UNIVERSIDAD NACIONAL AUTÓNOMA DE NICARAGUA
UNAN-LEÓN
FACULTAD DE CIENCIAS Y TECNOLOGÍA
DEPARTAMENTO DE COMPUTACIÓN

TESIS PARA OPTAR AL TÍTULO DE INGENIERO EN SISTEMAS DE INFORMACIÓN

TEMA:

DESARROLLO DE UN SISTEMA DE INFORMACIÓN PARA EL CONTROL Y GESTIÓN DEL VIDEO-RENTAL TAURO VIDEO

PRESENTADO POR:

- Delvis Ramón Lezama Maravilla.
- Carlos Ariel Saldaña Mendoza.
- Julio César Varela.

Tutor: Ing. Karina Esquivel

León, Julio del 2010

AGRADECIMIENTO

A *DIOS* nuestro creador por darnos la vida, guiar cada uno de nuestros pasos y habernos permitido culminar nuestros estudios universitarios.

A *nuestros padres* que con mucho esfuerzo y sacrificio nos apoyaron incondicionalmente en este largo camino para ver nuestro sueño hecho realidad.

A *nuestros profesores* por guiarnos y brindarnos muchos conocimientos, lo que hizo posible que hoy estemos finalizando con nuestros estudios, por la paciencia que nos tuvieron y el tiempo que nos dedicaron.

A *nuestro tutor Ing. Karina Juliett Esquivel Alvarado* por su valiosa guía, colaboración y profesionalismo que nos permitió realizar nuestro trabajo monográfico.

A *nuestros amigos* por animarnos a seguir siempre adelante y a *nuestros compañeros* por compartir sus experiencias.

Al *Sr. Esteban Jirón Bolaños*, propietario del Video-rental Tauro Video por confiar en nosotros para la elaboración del sistema de información para su negocio.

A todas aquellas personas que no mencionamos pero que de manera directa o indirecta nos ayudaron a lo largo de nuestra carrera.

DEDICATORIA

A Dios mi Padre Celestial

Por darme la vida, la sabiduría y la fortaleza para seguir adelante, por estar siempre a mi lado en los momentos más difíciles y darme la oportunidad de alcanzar mis metas. Sobre todo por ser la Luz que ilumina el camino de mi vida y haberme dado una familia que me ama.

A mi madre

Nimia Varela Macías por enseñarme cada día con su ejemplo el amor a Dios, la honestidad y la perseverancia. Por depositar su confianza en mí y con su trabajo y esfuerzo me ha permitido cumplir mis sueños de ser un profesional en la vida.

A mi hermana Deyanira Varela y Diego Suárez

Por haberme apoyado todos estos años de estudio y quienes día a día me motivaban a seguir adelante.

A mis Hermanos y Familiares

Por siempre confiar en mí y por brindarme su ayuda en todos los momentos de mi vida.

Julio César Varela.

DEDICATORIA

A DIOS Padre

Por estar siempre a mi lado en las buenas y en las malas dándome sabiduría, entendimiento y fortaleza para lograr mis metas. La misericordia y el amor de nuestro señor es tan grande que si uno confía en él jamás seremos defraudado muchas gracias PADRE.

A mis Padres

Victoria Maravilla Quezada y Delvis Iván Lezama Silva, por haberme dado todo lo necesario para lograr mis estudios, sus consejos, dedicación, amor, apoyo moral y económico, todo eso y mucho más he aquí el fruto de todos sus sacrificios.

A mi Hermano Iván Antonio Lezama Maravilla

Por ser la motivación más grande, el angelito de la casa y sobre todo la fuente de agua que me inspira a ser una persona responsable y mejor.

A mis Amigos

Por estar siempre conmigo en tiempo difíciles, son muchos los amigos que aquí no podré mencionar, pero dentro de mi corazón los llevaré presente gracias por su ayuda y a todas las demás personas que directa e indirectamente me brindaron su apoyo para alcanzar mis metas.

Delvis Ramón Lezama Maravilla.

INDICE

I. INTRODUCCIÓN

Vivimos en una época dominada por la tecnología en donde las computadoras son herramientas indispensables para el desarrollo comercial, intelectual y cultural de la humanidad. Como consecuencia del incremento de información, se hace cada vez más difícil el manejo de la misma, por tanto es necesario automatizar la gestión de las empresas.

Junto al crecimiento comercial, el avance tecnológico ha permitido idear tecnologías para compartir información dentro de las empresas. Esto ha conllevado a crear redes de computadoras. Nosotros aprovecharemos estas tecnologías para desarrollar un sistema cliente/servidor que sea capaz de trabajar con esta infraestructura.

Nuestro sistema será conocido con el nombre *Sistema de Información para el Control y Gestión del Video-Rental Tauro Video*, el cual permitirá la automatización de los procesos que lleva a cabo dicho video-rental, además de la generación de reportes que ayuden a la gestión administrativa del negocio.

El sistema deberá garantizar aspectos como la eficiencia y exactitud de la información generada. Implementando la metodología de Ingeniería del Software lograremos cumplir este objetivo.

II. ANTECEDENTES

El Video-rental Tauro Video inició hace aproximadamente quince años como un pequeño negocio. Llevar el control de las películas en existencia, los clientes, los alquileres y las filmaciones en ese tiempo, era mucho más sencillo debido al poco volumen de información.

Anteriormente el control se llevaba de manera manual utilizando un sistema que ellos denominaban “tarjetas de control con formato” el cual consistía en un recibo que se le entregaba al cliente en donde se anotaban los códigos y nombres de las películas que el cliente alquilaba y se le retenía su cédula. Actualmente registran los alquileres y las filmaciones en un cuaderno de apuntes.

En este negocio nunca se ha implementado un sistema computarizado que les ayude a mejorar el servicio que ofrecen a sus clientes, así como obtener información de interés para la administración.

