Renta de Videos
// Typelib: IdbParametro.tlb
// Descripcion: Componente de Datos de Parametros
// Fecha: 2002/Jun/09

[
  uuid(6099C540-31E8-4fef-9425-519C2BB7F5AC),
  version(1.0),
  helpstring("IdbParametro 1.0 Type Library")
]
library IdbParametroLib
{

    // TypeLib: OLE Automation

    importlib("stdole2.tlb");

    // TypeLib: Microsoft ActiveX Data Objects 2.5 Library

    importlib("..\msadol5.dll");


    // Forward declare all types defined in this typelib

    interface IdbParametro ;

    [
      odl,
      uuid(184F5DB9-8D54-4915-8F97-87C0161DD49A),
      version(1.0),
      helpstring("IdbParametro Interface"),
      nonextensible,
      oleautomation
    ]

    interface IdbParametro : IUnknown {
        [id(0), helpstring("method Add")]
        HRESULT Add(
            [in] BSTR      ParId,
            [in] BSTR      ParDescripcion,
            [in] CURRENCY  ParValor,
            [in] BSTR      ParTexto
                      );

        [id(1), helpstring("method Delete")]
        HRESULT Delete(
            [in] BSTR      ParId
                      );

        [id(2), helpstring("method Update")]
        HRESULT Update(
            [in] BSTR      ParId,
            [in] BSTR      ParDescripcion,
            [in] CURRENCY  ParValor,
            [in] BSTR      ParTexto
                      );

        [id(3), helpstring("method GetById")]
        HRESULT GetById(
            [in] BSTR      ParId,
            [out, retval] _Recordset**
                      );

        [id(4), helpstring("method ListAll")]
```

```
        HRESULT ListAll(
            [out, retval] _Recordset**
                        );
    };
};
```

Esta interface define los métodos para las operaciones de Add (Agregar), Delete (Eliminar), Update (Actualizar), GetById (Para recuperar un registro de la tabla de parámetros) y ListAll (Para obtener todos los registros de la tabla de parámetros en un recordset).

Interface de db_PreRegistroC.PreRegistro

```
// Proyecto: Renta de Videos
// Typelib: IdbPreRegistro.tlb
// Descripcion: Componente de Datos de PreRegistros
// Fecha: 2002/May/29

[
  uuid(80800832-4246-4d06-9B39-0AF77CCB9EBD),
  version(1.0),
  helpstring("IdbPreRegistro 1.0 Type Library")
]

library IdbPreRegistroLib
{

    // TypeLib: OLE Automation

    importlib("stdole2.tlb");

    // TypeLib: Microsoft ActiveX Data Objects 2.5 Library

    importlib("..\msadol5.dll");


    interface IdbPreRegistro ;

    [
      odl,
      uuid(E324835B-C72E-40fa-A5AC-0D53537D7E3D),
      version(1.0),
      helpstring("IdbPreRegistro Interface"),
      nonextensible,
      oleautomation
    ]

    interface IdbPreRegistro : IUnknown {
        [id(0), helpstring("method Add")]
        HRESULT Add(
                        [in] BSTR  PrgTipoIdentif,
                        [in] BSTR  PrgNumIdentif,
                        [in] BSTR  PrgLogin,
                        [in] BSTR  PrgClave,
```

```
                [in] BSTR  PrgNombre,
                [in] BSTR  PrgApellido,
                [in] short PrgSectorId,
                [in] BSTR  PrgDireccion,
                [in] BSTR  PrgTelefono,
                [in] BSTR  PrgEmail,
                [in] short PrgFormaPagoId,
                [in] short PrgTarjetaId,
                [in] BSTR  PrgTarjetaNum,
                [in] BSTR  PrgTarjetaVenc,
                [in] BSTR  PrgTarjetaTitular,
                [in] BSTR  PrgServEntrega,
                [in] BSTR  PrgServRetiro,
                [in] BSTR  PrgEstadoReg
                );

        [id(1), helpstring("method Update")]
        HRESULT Update(
                [in] BSTR  PrgTipoIdentif,
                [in] BSTR  PrgNumIdentif,
                [in] BSTR  PrgLogin,
                [in] BSTR  PrgClave,
                [in] BSTR  PrgNombre,
                [in] BSTR  PrgApellido,
                [in] short PrgSectorId,
                [in] BSTR  PrgDireccion,
                [in] BSTR  PrgTelefono,
                [in] BSTR  PrgEmail,
                [in] short PrgFormaPagoId,
                [in] short PrgTarjetaId,
                [in] BSTR  PrgTarjetaNum,
                [in] BSTR  PrgTarjetaVenc,
                [in] BSTR  PrgTarjetaTitular,
                [in] BSTR  PrgServEntrega,
                [in] BSTR  PrgServRetiro,
                [in] BSTR  PrgEstadoReg
                );

        [id(2), helpstring("method GetById")]
        HRESULT GetById(
                [in] BSTR  PrgTipoIdentif,
                [in] BSTR  PrgNumIdentif,
                [out, retval] _Recordset**
                );

        [id(3), helpstring("method GetByName")]
        HRESULT GetByName(
                [in] BSTR  PrgApellido,
                [out, retval] _Recordset**
                );

        [id(4), helpstring("method GetByCliId")]
        HRESULT GetByCliId(
                [in] long  PrgCliId,
                [out, retval] _Recordset**
                );

        [id(5), helpstring("method ListByEstado")]
        HRESULT ListByEstado(
                [in] BSTR  PrgEstadoReg,
                [out, retval] _Recordset**
                );

        [id(6), helpstring("method UpdEvaluacion")]
        HRESULT UpdEvaluacion(
                [in] BSTR  PrgTipoIdentif,
                [in] BSTR  PrgNumIdentif,
                [in] DATE  PrgEvalFecha,
                [in] BSTR  PrgEvalCalif,
```

```
                    [in] BSTR  PrgEvalTexto,
                    [in] BSTR  PrgEvalUsuId,
                    [in] long  PrgEvalCliId,
                    [in] BSTR  PrgEstadoReg
                    );
    };
};
```

Esta interface define los métodos para las operaciones de Add (Agregar), Delete (Eliminar), Update (Actualizar), GetById (Para recuperar un registro de la tabla de solicitudes de membresía), GetByName (Para obtener en un recordset las solicitudes que correspondan a un nombre de persona), GetByCliId (Para obtener en un recordset los datos que correspondan a un código de cliente), ListByEstado (Para retornar un recordset de las solicitudes que sean de un status dado) y UpdEvaluacion (Para actualizar los datos de una solicitud en el proceso de evaluación de las mismas).

Interface de db_PubliWebC.PubliWeb

```
// Proyecto: Renta de Videos
// Typelib: IdbPubliWeb.tlb
// Descripcion: Componente de Datos de Publicación en Web
// Fecha: 2002/Jun/09

[
  uuid(9D3E25E8-723B-4b0d-A8C3-EF6F135FFF12),
  version(1.0),
  helpstring("IdbPubliWeb 1.0 Type Library")
]
library IdbPubliWebLib
{
    // TypeLib: OLE Automation

    importlib("stdole2.tlb");

    // TypeLib: Microsoft ActiveX Data Objects 2.5 Library

    importlib("..\msadol5.dll");


    // Forward declare all types defined in this typelib
```

```
    interface IdbPubliWeb ;

    [
      odl,
      uuid(11A04860-45A0-4551-AB6B-17449AFA12D3),
      version(1.0),
      helpstring("IdbPubliWeb Interface"),
      nonextensible,
      oleautomation
    ]


    interface IdbPubliWeb : IUnknown {
        [id(0), helpstring("method Add")]
        HRESULT Add(
            [in] short PubCodigo,
            [in] short PubCatId,
            [in] BSTR  PubTitulo,
            [in] BSTR  PubDescripcion,
            [in] BSTR  PubRutaFoto,
            [in] BSTR  PubEstadoReg,
            [in] long  PubTitId
                    );

        [id(1), helpstring("method Delete")]
        HRESULT Delete(
            [in] short PubCodigo
                    );

        [id(2), helpstring("method Update")]
        HRESULT Update(
            [in] short PubCodigo,
            [in] short PubCatId,
            [in] BSTR  PubTitulo,
            [in] BSTR  PubDescripcion,
            [in] BSTR  PubRutaFoto,
            [in] BSTR  PubEstadoReg,
            [in] long  PubTitId
                    );

        [id(3), helpstring("method GetById")]
        HRESULT GetById(
            [in] short PubCodigo,
            [out, retval] _Recordset**
                    );

        [id(4), helpstring("method ListAll")]
        HRESULT ListAll(
           [out, retval] _Recordset**
                    );

        [id(5), helpstring("method ListByCat")]
        HRESULT ListByCat(
           [in] short PubCatId,
           [out, retval] _Recordset**
                    );
    };
};
```

Esta interface define los métodos para las operaciones de Add (Agregar), Delete (Eliminar), Update (Actualizar), GetById (Para recuperar un

registro de la tabla de publicaciones), ListAll (Para obtener todos los registros de la tabla de publicaciones en un recordset) y ListByCat (Para obtener las publicaciones que correspondan a un código de categoría dado).

Interface de db_SectorC.Sector

```
// Proyecto: Renta de Videos
// Typelib: IdbSector.tlb
// Descripcion: Componente de Datos de Sectores
// Fecha: 2002/Jun/09


[
  uuid(A6712A37-2845-41ed-91D7-EEB124D816F2),
  version(1.0),
  helpstring("IdbSector 1.0 Type Library")
]
library IdbSectorLib
{

    // TypeLib: OLE Automation

    importlib("stdole2.tlb");

    // TypeLib: Microsoft ActiveX Data Objects 2.5 Library

    importlib("..\msadol5.dll");


    // Forward declare all types defined in this typelib

    interface IdbSector ;

    [
      odl,
      uuid(F15EEE89-309C-4b8c-9D00-FED1E4F4FD50),
      version(1.0),
      helpstring("IdbSector Interface"),
      nonextensible,
      oleautomation
    ]

    interface IdbSector : IUnknown {
        [id(0), helpstring("method Add")]
        HRESULT Add(
            [in] short SctId,
            [in] BSTR  SctDescripcion
                    );

        [id(1), helpstring("method Delete")]
        HRESULT Delete(
            [in] short SctId
                    );

        [id(2), helpstring("method Update")]
        HRESULT Update(
            [in] short SctId,
```

```
            [in] BSTR  SctDescripcion
                        );

        [id(3), helpstring("method GetById")]
        HRESULT GetById(
            [in] long SctId,
            [out, retval] _Recordset**
                        );

        [id(4), helpstring("method ListAll")]
        HRESULT ListAll(
             [out, retval] _Recordset**
                        );
    };
};
```

Esta interface define los métodos para las operaciones de Add (Agregar), Delete (Eliminar), Update (Actualizar), GetById (Para recuperar un registro de la tabla de sectores) y ListAll (Para obtener todos los registros de la tabla de sectores en formato de recordset).

Interface de db_SecuencialC.Secuencial

```
// Proyecto: Renta de Videos
// Typelib: IdbSecuencial.tlb
// Descripcion: Componente de Numeros de Secuencia
// Fecha: 2002/May/20

[
  uuid(1484F5D0-401D-40d0-AA1D-A4E541A346E3),
  version(1.0),
  helpstring("IdbSecuencial 1.0 Type Library")
]
library IdbSecuencialLib
{

    // TypeLib: OLE Automation

    importlib("stdole2.tlb");

    // TypeLib: Microsoft ActiveX Data Objects 2.5 Library

    importlib("..\msado15.dll");


    // Forward declare all types defined in this typelib

    interface IdbSecuencial ;

    [
      odl,
      uuid(6E1F5313-4DAD-4be3-9E1E-736D5FC7FEBC),
      version(1.0),
      helpstring("IdbSecuencial Interface"),
```

```
      nonextensible,
      oleautomation
    ]

    interface IdbSecuencial : IUnknown {
        [id(0), helpstring("method Add")]
        HRESULT Add(
                        [in] long SecId,
                        [in] BSTR SecDesc,
                        [in] long SecValor);

        [id(1), helpstring("method Delete")]
        HRESULT Delete(
                        [in] long SecId);

        [id(2), helpstring("method Update")]
        HRESULT Update(
                        [in] long SecId,
                        [in] BSTR SecDesc,
                        [in] long SecValor);

        [id(3), helpstring("method GetById")]
        HRESULT GetById(
                        [in] long SecId,
                        [out, retval] _Recordset** );

        [id(4), helpstring("method GetNextNum")]
        HRESULT GetNextNum(
                        [in] long SecId,
                        [out, retval] long* );

        [id(5), helpstring("method ListAll")]
        HRESULT ListAll([out, retval] _Recordset** );
    };
};
```

Esta interface define los métodos para las operaciones de Add (Agregar), Delete (Eliminar), Update (Actualizar), GetById (Para recuperar un registro de la tabla de secuencias), GetNextNum (Para retornar el siguiente número de secuencia de una entidad dada) y ListAll (Para obtener todos los registros de la tabla de secuencias en un recordset).

Interface de db_TablasNumC.TablasNum

```
// Proyecto: Renta de Videos
// Typelib: IdbTablasNum.tlb
// Descripcion: Componente de Datos de Tablas Maestras (clave numérica)
// Fecha: 2002/Jun/08

[
  uuid(67502985-12C9-4680-B8C2-2E06B70E89C0),
  version(1.0),
```

```
  helpstring("IdbTablasNum 1.0 Type Library")
]
library IdbTablasNumLib
{

    // TypeLib: OLE Automation

    importlib("stdole2.tlb");

    // TypeLib: Microsoft ActiveX Data Objects 2.5 Library

    importlib("..\msadol5.dll");


    // Forward declare all types defined in this typelib

    interface IdbTablasNum ;

    [
      odl,
      uuid(83C35303-1DEE-4467-8625-E5965BF42615),
      version(1.0),
      helpstring("IdbTablasNum Interface"),
      nonextensible,
      oleautomation
    ]

    interface IdbTablasNum : IUnknown {
        [id(0), helpstring("method Add")]
        HRESULT Add(
             [in] long     TabNum,
             [in] long     TabId,
             [in] BSTR     TabDescripcion
                       );

        [id(1), helpstring("method Delete")]
        HRESULT Delete(
             [in] long     TabNum,
             [in] long     TabId
                       );

        [id(2), helpstring("method Update")]
        HRESULT Update(
             [in] long     TabNum,
             [in] long     TabId,
             [in] BSTR     TabDescripcion
                       );

        [id(3), helpstring("method GetById")]
        HRESULT GetById(
             [in] long      TabNum,
             [in] long      TabId,
             [out, retval]  _Recordset**
                       );

        [id(4), helpstring("method ListAll")]
        HRESULT ListAll(
             [in] long      TabNum,
             [out, retval]  _Recordset**
                       );
    };
};
```

Esta interface define los métodos para las operaciones de Add (Agregar), Delete (Eliminar), Update (Actualizar), GetById (Para recuperar un registro de la tabla genérica) y ListAll (Para obtener todos los registros de la tabla genérica en formato de recordset).

Interface de db_TalonarioC.Talonario

```
// Proyecto: Renta de Videos
// Typelib: IdbTalonario.tlb
// Descripcion: Componente de Datos de Talonarios
// Fecha: 2002/Jun/22

[
  uuid(67AE0F19-0FE8-48ee-94B7-BB3CCFD9A32D),
  version(1.0),
  helpstring("IdbTalonario 1.0 Type Library")
]
library IdbTalonarioLib
{

    // TypeLib: OLE Automation

    importlib("stdole2.tlb");

    // TypeLib: Microsoft ActiveX Data Objects 2.5 Library

    importlib("..\msado15.dll");


    // Forward declare all types defined in this typelib

    interface IdbTalonario ;

    [
      odl,
      uuid(A8BA6C03-EBC3-469e-8D2F-F6C1489B6062),
      version(1.0),
      helpstring("IdbTalonario Interface"),
      nonextensible,
      oleautomation
    ]

    interface IdbTalonario : IUnknown {
        [id(0), helpstring("method Add")]
        HRESULT Add(
              [in] long     TalId,
              [in] long     TalCliId,
              [in] short    TalTtiId,
               [in] short    TalNumTickets,
               [in] CURRENCY TalValor,
               [in] short    TalFormaPagoId,
               [in] short    TalTarjId,
               [in] BSTR     TalNumTarjeta,
               [in] BSTR     TalTarjFecVencim,
               [in] BSTR     TalTarjTitular
               );
```

```
        [id(1), helpstring("method Delete")]
        HRESULT Delete(
            [in] long     TalId
               );


        [id(2), helpstring("method Update")]
        HRESULT Update(
               [in] long     TalId,
               [in] long     TalCliId,
               [in] short    TalTtiId,
               [in] short    TalNumTickets,
               [in] CURRENCY TalValor,
               [in] short    TalFormaPagoId,
               [in] short    TalTarjId,
               [in] BSTR     TalNumTarjeta,
               [in] BSTR     TalTarjFecVencim,
               [in] BSTR     TalTarjTitular
               );


        [id(3), helpstring("method GetById")]
        HRESULT GetById(
            [in] long   TalId,
            [out, retval] _Recordset**
               );


        [id(4), helpstring("method ListAll")]
        HRESULT ListAll(
               [out, retval] _Recordset**
               );
    };
};
```

Esta interface define los métodos para las operaciones de Add (Agregar), Delete (Eliminar), Update (Actualizar), GetById (Para recuperar un registro de la tabla de talonarios) y ListAll (Para obtener todos los registros de la tabla de talonarios en formato de recordset).

Interface de db_TalonTipoC.TalonTipo

```
// Proyecto: Renta de Videos
// Typelib: IdbTalonTipo.tlb
// Descripcion: Componente de Datos de Tipos de Talonarios
// Fecha: 2002/Jun/22

[
  uuid(68EEE822-E7D7-442f-9744-C5FD6274F837),
  version(1.0),
  helpstring("IdbTalonTipo 1.0 Type Library")
]
library IdbTalonTipoLib
{
```

```
    // TypeLib: OLE Automation

    importlib("stdole2.tlb");

    // TypeLib: Microsoft ActiveX Data Objects 2.5 Library

    importlib("..\msadol5.dll");


    // Forward declare all types defined in this typelib

    interface IdbTalonTipo ;

    [
      odl,
      uuid(71F4D461-47A5-4b1b-B5F2-6AB5A8E9ECE4),
      version(1.0),
      helpstring("IdbTalonTipo Interface"),
      nonextensible,
      oleautomation
    ]

    interface IdbTalonTipo : IUnknown {
        [id(0), helpstring("method Add")]
        HRESULT Add(
            [in] short      TtiId,
            [in] BSTR       TtiDescripcion,
            [in] short      TtiNumTickets,
            [in] CURRENCY   TtiValor
                      );

        [id(1), helpstring("method Delete")]
        HRESULT Delete(
            [in] short      TtiId
                      );

        [id(2), helpstring("method Update")]
        HRESULT Update(
            [in] short      TtiId,
            [in] BSTR       TtiDescripcion,
            [in] short      TtiNumTickets,
            [in] CURRENCY   TtiValor
                      );

        [id(3), helpstring("method GetById")]
        HRESULT GetById(
            [in] short TtiId,
            [out, retval] _Recordset**
                      );

        [id(4), helpstring("method ListAll")]
        HRESULT ListAll(
            [out, retval] _Recordset**
                      );
    };
};
```

Esta interface define los métodos para las operaciones de Add (Agregar), Delete (Eliminar), Update (Actualizar), GetById (Para recuperar un

registro de la tabla de tipos de talonarios) y ListAll (Para obtener todos los registros de la tabla de tipos de talonarios en formato de recordset).

Interface de db_TarjetaC.Tarjeta

```
// Proyecto: Renta de Videos
// Typelib: IdbTarjeta.tlb
// Descripcion: Componente de Datos de Tarjetas de Credito
// Fecha: 2002/Jun/08

[
  uuid(ABFE3104-530F-4c4c-BCF3-409F512FE398),
  version(1.0),
  helpstring("IdbTarjeta 1.0 Type Library")
]
library IdbTarjetaLib
{

    // TypeLib: OLE Automation

    importlib("stdole2.tlb");

    // TypeLib: Microsoft ActiveX Data Objects 2.5 Library

    importlib("..\msado15.dll");


    // Forward declare all types defined in this typelib

    interface IdbTarjeta ;

    [
      odl,
      uuid(37137654-D65B-4976-B38F-3F463A04CF68),
      version(1.0),
      helpstring("IdbTarjeta Interface"),
      nonextensible,
      oleautomation
    ]

    interface IdbTarjeta : IUnknown {
        [id(0), helpstring("method Add")]
        HRESULT Add(
            [in] short TrjId,
            [in] BSTR  TrjDescripcion
                    );

        [id(1), helpstring("method Delete")]
        HRESULT Delete(
            [in] short TrjId
                    );

        [id(2), helpstring("method Update")]
        HRESULT Update(
            [in] short TrjId,
            [in] BSTR  TrjDescripcion
                    );

        [id(3), helpstring("method GetById")]
        HRESULT GetById(
            [in] long TrjId,
            [out, retval] _Recordset**
```

```
                    );

        [id(4), helpstring("method ListAll")]
        HRESULT ListAll(
            [out, retval] _Recordset**
                    );
    };
};
```

Esta interface define los métodos para las operaciones de Add (Agregar), Delete (Eliminar), Update (Actualizar), GetById (Para recuperar un registro de la tabla de tarjetas) y ListAll (Para obtener todos los registros de la tabla de tarjetas en formato de recordset).

Interface de db_TituloC.Titulo

```
// Proyecto: Renta de Videos
// Typelib: IdbTitulo.tlb
// Descripcion: Componente de Datos de Titulos
// Fecha: 2002/Jun/01

[
  uuid(47F70CD4-BF88-4da7-9FF6-C43AE043E9D7),
  version(1.0),
  helpstring("IdbTitulo 1.0 Type Library")
]

library IdbTituloLib
{

    // TypeLib: OLE Automation

    importlib("stdole2.tlb");

    // TypeLib: Microsoft ActiveX Data Objects 2.5 Library

    importlib("..\msado15.dll");


    interface IdbTitulo ;

    [
      odl,
      uuid(B4261E46-B905-4161-9693-43AE17BB3190),
      version(1.0),
      helpstring("IdbTitulo Interface"),
      nonextensible,
      oleautomation
    ]
```

```
    interface IdbTitulo : IUnknown {
        [id(0), helpstring("method Add")]
        HRESULT Add(
            [in] long TitId,
            [in] BSTR TitNombre,
            [in] BSTR TitNombreOrig,
            [in] BSTR TitSinopsis,
            [in] BSTR TitProductor,
            [in] BSTR TitDirector,
            [in] BSTR TitReparto,
            [in] BSTR TitCensuraId,
            [in] BSTR TitGenero1Id,
            [in] BSTR TitGenero2Id,
            [in] BSTR TitGenero3Id,
            [in] BSTR TitNominado,
            [in] BSTR TitNominaciones,
            [in] BSTR TitPremiado,
            [in] BSTR TitPremios,
            [in] BSTR TitOscarGanado,
            [in] BSTR TitDuracion,
            [in] long TitProduccion,
            [in] BSTR TitLogo
                    );

        [id(1), helpstring("method Update")]
        HRESULT Update(
            [in] long TitId,
            [in] BSTR TitNombre,
            [in] BSTR TitNombreOrig,
            [in] BSTR TitSinopsis,
            [in] BSTR TitProductor,
            [in] BSTR TitDirector,
            [in] BSTR TitReparto,
            [in] BSTR TitCensuraId,
            [in] BSTR TitGenero1Id,
            [in] BSTR TitGenero2Id,
            [in] BSTR TitGenero3Id,
            [in] BSTR TitNominado,
            [in] BSTR TitNominaciones,
            [in] BSTR TitPremiado,
            [in] BSTR TitPremios,
            [in] BSTR TitOscarGanado,
            [in] BSTR TitDuracion,
            [in] long TitProduccion,
            [in] BSTR TitLogo
                    );

        [id(2), helpstring("method GetById")]
        HRESULT GetById(
            [in] long TitId,
            [out, retval] _Recordset**
                    );

        [id(3), helpstring("method ListAll")]
        HRESULT ListAll(
             [out, retval] _Recordset**
                    );

        [id(4), helpstring("method GetByParam")]
        HRESULT GetByParam(
            [in] long MedioId,
            [in] long OpcBusqueda,
            [in] BSTR ArgBusqueda,
            [out, retval] _Recordset**
                    );
    };
};
```

Esta interface define los métodos para las operaciones de Add (Agregar), Delete (Eliminar), Update (Actualizar), GetById (Para recuperar un registro de la tabla de títulos), ListAll (Para obtener todos los registros de la tabla de títulos en formato de recordset) y GetByParam (Para obtener todos los títulos que cumplan con el criterio suministrado).

Interface de db_TransaccionC.Transaccion