III. JUSTIFICACIÓN

Al analizar la realidad de nuestro país, es evidente que en la mayoría de las empresas existe la necesidad de automatizar sus procesos de gestión y control de los datos producidos en las actividades comerciales que realizan. Además, las empresas prefieren soluciones que se ajusten a la medida de sus requerimientos

Por ello hemos decidido desarrollar un sistema cliente/servidor que tendrá la capacidad de trabajar en una red local utilizando un servidor de bases de datos para agilizar las actividades que se llevan a cabo en el video-rental Tauro Video.

Nuestro sistema automatizará los procesos que se realizan de forma manual en el video-rental ubicándolo a la par de la tecnología de la informática moderna.

IV. OBJETIVOS

OBJETIVO GENERAL

- Desarrollar un sistema cliente/servidor para la automatización de los procesos de Gestión y Control del video-rental Tauro Video.

OBJETIVOS ESPECÍFICOS

- Crear un sistema de calidad, seguro y confiable aplicando la metodología de la Ingeniería del Software.
- Diseñar una interfaz de usuario amigable e intuitiva que permita facilidad de uso del sistema.
- Automatizar la información de películas, clientes, alquileres, devoluciones y filmaciones del video-rental.
- Generar reportes con información de interés para la administración del negocio.
- Utilizar el lenguaje de Programación C# y el Sistema Gestor de Base de Datos SQL Server 2000.

V. MARCO TEÓRICO

5.1 SISTEMA

Es un conjunto de elementos con relaciones de interacción e interdependencia que le confieren entidad propia al formar un todo unificado. También se puede definir como un conjunto de elementos organizados que se encuentran en interacción, que buscan alguna meta o metas comunes.

5.2 INTERFAZ DE USUARIO

Es uno de los componentes más importantes de cualquier sistema computacional, pues funciona como el vínculo entre el humano y la máquina, es responsable de solicitar comandos al usuario, y de desplegar los resultados de la aplicación de una manera comprensible. [Chiavenato, 1992]

5.3 BASE DE DATOS

Una base de datos se puede definir como un conjunto de información relacionada que se encuentra agrupada o estructurada en uno o varios archivos, los cuales son creados y manipulados por Sistemas Gestores de Bases de Datos como Microsoft Access, SQL Server, MySQL, Oracle, etc., los cuales utilizan el lenguaje SQL para realizar consultas a la base de datos. Estas consultas pueden ir desde la creación de la base de datos, tablas, vistas, hasta el guardado, actualización, y recuperación de la información. Desde el punto de vista informático, la base de datos es un sistema formado por un conjunto de datos almacenados en archivos que permiten el acceso directo a ellos y un conjunto de programas que manipulen ese conjunto de datos.

5.4 SOFTWARE Y HERRAMIENTAS PARA LA IMPLEMENTACIÓN DEL SISTEMA

5.4.1 MICROSOFT VISUAL C#

C# es el nuevo lenguaje de propósito general diseñado por Microsoft para su plataforma .NET. Aunque es posible escribir código para la plataforma .NET en otros lenguajes, C# es el único que ha sido diseñado específicamente para ser utilizado en ella, por lo que programar usando C# es mucho más sencillo e intuitivo que hacerlo con cualquiera de los otros lenguajes ya que C# carece de elementos

heredados innecesarios en .NET. Por esta razón, se suele decir que C# es el **lenguaje nativo de .NET.**

La sintaxis y estructuración de C# es muy similar a la C++, ya que la intención de Microsoft con C# es facilitar la migración de códigos escritos en estos lenguajes a C# y facilitar su aprendizaje a los desarrolladores habituados a ellos. Sin embargo, su sencillez y el alto nivel de productividad son equiparables a los de Visual Basic.

En resumen, C# es un lenguaje de programación que toma las mejores características de lenguajes preexistentes como Visual Basic, Java o C++ y las combina en uno solo. [Soto, 2009]

5.4.1.1CARACTERÍSTICAS DE C#

A continuación se resumen las principales características de C#. Algunas de las características aquí señaladas no son exactamente propias del lenguaje sino de la plataforma .NET en general.

- **Sencillez:** C# elimina muchos elementos que otros lenguajes incluyen y que son innecesarios en .NET. Por ejemplo:
    - El código escrito en C# es **autocontenido**, lo que significa que no necesita de ficheros adicionales al propio fuente tales como ficheros de cabecera o ficheros IDL (Interface Definition Language).
    - El tamaño de los tipos de datos básicos es fijo e independiente del compilador, sistema operativo o máquina para quienes se compile (no como en C++), lo que facilita la portabilidad del código.
    - No se incluyen elementos poco útiles de lenguajes como C++ tales como macros, herencia múltiple o la necesidad de un operador diferente del punto (**.**) para acceder a miembros de espacios de nombres (**::**).

- **Modernidad:** C# incorpora en el propio lenguaje elementos que a lo largo de los años ha ido demostrándose son muy útiles para el desarrollo de aplicaciones y que en otros lenguajes como Java o C++ hay que simular, como un tipo básico **decimal** que permita realizar operaciones de alta precisión con reales de 128 bits (muy útil en el mundo financiero), la inclusión de una instrucción **foreach** que permita recorrer colecciones con facilidad y es extensible a tipos definidos por el

usuario, la inclusión de un tipo básico **string** para representar cadenas o la distinción de un tipo **bool** específico para representar valores lógicos.

- **Orientación a objetos:** Como todo lenguaje de programación de propósito general actual, C# es un lenguaje orientado a objetos, aunque eso es más bien una característica del CTS (Common Type System) que de C#. Una diferencia de este enfoque orientado a objetos respecto al de otros lenguajes como C++ es que el de C# es más puro en tanto que no admiten ni funciones ni variables globales sino que todo el código y datos han de definirse dentro de definiciones de tipos de datos, lo que reduce problemas por conflictos de nombres y facilita la legibilidad del código.

  C# soporta todas las características propias del paradigma de programación orientada a objetos: **encapsulación**, **herencia simple** y **polimorfismo**.

  En lo referente a la encapsulación es importante señalar que aparte de los típicos modificadores **public**, **private** y **protected**, C# añade un cuarto modificador llamado **internal**, que puede combinarse con **protected** e indica que al elemento a cuya definición precede sólo puede accederse desde su mismo ensamblado.

  Respecto a la herencia, a diferencia de C++ y al igual que Java, C# sólo admite herencia simple de clases ya que la múltiple provoca más complicaciones que facilidades y en la mayoría de los casos su utilidad puede ser simulada con facilidad mediante herencia múltiple de interfaces. De todos modos, esto vuelve a ser más bien una característica propia del CTS (Common Type System) que de C#.

  Por otro lado y a diferencia de Java, en C# se ha optado por hacer que todos los métodos sean por defecto sellados y que los redefinibles hayan de marcarse con el modificador **virtual** (como en C++), lo que permite evitar errores derivados de redefiniciones accidentales. Además, un efecto secundario de esto es que las llamadas a los métodos serán más eficientes por defecto al no tener que buscar en la tabla de funciones virtuales la implementación de los mismos a la que se ha de llamar. Otro efecto secundario es que permite que las llamadas a los métodos

virtuales se puedan hacer más eficientemente al contribuir a que el tamaño de dicha tabla se reduzca.

- **Orientación a componentes:** La propia sintaxis de C# incluye elementos propios del diseño de componentes que otros lenguajes tienen que simular mediante construcciones más o menos complejas. Es decir, la sintaxis de C# permite definir cómodamente **propiedades** (similares a campos de acceso controlado), **eventos** (asociación controlada de funciones de respuesta a notificaciones) o **atributos** (información sobre un tipo o sus miembros).

- **Gestión automática de memoria:** Como ya se comentó, todo lenguaje de .NET tiene a su disposición el recolector de basura del CLR (Common Language Runtime). Esto tiene el efecto en el lenguaje de que no es necesario incluir instrucciones de destrucción de objetos. Sin embargo, dado que la destrucción de los objetos a través del recolector de basura es indeterminista y sólo se realiza cuando éste se active ya sea por falta de memoria, finalización de la aplicación o solicitud explícita en el fuente, C# también proporciona un mecanismo de liberación de recursos determinista a través de la instrucción **using**.

- **Seguridad de tipos:** C# incluye mecanismos que permiten asegurar que los accesos a tipos de datos siempre se realicen correctamente, lo que evita que se produzcan errores difíciles de detectar por acceso a memoria no perteneciente a ningún objeto y es especialmente necesario en un entorno gestionado por un recolector de basura. Para ello se toman medidas del tipo:

    - No se pueden usar **variables no inicializadas**. El compilador da a los campos un valor por defecto consistente en ponerlos a cero y controla mediante análisis de control de flujo, que no se lea ninguna variable local sin que se le haya asignado previamente algún valor.
    - Se comprueba que todo **acceso a los elementos de una tabla** se realice con índices que se encuentren dentro del rango de la misma.
    - A diferencia de Java, C# incluye **delegados**, que son similares a los punteros a funciones de C++ pero siguen un enfoque orientado a objetos, pueden almacenar referencias a varios métodos simultáneamente, y se

comprueba que los métodos a los que apunten tengan parámetros y valor de retorno del tipo indicado al definirlos.

- Pueden definirse métodos que admitan un número indefinido de parámetros de un cierto tipo, y a diferencia lenguajes como C/C++, en C# siempre se comprueba que los valores que se les pasen en cada llamada sean de los tipos apropiados.

- **Instrucciones seguras:** Para evitar errores, en C# se han impuesto una serie de restricciones en el uso de las instrucciones de control más comunes. Por ejemplo, la guarda de toda condición ha de ser una expresión condicional y no aritmética, con lo que se evitan errores por confusión del operador de igualdad (**==**) con el de asignación (**=**); y todo caso de un **switch** ha de terminar en un **break** o **goto** que indique cuál es la siguiente acción a realizar, lo que evita la ejecución accidental de casos y facilita su reordenación.

- **Sistema de tipos unificado:** A diferencia de C++, en C# todos los tipos de datos que se definan siempre derivarán, aunque sea de manera implícita, de una clase base común llamada **System.Object**, por lo que dispondrán de todos los miembros definidos en ésta clase (es decir, serán "objetos").

  A diferencia de Java, en C# esto también es aplicable a los tipos de datos básicos. Además, para conseguir que ello no tenga una repercusión negativa en su nivel de rendimiento, se ha incluido un mecanismo transparente de **boxing** y **unboxing** con el que se consigue que sólo sean tratados como objetos cuando la situación lo requiera, y mientras tanto puede aplicárseles optimizaciones específicas.

  El hecho de que todos los tipos del lenguaje deriven de una clase común facilita enormemente el diseño de colecciones genéricas que puedan almacenar objetos de cualquier tipo.

- **Sobrecarga de operadores:** Para facilitar la legibilidad del código y conseguir que los nuevos tipos de datos básicos que se definan a través de estructuras estén al mismo nivel que los básicos predefinidos en el lenguaje, al igual que C++ y a diferencia de Java, C# permite redefinir el significado de la mayoría de

los operadores incluidos los de conversión, tanto para conversiones implícitas como explícitas cuando se apliquen a diferentes tipos de objetos.

- **Eficiente:** En principio, en C# todo el código incluye numerosas restricciones para garantizar su seguridad y no permite el uso de punteros. Sin embargo, y a diferencia de Java, en C# es posible saltarse dichas restricciones manipulando objetos a través de punteros.