```
// Proyecto: Renta de Videos
// Typelib: IdbTransaccion.tlb
// Descripcion: Componente de Datos de Transacciones
// Fecha: 2002/Jun/22

[
  uuid(FE5BE017-4433-4ee1-AFF3-33B86D7C2029),
  version(1.0),
  helpstring("IdbTransaccion 1.0 Type Library")
]
library IdbTransaccionLib
{

    // TypeLib: OLE Automation

    importlib("stdole2.tlb");

    // TypeLib: Microsoft ActiveX Data Objects 2.5 Library

    importlib("..\msado15.dll");


    // Forward declare all types defined in this typelib

    interface IdbTransaccion ;

    [
      odl,
      uuid(51EADE5D-BB69-489e-AF14-C51EADDE65DA),
      version(1.0),
      helpstring("IdbTransaccion Interface"),
      nonextensible,
      oleautomation
    ]

    interface IdbTransaccion : IUnknown {
        [id(0), helpstring("method GetById")]
        HRESULT GetById(
                        [in] long      TrnId,
                        [out, retval]  _Recordset**
                        );

        [id(1), helpstring("method ListByEstado")]
        HRESULT ListByEstado(
                        [in] BSTR      Estado,
                        [in] VARIANT   CliId,
                        [out, retval]  _Recordset**
                        );
```

```
        [id(2), helpstring("method EntregarItem")]
        HRESULT EntregarItem(
            [in] long               TrnId,
                        [in] DATE       FecEntrega,
                        [in] BSTR       TipoEntrega
                        );

        [id(3), helpstring("method RecibirItem")]
        HRESULT RecibirItem(
                        [in] long       TrnId,
                        [in] DATE       FecRecepcion
                        );

        [id(4), helpstring("method AlquilarItem")]
        HRESULT AlquilarItem(
                        [in, out]       long*   TrnId,
                        [in]            long    CliId,
                        [in]            long    TitId,
                        [in]            short   MedId,
                        [in]            BSTR     ServEntrega,
                        [in]            BSTR     ServRetiro
                        );
    };
};
```

Esta interface define los métodos para las operaciones de GetById (para obtener los datos de una transacción), ListByEstado (para obtener todas las transacciones de un cliente que tengan un estado determinado), EntregarItem (para actualizar una transacción en el proceso de entrega), RecibirItem (para actualizar una transacción en el proceso de recepción de items) y AlquilarItem (para actualizar las entidades en el proceso de alquiler de video).

6.5.2. Interfaces de Objetos de negocios.

Interface de bus_BancoC.Banco

```
// Proyecto: Renta de Videos
// Typelib: IbusBanco.tlb
// Descripcion: Componente de Negocios de Bancos
// Fecha: 2002/Jun/08

[
  uuid(0B50FB9C-0499-44c0-B818-4004EBC3982A),
  version(1.0),
  helpstring("IbusBanco 1.0 Type Library")
```

```
]

library IbusBancoLib
{

    // TypeLib: OLE Automation

    importlib("stdole2.tlb");

    // TypeLib: Microsoft ActiveX Data Objects 2.5 Library

    importlib("..\msado15.dll");


    interface IbusBanco ;

    [
      odl,
      uuid(33EBFCC8-C440-4cd8-8DD1-5A9AF3799291),
      version(1.0),
      helpstring("IbusBanco Interface"),
      dual,
      nonextensible,
      oleautomation
    ]

    interface IbusBanco : IDispatch {
        [id(0), helpstring("metodo Agregar")]
        HRESULT Agregar(
                        [in, out] short*  BcoId,
                        [in] BSTR  BcoNombre
                        );

        [id(1), helpstring("method Actualizar")]
        HRESULT Actualizar(
                        [in] short BcoId,
                        [in] BSTR  BcoNombre
                        );

        [id(2), helpstring("method Eliminar")]
        HRESULT Eliminar(
                        [in] short  BcoId
                        );

        [id(3), helpstring("method ObtenerRegistro")]
        HRESULT ObtenerRegistro(
            [in] short BcoId,
                        [out, retval] _Recordset**
                        );

        [id(4), helpstring("method ObtenerTodos")]
        HRESULT ObtenerTodos(
                        [out, retval] _Recordset**
                        );
    };
};
```

Esta interface define los métodos para los procesos de Agregar, Actualizar y Eliminar (bancos), ObtenerRegistro (para recuperar la

información de un banco) y ObtenerTodos (para obtener todos los registros de bancos).

Interface de bus_CategoriaWebC.CategoriaWeb

```
// Proyecto: Renta de Videos
// Typelib: IbusCategoriaWeb.tlb
// Descripcion: Componente de Negocios de Categoria en Web
// Fecha: 2002/Jun/09

[
  uuid(2E7D73E4-D1A8-4e77-89AE-645C78C9C53E),
  version(1.0),
  helpstring("IbusCategoriaWeb 1.0 Type Library")
]
library IbusCategoriaWebLib
{

    // TypeLib: OLE Automation

    importlib("stdole2.tlb");

    // TypeLib: Microsoft ActiveX Data Objects 2.5 Library

    importlib("..\msado15.dll");


    // Forward declare all types defined in this typelib

    interface IbusCategoriaWeb ;

    [
      odl,
      uuid(E1DE3405-7D15-4107-AB3E-274BCFA5DDD2),
      version(1.0),
      helpstring("IbusCategoriaWeb Interface"),
      dual,
      nonextensible,
      oleautomation
    ]

    interface IbusCategoriaWeb : IDispatch {
        [id(0), helpstring("method Agregar")]
        HRESULT Agregar(
            [in, out] short* CatId,
            [in] BSTR  CatDescripcion
                      );

        [id(1), helpstring("method Eliminar")]
        HRESULT Eliminar(
            [in] short CatId
                      );

        [id(2), helpstring("method Actualizar")]
        HRESULT Actualizar(
            [in] short CatId,
            [in] BSTR  CatDescripcion
                      );

        [id(3), helpstring("method ObtenerRegistro")]
        HRESULT ObtenerRegistro(
            [in] short CatId,
```

```
            [out, retval] _Recordset**
                        );

        [id(4), helpstring("method ObtenerTodos")]
        HRESULT ObtenerTodos(
                [out, retval] _Recordset**
                        );
    };
};
```

Esta interface define los métodos para los procesos de Agregar, Actualizar y Eliminar (categorías en web), ObtenerRegistro (para recuperar la información de una categoría de web) y ObtenerTodos (para obtener todos los registros de categorías).

Interface de bus_CensuraC.Censura

```
// Proyecto: Renta de Videos
// Typelib: IbusCensura.tlb
// Descripcion: Componente de Negocios de Censuras
// Fecha: 2002/Jun/10

[
  uuid(20940170-ED9E-4781-A08A-33E4FDC17BE2),
  version(1.0),
  helpstring("IbusCensura 1.0 Type Library")
]
library IbusCensuraLib
{

    // TypeLib: OLE Automation

    importlib("stdole2.tlb");

    // TypeLib: Microsoft ActiveX Data Objects 2.5 Library

    importlib("..\msado15.dll");


    // Forward declare all types defined in this typelib

    interface IbusCensura ;

    [
      odl,
      uuid(63F79E31-EBE6-4a96-B605-AC50761F9EC9),
      version(1.0),
      helpstring("IbusCensura Interface"),
      dual,
      nonextensible,
```

```
      oleautomation
    ]

    interface IbusCensura : IDispatch {

        [id(0), helpstring("method Agregar")]
        HRESULT Agregar(
            [in] BSTR  CenId,
            [in] BSTR  CenDescripcion
                      );


        [id(1), helpstring("method Eliminar")]
        HRESULT Eliminar(
            [in] BSTR  CenId
                      );


        [id(2), helpstring("method Actualizar")]
        HRESULT Actualizar(
            [in] BSTR  CenId,
            [in] BSTR  CenDescripcion
                      );


        [id(3), helpstring("method ObtenerRegistro")]
        HRESULT ObtenerRegistro(
            [in] BSTR  CenId,
            [out, retval] _Recordset**
                      );

        [id(4), helpstring("method ObtenerTodos")]
        HRESULT ObtenerTodos(
            [out, retval] _Recordset**
                      );
    };
};
```

Esta interface define los métodos para los procesos de Agregar, Actualizar y Eliminar (censuras), ObtenerRegistro (para recuperar la información correspondiente a un código de censura) y ObtenerTodos (para obtener todos los registros de censuras).

Interface de bus_ClienteC.Cliente

```
// Proyecto: Renta de Videos
// Typelib: IbusCliente.tlb
// Descripcion: Componente de Negocios de Clientes
// Fecha: 2002/Jun/01

[
  uuid(57513437-9B90-45f2-B1AF-661AE42006D8),
  version(1.0),
  helpstring("IbusCliente 1.0 Type Library")
```

```
]
library IbusClienteLib
{

    // TypeLib: OLE Automation

    importlib("stdole2.tlb");

    // TypeLib: Microsoft ActiveX Data Objects 2.5 Library

    importlib("..\msadol5.dll");


    interface IbusCliente ;

    [
      odl,
      uuid(8B99B338-6F36-4c6c-8AD9-AC3A8EAFB24F),
      version(1.0),
      helpstring("IbusCliente Interface"),
      dual,
      nonextensible,
      oleautomation
    ]

    interface IbusCliente : IDispatch {
        [id(0), helpstring("method Agregar")]
        HRESULT Agregar(
                        [in, out] long*  CliId,
                        [in] BSTR  CliNombre,
                        [in] BSTR  CliApellido,
                        [in] BSTR  CliTipoIdentificacion,
                        [in] BSTR  CliNumIdentificacion,
                        [in] BSTR  CliCiudad,
                        [in] BSTR  CliDireccion,
                        [in] short CliSectorId,
                        [in] BSTR  CliTelefono,
                        [in] BSTR  CliEmail,
                        [in] short CliFormaPagoId,
                        [in] short CliBcoId,
                        [in] BSTR  CliNumCtaBco,
                        [in] short CliTarjetaId,
                        [in] BSTR  CliNumTarjeta,
                        [in] DATE  CliTarjFechaVencim,
                        [in] BSTR  CliServicioEntrega,
                        [in] BSTR  CliServicioRetiro,
                        [in] BSTR  CliLogin,
                        [in] BSTR  CliClave,
                        [in] BSTR  CliEstadoReg
                        );

        [id(1), helpstring("method Actualizar")]
        HRESULT Actualizar(
                        [in] long  CliId,
                        [in] BSTR  CliNombre,
                        [in] BSTR  CliApellido,
                        [in] BSTR  CliTipoIdentificacion,
                        [in] BSTR  CliNumIdentificacion,
                        [in] BSTR  CliCiudad,
                        [in] BSTR  CliDireccion,
                        [in] short CliSectorId,
                        [in] BSTR  CliTelefono,
                        [in] BSTR  CliEmail,
                        [in] short CliFormaPagoId,
                        [in] short CliBcoId,
                        [in] BSTR  CliNumCtaBco,
                        [in] short CliTarjetaId,
                        [in] BSTR  CliNumTarjeta,
                        [in] DATE  CliTarjFechaVencim,
```

```
                    [in] BSTR  CliServicioEntrega,
                    [in] BSTR  CliServicioRetiro,
                    [in] BSTR  CliLogin,
                    [in] BSTR  CliClave,
                    [in] BSTR  CliEstadoReg
                    );

        [id(2), helpstring("method ObtenerRegistro")]
        HRESULT ObtenerRegistro(
            [in] long CliId,
            [out, retval] _Recordset**
                    );

        [id(3), helpstring("method ObtenerPorNombre")]
        HRESULT ObtenerPorNombre(
            [in] BSTR CliNombre,
            [out, retval] _Recordset**
                    );

        [id(4), helpstring("method ObtenerTodos")]
        HRESULT ObtenerTodos(
            [out, retval] _Recordset**
                    );

        [id(5), helpstring("method ValidarLogin")]
        HRESULT ValidarLogin(
            [in] BSTR CliLogin,
            [in] BSTR CliClave,
            [out, retval] long*
                    );

        [id(6), helpstring("method ObtenerInfoCuenta")]
        HRESULT ObtenerInfoCuenta(
            [in] long CliId,
            [out, retval] _Recordset**
                    );
    };
};
```

Esta interface define los métodos para los procesos de Agregar y Actualizar (clientes), ObtenerRegistro (para recuperar la información correspondiente a un código de cliente), ObtenerPorNombre (para recuperar los datos de un cliente por su nombre), ObtenerTodos (para obtener todos los registros de clientes), ValidarLogin (para verificar la contraseña de un cliente en la web) y ObtenerInfoCuenta (para retornar información de la cuenta del cliente, como el número de tickets disponibles).

Interface de bus_FormaPagoC.FormaPago

```
// Proyecto: Renta de Videos
// Typelib: IbusFormaPago.tlb
// Descripcion: Componente de Negocios de Formas de Pago
// Fecha: 2002/Jun/08

[
  uuid(6279AB8B-734C-47f7-9672-1BBDD855D660),
  version(1.0),
  helpstring("IbusFormaPago 1.0 Type Library")
]
library IbusFormaPagoLib
{

    // TypeLib: OLE Automation

    importlib("stdole2.tlb");

    // TypeLib: Microsoft ActiveX Data Objects 2.5 Library

    importlib("..\msado15.dll");


    // Forward declare all types defined in this typelib

    interface IbusFormaPago ;

    [
      odl,
      uuid(0485CED8-AEB7-41e8-97A5-5550E02FF139),
      version(1.0),
      helpstring("IbusFormaPago Interface"),
      dual,
      nonextensible,
      oleautomation
    ]

    interface IbusFormaPago : IDispatch {
        [id(0), helpstring("method Agregar")]
        HRESULT Agregar(
            [in, out] short* FpgId,
            [in] BSTR  FpgDescripcion,
            [in] BSTR  FpgValidezLocal,
            [in] BSTR  FpgValidezWeb
                      );

        [id(1), helpstring("method Eliminar")]
        HRESULT Eliminar(
            [in] short FpgID
                      );

        [id(2), helpstring("method Actualizar")]
        HRESULT Actualizar(
            [in] short FpgId,
            [in] BSTR  FpgDescripcion,
            [in] BSTR  FpgValidezLocal,
            [in] BSTR  FpgValidezWeb
                      );

        [id(3), helpstring("method ObtenerRegistro")]
        HRESULT ObtenerRegistro(
            [in] short FpgId,
            [out, retval] _Recordset**
                      );
```

```
        [id(4), helpstring("method ObtenerTodos")]
        HRESULT ObtenerTodos(
            [out, retval] _Recordset**
                    );

        [id(5), helpstring("method ObtenerPorLugar")]
        HRESULT ObtenerPorLugar(
            [in] BSTR FpgLocation,
            [out, retval] _Recordset**
                    );
    };
};
```

Esta interface define los métodos para los procesos de Agregar, Actualizar y Eliminar (formas de pago), ObtenerRegistro (para recuperar la información correspondiente a un código de forma de pago), ObtenerTodos (para obtener todos los registros de formas de pago) y ObtenerPorLugar (para retornar las formas de pago válidas en una ubicación dada).

Interface de bus_GeneroC.Genero

```
// Proyecto: Renta de Videos
// Typelib: IbusGenero.tlb
// Descripcion: Componente de Datos de Generos de Videos
// Fecha: 2002/Jun/09

[
  uuid(D7278494-185B-4e7c-9438-F446CA0BE9C8),
  version(1.0),
  helpstring("IbusGenero 1.0 Type Library")
]
library IbusGeneroLib
{

    // TypeLib: OLE Automation

    importlib("stdole2.tlb");

    // TypeLib: Microsoft ActiveX Data Objects 2.5 Library

    importlib("..\msado15.dll");


    // Forward declare all types defined in this typelib

    interface IbusGenero ;

    [
      odl,
```

```
    uuid(1A791B6D-49A9-4ac1-9AD9-1EF7BB0D5E46),
    version(1.0),
    helpstring("IbusGenero Interface"),
    dual,
    nonextensible,
    oleautomation
]

interface IbusGenero : IDispatch {
    [id(0), helpstring("method Agregar")]
    HRESULT Agregar(
        [in] BSTR  GenId,
        [in] BSTR  GenDescripcion
                );

    [id(1), helpstring("method Eliminar")]
    HRESULT Eliminar(
        [in] BSTR  GenId
                );

    [id(2), helpstring("method Actualizar")]
    HRESULT Actualizar(
        [in] BSTR  GenId,
        [in] BSTR  GenDescripcion
                );

    [id(3), helpstring("method ObtenerRegistro")]
    HRESULT ObtenerRegistro(
        [in] BSTR  GenId,
        [out, retval] _Recordset**
                );

    [id(4), helpstring("method ObtenerTodos")]
    HRESULT ObtenerTodos(
        [out, retval] _Recordset**
                );
};
};
```

Esta interface define los métodos para los procesos de Agregar, Actualizar y Eliminar (géneros), ObtenerRegistro (para recuperar la información correspondiente a un código de género) y ObtenerTodos (para obtener todos los registros de géneros).

Interface de bus_IdiomaC.Idioma

```
// Proyecto: Renta de Videos
// Typelib: IbusIdioma.tlb
// Descripcion: Componente de Datos de Idiomas
// Fecha: 2002/Jun/10

[
  uuid(99EB1BE1-864F-431c-88B7-6F96FDEC1960),
  version(1.0),
```

```
  helpstring("IbusIdioma 1.0 Type Library")
]
library IbusIdiomaLib
{

    // TypeLib: OLE Automation

    importlib("stdole2.tlb");

    // TypeLib: Microsoft ActiveX Data Objects 2.5 Library

    importlib("..\msado15.dll");


    // Forward declare all types defined in this typelib

    interface IbusIdioma ;

    [
      odl,
      uuid(2906EAB5-DE08-4e66-B9E1-3B737FF23ED1),
      version(1.0),
      helpstring("IbusIdioma Interface"),
      dual,
      nonextensible,
      oleautomation
    ]

    interface IbusIdioma : IDispatch {
        [id(0), helpstring("method Agregar")]
        HRESULT Agregar(
            [in, out] short* IdiId,
            [in] BSTR  IdiDescripcion
                      );

        [id(1), helpstring("method Eliminar")]
        HRESULT Eliminar(
            [in] short IdiId
                      );

        [id(2), helpstring("method Actualizar")]
        HRESULT Actualizar(
            [in] short IdiId,
            [in] BSTR  IdiDescripcion
                      );

        [id(3), helpstring("method ObtenerRegistro")]
        HRESULT ObtenerRegistro(
            [in] short IdiId,
            [out, retval] _Recordset**
                      );

        [id(4), helpstring("method ObtenerTodos")]
        HRESULT ObtenerTodos(
            [out, retval] _Recordset**
                      );
    };
};
```

Esta interface define los métodos para los procesos de Agregar, Actualizar y Eliminar (idiomas), ObtenerRegistro (para recuperar la

información correspondiente a un código de idioma) y ObtenerTodos (para obtener todos los registros de idiomas).

Interface de bus_ItemC.Item

```
// Proyecto: Renta de Videos
// Typelib: IbusItem.tlb
// Descripcion: Componente de Negocios de Items
// Fecha: 2002/Jun/01

[
  uuid(CB912A1A-A493-4b5c-8428-177AD023D7D4),
  version(1.0),
  helpstring("IbusItem 1.0 Type Library")
]

library IbusItemLib
{

    // TypeLib: OLE Automation

    importlib("stdole2.tlb");

    // TypeLib: Microsoft ActiveX Data Objects 2.5 Library

    importlib("..\msado15.dll");


    interface IbusItem ;

    [
      odl,
      uuid(135C7AED-4192-46f0-B055-D14558120F65),
      version(1.0),
      helpstring("IbusItem Interface"),
      dual,
      nonextensible,
      oleautomation
    ]

    interface IbusItem : IDispatch {
        [id(0), helpstring("method Agregar")]
        HRESULT Agregar(
            [in, out] long*  ItmId,
            [in] long        ItmTitId,
            [in] short       ItmMedioId,
            [in] short       ItmIdiomaId,
            [in] BSTR        ItmEstadoRegistro,
            [in] BSTR        ItmEstadoRenta,
            [in] BSTR        ItmMotivoIngreso
                      );

        [id(1), helpstring("method Actualizar")]
        HRESULT Actualizar(
            [in] long        ItmId,
            [in] long        ItmTitId,
            [in] short       ItmMedioId,
            [in] short       ItmIdiomaId,
            [in] BSTR        ItmEstadoRegistro,
            [in] BSTR        ItmEstadoRenta,
            [in] BSTR        ItmMotivoIngreso
                      );
```

```
        [id(2), helpstring("method ObtenerRegistro")]
        HRESULT ObtenerRegistro(
            [in] long         ItmId,
            [out, retval]     _Recordset**
                      );

        [id(3), helpstring("method ObtenerTodos")]
        HRESULT ObtenerTodos(
            [out, retval]     _Recordset**
                      );

        [id(4), helpstring("method ObtenerItemDisponible")]
        HRESULT ObtenerItemDisponible(
            [in] short        MedioId,
            [in] long         TituloId,
            [out, retval]     long*);
    };
};
```

Esta interface define los métodos para los procesos de Agregar y Actualizar (registros de ítems), ObtenerRegistro (para recuperar la información que corresponda a un código de ítem de inventario), ObtenerItemDisponible (para retornar un código de ítem que esté disponible para ser alquilado) y ObtenerTodos (para obtener todos los registros de ítems).

Interface de bus_MedioC.Medio

```
// Proyecto: Renta de Videos
// Typelib: IbusMedio.tlb
// Descripcion: Componente de Negocios de Medios
// Fecha: 2002/May/15

[
  uuid(2CC27258-80F3-40de-9AA0-1446E51D58A9),
  version(1.0),
  helpstring("IbusMedio 1.0 Type Library")
]
library IbusMedioLib
{

    // TypeLib: OLE Automation

    importlib("stdole2.tlb");

    // TypeLib: Microsoft ActiveX Data Objects 2.5 Library

    importlib("..\msado15.dll");
```

```
    // Forward declare all types defined in this typelib

    interface IbusMedio ;

    [
      odl,
      uuid(98DCB86C-5937-4fb6-BA57-1F990B2F478B),
      version(1.0),
      helpstring("IbusMedio Interface"),
      dual,
      nonextensible,
      oleautomation
    ]

    interface IbusMedio : IDispatch {
        [id(0), helpstring("method Agregar")]
        HRESULT Agregar(
            [in, out] long* MedID,
            [in] BSTR MedDesc,
            [in] CURRENCY MedCostoRepos,
            [in] BSTR MedLogo
                      );


        [id(1), helpstring("method Eliminar")]
        HRESULT Eliminar(
            [in] long MedID
                      );

        [id(2), helpstring("method Actualizar")]
        HRESULT Actualizar(
             [in] long MedID,
             [in] BSTR MedDesc,
             [in] CURRENCY MedCostoRepos,
             [in] BSTR MedLogo
                      );

        [id(3), helpstring("method ObtenerRegistro")]
        HRESULT ObtenerRegistro(
            [in] long MedID,
            [out, retval] _Recordset**
                      );

        [id(4), helpstring("method ObtenerTodos")]
        HRESULT ObtenerTodos(
            [out, retval] _Recordset**
                      );
    };
};
```

Esta interface define los métodos para los procesos de Agregar, Actualizar y Eliminar (registros de medios), ObtenerRegistro (para recuperar la información correspondiente a un código de medio) y ObtenerTodos (para obtener todos los registros de medios).

Interface de bus_ParametroC.Parametro

```
// Proyecto: Renta de Videos
// Typelib: IbusParametro.tlb
// Descripcion: Componente de Negocios de Parametros
// Fecha: 2002/Jun/09

[
  uuid(7A5E4E6C-AC70-4408-8A15-1D8F594A2765),
  version(1.0),
  helpstring("IbusParametro 1.0 Type Library")
]
library IbusParametroLib
{

    // TypeLib: OLE Automation

    importlib("stdole2.tlb");

    // TypeLib: Microsoft ActiveX Data Objects 2.5 Library

    importlib("..\msado15.dll");


    // Forward declare all types defined in this typelib

    interface IbusParametro ;

    [
      odl,
      uuid(750017CD-4B35-47d5-85F0-6B7756E8C10A),
      version(1.0),
      helpstring("IbusParametro Interface"),
      dual,
      nonextensible,
      oleautomation
    ]

    interface IbusParametro : IDispatch {
        [id(0), helpstring("method Agregar")]
        HRESULT Agregar(
            [in] BSTR      ParId,
            [in] BSTR      ParDescripcion,
            [in] CURRENCY  ParValor,
            [in] BSTR      ParTexto
                      );

        [id(1), helpstring("method Eliminar")]
        HRESULT Eliminar(
            [in] BSTR      ParId
                      );