- **Compatible:** Para facilitar la migración de programas, C# no sólo mantiene una sintaxis muy similar a C++ o Java sino que permite incluir directamente en código escrito en C# fragmentos de código escrito en estos lenguajes. [Soto, 2009]

5.4.2 MICROSOFT SQL SERVER 2000

Microsoft SQL Server 2000 es un Sistema Gestor de Bases de Datos relacional (SGBDR). Una base de datos relacional proporciona una forma de organizar la información almacenándola en tablas de bases de datos. La información relacional se puede agrupar en tablas, y también se pueden definir relaciones entre tablas; de ahí el nombre, base de datos relacional. Los usuarios acceden a la información que está en el servidor a través de una aplicación. Los administradores acceden al servidor directamente para realizar tareas administrativas, de configuración y de mantenimiento de la base de datos. SQL Server es una base de datos dimensionable, lo que quiere decir que puede almacenar grandes cantidades de datos y muchos usuarios accediendo a los datos al mismo tiempo**.**

SQL Server nació en 1989 y ha cambiado de forma significativa desde entonces. Se han realizado grandes mejoras de dimensionabilidad, integridad, facilidad de administración, rendimiento y las características del producto.

5.4.2.1CARACTERÍSTICAS DE SQL SERVER 2000

Entre las principales características de SQL Server figuran:

- **Soporte de transacciones:** Se encarga de coordinar las transacciones que abarcan múltiples administradores de recursos, como bases de datos y las colas de mensajes, además de utilizarla como si fuera un servicio.

- **Estabilidad:** SQL Server es mucho más resistente a la corrupción de los datos. Los datos de SQL Server van a través de múltiples puestos de control y SQL Server recuerda si el proceso se cierra sin previo aviso.

- **Seguridad:** SQL Server permite establecer los permisos NTFS (New Technology File System) en la base de datos de todos los archivos relacionados, incluida los de archivos binarios, para permitir accesos sólo de lectura y ejecución a usuarios autenticados, y completo control a los administradores y la cuenta de servicio de SQL Server.

- **Soporta procedimientos almacenados:** SQL Server permite encapsular la lógica necesaria para realizar una tarea rutinaria en un procedimiento almacenado, para de esta manera diseñar, codificar y probar la tarea una sola vez.

- **Entorno gráfico de administración:** SQL Server incluye también un potente entorno gráfico de administración, que permite el uso de comandos DDL (Data Definition Language) y DML (Data Manipulation Language) gráficamente.

SQL Server usa la arquitectura Cliente/Servidor para separar la carga de trabajo en tareas que corran en computadoras tipo Servidor y tareas que corran en computadoras tipo Cliente:

El Cliente es responsable de la parte lógica y de presentar la información al usuario. Generalmente, el cliente corre en una o más computadoras Cliente, aunque también puede correr en una computadora Servidor con SQL Server.

SQL Server administra Bases de Datos y distribuye los recursos disponibles del servidor (tales como memoria, operaciones de disco, etc.) entre las múltiples peticiones. [Elmoo, 2000]

5.4.3 ARQUITECTURA CLIENTE/SERVIDOR

Con respecto a la definición de arquitectura cliente/servidor se encuentran las siguientes definiciones:

- Cualquier combinación de sistemas que pueden colaborar entre sí, para dar a los usuarios toda la información que ellos necesiten sin que tengan que saber donde está ubicada.

- Es una arquitectura de procesamientos cooperativo donde uno de los componentes pide servicios a otro.

- Es un procesamiento de datos de índole colaborativo entre dos o más computadoras conectadas a una red.

- El término cliente/servidor es originalmente aplicado a la arquitectura de software que describe el procesamiento entre dos o más programas: una aplicación y un servicio soportante.

- IBM define al modelo Cliente/Servidor como la tecnología que proporciona al usuario final el acceso transparente a las aplicaciones, datos, servicios de cómputo o cualquier otro recurso del grupo de trabajo, en múltiples plataformas. El modelo soporta un medio ambiente distribuido en el cual los requerimientos de servicio hechos por estaciones de trabajo inteligentes o "clientes", resultan en un trabajo realizado por otros computadores llamados "servidores".

- "Es un modelo para construir sistemas de información, que se sustenta en la idea de repartir el tratamiento de la información y los datos por todo el sistema informático, permitiendo mejorar el rendimiento del sistema global de información".

5.4.3.1 ELEMENTOS PRINCIPALES DE LA ARQUITECTURA CLIENTE/SERVIDOR

"Los elementos principales de la arquitectura cliente servidor son justamente el elemento llamado cliente y el otro elemento llamado servidor". Por ejemplo dentro de un ambiente multimedia, el elemento cliente sería el dispositivo que puede observar el vídeo, cuadros y texto, o reproduce el audio distribuido por el elemento servidor. Por otro lado el cliente también puede ser una computadora personal o una televisión inteligente que posea la capacidad de entender datos digitales. Dentro de este caso el elemento servidor es el depositario del vídeo digital, audio, fotografías digitales y texto y los distribuye bajo demanda de ser una máquina que

cuenta con la capacidad de almacenar los datos y ejecutar todo el software que brindan al cliente.

5.4.3.2QUE ES UN CLIENTE

Es el que inicia un requerimiento de servicio. El requerimiento inicial puede convertirse en múltiples requerimientos de trabajo a través de redes LAN's (Local Area Network) o WAN's (Wide Area Network). La ubicación de los datos o de las aplicaciones es totalmente transparente para el cliente.

5.4.3.3QUE ES UN SERVIDOR

Es cualquier recurso de cómputo dedicado a responder a los requerimientos del cliente. Los servidores pueden estar conectados a los clientes a través de redes LAN's (Local Area Network) o WAN's (Wide Area Network), para proveer múltiples servicios tales como impresión, acceso a bases de datos, fax, procesamiento de imágenes, etc. [12]

5.4.3.4CARACTERÍSTICAS DEL MODELO CLIENTE/SERVIDOR

El modelo CLIENTE/SERVIDOR tiene las siguientes características:

1. El Cliente y el Servidor pueden actuar como una sola entidad y también pueden actuar como entidades separadas, realizando actividades o tareas independientes.

2. Las funciones de Cliente y Servidor pueden estar en plataformas separadas, o en la misma plataforma.

3. Un servidor da servicio a múltiples clientes en forma concurrente.

4. Cada plataforma puede ser escalable independientemente. Los cambios realizados en las plataformas de los Clientes o de los Servidores, ya sean por actualización o por reemplazo tecnológico, se realizan de una manera transparente para el usuario final.