        [id(2), helpstring("method Actualizar")]
        HRESULT Actualizar(
            [in] BSTR      ParId,
            [in] BSTR      ParDescripcion,
            [in] CURRENCY  ParValor,
            [in] BSTR      ParTexto
                      );

        [id(3), helpstring("method ObtenerRegistro")]
        HRESULT ObtenerRegistro(
            [in] BSTR      ParId,
            [out, retval] _Recordset**
                      );
```

```
        [id(4), helpstring("method ObtenerTodos")]
        HRESULT ObtenerTodos(
            [out, retval] _Recordset**
                        );
    };
};
```

Esta interface define los métodos para los procesos de Agregar, Actualizar y Eliminar (registros de parámetros), ObtenerRegistro (para recuperar la información correspondiente a un código de parámetro) y ObtenerTodos (para obtener todos los registros de parámetros).

Interface de bus_PreRegistroC.PreRegistro

```
// Proyecto: Renta de Videos
// Typelib: IbusPreRegistro.tlb
// Descripcion: Componente de Negocios de PreRegistros
// Fecha: 2002/Jun/12

[
  uuid(9C7D5343-FE77-4501-B77A-7344F22D23A9),
  version(1.0),
  helpstring("IbusPreRegistro 1.0 Type Library")
]

library IbusPreRegistroLib
{

    // TypeLib: OLE Automation

    importlib("stdole2.tlb");

    // TypeLib: Microsoft ActiveX Data Objects 2.5 Library

    importlib("..\msado15.dll");


    interface IbusPreRegistro ;

    [
      odl,
      uuid(674875A3-11D6-4f69-B957-D627F52A1E27),
      version(1.0),
      helpstring("IbusPreRegistro Interface"),
      dual,
      nonextensible,
      oleautomation
    ]

    interface IbusPreRegistro : IDispatch {
        [id(0), helpstring("method Agregar")]
        HRESULT Agregar(
                        [in] BSTR  PrgTipoIdentif,
                        [in] BSTR  PrgNumIdentif,
```

```
                [in] BSTR  PrgLogin,
                [in] BSTR  PrgClave,
                [in] BSTR  PrgNombre,
                [in] BSTR  PrgApellido,
                [in] short PrgSectorId,
                [in] BSTR  PrgDireccion,
                [in] BSTR  PrgTelefono,
                [in] BSTR  PrgEmail,
                [in] short PrgFormaPagoId,
                [in] short PrgTarjetaId,
                [in] BSTR  PrgTarjetaNum,
                [in] BSTR  PrgTarjetaVenc,
                [in] BSTR  PrgTarjetaTitular,
                [in] BSTR  PrgServEntrega,
                [in] BSTR  PrgServRetiro,
                [in] BSTR  PrgEstadoReg
                );


        [id(1), helpstring("method Actualizar")]
        HRESULT Actualizar(
                [in] BSTR  PrgTipoIdentif,
                [in] BSTR  PrgNumIdentif,
                [in] BSTR  PrgLogin,
                [in] BSTR  PrgClave,
                [in] BSTR  PrgNombre,
                [in] BSTR  PrgApellido,
                [in] short PrgSectorId,
                [in] BSTR  PrgDireccion,
                [in] BSTR  PrgTelefono,
                [in] BSTR  PrgEmail,
                [in] short PrgFormaPagoId,
                [in] short PrgTarjetaId,
                [in] BSTR  PrgTarjetaNum,
                [in] BSTR  PrgTarjetaVenc,
                [in] BSTR  PrgTarjetaTitular,
                [in] BSTR  PrgServEntrega,
                [in] BSTR  PrgServRetiro,
                [in] BSTR  PrgEstadoReg
                );

        [id(2), helpstring("method ObtenerRegistro")]
        HRESULT ObtenerRegistro(
                [in] BSTR  PrgTipoIdentif,
                [in] BSTR  PrgNumIdentif,
                [out, retval] _Recordset**
                );

        [id(3), helpstring("method ObtenerPorApellido")]
        HRESULT ObtenerPorApellido(
                [in] BSTR  PrgApellido,
                [out, retval] _Recordset**
                );

        [id(4), helpstring("method ObtenerPorCliente")]
        HRESULT ObtenerPorCliente(
                [in] long  PrgCliId,
                [out, retval] _Recordset**
                );

        [id(5), helpstring("method ListarPorEstado")]
        HRESULT ListarPorEstado(
                [in] BSTR  PrgEstadoReg,
                [out, retval] _Recordset**
                );

        [id(6), helpstring("method Evaluar")]
        HRESULT Evaluar(
                [in] BSTR  PrgTipoIdentif,
```

```
                        [in] BSTR  PrgNumIdentif,
                        [in] DATE  PrgEvalFecha,
                        [in] BSTR  PrgEvalCalif,
                        [in] BSTR  PrgEvalTexto,
                        [in] BSTR  PrgEvalUsuId,
                        [in, out] long*  PrgEvalCliId,
                        [in] BSTR  PrgEstadoReg
                        );
    );
);
```

Esta interface define los métodos para los procesos de Agregar y Actualizar (registros de solicitudes de membresía), ObtenerRegistro (para recuperar la información correspondiente a la identificación de una persona), ObtenerPorApellido (para retornar todos las solicitudes que coincidan con un apellido dado), ObtenerPorCliente (para retornar la información de la solicitud de un cliente), ListarPorEstado (para retornar todas las solicitudes con un estado en particular) y Evaluar (para facilitar el proceso de evaluación de solicitudes de membresía).

Interface de bus_PubliWebC.PubliWeb

```
// Proyecto: Renta de Videos
// Typelib: IbusPubliWeb.tlb
// Descripcion: Componente de Negocios de Publicación en Web
// Fecha: 2002/Jun/09

[
  uuid(53F45E05-232B-45fc-8560-E7721502B839),
  version(1.0),
  helpstring("IbusPubliWeb 1.0 Type Library")
]

library IbusPubliWebLib
{

    // TypeLib: OLE Automation

    importlib("stdole2.tlb");

    // TypeLib: Microsoft ActiveX Data Objects 2.5 Library

    importlib("..\msadol5.dll");


    // Forward declare all types defined in this typelib
```

```
interface IbusPubliWeb ;

[
  odl,
  uuid(12D20F86-5466-475a-9F71-ED708FF0334D),
  version(1.0),
  helpstring("IbusPubliWeb Interface"),
  dual,
  nonextensible,
  oleautomation
]


interface IbusPubliWeb : IDispatch {
    [id(0), helpstring("method Agregar")]
    HRESULT Agregar(
        [in, out] short* PubCodigo,
        [in] short PubCatId,
        [in] BSTR  PubTitulo,
        [in] BSTR  PubDescripcion,
        [in] BSTR  PubRutaFoto,
        [in] BSTR  PubEstadoReg,
        [in] long  PubTitId
                );


    [id(1), helpstring("method Eliminar")]
    HRESULT Eliminar(
        [in] short PubCodigo
                );


    [id(2), helpstring("method Actualizar")]
    HRESULT Actualizar(
        [in] short PubCodigo,
        [in] short PubCatId,
        [in] BSTR  PubTitulo,
        [in] BSTR  PubDescripcion,
        [in] BSTR  PubRutaFoto,
        [in] BSTR  PubEstadoReg,
        [in] long  PubTitId
                );


    [id(3), helpstring("method ObtenerRegistro")]
    HRESULT ObtenerRegistro(
        [in] short PubCodigo,
        [out, retval] _Recordset**
                );

    [id(4), helpstring("method ObtenerTodos")]
    HRESULT ObtenerTodos(
        [out, retval] _Recordset**
                );


    [id(5), helpstring("method ObtenerPorCategoria")]
    HRESULT ObtenerPorCategoria(
        [in] short PubCatId,
        [out, retval] _Recordset**
                );
};
};
```

Esta interface define los métodos para los procesos de Agregar, Actualizar y Eliminar (registros de publicaciones en la web), ObtenerRegistro (para recuperar la información correspondiente a un código de publicación), ObtenerPorCategoria (para retornar todas los registros de publicaciones que coincidan con un código de categoría dado) y ObtenerTodos (para obtener todos los registros de publicaciones en web).

Interface de bus_SectorC.Sector

```
// Proyecto: Renta de Videos
// Typelib: IbusSector.tlb
// Descripcion: Componente de Negocios de Sectores
// Fecha: 2002/Jun/09

[
  uuid(20A60492-DEAB-4cfc-B437-60BEB7A45CD1),
  version(1.0),
  helpstring("IbusSector 1.0 Type Library")
]


library IbusSectorLib
{

    // TypeLib: OLE Automation

    importlib("stdole2.tlb");

    // TypeLib: Microsoft ActiveX Data Objects 2.5 Library

    importlib("..\msado15.dll");


    // Forward declare all types defined in this typelib

    interface IbusSector ;

    [
      odl,
      uuid(CE95F8A0-7089-417d-9BE7-3B911E1D570D),
      version(1.0),
      helpstring("IbusSector Interface"),
      dual,
      nonextensible,
      oleautomation
    ]

    interface IbusSector : IDispatch {
        [id(0), helpstring("method Agregar")]
        HRESULT Agregar(
            [in, out] short* SctId,
```

```
            [in] BSTR  SctDescripcion
                    );

        [id(1), helpstring("method Eliminar")]
        HRESULT Eliminar(
            [in] short SctId
                    );

        [id(2), helpstring("method Actualizar")]
        HRESULT Actualizar(
            [in] short SctId,
            [in] BSTR  SctDescripcion
                    );

        [id(3), helpstring("method ObtenerRegistro")]
        HRESULT ObtenerRegistro(
            [in] long SctId,
            [out, retval] _Recordset**
                    );

        [id(4), helpstring("method ObtenerTodos")]
        HRESULT ObtenerTodos(
             [out, retval] _Recordset**
                    );
    };
};
```

Esta interface define los métodos para los procesos de Agregar, Actualizar y Eliminar (registros de sectores), ObtenerRegistro (para recuperar la información correspondiente a un código de sector) y ObtenerTodos (para obtener todos los registros de sectores).

Interface de bus_SecuencialC.Secuencial

```
// Proyecto: Renta de Videos
// Typelib: IbusSecuencial.tlb
// Descripcion: Componente de Negocios de Secuenciales
// Fecha: 2002/May/23

[
  uuid(7FAF571D-DF9C-41d5-BBEA-E481A3702C24),
  version(1.0),
  helpstring("IbusSecuencial 1.0 Type Library")
]

library IbusSecuencialLib
{

    // TypeLib: OLE Automation

    importlib("stdole2.tlb");

    // TypeLib: Microsoft ActiveX Data Objects 2.5 Library
```

```
    importlib("..\msado15.dll");


    // Forward declare all types defined in this typelib

    interface IbusSecuencial ;

    [
      odl,
      uuid(CABFA5E6-F60D-421e-8987-A1D8FC09F6DE),
      version(1.0),
      helpstring("IbusSecuencial Interface"),
      dual,
      nonextensible,
      oleautomation
    ]

    interface IbusSecuencial : IDispatch {
        [id(0), helpstring("method Agregar")]
        HRESULT Agregar(
            [in] long SecId,
            [in] BSTR SecDesc,
            [in] long SecValor
                      );


        [id(1), helpstring("method Eliminar")]
        HRESULT Eliminar(
            [in] long SecId
                      );

        [id(2), helpstring("method Actualizar")]
        HRESULT Actualizar(
            [in] long SecId,
            [in] BSTR SecDesc,
            [in] long SecValor
                      );

        [id(3), helpstring("method ObtenerRegistro")]
        HRESULT ObtenerRegistro(
            [in] long SecId,
            [out, retval] _Recordset**
                      );

        [id(4), helpstring("method ObtenerSiguiente")]
        HRESULT ObtenerSiguiente(
            [in] long SecId,
            [out, retval] long*
                      );

        [id(5), helpstring("method ObtenerTodos")]
        HRESULT ObtenerTodos(
               [out, retval] _Recordset**
                      );
    };
};
```

Esta interface define los métodos para los procesos de Agregar, Actualizar y Eliminar (registros de secuencias), ObtenerRegistro (para recuperar la información correspondiente a un código de secuencia),

ObtenerSiguiente (para obtener el siguiente número de secuencia de una entidad dada) y ObtenerTodos (para obtener todos los registros de secuencias).

Interface de bus_TablasNumC.TablasNum

```
// Proyecto: Renta de Videos
// Typelib: IbusTablasNum.tlb
// Descripcion: Componente de Negocios de Tablas Maestras (clave numérica)
// Fecha: 2002/Jun/08

[
  uuid(1C70FA49-FE87-4938-917C-B1A2B5D6CAC6),
  version(1.0),
  helpstring("IbusTablasNum 1.0 Type Library")
]
library IbusTablasNumLib
{

    // TypeLib: OLE Automation

    importlib("stdole2.tlb");

    // TypeLib: Microsoft ActiveX Data Objects 2.5 Library

    importlib("..\msadol5.dll");


    // Forward declare all types defined in this typelib

    interface IbusTablasNum ;

    [
      odl,
      uuid(0A624E69-2780-4e37-8EE1-6897D857DFDC),
      version(1.0),
      helpstring("IbusTablasNum Interface"),
      dual,
      nonextensible,
      oleautomation
    ]

    interface IbusTablasNum : IDispatch {
        [id(0), helpstring("method Agregar")]
        HRESULT Agregar(
            [in] long         TabNum,
            [in, out] long*   TabId,
            [in] BSTR         TabDescripcion
                      );


        [id(1), helpstring("method Eliminar")]
        HRESULT Eliminar(
            [in] long         TabNum,
            [in] long         TabId
                      );

        [id(2), helpstring("method Actualizar")]
        HRESULT Actualizar(
            [in] long         TabNum,
```

```
            [in] long          TabId,
            [in] BSTR          TabDescripcion
                        );

        [id(3), helpstring("method ObtenerRegistro")]
        HRESULT ObtenerRegistro(
            [in] long          TabNum,
            [in] long          TabId,
            [out, retval]      _Recordset**
                        );

        [id(4), helpstring("method ObtenerTodos")]
        HRESULT ObtenerTodos(
            [in] long          TabNum,
            [out, retval]      _Recordset**
                        );
    };
};
```

Esta interface define los métodos para los procesos de Agregar, Actualizar y Eliminar (registros de una tabla), ObtenerRegistro (para recuperar la información correspondiente a un código de una tabla) y ObtenerTodos (para obtener todos los registros de una tabla dada).

Interface de bus_TalonarioC.Talonario

```
// Proyecto: Renta de Videos
// Typelib: IbusTalonario.tlb
// Descripcion: Componente de Negocios de Talonarios
// Fecha: 2002/Jun/22

[
  uuid(3DC6DA34-FC2C-4477-9952-2753D73152A5),
  version(1.0),
  helpstring("IbusTalonario 1.0 Type Library")
]
library IbusTalonarioLib
{

    // TypeLib: OLE Automation

    importlib("stdole2.tlb");

    // TypeLib: Microsoft ActiveX Data Objects 2.5 Library

    importlib("..\msado15.dll");


    // Forward declare all types defined in this typelib

    interface IbusTalonario ;

    [
      odl,
```

```
    uuid(F3DC7C55-E79A-46bb-A39E-466BB070F007),
    version(1.0),
    helpstring("IbusTalonario Interface"),
    dual,
    nonextensible,
    oleautomation
  ]

  interface IbusTalonario : IDispatch {
      [id(0), helpstring("method Agregar")]
      HRESULT Agregar(
                    [in, out] long*              TalId,
                    [in] long                    TalCliId,
                    [in] DATE                    TalFecCompra,
                    [in] short                   TalTtiId,
                    [in] short                   TalNumTickets,
                    [in] short                   TalSaldoTickets,
                    [in] CURRENCY                TalValor,
                    [in] long                    TalTransId,
                    [in] short                   TalFormaPagoId,
                    [in] short                   TalTarjId,
                    [in] BSTR                    TalNumTarjeta,
                    [in] BSTR                    TalTarjFecVencim,
                    [in] BSTR                    TalTarjTitular,
                    [in] BSTR                    TalTarjNumAutoriza
                    );


      [id(1), helpstring("method Eliminar")]
      HRESULT Eliminar(
                    [in] long     TalId
                    );


      [id(2), helpstring("method Actualizar")]
      HRESULT Actualizar(
                    [in] long       TalId,
                    [in] long       TalCliId,
                    [in] DATE       TalFecCompra,
                    [in] short      TalTtiId,
                    [in] short      TalNumTickets,
                    [in] short      TalSaldoTickets,
                    [in] CURRENCY   TalValor,
                    [in] long       TalTransId,
                    [in] short      TalFormaPagoId,
                    [in] short      TalTarjId,
                    [in] BSTR       TalNumTarjeta,
                    [in] BSTR       TalTarjFecVencim,
                    [in] BSTR       TalTarjTitular,
                    [in] BSTR       TalTarjNumAutoriza
                    );


      [id(3), helpstring("method ObtenerRegistro")]
      HRESULT ObtenerRegistro(
                    [in] long        TalId,
                    [out, retval]    _Recordset**
                    );


      [id(4), helpstring("method ObtenerTodos")]
      HRESULT ObtenerTodos(
                    [out, retval] _Recordset**
                    );
  };
};
```

Esta interface define los métodos para los procesos de Agregar, Actualizar y Eliminar (registros de talonarios), ObtenerRegistro (para recuperar la información correspondiente a un código de talonario) y ObtenerTodos (para obtener todos los registros de talonarios).

Interface de bus_TalonTipoC.TalonTipo

```
// Proyecto: Renta de Videos
// Typelib: IbusTalonTipo.tlb
// Descripcion: Componente de Negocios de Tipos de Talonarios
// Fecha: 2002/Jun/08

[
  uuid(990ECA4A-8CDE-4bcc-AB67-9B7DCDF6B14F),
  version(1.0),
  helpstring("IbusTalonTipo 1.0 Type Library")
]

library IbusTalonTipoLib
{

    // TypeLib: OLE Automation

    importlib("stdole2.tlb");

    // TypeLib: Microsoft ActiveX Data Objects 2.5 Library

    importlib("..\msado15.dll");


    interface IbusTalonTipo ;

    [
      odl,
      uuid(9DB5C3EF-FC2E-4b9a-B0EB-92DB9482BDD3),
      version(1.0),
      helpstring("IbusTalonTipo Interface"),
      dual,
      nonextensible,
      oleautomation
    ]

    interface IbusTalonTipo : IDispatch {
        [id(0), helpstring("metodo Agregar")]
        HRESULT Agregar(
                        [in, out] short*      TtiId,
                        [in] BSTR             TtiDescripcion,
                        [in] short            TtiNumTickets,
                        [in] CURRENCY         TtiValor
                        );

        [id(1), helpstring("method Actualizar")]
        HRESULT Actualizar(
                        [in] short            TtiId,
                        [in] BSTR             TtiDescripcion,
                        [in] short            TtiNumTickets,
                        [in] CURRENCY         TtiValor
                        );
```

```
        [id(2), helpstring("method Eliminar")]
        HRESULT Eliminar(
                        [in] short              TtiId
                        );

        [id(3), helpstring("method ObtenerRegistro")]
        HRESULT ObtenerRegistro(
                        [in] short              TtiId,
                        [out, retval]           _Recordset**
                        );

        [id(4), helpstring("method ObtenerTodos")]
        HRESULT ObtenerTodos(
                        [out, retval]           _Recordset**
                        );
    };
};
```

Esta interface define los métodos para los procesos de Agregar, Actualizar y Eliminar (registros de tipos de talonarios), ObtenerRegistro (para recuperar la información correspondiente a un código de tipo de talonario) y ObtenerTodos (para obtener todos los registros de tipos de talonarios).

Interface de bus_TarjetaC.Tarjeta

```
// Proyecto: Renta de Videos
// Typelib: IbusTarjeta.tlb
// Descripcion: Componente de Negocios de Tarjetas de Credito
// Fecha: 2002/Jun/08

[
  uuid(B0753024-EF56-4894-B5D3-B3BFB762F86B),
  version(1.0),
  helpstring("IbusTarjeta 1.0 Type Library")
]
library IbusTarjetaLib
{

    // TypeLib: OLE Automation

    importlib("stdole2.tlb");

    // TypeLib: Microsoft ActiveX Data Objects 2.5 Library

    importlib("..\msado15.dll");


    // Forward declare all types defined in this typelib

    interface IbusTarjeta ;
```

```
    [
      odl,
      uuid(797FC9DE-9FC2-49f1-8ADC-AB3C565CD5C3),
      version(1.0),
      helpstring("IbusTarjeta Interface"),
      dual,
      nonextensible,
      oleautomation
    ]

    interface IbusTarjeta : IDispatch {
        [id(0), helpstring("method Agregar")]
        HRESULT Agregar(
            [in] short TrjId,
            [in] BSTR  TrjDescripcion
                      );

        [id(1), helpstring("method Eliminar")]
        HRESULT Eliminar(
            [in] short TrjId
                      );

        [id(2), helpstring("method Actualizar")]
        HRESULT Actualizar(
            [in] short TrjId,
            [in] BSTR  TrjDescripcion
                      );

        [id(3), helpstring("method ObtenerRegistro")]
        HRESULT ObtenerRegistro(
            [in] long TrjId,
            [out, retval] _Recordset**
                      );

        [id(4), helpstring("method ObtenerTodos")]
        HRESULT ObtenerTodos(
             [out, retval] _Recordset**
                      );

        [id(5), helpstring("method AutorizarTransTarjeta")]
        HRESULT AutorizarTransTarjeta(
            [in] short       TrjId,
            [in] BSTR        TrjNumero,
            [in] DATE        TrjFecVencim,
            [in] CURRENCY    TrjValor,
            [in] BSTR        TrjTitular,
             [in, out] long*  TrjAutorizacion,
             [out, retval] long*
                      );
    };
};
```

Esta interface define los métodos para los procesos de Agregar, Actualizar y Eliminar (registros de tarjetas), ObtenerRegistro (para recuperar la información correspondiente a un código de tarjeta), ObtenerTodos (para obtener todos los registros de tarjetas) y

AutorizarTransTarjeta (para validar la información de transacciones con tarjeta de crédito).

Interface de bus_TituloC.Titulo

```
// Proyecto: Renta de Videos
// Typelib: IbusTitulo.tlb
// Descripcion: Componente de Negocios de Titulos
// Fecha: 2002/Jun/01

[
  uuid(3D9A6594-9BA9-40b1-9A06-9638A0B3F1AD),
  version(1.0),
  helpstring("IbusTitulo 1.0 Type Library")
]

library IbusTituloLib
{

    // TypeLib: OLE Automation

    importlib("stdole2.tlb");

    // TypeLib: Microsoft ActiveX Data Objects 2.5 Library

    importlib("..\msado15.dll");


    interface IbusTitulo ;

    [
      odl,
      uuid(638E324F-486F-4fd8-A489-68EA52FB1660),
      version(1.0),
      helpstring("IbusTitulo Interface"),
      dual,
      nonextensible,
      oleautomation
    ]

    interface IbusTitulo : IDispatch {
        [id(0), helpstring("method Agregar")]
        HRESULT Agregar(
            [in, out] long* TitId,
            [in] BSTR TitNombre,
            [in] BSTR TitNombreOrig,
            [in] BSTR TitSinopsis,
            [in] BSTR TitProductor,
            [in] BSTR TitDirector,
            [in] BSTR TitReparto,
            [in] BSTR TitCensuraId,
            [in] BSTR TitGenero1Id,
            [in] BSTR TitGenero2Id,
            [in] BSTR TitGenero3Id,
            [in] BSTR TitNominado,
            [in] BSTR TitNominaciones,
            [in] BSTR TitPremiado,
            [in] BSTR TitPremios,
            [in] BSTR TitOscarGanado,
            [in] BSTR TitDuracion,
            [in] long TitProduccion,
            [in] BSTR TitLogo
```

```
                    );

        [id(1), helpstring("method Actualizar")]
        HRESULT Actualizar(
            [in] long TitId,
            [in] BSTR TitNombre,
            [in] BSTR TitNombreOrig,
            [in] BSTR TitSinopsis,
            [in] BSTR TitProductor,
            [in] BSTR TitDirector,
            [in] BSTR TitReparto,
            [in] BSTR TitCensuraId,
            [in] BSTR TitGenero1Id,
            [in] BSTR TitGenero2Id,
            [in] BSTR TitGenero3Id,
            [in] BSTR TitNominado,
            [in] BSTR TitNominaciones,
            [in] BSTR TitPremiado,
            [in] BSTR TitPremios,
            [in] BSTR TitOscarGanado,
            [in] BSTR TitDuracion,
            [in] long TitProduccion,
            [in] BSTR TitLogo
                    );

        [id(2), helpstring("method ObtenerRegistro")]
        HRESULT ObtenerRegistro(
            [in] long TitId,
            [out, retval] _Recordset**
                    );

        [id(3), helpstring("method ObtenerTodos")]
        HRESULT ObtenerTodos(
            [out, retval] _Recordset**
                    );

        [id(4), helpstring("method ObtenerSegunCriterio")]
        HRESULT ObtenerSegunCriterio(
            [in] long MedioId,
            [in] long OpcBusqueda,
            [in] BSTR ArgBusqueda,
            [out, retval] _Recordset**
                    );
    };
};
```

Esta interface define los métodos para los procesos de Agregar y Actualizar (registros de títulos), ObtenerRegistro (para recuperar la información correspondiente a un código de título), ObtenerSegunCriterio (para obtener todos los títulos que conformen los criterios indicados por los parámetros dados) y ObtenerTodos (para obtener todos los registros de títulos).

Interface de bus_TransaccionC.Transaccion

```
// Proyecto: Renta de Videos
// Typelib: IbusTransaccion.tlb
// Descripcion: Componente de Negocios de Transacciones
// Fecha: 2002/Jun/22

[
  uuid(112A6BD0-CF8C-4953-9F1F-535FE29A6B2F),
  version(1.0),
  helpstring("IbusTransaccion 1.0 Type Library")
]
library IbusTransaccionLib
{

    // TypeLib: OLE Automation

    importlib("stdole2.tlb");

    // TypeLib: Microsoft ActiveX Data Objects 2.5 Library

    importlib("..\msado15.dll");


    // Forward declare all types defined in this typelib

    interface IbusTransaccion ;

    [
      odl,
      uuid(8C30DA7C-49C3-4158-A464-2F0EBE378340),
      version(1.0),
      helpstring("IbusTransaccion Interface"),
      dual,
      nonextensible,
      oleautomation
    ]

    interface IbusTransaccion : IDispatch {
        [id(0), helpstring("method VenderTalonario")]
        HRESULT VenderTalonario(
                        [in, out] long*             TalId,
                        [in] long                   TalCliId,
                        [in] short                  TalTtiId,
                        [in] short                  TalNumTickets,
                        [in] CURRENCY               TalValor,
                        [in] short                  TalFormaPagoId,
                        [in] short                  TalTarjId,
                        [in] BSTR                   TalNumTarjeta,
                        [in] BSTR                   TalTarjFecVencim,
                        [in] BSTR                   TalTarjTitular
                        );

        [id(1), helpstring("method AlquilarListaItems")]
        HRESULT AlquilarListaItems(
                        [in] long                   CliId,
                        [in] BSTR                   ListaItems,
                        [in] BSTR                   ServEntrega,
                        [in] BSTR                   ServRetiro,
                        [in, out] BSTR*             ListaRespuesta
                        );

        [id(2), helpstring("method EntregarItem")]
        HRESULT EntregarItem(
                        [in] long                   TrnId,
                        [in] DATE                   FecEntrega,
```

```
                    [in] BSTR                   TipoEntrega
                    );

        [id(3), helpstring("method RecibirItem")]
        HRESULT RecibirItem(
                    [in] long                   TrnId,
                    [in] DATE                   FecRecepcion
                    );

        [id(4), helpstring("method ListarItemsPorEstado")]
        HRESULT ListarItemsPorEstado(
                    [in] BSTR                    Estado,
                    [in] VARIANT                CliId,
                    [out, retval]               _Recordset**
                    );