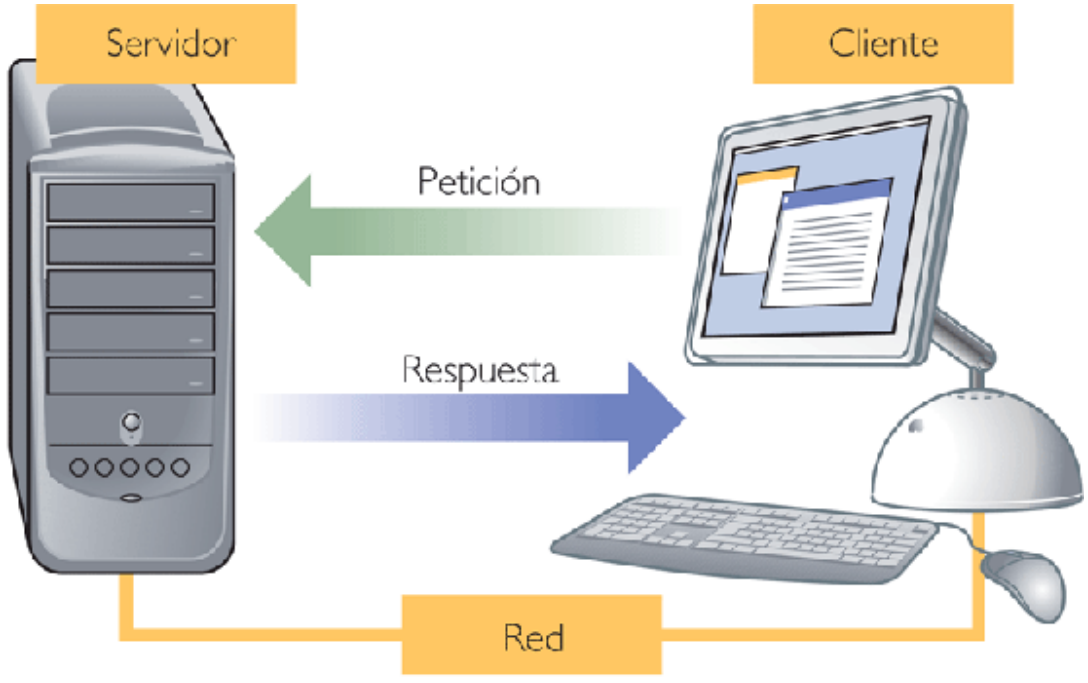
5. La interrelación entre hardware y software están basados en una infraestructura poderosa, de tal forma que el acceso a los recursos de la red no muestra la complejidad de los diferentes tipos de formatos de datos y de los protocolos.

6. Un sistema de servidores realiza múltiples funciones al mismo tiempo que presenta una imagen de un solo sistema a las estaciones Clientes. Esto se logra combinando los recursos de cómputo que se encuentran físicamente separados en un solo sistema lógico, proporcionando de esta manera el servicio más efectivo para el usuario final.

7. Designa un modelo de construcción de sistemas informáticos de carácter distribuido.

Su representación típica es un centro de trabajo (PC), en donde el usuario dispone de sus propias aplicaciones de oficina y sus propias bases de datos, sin dependencia directa del sistema central de información de la organización, al tiempo que pueden acceder a los recursos de este servidor y otros sistemas de la organización. [12]

Representación gráfica simplificada del Modelo Cliente/Servidor



VI. DISEÑO METODOLÓGICO

En la construcción y desarrollo de proyectos se aplican métodos y técnicas para resolver los problemas, la informática aporta herramientas y procedimientos sobre los que se apoya la ingeniería de software.

La Ingeniería del Software es la aplicación práctica del conocimiento científico en el diseño y construcción de programas de computadoras y la documentación asociada requerida para desarrollarlos, operarlos y mantenerlos. Se conoce también como desarrollo de software o producción de software. [Pressman, 2002]

La Ingeniería del Software está compuesta de tres partes:

- **Herramientas:** Son los ambientes de apoyo necesario para automatizar las prácticas de Ingeniería de Software.

- **Métodos:** Son las maneras que se efectúan las tareas de Ingeniería de Software o las actividades del ciclo de vida.

- **Procedimientos:** Definen la secuencia en la que se aplican los métodos, las entregas que se requieren y los controles que ayuden a asegurar la calidad y coordinar los cambios y las guías que facilitan a los gestores de software establecer el desarrollo. Facilitan un desarrollo racional y oportuno del software de computadoras. [Ibarra, 2007]

6.1 MATERIALES

a. Hardware: En nuestro proyecto, a nivel de hardware emplearemos las siguientes herramientas:

- 1 PC con las siguientes características:
  - Memoria RAM de 3 GB
  - 190 GB de Disco Duro
  - Procesador AMD Turion Dual-Core RM-70 de 2,0 GHz

b. Software

- Microsoft Word y Microsoft Office Visio 2007
- Easy Case Professional Versión 4.21
- HelpMaker 7.4.4.0
- Windows XP Professional SP2(Sistema operativo)
- SQL Server 2000 Personal Edition (Sistema Gestor de Base de Datos)
- Microsoft Visual C# (Lenguaje de Programación)
- Crystal Reports for Visual Studio 2005

6.2 MÉTODOS

Un ciclo de vida para un proyecto se compone de fases sucesivas compuestas por tareas planificables. Según el modelo de ciclo de vida, la sucesión de fases puede ampliarse con bucles de realimentación, de manera que lo que conceptualmente se considera una misma fase se pueda ejecutar más de una vez a lo largo de un proyecto, recibiendo en cada pasada de ejecución aportaciones de los resultados intermedios que se van produciendo.

Para un adecuado control de la progresión de las fases de un proyecto se hace necesario especificar con suficiente precisión los resultados evaluables, o sea, productos intermedios que deben resultar de las tareas incluidas en cada fase.