        [id(5), helpstring("method AlquilarItem")]
        HRESULT AlquilarItem(
                    [in, out] long*             TrnId,
                    [in] long                   CliId,
                    [in] long                   TitId,
                    [in] short                  MedId,
                    [in] BSTR                   ServEntrega,
                    [in] BSTR                   ServRetiro
                    );
    };
};
```

Esta interface define los métodos para los procesos de VenderTalonario (para la venta de talonarios a los clientes), AlquilarListaItems (para el procedimiento de alquiler en la web), AlquilarItem (para alquilar un solo ítem), EntregarItem (para el proceso de entrega de videos), RecibirItem (para el proceso de recepción de los videos) y ListarItemsPorEstado (para retornar todos los items que tengan un status determinado).

6.6. Comunicación entre Objetos distribuídos.

Para la comunicación entre objetos se utiliza la infraestructura de comunicaciones propia de COM. En este caso se utiliza la característica de COM de transparencia de ubicación para instanciar objetos residentes en diferentes ubicaciones.

En base a la type library de los componentes, COM localiza en el registro de Windows la información de la ubicación real de un componente.

En el caso de las llamadas hechas desde los scripts ASP, se hacen invocaciones a procedimientos públicos que imitan la sintaxis de llamada de los procedimientos privados equivalentes. De esta manera se logra evitar la duplicación de código.

La mayoría de los parámetros de los métodos utilizan las llamadas por valor para lograr un mejor rendimiento. Sólo en los casos indispensables se han usado parámetros por referencia. En los procedimientos públicos los parámetros de referencia se establecieron de tipo Variant para evitar incompatibilidades en las llamadas desde los scripts ASP.

6.7. Proceso servidor transaccional.

El servidor transaccional fue implementado en el entorno COM+ que viene integrado con el Windows 2000.

A la aplicación COM+ se la denominó "RentaVideo".

Los componentes de negocios y datos se incorporaron a la aplicación COM+ con la opción "Instalar nuevo componente". Se agregaron tanto la DLL como la TLB de cada componente.

Los componentes de negocios están cargados con la opción "Requiere Transacciones" y los componentes de datos con la opción "Compatible con Transacciones".

El servidor transaccional se encarga de atender las solicitudes de los clientes standalone y de los scripts ASP para la instanciación de objetos. Para ello crea contextos de ejecución en donde se realizan las conexiones a la base de datos y se inician y finalizan las transacciones. Se encarga también de deshacer las actualizaciones a la base de datos en caso de que ocurra una llamada al procedimiento SetAbort del contexto.

También se utilizan los servicios de COM+ cuando se instancian los objetos de datos desde los objetos de negocios.

7

IMPLEMENTACIÓN DE LA CAPA DE PRESENTACIÓN

7. IMPLEMENTACIÓN DE LA CAPA DE PRESENTACIÓN.

7.1 Diseño del Sitio Web – Cliente basado en browser.

El diseño del sitio Web se hizo de acuerdo a los requerimientos y necesidades de nuestro cliente, una empresa de renta de videos. Se contemplo para el diseño de la aplicación el seguimiento de una línea de imagen corporativa o institucional, basándonos en el negocio de una empresa que provee servicios.

El objetivo de crear esta aplicación es que por medio del e-commerce podemos expandir y desarrollar la empresa o negocio vendiendo sus productos y/o servicios desde su propia tienda virtual (sitio) hacia todo el mundo. Además de ampliar el mercado al llegar potencialmente a todas partes, reducirá costos y brindará un servicio cómodo, exclusivo, eficiente y seguro para todos sus clientes.

El diseño del sitio se apega a los requerimientos que exije el mundo actual del Internet, es decir un sitio dinámico, facil de navegar, interactivo, amigable con el usuario y muy vistoso, lo mismo que lo hace un sistema que permite mostrar la información a nuestros clientes de una manera ordenada y clasificada, permitiéndole al usuario que sea él quien tenga el control sobre la aplicación.

El sitio está conformado básicamente por dos grupos de páginas:

1.- Las páginas dinámicas (ASP)

2.- Las páginas estáticas (HTML)

Siendo las herramientas usadas para su diseño las siguientes: Macromedia Flash, Macromedia Fireworks, Swift 3D, Adobe Photoshop y Adobe Ilustrator

Y para el desarrollo de las páginas dinámicas y estáticas las siguientes aplicaciones: Visual InterDev y Macromedia DreamWeaver

7.2. Cliente standalone.

La aplicación standalone está dirigida al personal de Ecuacinema que realice la gestión operativa de Renta de Videos, y la hemos llamado **Módulo de Renta de Videos**.

Para una mejor comprensión de la aplicación la vamos ir recorriendo de acuerdo a las bondades que nos ofrece.

7.2.1. Acceso al Sistema.

Al ingresar a la aplicación ésta le solicita que ingrese el usuario y la contraseña, y una vez que hayan sido verificados sus datos el sistema se habilita.

Figura 7-1. Pantalla de login de usuario

7.2.2. Menú de la aplicación.

La aplicación muestra una barra de menús y las opciones más comunes en una barra de herramientas.

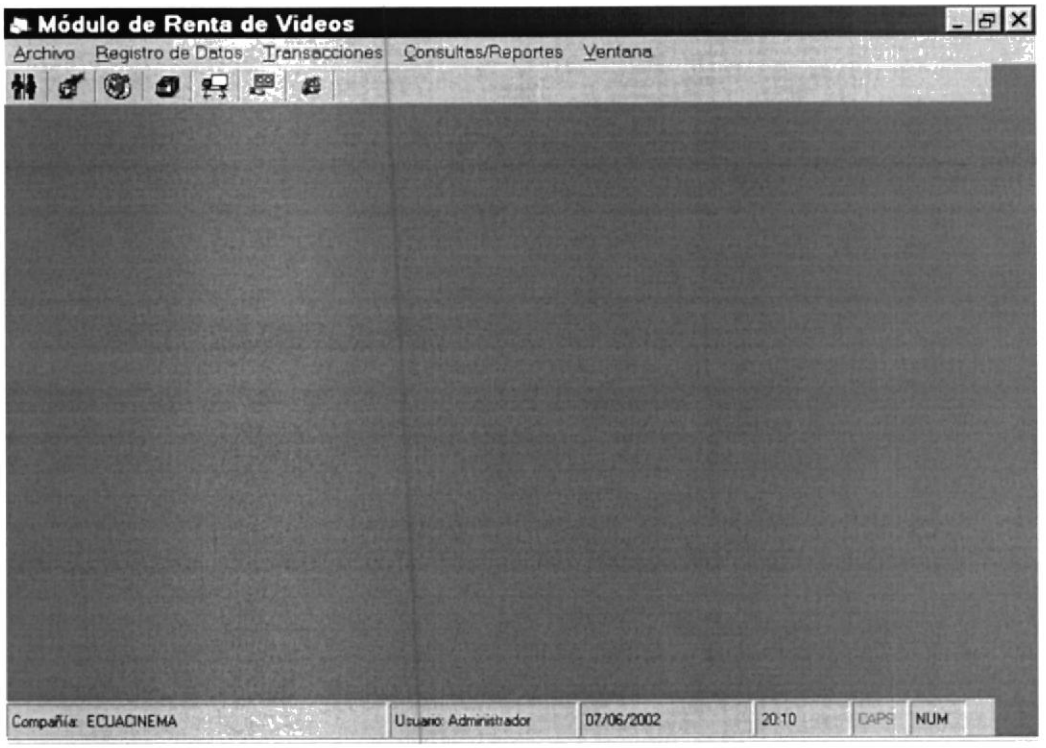


Figura 7-2. Menú principal de Módulo de Renta de Videos

La barra de menús nos muestra las siguientes opciones principales del sistema standalone que son:

- Registro de Datos
- Transacciones
- Consultas / Reportes

7.2.3. Registro de Datos.

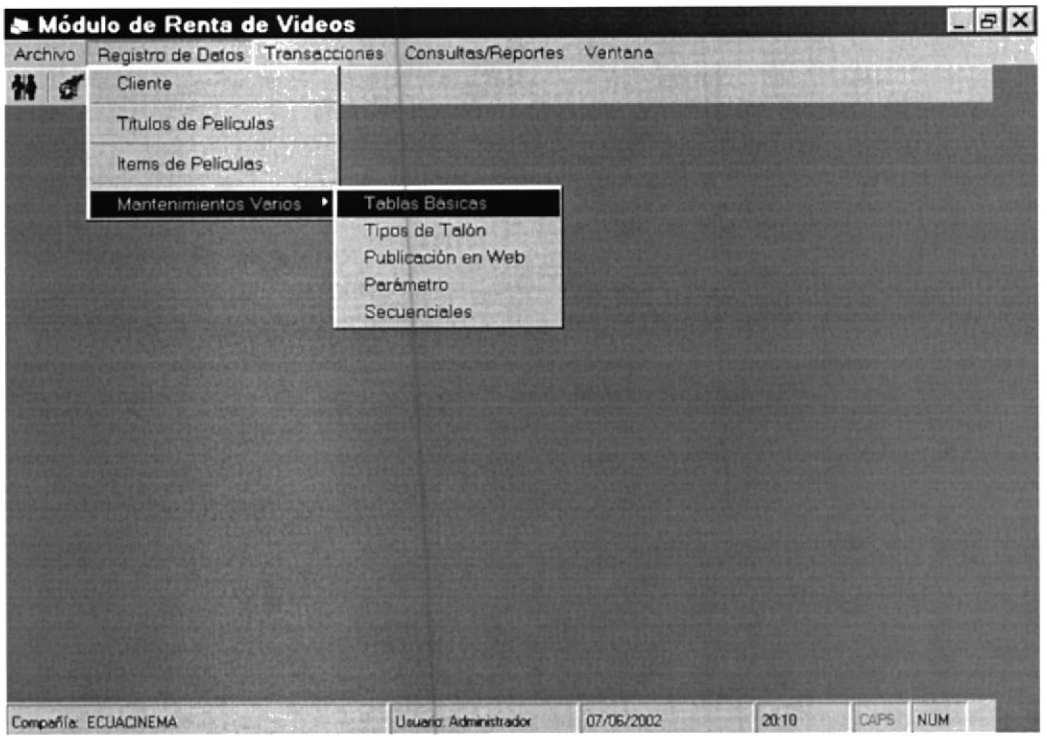


Figura 7-3. Menú de Mantenimientos Varios

En el menú de Registro de Datos, nos permite realizar el mantenimiento a información relevante como: clientes, títulos de películas, ítems de películas, emisoras de tarjetas de crédito, tipos de censura, tipos de géneros, tipos de talonarios, etc.

7.2.3.1. Mantenimiento de Clientes.

Como se aprecia en la pantalla a continuación, en ella se permite ingresar toda la información referente al cliente y su relación con el negocio de la empresa.

Los campos que se requieren son: Tipo de identificación (cédula ó RUC), número de identificación, nombres, apellidos, sector de la ciudad donde reside, dirección domiciliaria, teléfono, e-mail, forma de pago (efectivo ó tarjeta de crédito), si eligió tarjeta de crédito debe incluir el emisor de la tarjeta, número y fecha de vencimiento. Adicionalmente se le permite seleccionar los servicios de entrega y retiro de películas a domicilio, como también el login de acceso al sitio Web de Renta de videos.

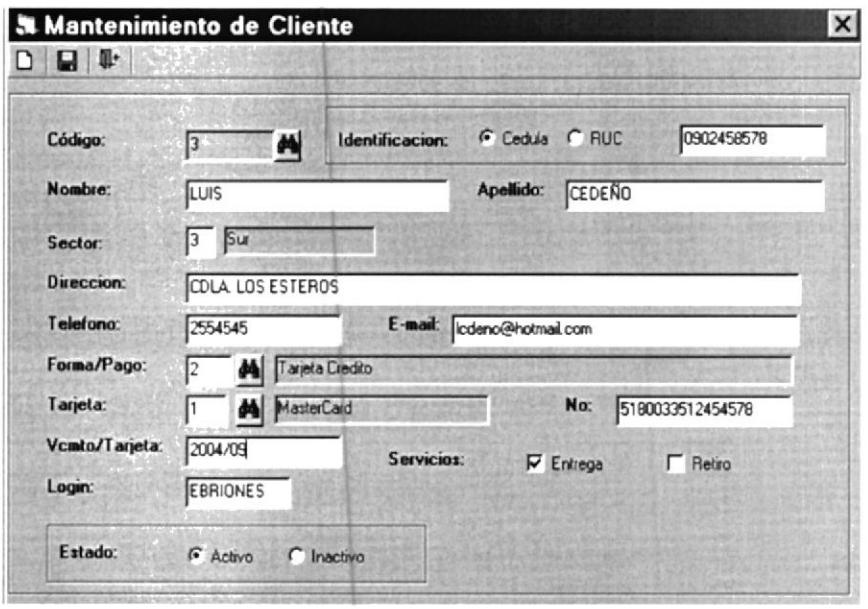


Figura 7-4. Pantalla de Mantenimiento de Clientes

7.2.3.2. Mantenimiento de Títulos.

Los campos que se requieren son: nombre original de la película, nombre de la película en español, sinopsis de la película, productor, director, reparto, censura, tipos de géneros (máximo hasta tres),

identificar si la película ha sido nominada y registrar la referencia de esa nominación, identificar si la película ha sido premiada y registrar la referencia de esa premiación, identificar si ha ganado el Oscar, registrar su tiempo de duración (hh:mm), el año de producción de la película y por último la ruta en donde almacenaremos un archivo de imagen de la película.

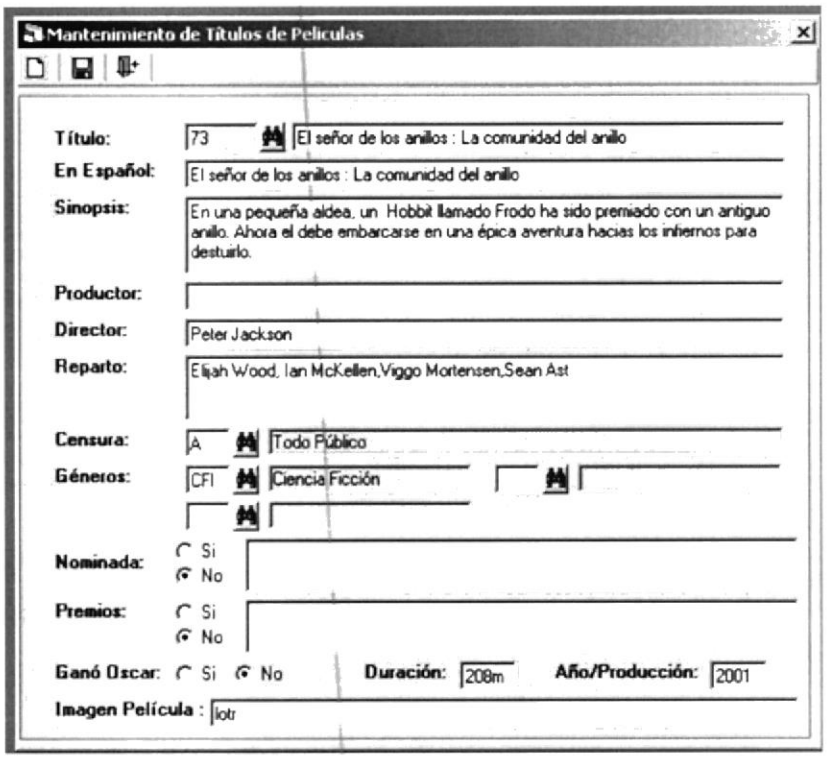


Figura 7-5. Pantalla de Mantenimiento de Títulos

7.2.3.3. Mantenimiento de Ítems de Películas.

Los ítems van a representar los diferentes medios físicos en donde se encuentra grabada la película, y entre los datos que debe contener su

registro tenemos: Un código único que lo dará el sistema al crear cada nuevo ítem, su código título de la película, el código del tipo de medio (VHS ó DVD), además el idioma en que está editada la película.

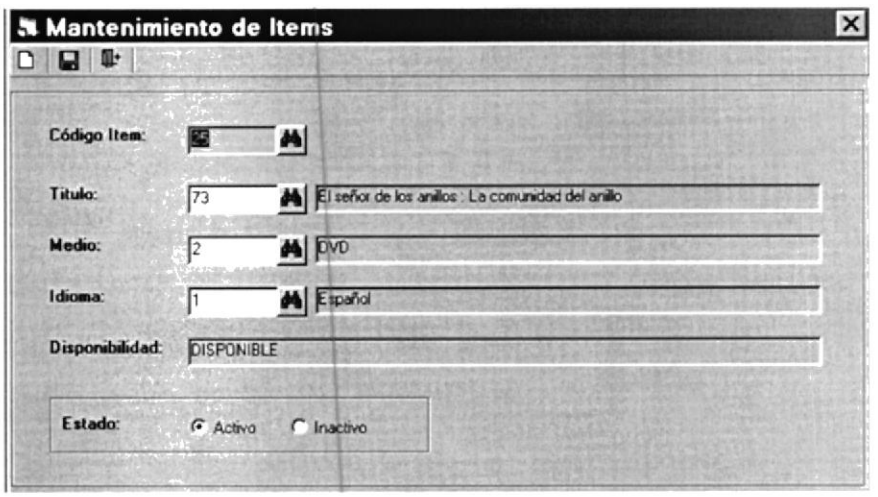


Figura 7-6. Pantalla de Mantenimiento de Ítems

7.2.3.4. Mantenimientos Varios.

En este menú se encuentran las opciones para dar mantenimiento a la información de:

- Tablas básicas, como: sectores de la ciudad, tarjetas de crédito, idiomas, categorías para la publicación del sitio Web.
- Tipos de talonarios
- Detalles de las categorías para publicación en el sitio Web.

- Parámetros de la empresa
- Secuenciales de todas las tablas del Sistema.

7.2.4. Transacciones.

En el menú de transacciones existen las opciones que nos permiten gestionar las operaciones del registro de entrega y recibo de películas, como también realizar la evaluación de las solicitudes ingresadas desde el sitio Web de Ecuacinema.

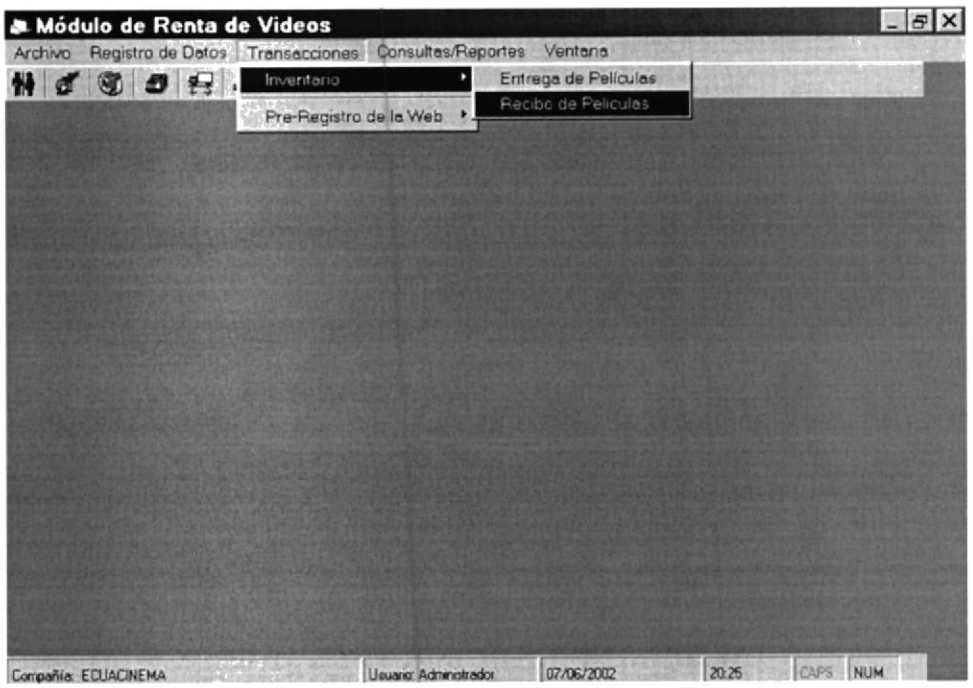


Figura 7-7. Pantalla de menú de Transacciones

7.2.4.1. Entrega de Películas.

En esta pantalla se le permitirá al usuario seleccionar todas aquellas películas que se han alquilado y no han sido entregadas al cliente. Para realizar esto, se le permite seleccionar todas las películas que se encuentren en esta situación, simplemente dándole un clic en el casillero de la columna seleccionar, debe también indicar el lugar en donde fue entregado, sea este el local ó en el domicilio del cliente y para finalizar debe dar otro clic en icono del diskette para grabar su confirmación.

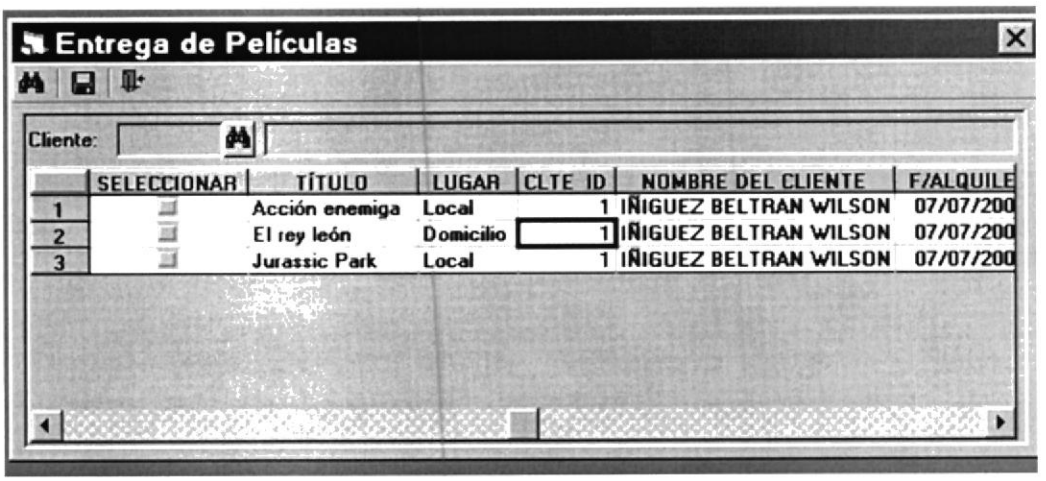


Figura 7-8. Pantalla de Entrega de Películas

7.2.4.2. Recibo de Películas.

En ésta pantalla se le permitirá al usuario seleccionar todas aquellas películas que se han entregado al cliente, pero que no han sido devueltas a Ecuacinema. En esta pantalla se le permite seleccionar

todas las películas que se encuentren en esa situación, simplemente dándole un clic en el casillero de la columna seleccionar y después otro clic en icono del diskette para grabar su confirmación.

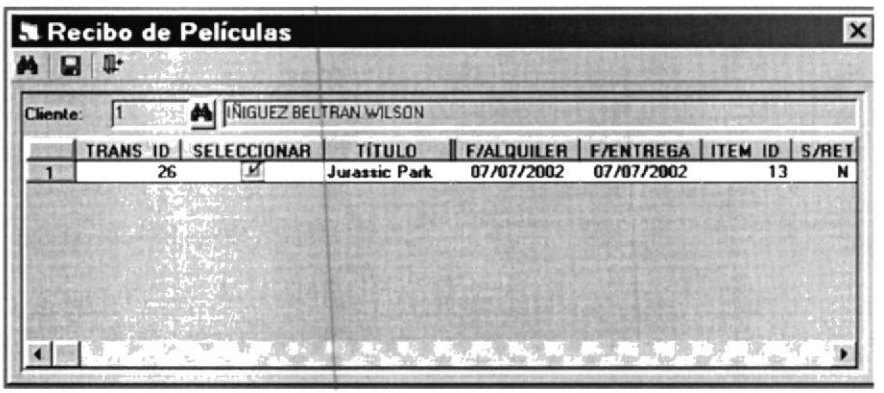


Figura 7-9. Pantalla de Recibo de Películas

7.2.4.3. Evaluación de Solicitudes.

Esta opción nos permite evaluar la información del Pre-registro que se hizo en el sitio Web, y si la información es de una persona idónea se procederá a su aprobación.

7.2.5. Consultas / Reportes.

En este menú la aplicación le ofrece los siguientes reportes de las operaciones que se realizan.

Figura 7-10. Pantalla de Evaluación de Solicitudes de Membresía

7.2.5.1. De Clientes.

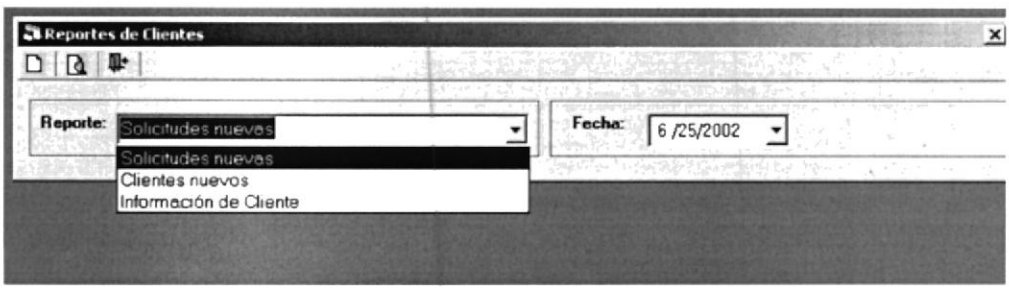
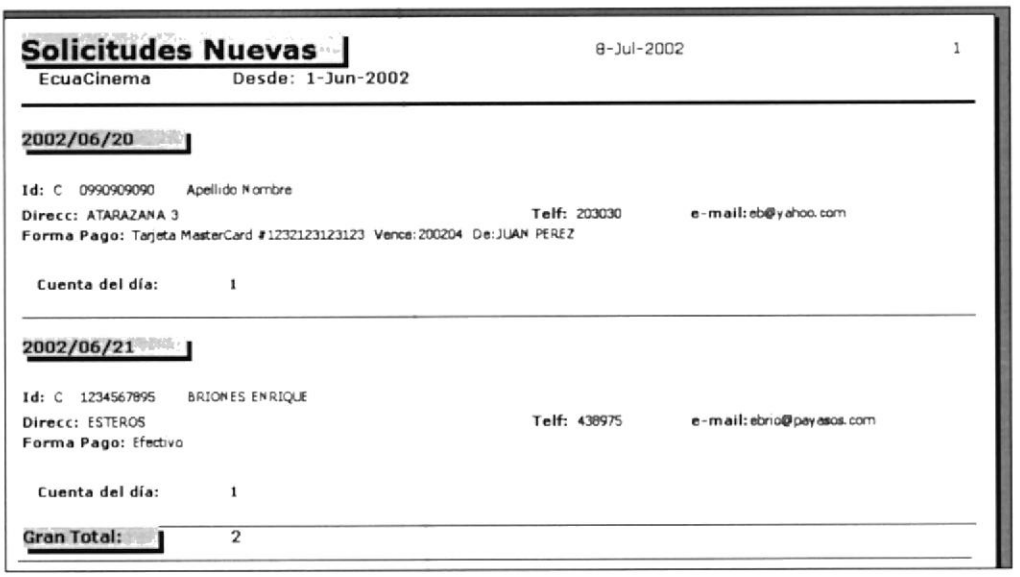


Figura 7-11. Pantalla de Reportes de Clientes

En esta opción se le permite al usuario emitir los reportes de:

- Solicitudes nuevas
- Clientes nuevos
- Información del cliente



Solicitudes Nuevas 8-Jul-2002 1

EcuaCinema Desde: 1-Jun-2002

2002/06/20

Id: C 0990909090 Apellido Nombre

Direcc: ATARAZANA 3 **Telf:** 203030 **e-mail:** eb@yahoo.com

Forma Pago: Tarjeta MasterCard #1232123123123 Vence:200204 De:JUAN PEREZ

Cuenta del día: 1

2002/06/21

Id: C 1234567895 BRIONES ENRIQUE

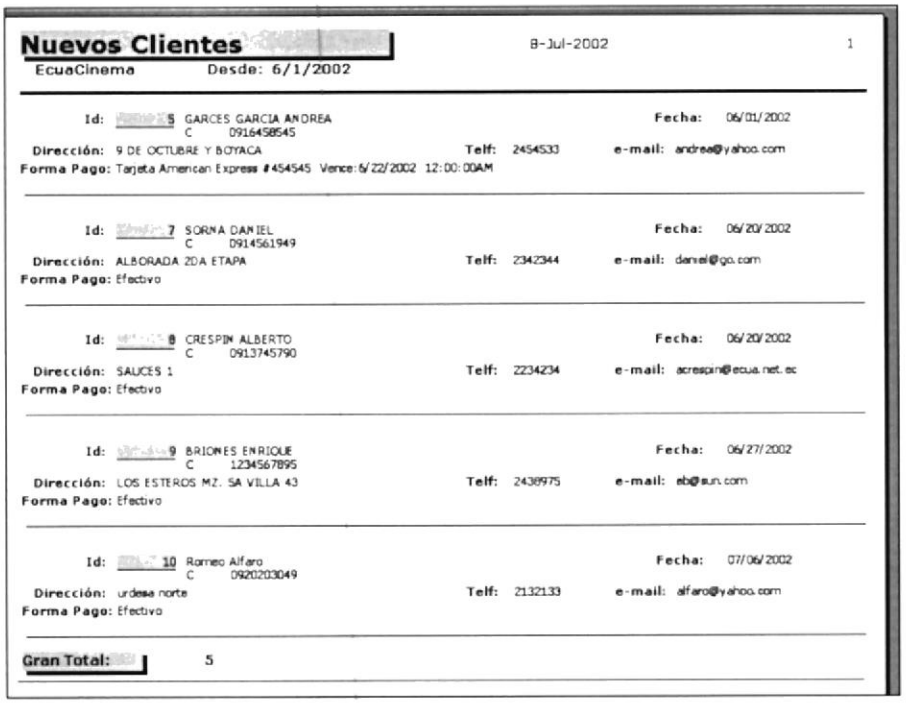
Direcc: ESTEROS **Telf:** 438975 **e-mail:** ebrio@payasos.com

Forma Pago: Efectivo

Cuenta del día: 1

Gran Total: 2

Figura 7-12. Reporte de Solicitudes Nuevas



Nuevos Clientes 8-Jul-2002 1

EcuaCinema Desde: 6/1/2002

Id: 5 GARCES GARCIA ANDREA Fecha: 06/01/2002
C 0916458545
Dirección: 9 DE OCTUBRE Y BOYACA Telf: 2454533 e-mail: andrea@yahoo.com
Forma Pago: Tarjeta American Express #454545 Vence: 6/22/2002 12:00:00AM

Id: 7 SORNA DANIEL Fecha: 06/20/2002
C 0914561949
Dirección: ALBORADA 2DA ETAPA Telf: 2342344 e-mail: daniel@go.com
Forma Pago: Efectivo

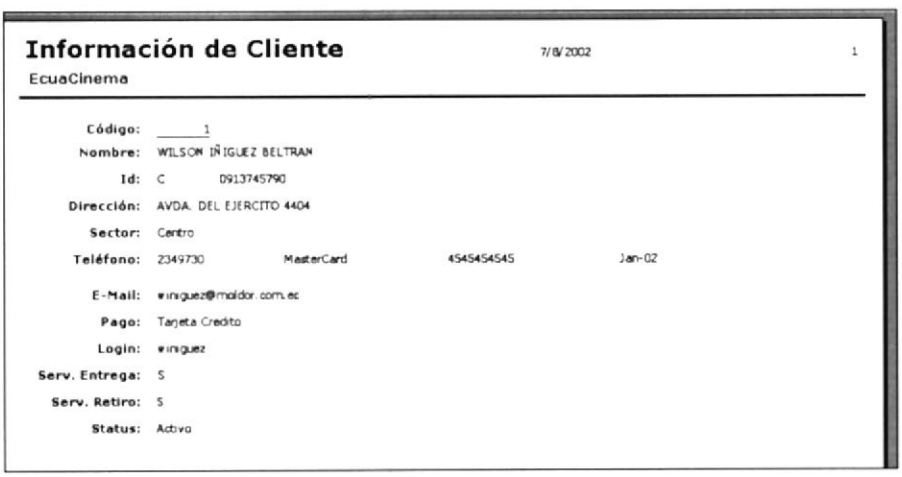
Id: 8 CRESPIN ALBERTO Fecha: 06/20/2002
C 0913745790
Dirección: SAUCES 1 Telf: 2234234 e-mail: acrespin@ecua.net.ec
Forma Pago: Efectivo

Id: 9 BRIONES ENRIQUE Fecha: 06/27/2002
C 1234567895
Dirección: LOS ESTEROS MZ. SA VILLA 43 Telf: 2438975 e-mail: eb@sun.com
Forma Pago: Efectivo

Id: 10 Romeo Alfaro Fecha: 07/06/2002
C 0920203049
Dirección: urdesa norte Telf: 2132133 e-mail: alfaro@yahoo.com
Forma Pago: Efectivo

Gran Total: 5

Figura 7-13. Reporte de Clientes Nuevos



Información de Cliente 7/8/2002 1

EcuaCinema

Código: 1
Nombre: WILSON IÑIGUEZ BELTRAN
Id: C 0913745790
Dirección: AVDA. DEL EJERCITO 4404
Sector: Centro
Teléfono: 2349730 MasterCard 4545454545 Jan-02
E-Mail: winiguez@moldor.com.ec
Pago: Tarjeta Credito
Login: winiguez
Serv. Entrega: S
Serv. Retiro: S
Status: Activo

Figura 7-14. Reporte de Información de Cliente

7.2.5.2. De Transacciones.

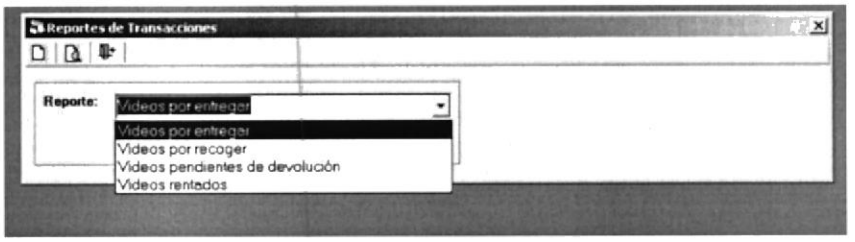


Figura 7-15. Pantalla de Reportes de Transacciones

En esta opción se le permite al usuario emitir los reportes de:

- Videos por entregar
- Videos por recoger
- Videos pendientes de devolución
- Videos rentados

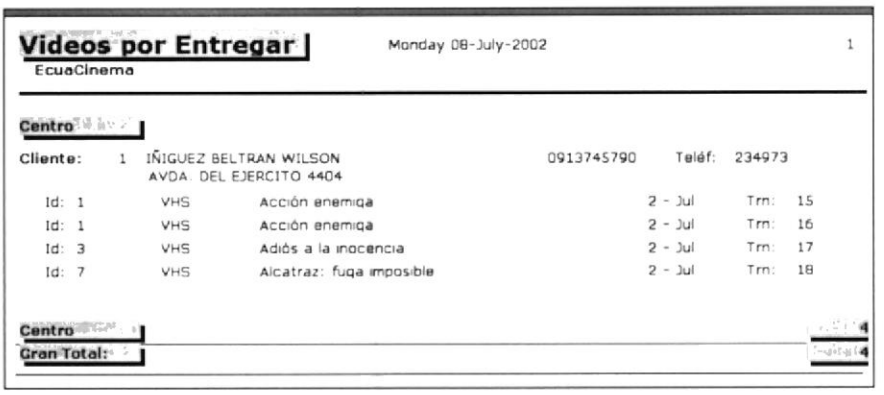


Figura 7-16. Reporte de Videos por Entregar

Videos por Recoger Friday 12-July-2002 1

EcuaCinema

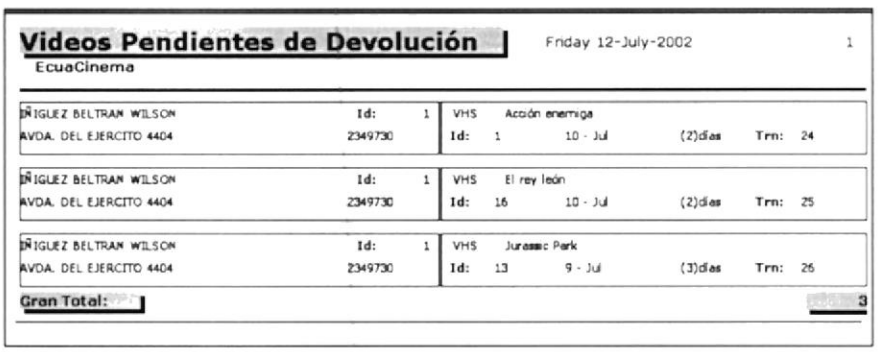
Centro

Cliente: 1 IÑIGUEZ BELTRAN WILSON 0913745790 Teléf: 234973
AVDA. DEL EJERCITO 4404

Id: 1	VHS	Acción enemiga	10 - Jul	Trn: 24
Id: 16	VHS	El rey león	10 - Jul	Trn: 25
Id: 13	VHS	Jurassic Park	9 - Jul	Trn: 26

Centro 3
Gran Total: 3

Figura 7-17. Reporte de Videos por Recoger



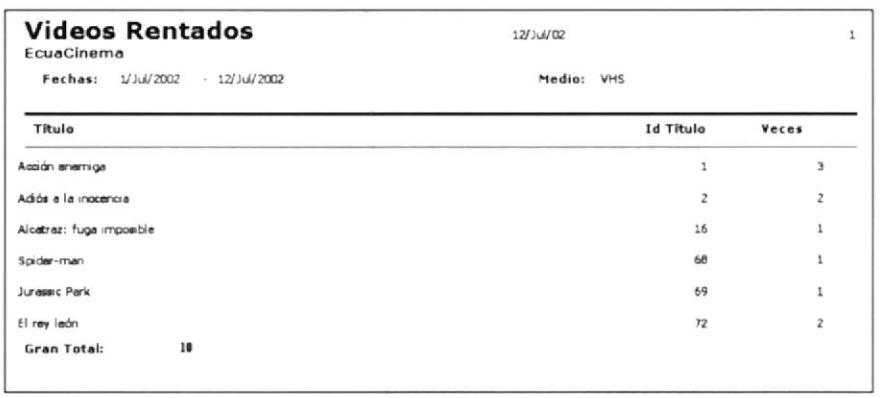
Videos Pendientes de Devolución Friday 12-July-2002 1

EcuaCinema

IÑIGUEZ BELTRAN WILSON AVDA. DEL EJERCITO 4404	Id: 1 2349730	VHS Acción enemiga Id: 1	10 - Jul	(2)días	Trn: 24
IÑIGUEZ BELTRAN WILSON AVDA. DEL EJERCITO 4404	Id: 1 2349730	VHS El rey león Id: 16	10 - Jul	(2)días	Trn: 25
IÑIGUEZ BELTRAN WILSON AVDA. DEL EJERCITO 4404	Id: 1 2349730	VHS Jurassic Park Id: 13	9 - Jul	(3)días	Trn: 26

Gran Total: 3

Figura 7-18. Reporte de Videos Pendientes de Devolución



Videos Rentados 12/Jul/02 1

EcuaCinema

Fechas: 1/Jul/2002 - 12/Jul/2002 **Medio:** VHS

Título	Id Título	Veces
Acción enemiga	1	3
Adiós a la inocencia	2	2
Alcatraz: fuga imposible	16	1
Spider-man	68	1
Jurassic Park	69	1
El rey león	72	2

Gran Total: 10

Figura 7-19. Reporte de Videos Rentados

7.3. Cliente HTML.

Página de Inicio de Ecuacinema.com

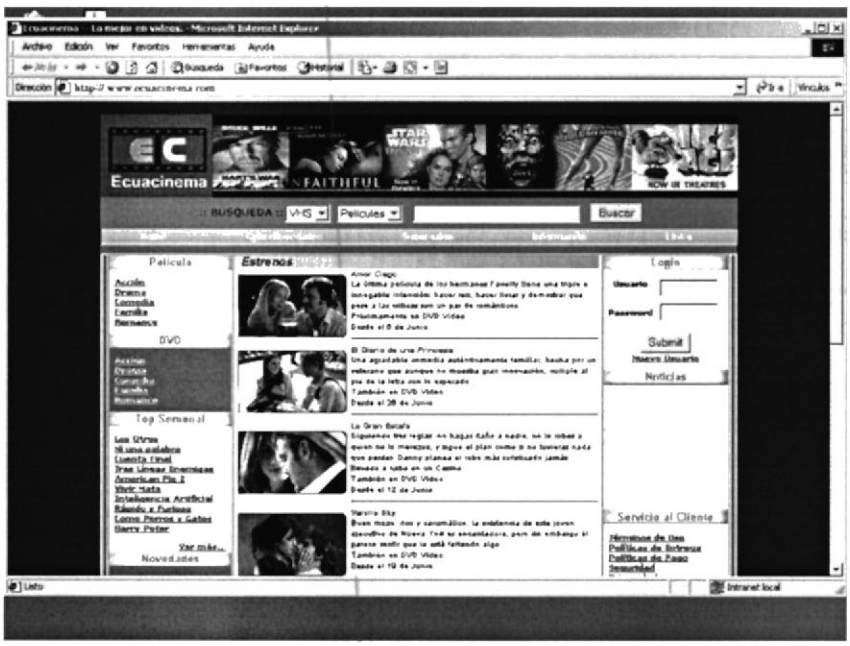


Figura 7-20. Página de inicio de Ecuacinema

A partir del diseño de este sitio - Ecuacinema podemos:

1. Establecer y mejorar la relación entre sus clientes y nuestra empresa.

2. Reforzar y mejorar la comunicación en el mundo empresarial (el futuro real de Internet está en el B2B (business to business o relaciones entre empresas).

3. Competir y ofrecer sus servicios o productos en igualdad de condiciones que una gran multinacional y a nivel mundial

(independientemente de que el ámbito de su empresa sea local o nacional).

4. Establecer, reforzar y mejorar la imagen de la empresa (los beneficios que le pueden reportar un sitio web -información y publicidad de su empresa 24 horas 365 días al año como mínimo- hacen que esté considerada como una de las mejores inversiones que puede hacer una empresa en tecnología e innovación).

5. Tener abierta una "oficina" permanentemente para informar, vender o atender a sus clientes sin tener que estar físicamente en ella, además de estar integrada y coordinada con sus otras oficinas o sede central (ejemplo: las compras se registran automáticamente en la gestión de almacén y contabilidad).

Objetivos del sitio y estrategias de diseño.

Existen diferentes estrategias de diseño en función de los objetivos del sitio Ecuacinema. En un medio tan competitivo, nuestro objetivo ha sido el de comunicar una información, promocionar, enseñar, tanto es asi que nos vamos a encontrar con diferentes exigencias en el desarrollo de el web en sus diferentes aspectos, que se tomaron en cuenta tales como contenido, tono e imagen.

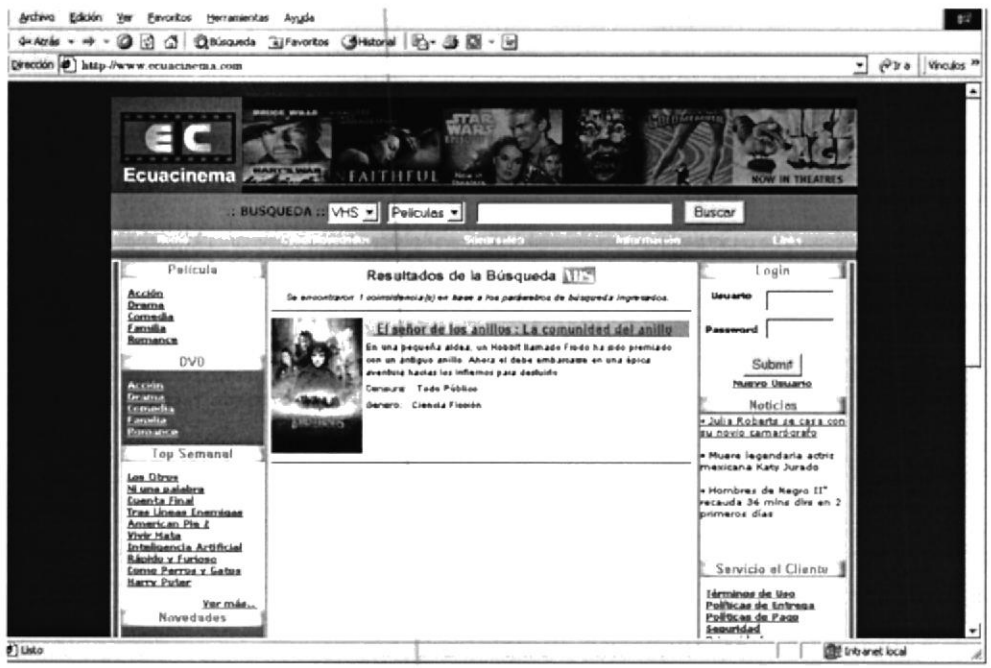


Figura 7-21. Ecuacinema – búsqueda de videos

La consistencia.

El visitante o usuario puede tener a través de este sitio una noción clara de dónde se encuentra, de dónde viene y a dónde va. Para ello proporcionamos pistas visuales que le comunican esta información de una forma sutil, sin distraerlo de los contenidos. La mejor técnica que se tomó en consideración fue el de planear una identidad consistente en las diferentes partes del web, de manera que el usuario conozca que está aún ahí, y reforzar la idea de situación con los elementos de navegación de cada página.

Para conseguir una web con consistencia en su diseño y aspecto profesional, tenemos varios puntos que se deben cuidar. Si tenemos éxito en su tratamiento, el web será un éxito tanto por su funcionalidad como por su atractivo visual. Es así que la consistencia de nuestro sitio se fundamenta en base a:

Paleta de colores. Se seleccionó una paleta de colores limitada, seleccionando cuidadosamente colores armónicos entre sí.

Fuentes tipográficas. Nuevamente, optamos por una paleta de tipos de letra limitada.

Ilustraciones. Cuya presentación imprimen en el sitio un marcado carácter, en función de su estilo y al tipo de empresa que es la de Renta de Video.

Fotografías. A traves de las fotografías logramos manipular y unificar el aspecto del sitio y darle una presentación común: por ejemplo, se usaron fotos e imágenes tintadas en un color corporativo propio de la entidad, empresa o proyecto.

7.4. Cliente Script – DHTML

El uso de scripts del lado del cliente nos ha permitido controlar y reutilizar una serie de validaciones y de interacciones que existen en cada una de las páginas, adicionalmente a esto, el uso de los applets de Java en la barra de tareas da una vistosidad a la aplicación, todo esto unido a las animaciones interactivas de tipo vectorial, extremadamente compactas, hacen posible de que este sitio sea amigable, fácil de usar, vistoso, seguro y confiable.

7.5. Server Script - ASP.

Las páginas ASP nos permiten interactuar con la información almacenada dentro de la base de datos en donde se almacena la información del negocio.

Podemos anotar que para las consultas de información existen páginas cuyo objetivo es el de capturar información y pasarlo a una página que se encargará de procesar la información ingresada, y también existen

páginas dentro de la aplicación que nos permiten consultar de una manera directa sin necesidad de ingresar algún parámetro de búsqueda.

A continuación procedemos a presentar la estructura de la aplicación Web con las carpetas que componen en proyecto y una lista de las páginas ASP usadas en el sistema, con una breve explicación de su uso.

Figura 7-22. Arbol de Directorio

Listado de las páginas ASP que componen el Sistema

Página ASP	Descripción de la página
index.asp	Página principal del Sitio de Ecuacinema.
buyticket.asp	Permite capturar la información necesaria para comprar los talonarios para la renta de videos.
Estrenos.asp	Permite consultar la información referente a los últimos videos que son estrenos.
Alquiler.asp	Página usada para presentar los items seleccionados para alquilar, previo a ser registrados en el sistema.
Logon.asp	Página que permite que un usuario pueda ser

	reconocido por el sistema a través de un usuario y un password.
Noticias.asp	Muestra la información detallada de las noticias que se presentan de manera interactiva en la página principal y en sus páginas anexas.
Outlogon.asp	Cierra la sesión del usuario que ha sido reconocido por el sistema a través de su nombre de usuario y su password.
Preview.asp	Muestra información detallada del video.
Proximos.asp	Permite consultar la información referente a los próximos videos que podrán ser alquilados.
Regalquiler.asp	Página final de registro de alquiler de videos que le indica al usuario si la transacción fue o no realizada con éxito.
Regbuyticket.asp	Página final de registro de talonarios para alquiler de videos que indica al usuario si la transacción fue o no realizada con éxito.
Regcliente.asp	Página final de registro de clientes que le indica al usuario si la transacción fue o no realizada con éxito.
Registro.asp	Captura la información referente a los datos del cliente próximo a ser registrado.
Search.asp	Muestra la información de los videos consultados a partir de los parámetros de búsqueda que son: El tipo de medio (VHS, DVD), además si se trata de una película, director, o reparto, y el argumento.
Secciones.asp	Permite consultar la información referente a los diversas secciones que componen la página principal y las páginas anexas del sistema.
Sysalquiler.asp	Página de uso transparente para el cliente que

	permite referenciar al componente que se encargará de registrar la información de los items seleccionados para el alquiler.
Sysbuyticket	Página de uso transparente para el cliente que permite referenciar al componente que se encargará de registrar la información de los talonarios.
Sysregistro.asp	Página de uso transparente para el cliente que permite referenciar al componente que se encargará de registrar la información del cliente.

7.6. CSS.

Una parte importante en el desarrollo de este sistema corresponde a su orden, contenido y estructura, para que ello ocurra se ha tratado de simplificar la cantidad de código que tenga que ser editado en cada cambio, ya sea éste de formato en el tamaño de las letras, color, fondo y demás, es por esto que el archivo usado toma el nombre de **Styles.css** y el cual nos ha permitido estandarizar todos y cada uno de las características del formato que presenta el sitio Web.

7.7. Comunicación con componentes de negocios.

Nuestro sistema -basado en la arquitectura de tres capas-, posee una capa de negocios que sirve de enlace entre la capa de presentación y la capa de acceso a datos, permitiendo ademas ocultar la compleja interacción entre las reglas de negocios y las interfaces del sistema.

La comunicación con dicha capa fue estructurada en dos segmentos dependiendo del manejo o tarea que llevaban a cabo los componentes, es decir si la llamada a estos implicaba de una forma transparente la consulta, inserción, modificación o eliminación de datos.

Así tenemos que para el caso de nuestra aplicación las consultas realizadas en las páginas ASP, la referencia a dichos componentes se los hacía in situ, es decir si la llamada a dicho componente implicaba la consulta de información, esta referencia se la hacía dentro de dicha página dinámica, y en caso de que se tratara de una inserción o modificación de datos, dicha llamada al componente de negocios se lo hacía a través de una pagina intermedia o de referencia en cuya estructura no se posee nada más que el código que permite crear el objeto componente, referenciarlo y obtener de él la información sobre el estado de la transacción solicitada, para luego mostrar el resultado de dicha operación en una página expresamente creada para la presentación de la respuesta de la transacción.