6.3 DESCRIPCIÓN DEL CICLO DE VIDA

El modelo en espiral fue propuesto originalmente por Boehm, es un modelo de proceso de software evolutivo que conjuga la naturaleza iterativa de construcción de prototipos con los aspectos controlados y sistemáticos del modelo lineal secuencial. Proporciona el potencial para el desarrollo rápido de versiones incrementales del software.

En este modelo, el software se desarrolla en una serie de versiones incrementales. Durante las primeras iteraciones, la versión incremental podría ser un modelo en papel o un prototipo. Durante las últimas iteraciones, se producen versiones cada vez más completas del sistema diseñado.

El modelo en espiral se divide en un número de actividades de marco de trabajo, también llamadas “regiones de tareas”. Generalmente, existen entre tres y seis regiones de tareas:

- **Comunicación con el cliente:** Las tareas requeridas para establecer comunicación entre el desarrollador y el cliente.

- **Planificación:** Las tareas requeridas para definir recursos, el tiempo y otra información relacionadas con el proyecto.

- **Análisis de Riegos:** Las tareas requeridas para construir una o más representaciones de la aplicación.

- **Construcción y acción:** Las tareas requeridas para construir probar instalar y proporcionar soporte al usuario (por ejemplo: documentación y práctica)

- **Evaluación del Cliente:** Las tareas requeridas para obtener la reacción del cliente según la evaluación de las representaciones del software creadas durante la etapa de ingeniería e implementada durante la etapa de instalación.

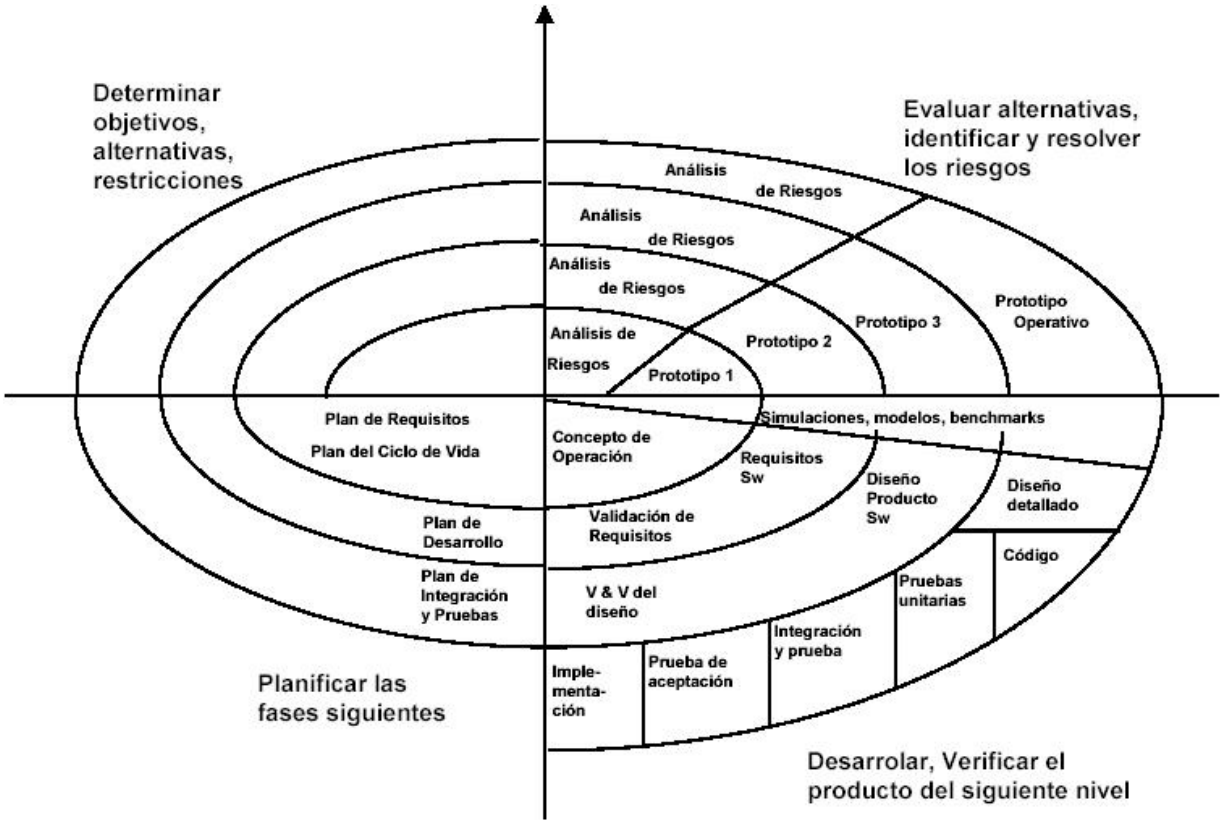
Cada una de las regiones está compuesta por un conjunto de tareas de trabajo, llamado conjunto de tareas, que se adaptan a las características del proyecto que va a emprenderse. Para proyectos pequeños, el número de tareas de trabajo y su formalidad es bajo. Para proyecto mayores y más críticos cada región de tareas contiene tareas de trabajo que se definen para lograr un nivel más alto de formalidad. En todos los casos, se aplican las actividades de protección (por ejemplo: gestión de configuración del software y garantía de calidad del software).

Cuando empieza este proceso evolutivo, el equipo de ingeniería del software gira alrededor de la espiral en la dirección de las agujas del reloj, comenzando por el centro. El primer circuito de la espiral puede producir el desarrollo de una especificación de productos; los pasos siguientes en la espiral se podrían utilizar para desarrollar un prototipo y progresivamente versiones más sofisticadas del software. Cada paso por la región de planificación produce ajustes en el plan del proyecto. El coste y la planificación se ajustan con la realimentación ante la evaluación del cliente. Además, el gestor del proyecto ajusta el número planificado de iteraciones requeridas para completar el software.