A continuación presentamos un listado de nuestras páginas dinámicas asociadas con los diversos componentes de negocios que son referenciados por ellas.

Página	Componente de Negocios	Método Publico Invocado
Index.asp	bus_MedioC.Medio	ObtenerTodos()
search.asp	bus_MedioC.Medio	ObtenerTodos()
	bus_TituloC.Titulo	ObtenerSegunCriterio()
alquiler.asp	bus_TituloC.Titulo	ObtenerRegistro()
buyticket.asp	busMedioC.Medio	ObtenerTodos()
	bus_TalonTipoC.TalonTipo	ObtenerTodos()
	bus_tarjetaC.Tarjeta	ObtenerTodos()
estrenos.asp	bus_PubliWebC.PubliWeb	ObtenerPorCategoria()
logon.asp	bus_clienteC.Cliente	ValidarLogin()
		ObtenerInfoCuenta()
noticias.asp	bus_PubliWebC.PubliWeb	ObtenerPorCategoria()
outlogon.asp	Ninguno	
preview.asp	bus_Medio_C.Medio	ObtenerTodos()
	bus_tituloC.titulo	ObtenerRegistro()
proximos.asp	bus_PubliWeb.PubliWeb	ObtenerPorCategoria()
Regalquiler.asp	bus_medioc.Medio	ObtenerTodos()
regbuyticket.asp	bus_medioc.medio	ObtenerTodos()

regcliente.asp	bus_medioc.medio	ObtenerTodos()
registro.asp	bus_medioc.medio	OntenerTodos()
	bus_sectorc.sector	ObtenerTodos()
	bus_tarjetaC.Tarjeta	ObtenerTodos()

7.8. Proceso Servidor Web.

El proceso servidor Web tal como lo hemos venido señalando dentro de este manual, es una parte fundamental dentro de la arquitectura en tres capas del sistema. Es así que el usuario que desee interactuar con el sistema deberá de "levantar" el browser y digitar la dirección url, siendo el Web browser el proceso cliente que envía requerimientos HTTP al proceso servidor que es el Web Server.

La solicitud es por una página estática HTML o por un programa que se encarge de generar dinámicamente el codigo HTML que posteriormente será enviado al browser.

El producto utilizado para el desarrollo y posteriormente para poner en producción al Sistema de Renta de Videos (Ecuacinema) es el Microsoft Internet Information Server versión 5 corriendo como parte integrada del Sistema Operativo Windows 2000 Server.

Instalación

La instalación del IIS 5.0 se la realizó directamente cuando se instaló el sistema operativo Windows 2000. Una vez realizada la instalación del Windows 2000 Server -que es la plataforma en el cual correrán todas las aplicaciones y procesos-, sé procedio a instalar el IIS de la siguiente forma:

Dando un clic en el boton de Inicio, en Programas, Herramientas Administrativas, Configuración del Servidor, lo cual nos abre una ventana que nos permitirá agregar o quitar aplicaciones, encontrándose en el lado izquierdo de dicha ventana con un enlace a la opción Windows Component Wizard. Al dar un clic en dicha opción se muestra el listado de los componentes que están disponibles para su instalación, encontrándose en aquellos el llamado "Servicios de Internet Information Server".

Configuración del Internet Information Server (IIS)

Iniciamos el Administrador de Servicios de Internet. Ubicado tradicionalmente en la siguiente secuencia de opciones:

Inicio - Programas – Herramientas Administrativas, apareciendo la pantalla que aparece a continuación.

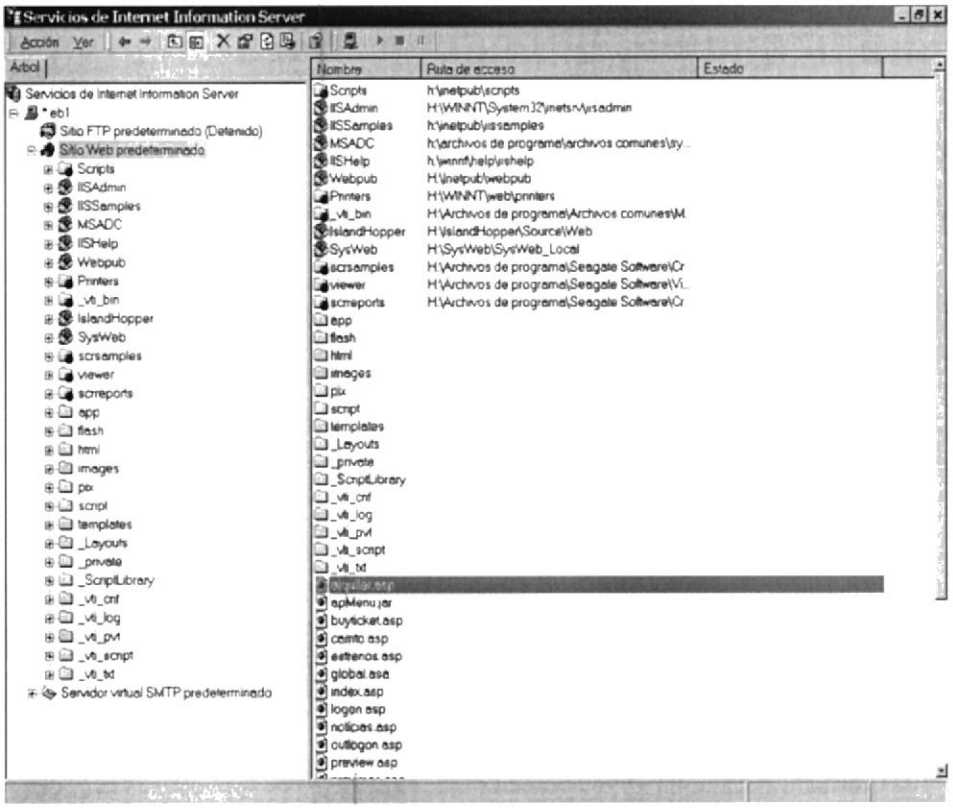


Figura 7-23. Internet Information Server (IIS)

Una vez iniciado el IIS (Internet Information Server), procedemos a crear un nuevo Sitio Web que corresponde a una dirección virtual que permite publicar el sistema en el entorno Web, para lo cual procedemos a seleccionar el elemento Servidor que para nuestro caso se denomina Eb1, dando un clic derecho con el mouse y escogiendo la opción Nuevo Sitio Web, tal como se muestra a continuación:

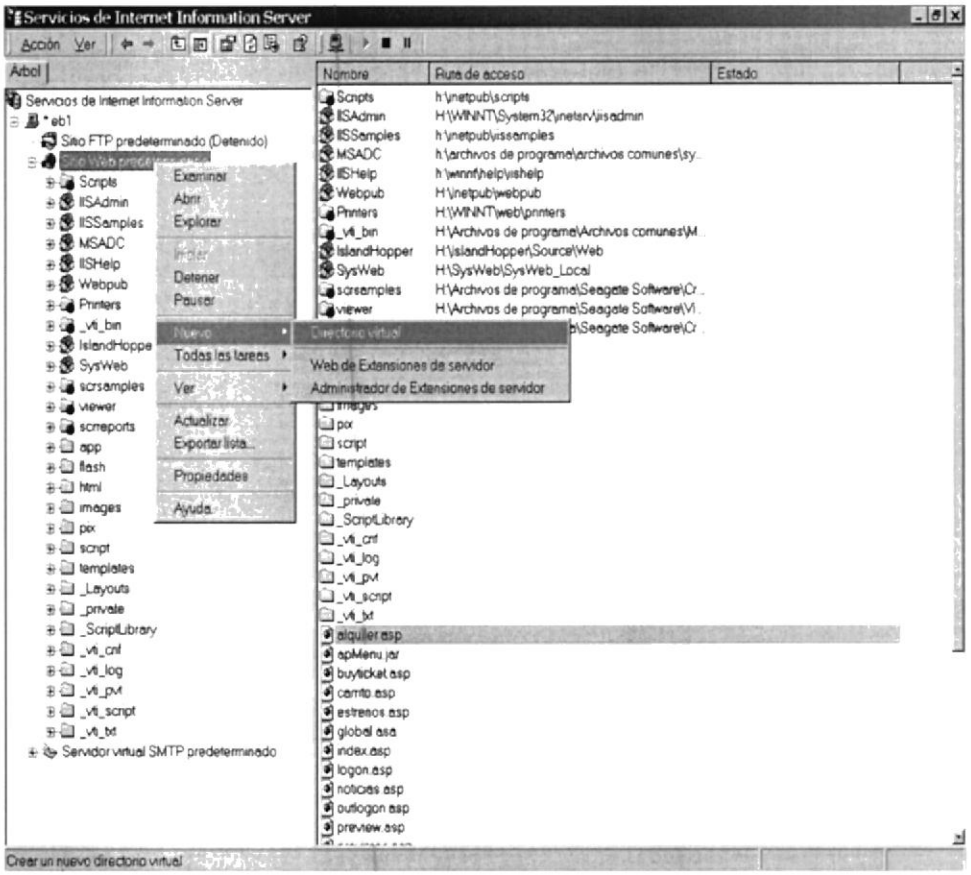


Figura 7-24. IIS – Creación de Sitio Virtual

Realizado el paso anterior se ejecuta el asistente para sitios web, ingresando en el cuadro de texto Descripción del Sitio el nombre que se le va a dar a la aplicación que para nuestro caso es Ecuacinema.

A continuación procedimos a seleccionar la dirección IP en donde nuestra aplicación va a estar alojada.

En el siguiente paso, escribimos la dirección física del sitio c:\inetpub\wwwroot\Ecuacinema.

La siguiente pantalla en aparecer nos permitio asociar dicha dirección virtual con nuestra ruta física del sistema, para lo cual seleccionamos la ruta en la cual se halla almacenado el sistema.

En la pantalla de permisos seleccionamos los permisos de lectura, escritura y de ejecución de los comandos de script para las paginas dinamicas.

En este momento habremos terminado de configurar nuestro sitio web, la carpeta SYSWEB, que hemos configurado como directorio virtual contiene páginas HTML y ASP que están almacenadas en carpetas individuales.

Ahora bien, para que nuestro sitio Web como carpeta de desarrollo esté en ejecución, lo que hacemos es seleccionar el directorio virtual y con el botón derecho del mouse se selecciona y se da clic en la opción inicio o start, tal como se muestra a continuación. Cabe mencionar que si otro sitio Web estuviera ya iniciado debe de ser detenido antes de ejecutar esta opción.

De esta forma habremos definido nuestro sitio web dentro de IIS seleccionando el sitio que hemos instalado, haciendo clic con el botón

derecho con el mouse, y escojiendo la opcion de Directorio Virtual, tal como se muestra a continuación.

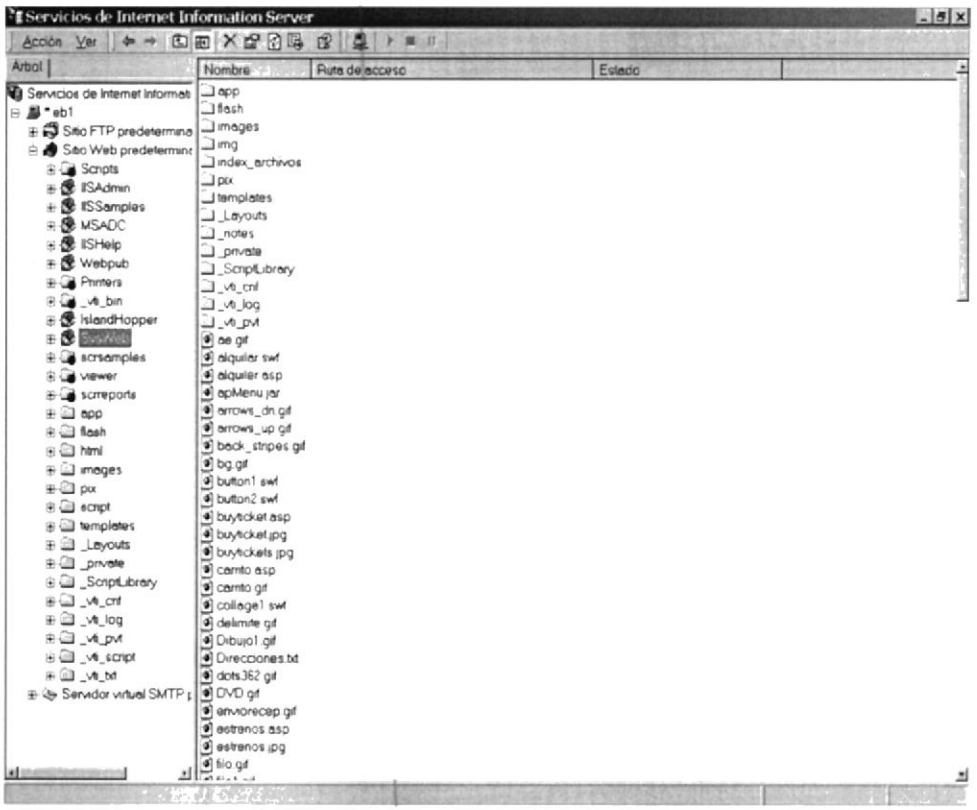


Figura 7-25. IIS – Directorio Virtual SysWeb

Una vez creado el directorio virtual de la aplicación es importante aprovechar las ventajas que da el IIS para la ejecución de los procesos transaccionales agrupados o aislados en segmentos de memoria lo cual permite usar todas las ventajas que el modelo de tres capas provee a todo entorno de desarrollo a más de proteger al Sistema de manera

independiente de toda transacción que resultara errónea y que pueda hacer colapsar al sistema.

Para poder hacer esto damos un clic con el botón derecho del ratón y seleccionamos la opción de propiedades y en la pestaña denominada ***directorio virtual*** buscamos la opción protección de la aplicación y seleccionamos el *nivel alto(asilado) de protección,* de la forma como podemos ver en la siguiente figura.

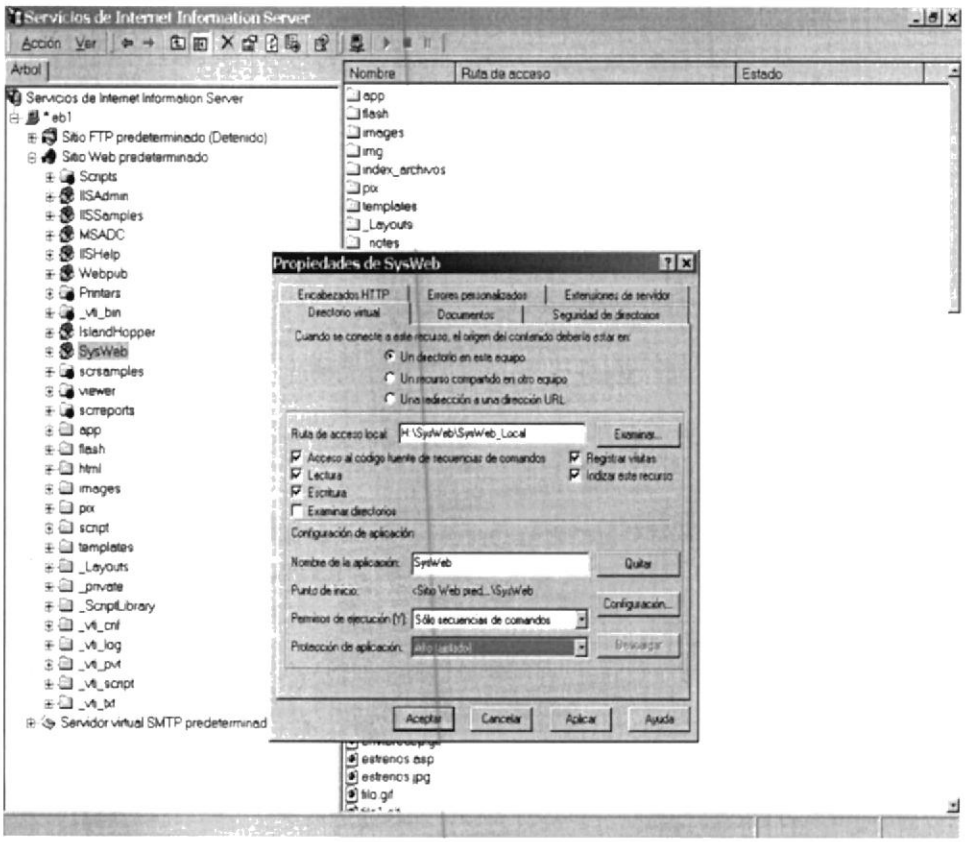


Figura 7-26. IIS – Selección del nivel de protección

Para poder configurar que las extensiones que el servidor Web podrá ejecutar, vamos a la opción de propiedades y en el tab de Extensiones del servidor marcamos la opción de "Habilitar Autorizaciones" para que se puedan ejecutar scripts del lado del servidor, que para nuestro caso corresponderia a la ejecución de JavaScript. Esto además habilita la opción de actualizar archivos al servidor a partir del InterDev.

Otra opción de configuración que nos ayudó en el desarrollo de la aplicación fue la de depuración de las páginas dinámicas, permitiendo depurar la aplicación y enviando la información de error tanto al servidor como al cliente, lo que nos permite tener un control sobre la ejecución del programa. Esta opción la podemos configurar a traves de la ventana de propiedades en la pestaña denominada directorio virtual y dando clic en el botón de configuración de la aplicación, abriéndose una pantalla adicional en la cual buscaremos la pestaña de depuración de la aplicación y marcando las opciones llamadas: Habilitar depuración de secuencia de comandos ASP en el servidor y Habilitar depuración de secuencia de comandos ASP en el cliente.

Todas estas opciones de configuración que hemos visto nos permitirán tener un mayor control de la aplicación tanto a nivel de desarrollo como a nivel de producción, permitiendo aprovechar las características y entorno con el que fue programado el Sistema (modelo de 3 capas).

CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

El desarrollo de una aplicación de 3 capas es un proyecto complejo que requiere profesionales que dominen muchas herramientas. Además requiere una disciplina y organización superior a la que se necesita para el desarrollo de aplicaciones convencionales.

Sin embargo las aplicaciones que resultan tienen un futuro mucho más largo y prometedor. Es mucho más agradable el mantener una aplicación de 3 capas, debido al hecho de que las piezas de código a modificar suelen ser de pocas líneas.

Asignar al profesional adecuado a la tarea es otro factor importantísimo para el avance acelerado del proyecto.

RECOMENDACIONES

Como toda aplicación, las fases de análisis y diseño son claves para el éxito de cualquier proyecto de desarrollo de software. En el caso de las aplicaciones de 3 capas es importantísimo tener un diseño de base de datos que no sufra muchas alteraciones durante la fase de desarrollo ya que causará muchos retrasos al proyecto debido a que se tendrán que volver a generar interfaces, componentes de datos, de negocios y programas clientes.

Mantener una red de computadoras para el desarrollo del proyecto, con software estandarizado y con las últimas actualizaciones de las correcciones que se encuentran en la Internet es importante para minimizar el tiempo perdido.

Los componentes de negocios y datos deben ser probados independientemente antes de ser puestos en producción. Esto le evita molestias y retrasos a los desarrolladores de la capa de presentación.

BIBLIOGRAFÍA

1. HOMER, SUSSMAN, FRANCIS, *Professional Active Server Pages 3.0*, WROX, 1999.

2. PATTISON, *Programming Distributed Applications with COM+ and Microsoft Visual Basic 6.0 Second Edition*, Microsoft, 2000.

3. SOUKUP, DELANEY, *A fondo Microsoft SQL Server 7.0*, Microsoft Press, 1999.

4. KIRTLAND, *Designing Component-Based Applications*, Microsoft Press, 1998.

5. EVANS, MILLER, SPENCER, *Programming MS Visual Interdev 6.0*, Microsoft Press, 1999.

6. SCEPPA, Programming ADO, Microsoft, 2000.

7. ORÓS, Diseño de Páginas Web interactivas con JavaScript 2da. edición, Alfaomega, 2000

8. Sitio de Internet www.kbalertz.com (knowledge base de COM+).

9. Sitio de Internet de Blockbuster Méjico (www.blockbuster.com.mx).

10. Sitio de Internet de cine (www.usimdb.com).

11. Sitio de Internet de enciclopedia de términos computacionales (www.webopedia.com).

12. Sitio de Internet de Microsoft Developer Network (www.msdn.microsoft.com).

A

ARQUITECTURA CLIENTE / SERVIDOR

A. ARQUITECTURA CLIENTE / SERVIDOR

A.1. Arquitectura Host.

Los primeros modelos computacionales se caracterizaban por el uso de sistemas centrales (hosts) al cual se conectaban los terminales "tontos". En este modelo el sistema central era aprovechado al máximo y en cambio los equipos clientes se usaban únicamente como receptores y visualizadores de información. Los problemas de rendimiento en las aplicaciones sólo se podían solucionar realizando actualizaciones al hardware host, las cuales eran muy caras.

Las aplicaciones eran programas enormes que contenían toda la funcionalidad de la aplicación.

A.2. Arquitectura File-Server.

Posteriormente surgió un modelo con aplicaciones limitadas, el modelo File Server. Este se caracterizaba por constar de un equipo central que ofrecía su hardware a un conjunto de equipos clientes. El hardware ofrecido era principalmente de almacenamiento, tales como discos duros. Aquí, los equipos clientes realizaban todo el procesamiento computacional, dejando únicamente al equipo central las tareas de entrada y salida a los dispositivos de almacenamiento. En este modelo, el poder computacional del equipo central es desperdiciado.

A.3. Arquitectura Cliente / Servidor.

En una aplicación de negocios tradicional se distinguen claramente tres tipos de lógica o programación:

a) lógica de presentación
b) lógica de negocios
c) lógica de acceso a datos

La lógica de presentación se encarga de la interacción con el usuario: aceptar entrada, validarla y enviar respuestas al usuario.

La lógica de negocios gobierna las reglas del negocio o políticas comerciales u operativas.

La lógica de acceso a datos se encarga de "conversar" con el DBMS para realizar la apertura y cierre de conexiones, y para ordenar al DBMS la ejecución de instrucciones de consulta y actualización de los datos.

En los modelos iniciales, los tres tipos de lógicas estaban contenidas en un solo programa. Esto facilitaba la distribución pero dificulta en gran medida el mantenimiento de las aplicaciones. Tenía que ser revisado y compilado todo el programa y por sólo una persona. Es decir, se desaprovechan los grupos de trabajo, así como también, las modificaciones a los programas requieren más tiempo.

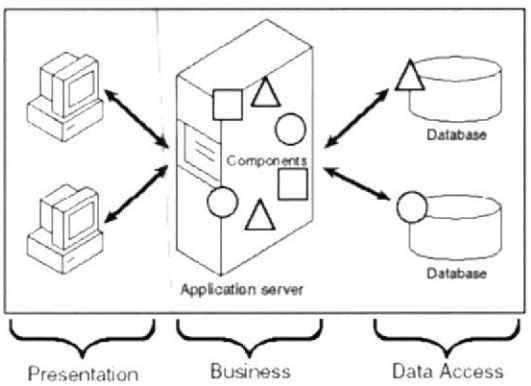


Figura A-1. Arquitectura Cliente/Servidor

La arquitectura cliente / servidor soluciona este problema, ya que divide estos diferentes tipos de procesamiento en procesos especializados e independientes. Constan de procesos servidores especializados que proveen diferentes servicios a uno o más procesos clientes. Estos procesos pueden correr en diferentes plataformas de hardware y software en una red LAN o WAN. Para comunicarse entre procesos clientes y servidores también necesitan de protocolos de comunicación efectivos.

A.4. Tipos de Procesos.

A.4.1. Proceso Servidor.

Es aquel proceso que brinda uno o más servicios especializados. Las características principales que deben tener es que siempre deben atender a los procesos clientes y además en el menor tiempo posible.

Algunos ejemplos de procesos servidores son: File Server, Database Server, Transaction Processing Server, Groupware Server, Web Server, Object Server, etc.

File Server: Ofrecen servicios para acceder a un sistema de archivos de una manera confiable, segura y rápida.

Database Server: Ofrecen servicios para acceso a bases de datos. Generalmente se apoyan en el lenguaje SQL. Garantizan la integridad, seguridad y acceso a los datos.

Object Server: Ofrecen el servicio de instanciar los objetos y de controlar sus contextos y seguridades.

Web Server: Ofrece el servicio de publicación de páginas HTML utilizando el protocolo HTTP para su transmisión.

A.4.2. Proceso Cliente.

Es aquel que interactúa con el usuario y que obtiene servicios de los procesos servidores. Se pueden clasificar en dependientes e independientes.

Clientes Independientes

Se ejecutan nativamente dentro del ambiente ofrecido por el sistema operativo. Están ligados a él. Ejemplo: un programa Win32 cualquiera.

Clientes Dependientes

Se ejecutan únicamente dentro de un ambiente o programa contenedor. Ejemplo, los programas que se ejecutan desde un browser de Internet (una aplicación Web).

Al ejecutarse dentro de un contenedor, son portables a cualquier ambiente que soporte el programa contenedor.

A.4.3. Middleware

Involucra a todos los procesos que posibilitan la comunicación entre los procesos clientes y servidores.

Se identifican generalmente como protocolos de comunicación, interfaces de acceso a bases de datos, etc.

Por su generalidad se pueden clasificar en genéricos y específicos.

Middleware genérico

Lo conforman componentes de software de comunicaciones y extensiones del sistema operativo.

Ejemplos, TCP/IP, NetBios, RPC, Sockets

Middleware específico

Son componentes de software que están íntimamente relacionados con el proceso servidor con el cual operan.

Algunos ejemplos son: en la Web (HTTP, SSL), para correo (MAPI, VIM, POP, SMTP), de objetos (CORBA, COM), de servicios de bases de datos (DAO, RDO, ADO, OLEDB, ODBC), etc.

A.5. Implementaciones de la arquitectura Cliente / Servidor

La arquitectura cliente / servidor tiene diferentes implementaciones, que son:

1) modelo de 2 capas
    a. Fat client (cliente "gordo")
    b. Thin client (cliente "delgado")
2) modelo de 3 capas
3) modelo de N capas

A.5.1. Modelos de 2 capas

Fat Client

En la primera "capa" se encuentra el proceso cliente, que contiene la lógica de: Presentación, Negocio y Acceso a Datos. En la segunda capa se encuentra el manejador de la base de datos (DBMS).

En este modelo, el proceso cliente usualmente se conecta a la base de datos desde el inicio hasta el fin de la aplicación.

Se recomienda este modelo para aplicaciones con pocos usuarios clientes y que no requieren cambios frecuentes.

Ventajas

- Requiere menos esfuerzo para el desarrollo inicial.
- Requiere menor cantidad de desarrolladores.
- Trabaja bien en ambientes controlados de hasta 100 usuarios.

Desventajas

- Se crean programas clientes monolíticos. Programas grandes a los cuales es complicado realizar mantenimiento.
- Cada usuario nuevo equivale a por lo menos una conexión a la base de datos. Esto complica la escalabilidad de la aplicación.
- Al radicar todos los programas en el cliente, se compromete la confidencialidad de las reglas del negocio.
- Genera una interdependencia entre programa cliente y tipo de base de datos.

Thin Client

Es una variación al modelo Fat Client. Se lo llama también 2 1/2 – tier o modelo Fat Server.

La diferencia principal está en el empleo de stored procedures en la base de datos. Estos son piezas de código en lenguaje propio del DBMS que permiten implementar la lógica de acceso a datos y de negocios dentro de la misma base de datos. Se ejecutan en el servidor DBMS y no en el proceso cliente. El proceso cliente recibe únicamente la respuesta o resultado de ejecución de los stored procedures.

Ventajas

- Provee un mejor rendimiento, los stored procedures una vez "compilados" son ejecutados mucho más rápido que cualquier proceso cliente convencional.
- Es más escalable que el modelo Fat Client.
- Las peticiones (tramas) a la base de datos son más cortas.
- Se consigue una mejor confidencialidad de las reglas del negocio, ya que residen en el servidor DBMS y no en el cliente.
- El proceso cliente no se preocupa de la estructura de los datos. Permite realizar cambios en la lógica de los stored procedures que no requieren re-compilación de los programas clientes.
- Se comparte la lógica de procesamiento entre el cliente y el servidor.

Desventajas

- El cliente se conecta directamente a la base de datos, por lo que se pierde flexibilidad y se utilizan demasiadas conexiones.
- Si los stored procedures son muy complejos, no liberan las conexiones rápidamente.
- Se crea dependencia al DBMS y su implementación particular, por lo que se complica la portabilidad de la aplicación.
- Requiere habilidades de programación tanto en el cliente como en el DBMS.
- El lenguaje de los stored procedures no es tan desarrollado como el que está disponible para las aplicaciones clientes.

A.5.2. Modelo de 3 capas

Se divide las aplicaciones en tres capas, que son:

1. Capa de presentación, encargada de interactuar con el usuario y de realizar validaciones de entrada.

2. Capa de negocios y acceso a datos, que contienen componentes que llevan a cabo las reglas del negocio, así como también la comunicación con el DBMS.

3. Capa de datos, que es llevada a cabo por el manejador de la base de datos o DBMS.

Ventajas

- Al tener centralizado en un sitio las reglas del negocio y de acceso a datos, se facilita en gran medida el mantenimiento de la aplicación.
- Facilita el reúso de los componentes centralizados.
- La capa de negocio y de acceso a datos puede estar ubicada en uno o más servidores, lo que permite un mejor escalado de la aplicación.
- Requiere menos conexiones concurrentes a la base de datos.
- Permite definir un control de seguridad a nivel de proceso o método y aplicado a roles de usuarios.

Desventajas

- Requiere de un equipo disciplinado de desarrolladores
- Requiere habilidades superiores a los desarrolladores
- Necesita de tiempo adicional en la fase de análisis y diseño inicial.

A.5.3. Modelo de N capas

Es una generalización del modelo de 3 capas. El concepto es que los componentes de la capa 2 se subdividen en capas especializadas en: interface de usuario, reglas de negocios y acceso a datos.

Este modelo es usado unicamente por las grandes aplicaciones corporativas.

B

PROGRAMACIÓN EN LA WEB

B. PROGRAMACIÓN EN LA WEB

B.1. HTML.

En sus inicios, la información que viajaba por Internet estaba constituída únicamente por páginas que estaban descritas en un lenguaje conocido como HTML.

HTML significa HyperText Markup Language y es un lenguaje para la creación de documentos en Internet. Define la estructura y el formato de un documento Web usando una amplia gama de "tags" o etiquetas y atributos. La mayoría de los tags requieren de un tag de cierre, que es el mismo tag que se le antepone una barra inclinada (slash).

Algunos ejemplos de tags comunes son:

<HTML> ... </HTML>	Delimitan el inicio y fin del documento HTML
<HEAD> ... </HEAD>	Define la cabecera o título del documento
<BODY> ... </BODY>	Delimita la información principal del documento

Otros tags sirven para dar formato al documento, como:

<P> ... </P>	Usado para crear párrafos
<I> ... </I>	Usado para indicar un tipo de letra itálica

Otros tags definen vínculos de hypertexto. Estos permiten navegar a otra páginas Web en el mismo sitio o en otro dominio de Internet.

Los documentos HTML residen en servidores WEB que utilizan el protocolo HTTP para enviar estos documentos a los browsers que los solicitan.

B.2. Protocolo HTTP.

El protocolo de comunicación entre el browser y el servidor web se denomina HTTP. Es una abreviatura de HyperText Transfer Protocol.

Se utiliza para acceder a recursos ubicados en servidores web que operan en una red TCP/IP.

Se caracteriza por ser un protocolo sin estado (stateless). Esto quiere decir que cada requerimiento del browser es independiente del siguiente. Ni el browser ni el servidor web "recuerdan" algo de la petición anterior.

En cada requerimiento, el browser establece una conexión con el servidor web, éste procesa el requerimiento y le envíe la página solicitada y cierra la conexión.

El documento HTML puede contener referencias a recursos adicionales como imágenes, sonidos, videos, es decir, objetos de diferentes tipo. Cada recurso es procesado, uno por uno, hasta que se envían todos al programa browser.

B.2.1. HTTP Request.

La solicitud HTTP se compone de típicamente de dos partes: Header y Data. Estas secciones están separados por una línea en blanco.

B.2.1.1. HTTP Header.

La cabecera de toda solicitud HTTP se compone de:

1. Request Header Line
2. Request Header Fields

Request Header Line

Está compuesto de 3 campos de texto separados por espacios en blanco. Tiene el siguiente formato:

```
METODO    RECURSO    VERSION
```

El primer campo especifica el tipo de requerimiento (método o comando) a ser aplicado a un recurso del lado del servidor.

El segundo campo especifica el recurso requerido por el cliente y su ubicación.

El tercer campo identifica la versión del protocolo HTTP usada por el cliente.

Ejemplo:

```
GET /libro/indice.html HTTP/1.0
```

El método más usado es el GET que recupera la página indicada en el nombre del recurso. Otros métodos son: HEAD, PUT, DELETE, POST.

Request Header Fields

Estos campos los utiliza el browser para enviar información adicional acerca del requerimiento y del cliente mismo.

Entre otras cosas, estos campos los utiliza para indicarle al servidor los formatos de archivos que soporta, el nombre y versión del programa, etc.

Cada campo consiste de un nombre seguido por el signo de dos puntos (:) y el valor del campo. Ejemplo:

```
Accept: text/plain
Accept: text/html
Accept: audio/x
User-agent: Netscape 6.2
```

B.2.1.2. HTTP Data.

Contiene sólo una sección denominada "Entity Body".

Entity Body

Se utiliza esta sección para enviar al servidor Web el contenido de las variables que se llenan en los formularios HTML cuando se utiliza el método POST.

B.2.2. HTTP Response.

La respuesta del servidor Web al browser se denomina HTTP Response. Se compone de dos partes:

1.Response Header

2. Response Data

Existe una línea en blanco que divide las partes.

B.2.2.1. Response Header.

La primera parte de la respuesta del servidor web se compone de dos elementos:

1. Response Header Line
2. Response Header Fields

Response Header Line.

Envía al browser información de la versión HTTP, el 'status' de la respuesta y una explicación del 'status' retornado.

Formato:

```
<HTTP Version> <result code> [<explanation>]<crlf>
```

Ejemplo:

```
HTTP/1.1 200 OK
```

Response Header Fields.

Retorna información que describe al servidor y al formato de la respuesta. De esta manera el browser puede discernir cómo debe procesar el documento. Ejemplo:

```
Server: NCSA/1.2
Content_type: text/html
Content_length: 5500
```

B.2.2.2. Response Data.

Esta sección contiene únicamente el "Entity Body".

Entity Body

Aquí se envía el documento HTML que el browser ha solicitado. También se envía aquí los diferentes recursos que indica la página (imágenes, sonidos, videos, etc.).

B.2.3. URL.

Para referenciar un documento dentro de un servidor Web nos valemos de su URL. Significa Uniform Resource Locator.

La sintaxis del URL para el caso en que se utilice como protocolo HTTP es:

```
http://<host>:<port>/<path>?<searchpart>
```

El método predeterminado es GET, por lo tanto no es necesario anteponerlo al nombre del URL.

La porción ?<searchpart> de un HTTP URL es opcional. Cuando está presente, indica un tipo de consulta especial que será ejecutada cuando el recurso sea accesado.

B.3. Procesamiento de Formularios Web.

Existen 2 formas usadas para pasar los datos ingresados en un formulario al servidor web.

1. Utilizando variables de ambiente, cuando se usa el método GET.
2. Utilizando la entrada estándar (stdin), cuando se usa el método POST.

El servidor Web genera una cadena de caracteres donde están todas las variables del formulario con sus respectivos valores ingresados por el usuario.

El formato que tiene esta cadena es:

```
variable1=valor1&variable2=valor2&...
```

Ciertos caracteres especiales no son permitidos en URLs, por lo que son reemplazados por el caracter '%' seguido por su valor hexadecimal en ASCII. Por ejemplo el caracter de espacio en blanco es %20.

Ejemplo:

```
nombre=Carlos%20Coronel&edad=34
```

B.3.1. Proceso del Request con el método GET.

1. El browser genera la cadena de variable-valor.
2. El browser coloca la cadena de variables luego del URL y de un signo de interrogación (?). La información viaja en el Request Header.
3. El servidor web extrae la trama y la ubica en una variable de ambiente llamada 'query_string'
4. El script de servidor accesa la variable Query_String y genera los resultados en la salida estándar (stdout).

El problema principal de esta implementación es que la trama puede ser truncada por el servidor web si se excede el tamaño máximo de las variables de ambiente definido en el servidor.

B.3.2. Proceso del Request con el método POST.

1. El browser genera la cadena de variable-valor.

2. El browser coloca la cadena en el cuerpo (entity body) del HTTP Request.
3. El servidor web procesa la trama y ubica las variables y sus contenidos en la entrada estándar (stdin).
4. El script procesa las variables.

La ventaja de usar este método es que no hay restricción en cuanto el tamaño de la información enviada (cadena variable-valor).

B.4. Creación de Formularios Web.

Los formularios web permiten al usuario ingresar información para que posteriormente sea enviada al servidor web para su procesamiento por parte de un script. El script recupera los valores de los campos del formulario y realiza las operaciones para la cual fue diseñado y envía los resultados de vuelta al browser.

B.4.1. Definición de formulario.

Se utiliza la etiqueta <FORM>. Se pueden tener tantos formularios como se quieran en una misma página. No se permiten formularios anidados.

La etiqueta <FORM> tiene dos atributos principales: el atributo METHOD, con el cual se indica el método de envío del formulario; y el atributo ACTION que especifica el script que procesará el formulario.

Ejemplo:

```
<FORM METHOD="POST"
ACTION="http://www.ecuacinema.com/procesar.asp>
      .
      .
</FORM>
```

B.4.2. Elementos de un Formulario.

Existen etiquetas para definir los diferentes tipos de controles de formularios. A continuación se describen los principales controles disponibles.

Elemento Text

Permite al usuario ingresar una simple línea de texto. Ejemplo:

```
<INPUT type="text" name="nombre"">
```

Elemento Password

Igual que el elemento "text", con la diferencia de que todos los caracteres ingresados aparecen en pantalla como "*". Ejemplo:

```
<INPUT type="password" name="pwd">
```

Elemento Radio Button

Muestra una lista de elementos, de los cuales sólo uno puede ser escogido. Se identifica a un grupo de "radio buttons" porque tienen el mismo valor de "name". Con el atributo CHECKED se especifica cuál opción es la predeterminada. Ejemplo:

```
<INPUT type="radio" name="colr" value=r checked>Rojo
<INPUT type="radio" name="colr" value=a> Azul
<INPUT type="radio" name="colr" value=v> Verde
```

Elemento Checkbox

Permite escoger múltiples elementos de una lista. También podemos usar el atributo CHECKED. Ejemplo:

```
<INPUT type="checkbox" name="perro" checked>Perro
<INPUT type="checkbox" name="gato">Gato
<INPUT type="checkbox" name="loro">Loro
```

Elemento Submit Button

Sirve para indicar que se deben enviar los datos del formulario al servidor web mediante un botón "submit". Ejemplo:

```
<INPUT type="submit" value="Enviar datos">
```

Elemento Reset Button

Con este botón, todos los elementos del formulario retoman sus valores "originales". Ejemplo:

```
<INPUT type="reset" value="Borrar">
```

Elemento Button

Es un botón simple al cual se le pueden asociar diferentes funcionalidades. Ejemplo:

```
<INPUT type="button" value="Aceptar">
```

Elemento Text Area

Permite al usuario ingresar varias líneas de texto en un sólo campo. Sirve para ingresar datos como: comentarios, observaciones, preferencias, etc. Este elemento sí requiere de etiqueta de finalización. Ejemplo:

```
<TEXTAREA name="comentario" rows="14" cols="50">
Ingrese sus comentarios aquí
</TEXTAREA>
```

Elemento Select

Se utilizan para permitir al usuario seleccionar uno o más elementos de una lista o combo. Ejemplo:

```
<SELECT name="calificacion">
<OPTION>Muy buena
<OPTION>Bueno
<OPTION>Regular
<OPTION>Malo
</SELECT>
```

Con el atributo SELECTED podemos indicar la opción predeterminada. Si se permite escoger más de un elemento, se debe usar el atributo MULTIPLE. Mediante el atributo SIZE se indica si debe aparecer como una lista o combo. Si no se especifica lo contrario, la opción predeterminada es que se muestre como un combo box.

Elemento Hidden

Permite implementar los "campos escondidos" que ayudan a mantener el estado en una aplicación web.

```
<INPUT type="hidden" name=codigocia>
```

Esto hace que el elemento no aparezca en el formulario, es decir, no se hace visible

B.5. Programación en la Web.

La especificación HTML es básicamente definida para páginas estáticas. Para implementar páginas dinámicas, validaciones de datos, acceder a datos externos, etc. se utilizan tecnologías complementarias.

Las tecnologías para el desarrollo de aplicaciones en la Web se las pueden clasificar de dos maneras, por su lugar de ejecución y por su forma de implementación.

B.5.1. Scripts según su lugar de ejecución.

Por su lugar de ejecución, se clasifican en:

a) Scripts de Cliente
b) Scripts de Servidor

Scripts de Cliente.

Son interpretados por el browser. Se utilizan principalmente para tareas de animación y validación de datos. El script completo "viaja" al cliente para que pueda ser ejecutado por el browser.

Algunos ejemplos de este tipo de scripts son Java Applets, Active X Controls, Java Scripts e incluso VB Script.

Scripts de Servidor.

Son interpretados por el servidor Web, por lo tanto son dependientes de esta tecnología. Efectúan procesamiento y envían resultados al browser en formato HTML puro. Se utilizan para todo tipo de procesamiento, validaciones, acceso a bases de datos y componentes COM, etc.

Ejemplos de este tipo de scripts son CGI, ASP, PHP, Java Servlets, etc.

B.5.2. Según su forma de implementación.

Por su forma de implementación se clasifican los programas Web en:

a) Programas aislados

b) Programas embebidos

Programas aislados

Se caracterizan porque los programas residen externamente a las páginas HTML. Algunos ejemplos de este tipo de programas son CGI (Python, Perl) y Servlets (programas en Java).

Programas embebidos

Como su nombre lo indica, este tipo de programas residen dentro de las mismas páginas HTML agregándole ciertos "tags" que le indicar al servidor Web el tipo de programa que se incluye.

Algunos tecnologías embebidas son: ASP (Active Server Pages), JSP, PHP.

B.6. DHTML

B.6.1. Concepto de DHTML.

Para realizar tareas tales como animaciones y validaciones de datos, se usa la tecnología DHTML. Dynamic HyperText Markup Language se refiere a todas los lenguajes y tecnologías que ejecutan scripts del lado del cliente por medio del browser. De esta manera se aprovecha de mejor manera los recursos del cliente y se disminuye la carga de la comunicación con el servidor web. Uno de los lenguajes más utilizados es el Javascript.

B.6.2. Características de DHTML.

Las principales características de DHTML son:

- Provee un DOM (Document Object Model) o modelo de objetos que facilita la programación de cualquier aspecto de la interfaz de usuario.
- Se programa en base a eventos.
- La implementación del DOM es muy particular al browser que se utilice.

B.6.3. Usos de DHTML.

Los principales usos de la tecnología DHTML son:

- Para validación de datos.
- Para modificar la presentación y el contenido del documento.
- Para realizar acciones que eviten el tener que "bajar" un nuevo documento completo del servidor web.
- Para realizar animaciones sin involucrar al servidor web.

B.6.4. Javascript.

Javascript Es un lenguaje interpretado desarrollado por Netscape para permitir la creación de páginas interactivas. Aunque se parece mucho a Java, este lenguaje fue creado independientemente. Actualmente es respaldado por muchas compañías de software.

Es un lenguaje abierto que no requiere licencia de uso. Soportado por los principales browsers de Netscape y Microsoft. La versión Javascript de Microsoft se denomina Jscript.

B.7. CSS.

CSS (Cascading Style Sheets) es una especificación adicional al lenguaje HTML, desarrollada por el W3C, que posibilita un mayor control

de las forma en que las páginas se presentan al usuario, a la vez que facilita la estandarización y mantenimiento de las mismas.

Usando CSS, los diseñadores pueden crear hojas de estilos que definen cómo los diferentes elementos de una página, tales como títulos, links y párrafos, deben mostrarse. De esta manera se centraliza y estandariza un sitio Web.

El término "cascading" radica en el hecho de que se pueden aplicar múltiples hojas de estilo a una misma página web.

Existen 3 maneras de usar estilos en una página Web:

1. In-line
2. Clases
3. Archivo externo

B.7.1. In-line CSS.

Permite aplicar estilos individuales a cualquier tag de una página. El problema con este método es que se pierde estandarización. Se debe usar sólo para establecer excepciones.

Ejemplo: El siguiente estilo establece un título en color azul.