A diferencia del modelo de proceso clásico que termina cuando se entrega el software, el modelo en espiral puede adaptarse y aplicarse a lo largo de la vida del software de computadora.

El modelo en espiral es un enfoque realista del desarrollo de sistemas y de software a gran escala. Como el software evoluciona, a medida que progresa el proceso el desarrollador y el cliente comprenden y reaccionan mejor ante riesgos en cada uno de los niveles evolutivos. El modelo en espiral utiliza la construcción de prototipos como mecanismo de reducción de riesgos, pero, lo que es más importante, permite a quien lo desarrolla aplicar el enfoque de construcción de prototipos en cualquier etapa de evolución del producto. Mantiene el enfoque sistemático de los pasos sugeridos por el ciclo de vida clásico, pero lo incorpora al marco de trabajo iterativo que refleja de forma más realista el mundo real. [Pressman, 2002]

Representación gráfica del Modelo en Espiral



VII. ANÁLISIS

ESPECIFICACIÓN DE REQUISITOS SOFTWARE (ERS)

1 Introducción

1.1 Propósito

Definición del conjunto de especificaciones de requisitos software que debe cumplir la aplicación para automatizar los procesos del video-rental Tauro Video que permitirá controlar y gestionar las actividades que se realizan en dicho negocio. El documento se dirige al dueño del video-rental y a los usuarios finales que deberán estudiarlo para su aprobación o desacuerdo.

1.2 Alcance

El nombre con el que se conocerá a esta aplicación será: "***Sistema de Información para el Control y Gestión del Video-Rental Tauro Video".***

El sistema será una herramienta que facilitará el trabajo de los usuarios finales al agilizar los procesos que se llevan a cabo en el video-rental de tal manera que sea más fácil su administración, control y gestión. Permitirá llevar el control de las películas alquiladas, de los eventos que están en agenda para filmarse y llevará el control de los ingresos percibidos por el negocio.

El producto realizará las siguientes funciones:

1. Creación de usuario del sistema.
2. Validar usuario.
3. Borrar usuario.
4. Modificar contraseña de usuario.
5. Conexión con el servidor.
6. Ingresar formato de películas.
7. Borrar formato de películas.
8. Modificar datos de formato de películas.

9. Ingresar género de películas.
10. Borrar género de películas.
11. Ingresar evento.
12. Borrar evento.
13. Registrar cliente.
14. Borrar cliente.
15. Modificar datos de cliente.
16. Buscar cliente.
17. Buscar clientes con películas alquiladas.
18. Registrar película.
19. Borrar película.
20. Modificar datos de películas.
21. Buscar película.
22. Registrar alquiler.
23. Registrar devolución.
24. Registrar filmación.
25. Cancelar filmación.
26. Modificar datos de filmación.
27. Generar informe de alquileres.
28. Generar informe de clientes con películas alquiladas.
29. Generar informe devoluciones.
30. Generar informe de filmaciones.
31. Generar informe de ingresos.
32. Generar informe de clientes morosos.
33. Generar informe de películas.
34. Generar informe de películas más alquiladas.
35. Respaldar base de datos.
36. Restaurar base de datos.

1.3 Definiciones, acrónimos y abreviaturas

1. **Adelanto:** Suma de dinero que el CLIENTE abona por adelantado cuando contrata una FILMACIÓN. En lo sucesivo ADELANTO.
2. **Cliente:** Persona que realiza un alquiler de películas o que contrata una filmación para un evento social en el video-rental. En lo sucesivo será conocido como CLIENTE.

3. **Cédula:** Documento con el cual, el CLIENTE se identificará para alquilar una película o para contratar una FILMACIÓN. En lo sucesivo CÉDULA.
4. **Disponibilidad:** Si existen copias de una película disponibles para realizar el alquiler de películas. En lo sucesivo DISPONIBILIDAD.
5. **Evento:** Representa un evento social (boda, quince años, etc.). En lo sucesivo será conocido como EVENTO.
6. **Filmación:** Servicio que ofrece el video-rental Tauro Video para filmar cualquier evento social. El resto del documento lo llamará FILMACIÓN.
7. **Formato:** Nombre de un formato de película (DVD, VCD, etc.). El resto del documento lo llamará FORMATO.
8. **Género:** Representa el género al que pertenece una PELÍCULA (Acción, Comedia, etc.). En lo sucesivo será conocido como GÉNERO.
9. **Película:** Representa una película en un determinado FORMATO y que pertenece a un determinado GENERO. En lo sucesivo será conocido como PELÍCULA.
10. **Recargo:** Suma de dinero adicional que se le cobra al CLIENTE si este devuelve la(s) película(s) después de la fecha de devolución. El resto del documento lo llamará RECARGO.
11. **Saldo:** Dinero que el CLIENTE adeuda al video-rental por una FILMACIÓN. En lo sucesivo SALDO.
12. **Usuario:** Representa una persona que hace uso del sistema. Habrán dos niveles de usuarios, un usuario Administrador que podrá acceder a todas las funciones del sistema y los usuarios Operadores que tendrán un acceso limitado. En lo sucesivo USUARIO.

1.4 Referencias

- Entrevistas realizadas al administrador del video-rental Tauro Video.
- Plan docente de Análisis y Diseño de Sistemas de Información. Suministrado por el Departamento de Computación de la UNAN-León
- MSc. Martín Ibarra. Apuntes de Ingeniería del software.

1.5 Visión General

Primeramente se realizará una Descripción General del producto que se desea desarrollar, para luego estudiar cada uno de los Requisitos Específicos.

2 Descripción general

2.1 Relaciones del producto

La aplicación no interactuará con ninguna otra aplicación existente.

El equipo servidor en el que se implantará la base de datos es:

- Procesador Intel Core Duo E6400 de 2,0 GHz.
- 2 GB RAM.
- 500 GB disco duro.

Las estaciones cliente en la que se implantará el producto final es:

- Procesador Intel® Pentium® 4 CPU de 2,4 GHz.
- 1 GB RAM.
- 80 GB Disco Duro.