```
<H2 style="color:blue">Titulo</H2>
```

B.7.2. Archivo externo CSS.

Se generan creando un archivo con la extensión .CSS y referenciando ese nombre en el tag HEAD del documento.

Ejemplo: Para usar un archivo llamado WDSTYLE.CSS como hoja de estilo, se lo hace de la siguiente manera:

```
<link rel=stylesheet href="wdstyle.css" type="text/css">
```

Este archivo puede contener especificaciones como:

```
H1    {
            color:red;
            font-family:"Arial";
            font-size:40;
      }
```

B.7.3. Clases CSS.

Permite definir estilos que pueden ser aplicados a los tags. Ofrece un control intermedio entre el método in-line y el método del archivo externo. Se los define y se los referencia de la misma manera que los estilos definidos en archivos externos.

Ejemplo: Se define una clase (subtitulo) que establece el color del texto en amarillo

```
.subtitulo
      {color:yellow;}
```

Para usar el estilo, se usa el atributo class:

```
<H4 class="subtitulo">Antecedentes  </H4>
```

C

ARQUITECTURA COM / DCOM / COM+

C. COM / DCOM / COM+

C.1. Antecedentes.

En los inicios de la década del 90', Microsoft tenía una tecnología que facilitaba la comunicación entre programas. Se llamaba OLE (Object Linking and Embedding), y aunque era funcional, tenía limitaciones en cuanto al rendimiento. Decidieron entonces mejorar esta tecnología, pero se dieron cuenta que se necesitaban bases sólidas sobre la cual edificar el nuevo OLE 2.

El fruto de este trabajo de reingeniería de software se lo denominó COM (Component Object Model).

C.2. Características de COM.

COM es una arquitectura de desarrollo de software que se fundamenta en el uso de piezas de código que colaboran entre sí.

Para lograr la comunicación entre diferentes objetos de código, COM establece estándares que deben ser seguidos estrictamente por desarrolladores y proveedores de herramientas de software para lograr que exista la comunicación adecuada entre componentes.

Como el estándar de COM se basa en compatibilidad a nivel binario, se puede usar cualquier lenguaje de programación que soporte el estándar

COM. De esta manera se aprovechan mejor los talentos de los grupos de desarrolladores. Entre los lenguajes más usados para desarrollar componentes COM están Visual Basic, C++, Java, Delphi, e incluso COBOL. Existen herramientas de programación que no dan soporte al desarrollo de objetos COM, pero sin embargo, pueden utilizar dichos objetos. Por ejemplo, programas scripts en ASP o Visual Basic for Applications.

Otra característica importante de COM es que se basa en la tecnología orientada a objetos. De esta manera permite implementar conceptos como el de encapsulación, herencia y polimorfismo cuando se usan las herramientas de desarrollo de software adecuadas.

Un requerimiento de OLE 2 era que la arquitectura debía permitir que procesos en diferentes espacios de memoria se comuniquen. La especificación COM específicamente establece la forma en la cual los programas clientes pueden crear objetos en espacios de memoria separados. COM es la tecnología base en la estrategia Microsoft para la computación distribuída.

C.3. Beneficios de COM.

Los beneficios que proporciona COM al desarrollo de aplicaciones son:

- Es independiente de una herramienta o lenguaje en particular. Permite usar el lenguaje de programación más adecuado a la tarea.
- Facilita la creación de código compacto y reutilizable.
- El mantenimiento de las aplicaciones se hace menos complejo. Los componentes encapsulan cierta funcionalidad. Si la funcionalidad debe cambiar, se actualiza sólo el componente y no todas las aplicaciones que usan el componente.
- Desarrollo más fácil. El dividir la aplicación en componentes proporciona la ventaja de desarrollar y probar pequeñas piezas de código de forma independiente.
- Múltiples desarrolladores pueden trabajar en un mismo proyecto permitiendo que las tareas sean distribuídas entre todos los miembros del equipo de desarrollo.
- Permite implementar aplicaciones escalables a miles de usuarios

C.4. DCOM.

El estándar COM original especificaba los requerimientos para comunicación entre objetos que "corrían" en una sola computadora. Distributed COM (DCOM), es una adición a esta especificación que facilita la comunicación entre objetos que existen dentro de una red de computadoras.

C.5. Clasificación de componentes.

Existen dos tipos de componentes de software:

- In-Process
- Out-of-Process

C.5.1. Componentes In-Process

Un componente in-process (en proceso) comparte el mismo espacio de memoria que su programa invocador. El cliente del componente puede ser un programa simple u otro componente. Este tipo de componentes se implementan en bibliotecas DLL (Dynamic Link Libraries).

La principal ventaja de las bibliotecas DLL está en que son rápidas en ejecución, y su desventaja es que si se "cae" la biblioteca, se cae toda la aplicación cliente.

C.5.2. Componentes Out-of-Process

Los componentes out-of-process (fuera de proceso), corren en su propio espacio de memoria, separado de su programa cliente. Estos componentes se implementan como programas EXE independientes. Aunque su rendimiento no es equivalente al de las bibliotecas DLL, en cambio ofrecen una protección de ejecución a la aplicación cliente.

C.6. Interfaces.

C.6.1. Concepto de interfaces

Uno de los pilares en que se basa COM es la programación basada en interfaces. El concepto de la programación basada en interfaces es el de ofrecer una vía indirecta para acceder a un objeto. Es como un intermediario entre la clase (tipo de datos) y el cliente que la usa.

Se dice entonces que una clase implementa una o más interfaces, y que una interface puede ser implementada por una o más clases.

Una vez que la interfase ha sido implementada por la clase, un cliente puede crear un objeto de la clase y comunicarse con ésta a través de una interface (referencia basada en interface).

Es un tipo de dato abstracto que no permite crear objetos directamente a partir de ella. También es un "contrato" ya que todos los métodos que estén en la interface deben ser implementados por las clases que adopten dicha interface.

C.6.2. Beneficios de usar interfaces.

El usar interfaces conlleva un trabajo adicional al desarrollador. Cuando existe una relación de uno a uno entre interfaces y clases es difícil justificar la creación de interfaces. En los otros casos, los beneficios de usar interfaces son:

- Para facilitar el manejo de versiones de objetos
- Para separar el desarrollo de programas clientes y componentes.
- Para implementar polimorfismo y herencia basada en interfaces.
- Para que en tiempo de ejecución los programas clientes puedan determinar el tipo de objeto con el cual trabajan (run-time type inspection).
- Para que los clientes no construyan dependencias sobre clases.

C.7. Creación de Componentes.

Los pasos para crear un componente COM son los siguientes:

1. El primer paso es definir mediante un análisis los métodos que tendrá el componente.
2. Se crea la interface en un archivo de texto común o utilizando un utilitario de edición como el Visual C++. Las interfaces se definen en un lenguaje llamado IDL (Interface Definition Language). Visual Basic

crea automáticamente una interface default en base a los métodos públicos definidos en las clases.

3. Al definir la interface se escoge la interface de la cual hereda la funcionalidad. Existen dos interfaces bases: IUnknown e IDispatch. La interface IUnknown provee acceso a clientes mediante una técnica llamada v-table binding que es un enlace de alto rendimiento, pero que es solo soportada por cierto tipo de clientes. La interface IDispatch está basada en la interface IUnknown, por lo tanto provee enlace "v-table binding" y también un acceso dinámico a todo tipo de clientes, especialmente clientes tipo scripts, pero que funciona mucho más lentamente. Este enlace se lo denomina COM Automation.
4. Se compila la interface utilizando un compilador llamado MIDL (Microsoft Interface Definition Language Compiler). Este compilador genera la "type library", que vendría a ser como el "contrato".
5. Se crea el componente que implementará los métodos de la interface. Se lo denomina también "coclass". Aquí se referencia la type library creada anteriormente.
6. Se compila el componente.

C.8. Type Library.

Una type library es un catálogo que describe las clases, interfaces y métodos definidos en una especificación IDL. Es el "contrato" que indica cual es la funcionalidad que ofrece un objeto.

Los archivos de "type library" tienen algunas posibles extensiones: .tlb, .dll, .exe, .olb y .ocx

C.9. Enlace de Componentes.

Una vez creado la type library del objeto COM, se puede empezar a crear el programa cliente. Existen tres maneras de acceder al objeto COM, a saber:

1. V-table binding
2. Early binding
3. Late binding

C.9.1. V-Table binding.

Este tipo de enlace se obtiene cuando en el programa objeto cliente se ubican punteros a las ubicaciones de los métodos del componente. Es el más rápido en rendimiento, pero requiere una referencia en tiempo de

compilación. Esta referencia se logra accesando la type library del componente.

C.9.2. Early binding.

Este enlace requiere que exista la type library del componente pero no realiza un vínculo en tiempo de compilación. De esta manera da soporte al pre-chequeo de sintaxis. Es usada por lenguajes o ambientes de desarrollo interpretados que soportan el establecimiento de referencias, ejemplo: Visual Interdev.

C.9.3. Late binding.

Cuando no se tenga acceso a una type library o cuando se referencie al componente de una manera dinámica – como un object genérico, se establece un enlace en tiempo de ejecución. Este enlace es dinámico y flexible, pero en cambio es el más lento de todos y no provee chequeo de sintaxis.

C.10. Uso de Componentes.

Los pasos necesarios para acceder a un componente son:

1. Registrar el componente. Este paso es obligatorio para que el cliente pueda localizar el componente en ejecución. Visual Basic automáticamente registra el componente luego de finalizar la compilación.
2. Crear una referencia al componente. Este paso se realiza solamente si disponemos de la type library del componente y si la herramienta de desarrollo que usamos soporta hacerlo.
3. Dimensionar la interface del componente que se desea utilizar. En otras palabras, establecer una variable con el tipo de la interface que vamos a usar para acceder al componente. Para este propósito, en Visual Basic se utiliza la instrucción DIM.
4. Instanciar la clase del componente que fue dimensionada. En Visual Basic esto se lo hace con la cláusula NEW.
5. Invocar los métodos del objeto creado. Se indica el método y los parámetros que se pasarán al objeto.

C.11. Comunicación entre procesos.

C.11.1. Objetos Proxy y Stub.

Para que dos objetos que corren en espacios de memoria diferentes se comuniquen, COM hace uso de un par de objetos llamados "proxy" y "stub". El objeto proxy se comunica con el cliente haciéndole creer que se trata del objeto. En cambio el stub se comunica con el objeto haciéndole creer que se trata del programa cliente. Estos objetos se comunican mediante un canal RPC, en donde se transmiten datos en ambos sentidos durante la ejecución de los métodos.

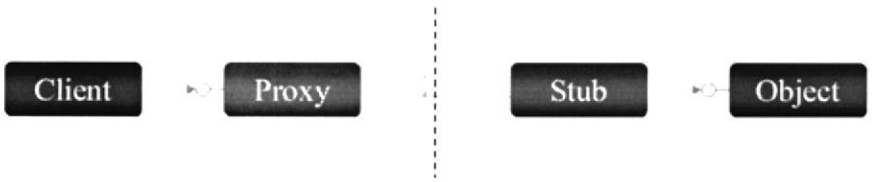


Figura C-1. Objetos Proxy y Stub

Estos objetos facilitan una de las principales características de COM que es la "transparencia de ubicación", ya que ni el objeto ni el programa cliente conocen en dónde se encuentra el objeto.

C.11.2. Marshaling.

Los parámetros transmitidos entre el programa cliente y el objeto, son afectados por un procedimiento denominado "marshaling" y efectuado

por el servicio "universal marshaler" implementado en la biblioteca OLEAUT32.DLL.

La llamada a un método remoto toma mucho más tiempo que una llamada a un método local. La creación de los objetos proxy/stub afectan la duración de los métodos.

C.12. COM+.

C.12.1. Antecedentes.

Hasta antes de aparecer el sistema operativo Windows 2000, los objetos COM distribuídos en servidores Windows NT residían en un servidor transaccional denominado MTS (Microsoft Transaction Server). Este programa ofrecía servicios de pool de objetos, manejo de transacciones, seguridad y administración.

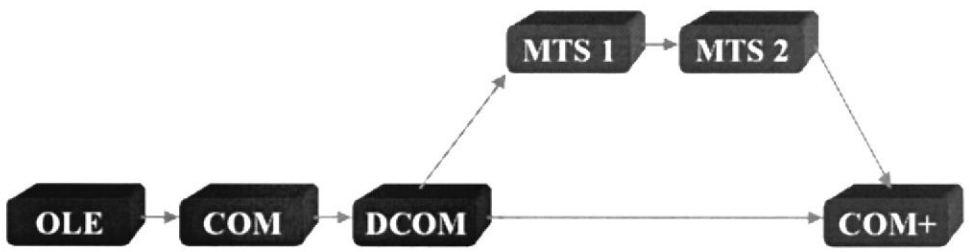


FIG. No. 2 EVOLUCIÓN DE COM A COM+

Con el desarrollo de Windows 2000, Microsoft decidió unir en un sólo producto los servicios de DCOM y de MTS, y denominó a este producto COM+.

C.12.2. Características

COM+ es DCOM y MTS, pero a la vez extiende los servicios de ambos y mantiene un gran nivel de compatibilidad con las aplicaciones anteriores. De hecho, se puede migrar fácilmente de la versión anterior.

La arquitectura distribuída de COM (DCOM), se mantiene de la misma manera en COM+.

C.12.3. Servicios de COM+.

Los principales servicios que ofrecen COM+, son entre otros:

- Soporte de Transacciones: Permite especificar cuáles componentes necesitan soporte de transacciones y el nivel de protección.
- Activación Just-in-Time: Los objetos se activan justo en el momento en que se hace la llamada al primer método.
- Concurrencia y sincronización: Para garantizar la integridad de las llamadas a objetos.

- Pool de objetos: Provee objetos pre-instanciados, útiles para mejorar el rendimiento de aplicaciones.
- Balanceo de carga: Permite establecer un grupo de servidores que ejecutarán los objetos de la aplicación de una manera balanceada. Esta técnica provee un mejor rendimiento y tolerancia a fallas.
- Seguridad: Ofrece servicios de autenticación y autorización.
- Queued Components: Posibilita comunicación asincrónica entre cliente y objetos.
- In-memory Database: Establece datos en memoria RAM basados en datos de una base de datos permanente. Acelera el acceso a datos de lectura.
- Administración: Permite administrar localmente o remotamente cualquier servidor COM+.

C.12.4. Aplicaciones COM+.

En COM+ se pueden definir dos tipos de aplicaciones.

- Aplicación de Biblioteca
- Aplicación de Servidor

Las aplicaciones de biblioteca corren como in-process (proceso local) y se ejecutan muy rápidamente. Se utilizan para implementar objetos

utilitarios. Su desventaja es que no pueden utilizar ciertos servicios de COM+ tales como seguridad y balanceo de carga.

Las aplicaciones de servidor corren dentro de una aplicación contenedora COM+.

D

INTERNET INFORMATION SERVER Y ACTIVE SERVER PAGES

D. INTERNET INFORMATION SERVER Y ACTIVE SERVER PAGES

D.1. Internet Information Server.

D.1.1. Internet Information Server.

IIS (Internet Information Server) es el servidor web de Microsoft. Actualmente en su versión 5.0, IIS se encarga del proceso de atender solicitudes de browsers que se comunican mediante el protocolo HTTP. Incluso soporta otros protocolos de comunicación como el FTP (File Transfer Protocol).

D.1.2. Características.

Las características principales que provee IIS son:

- Seguridad: Permite la autenticación segura de los usuarios a través de los servidores proxy y servidores de seguridad. Además de las autenticaciones anónimas, básica HTTP e integrada de Windows.

- Comunicaciones seguras: Mediante el empleo de SSL 3.0 (Secure sockets layer) y Transport Layer Security (TLS) proporciona una forma segura para intercambiar información entre clientes y servidores.

- Opciones de configuración: Se pueden establecer los permisos para las operaciones web de lectura, escritura, ejecución y secuencia de comandos en los sitios, directorios o archivos.

- Programabilidad: Provee compatibilidad completa con las páginas ASP, acceso a componentes COM y funcionalidad para el procesamiento de errores.

- Monitoreo de rendimiento: Se puede vigilar en tiempo real y de manera gráfica las estadísticas del tráfico del sitio, tales como peticiones diarias, peticiones por hora, visitantes diarios, visitantes por hora, etc.

- Administración centralizada: Se puede administrar un servidor IIS desde cualquier computadora con Windows 2000.

- Protección de aplicaciones: IIS incrementa la confiabilidad de las aplicaciones Web, ya que de manera predeterminada, ejecuta todas las aplicaciones en un proceso común o agrupado que está separado de los procesos del núcleo de IIS.

D.2. Active Server pages

D.2.1. Concepto.

ASP (Active Server Pages) es la tecnología de Microsoft para la implementación de scripts del lado del servidor web.

Expone un modelo de objetos mediante el cual se accede a toda la funcionalidad del servidor web.

D.2.2. Características.

Las principales características de ASP son:

- Permite crear contenido dinámico, es decir que puede generar HTML en tiempo de ejecución.
- Permite mantener el estado en una aplicación web, logrando así superar una limitación del protocolo HTTP.
- Es independiente del browser del cliente, ya que éste recibe únicamente los documentos HTML generados.
- Los scripts ASP pueden ser codificados en varios lenguajes, siendo los más usados el VB Script y el JScript.
- Permite usar componentes COM. De esta manera se abren las puertas a una gran cantidad de posibilidades, por ejemplo, acceso a base de datos, seguridad, utilitarios, etc.
- Acceso simplificado a bases de datos a través de ADO/OLEDB.

En resumen, ASP permite combinar HTML, scripts y componentes para el desarrollo de aplicaciones Web.

D.2.3. Desarrollo de una página ASP.

Las páginas ASP tienen la extensión .asp. Las páginas HTML estáticas también pueden tener la extensión .asp sin ninguna repercusión en cuanto al rendimiento.

Para indicar en donde empieza un script en una página ASP existen dos alternativas, usando la etiqueta <SCRIPT> ó usando el delimitador de inicio de script <% y de finalización %>.

Ejemplos:

```
<SCRIPT runat="SERVER"> … </SCRIPT>
<% ..... %>
```

Los scripts ASP se ejecutan de forma lineal de arriba hacia abajo justo antes de enviar la página web al browser.

D.3. Modelo de Objetos ASP.

El entorno de programación ASP provee objetos con los cuales los scripts ASP interactúan para implementar cualquier funcionalidad.

Estos objetos están inmediatamente disponibles al desarrollador, por lo tanto no necesita, ni debe instanciarlos.

Los objetos principales del entorno ASP son: Request, Response, Server, Session y Application. Además tiene dos objetos nuevos: ObjectContext y ASPError.

D.3.1. Objeto Request.

El script ASP utiliza el objeto Request para recibir toda la información que envía el browser. Esta información puede ser los campos en un formulario web o información de control del browser. Ejemplo: campos de formularios, variables de entorno, contenido de "cookies" y certificados.

El objeto Request ofrece algunas colecciones que proveen la información anteriormente indicada, tales como: ServerVariables, Form, QueryString, Cookies y ClientCertificate.

D.3.2. Objeto Response.

Utilizado para enviar al servidor web la salida o documento HTML que se enviará al browser.

El método write del objeto Response, es el más usado para enviar código HTML.

Tiene la colección Cookies, que sirve para modificar el contenido de las variables de cookies que residen en el cliente.

D.3.3. Objeto Server.

Se utiliza para acceder a los recursos que provee el servidor web. Su utilidad principal es la de acceder a componentes COM.

El método más utilizado es el `CreateObject`, el cual permite instanciar componentes COM recibiendo como parámetro el `ProgID` del componente.

D.3.4. Objeto Session.

Permite mantener el estado de una sesión o actividad de un usuario en un sitio web.

Permite crear variables y almacenar valores que existirán durante el tiempo de vida de la sesión de usuario y que podrán ser accesadas o "vistas" únicamente por esa sesión de usuario

D.3.5. Objeto Application.

Usado para leer y almacenar información sobre la aplicación web. Esta información es compartida entre todos los usuarios de la aplicación y existe durante el tiempo de vida de la misma.

D.3.6. Objeto ObjectContext.

Se utiliza el objeto ObjectContext para confirmar o anular una transacción COM, iniciada por una secuencia de comandos contenida en una página ASP.

Cuando una página ASP contiene la directiva @TRANSACTION, la página se ejecuta en una transacción y no finaliza el proceso hasta que la transacción se completa correctamente o se produce un error.

D.3.7. Objeto ASPError.

El objeto ASPError se utiliza para obtener información acerca de una condición de error en una secuencia de comandos de una página ASP.

D.4. Aplicaciones ASP.

D.4.1. Definición.

Una aplicación ASP ó ASA (ASP Application) está conformada por todos los archivos .asp, .htm y otros que forman parte de un sitio web.

En realidad se trata de un directorio virtual administrado por el servidor web y que contiene el archivo global.asa. La presencia de este archivo indica el sitio donde empieza una aplicación ASP.

D.4.2. Archivo GLOBAL.ASA.

Si un subdirectorio contiene un archivo global.asa, ése subdirectorio y todo su contenido son parte de una ASP application independiente.

Dentro de este archivo se puede codificar instrucciones para controlar los eventos de inicio y fin de aplicación y sesión. Los eventos suministrados son:

- Eventos de aplicaciones: `Application_OnStart` y `Application_OnEnd`.
- Eventos de sesiones: `Session_OnStart` y `Session_OnEnd`.

Una aplicación comienza cuando el primer usuario accesa al directorio virtual raíz y termina cuando expira la sesión del último usuario.

Una sesión empieza cuando un usuario entra por primera vez al directorio virtual raíz de una aplicación y finaliza cuando se lo indiquemos explícitamente o hasta que transcurra el tiempo máximo de inactividad.

D.5. Cookies.

D.5.1. Definición.

Un "cookie" es un archivo de texto que almacena información concerniente al usuario y a una conexión específica con una aplicación ASP.

D.5.2. Usos de Cookies.

Los cookies se utilizan para guardar información propia del usuario y que sirva para "ahorrar" tiempo o para reconocerlo en futuras conexiones.

Se usan para almacenar datos como identificadores de usuarios, contraseñas, últimas fechas, preferencias, datos de ubicación, etc.

D.5.3. Desventajas de los Cookies.

Las principales desventajas de los cookies son:

- No son soportados por todos los browsers.
- Pueden ser eliminados o alterados por el usuario.

GLOSARIO

ADO

ActiveX Data Objects

Interface de alto nivel de Microsoft para objetos de datos.

CGI

Common Gateway Interface

Especificación para transferir información entre un servidor Web y un programa CGI. El programa CGI puede ser escrito en cualquier lenguaje de programación, incluyendo C, Perl, Java y Visual Basic.

DLL

Dynamic Link Library

Biblioteca de funciones ejecutables que puede ser usada por una aplicación Windows.

IDL

Interface Definition Language

Lenguaje para describir las interfaces de objetos de software.

Interface

Una declaración tipo contrato de un conjunto de métodos y los argumentos de llamada que los componen.

Java

Lenguaje de programación de alto nivel orientado a objetos de propósito general, similar a C++, pero simplificado para eliminar características que del lenguaje C++ que suelen producir errores. Altamente portable, ya que existen ambientes de ejecución (llamadas máquinas virtuales Java) para la mayoría de los sistemas operativos.

Java Applets

Pequeñas aplicaciones Java que pueden ser descargadas del WWW y correr en el browser de una computadora cliente.

Lenguaje script

Lenguaje de programación simple interpretado.

OLTP

Online Transaction Processing
Tipo de procesamiento computacional en el que la computadora responde inmediatamente a las solicitudes de los usuarios.

Perl

Practical Extraction and Report Language.
Lenguaje de programación especialmente diseñado para procesamiento de textos. Es uno de los lenguajes más populares para escribir scripts CGI .

PHP

PHP Hypertext Preprocessor
Lenguaje de script embebido, del lado del servidor similar al lenguaje C utilizado para crear páginas dinámicas. Su fortaleza radica en la compatibilidad que provee con diferentes motores de bases de datos. Se incluye como parte de servidores Web como RedHat Linux.

Python

Lenguaje de programación interpretado y orientado a objetos altamente portable a diferentes plataformas.