El producto tendrá la capacidad de trabajar dentro de una red de área local. En la cual las estaciones cliente podrán acceder a la base de datos en el equipo servidor. El alcance de nuestro proyecto no incluye la configuración de la red.

2.2 Funcionalidad del producto

El producto final debe contener todas las tareas que realiza manualmente el personal del video-rental Tauro Video de forma diaria. Estas son:

1. Cuando un CLIENTE llega a alquilar películas, el USUARIO deberá buscar al CLIENTE y seleccionar la(s) película(s) para que el alquiler sea registrado en la base de datos.

2. Cuando un CLIENTE llega a devolver películas, el USUARIO deberá buscar al CLIENTE por nombre o por CEDULA, se buscarán las películas alquiladas, se verificará si el CLIENTE tiene algún RECARGO, en cuyo caso se le notificará al USUARIO para proceder al cobro y luego de esto se registrará la devolución.

3. Para registrar una FILMACION el USUARIO buscará al CLIENTE, si este no se encuentra, se registrará, además de todos los datos necesarios para la filmación.

4. El USUARIO podrá disponer de forma diaria de los siguientes listados:
    a. Listado de CLIENTES con películas alquiladas.
    b. Listado de CLIENTES que excedieron la fecha de devolución.
    c. Listado de alquileres.
    d. Listado de devoluciones.
    e. Listado de FILMACIONES programadas.
    f. Listado de películas.
    g. Listado de las películas más alquiladas.

5. La aplicación generará un informe de ingresos en cualquier momento que el USUARIO Administrador lo requiera y para cualquier intervalo de tiempo que sea especificado.

2.3 Características de los usuarios

Los usuarios finales de la aplicación serán personas cuya experiencia informática es media. Sin embargo, se incluirá un manual de ayuda en el producto final.

2.4 Restricciones

- Se deberán seguir los estándares de la programación orientada a objetos.
- Se aplicará el modelo Cliente/Servidor en el desarrollo de la aplicación, tomando en cuenta las características de su arquitectura. En donde los procesos clientes podrán acceder a la base de datos en el servidor, interconectados entre sí a través de una red LAN Ethernet o Wireless.

3 Requisitos específicos

3.1 Requisitos funcionales

3.1.1 Crear USUARIO del sistema

3.1.1.1 Especificación

3.1.1.1.1 Introducción

Este proceso deberá realizar la captura de los datos de un USUARIO que vaya a hacer uso de la aplicación.

3.1.1.1.2 Entradas

Por pantalla:

- Nombre de USUARIO
- Contraseña
- Confirmar contraseña

Datos proporcionados por el sistema:

- Tipo de acceso

3.1.1.1.3 Proceso

Se mostrará por pantalla el formulario para introducir los datos del nuevo USUARIO.

Los datos necesarios a introducir serán:

- **Nombre de USUARIO:** Dato obligatorio. Nombre con que el nuevo USUARIO será registrado en la base de datos.
- **Contraseña:** Dato obligatorio. El acceso del USUARIO estará protegido mediante este campo.
- **Confirmar contraseña:** Dato obligatorio. Para seguridad, el USUARIO deberá escribir su contraseña nuevamente en este campo.

Los datos suministrados por el sistema:

- **Tipo de Acceso:** Dato Obligatorio. Se deberá seleccionar el tipo de acceso que tendrá el USUARIO, que puede ser ya sea Operador o Administrador.

Los campos contraseña y confirmar contraseña deberán ser exactamente iguales, de lo contrario se indicará un mensaje de error. Se verificará que el USUARIO no exista en la base de datos, en cuyo caso se indicará un mensaje de error.

3.1.1.1.4 Salidas

Cuando los datos ingresados manualmente se hayan validado y verificado, se almacenará el registro en la base de datos.

3.1.2 Validar USUARIO

3.1.2.1 Especificación

3.1.2.1.1 Introducción

Este proceso permitirá el acceso de un USUARIO (Administrador/Operador) a la aplicación.

3.1.2.1.2 Entradas

Por pantalla:

- Nombre de USUARIO
- Contraseña

Datos proporcionados por el sistema:

- El sistema no proporcionará datos en este proceso.

3.1.2.1.3 Proceso

Se mostrará por pantalla el formulario para ingresar el nombre y la contraseña.

Los datos necesarios a introducir serán:

- **Nombre de USUARIO:** Dato obligatorio. Nombre con el que el USUARIO está registrado en la base de datos.
- **Contraseña: Dato obligatorio.** Clave de acceso de USUARIO.

Se comprobará que los datos obligatorios sean introducidos. Con estos datos se buscará el USUARIO(Administrador/Operador) en la base de datos, si no se encuentra, se indica con un mensaje y si excede tres intentos la aplicación se cerrará.

3.1.2.1.4 Salidas

Con los datos introducidos correctamente el USUARIO podrá acceder a la aplicación.

3.1.3 Borrar USUARIO

3.1.3.1 Especificación

3.1.3.1.1 Introducción

Este proceso le permitirá al administrador, eliminar un USUARIO del sistema.

3.1.3.1.2 Entradas

Por pantalla:

- No se ingresarán datos para este proceso.

Datos proporcionados por el sistema:

- Nombre de USUARIO
- Tipo de acceso

3.1.3.1.3 Proceso

Se mostrará por pantalla el formulario que contiene una lista con todos los USUARIOS registrados. El Administrador seleccionará el usuario que desea borrar y pulsará el botón correspondiente. Se pedirá confirmación de la acción antes de proceder.

3.1.3.1.4 Salidas

El proceso anterior borrará el registro de USUARIO de la base de datos.

3.1.4 Modificar contraseña de USUARIO

3.1.4.1 Especificación

3.1.4.1.1 Introducción

Este proceso deberá realizar la modificación de la clave de acceso de un USUARIO registrado en la base de datos.

3.1.4.1.2 Entradas

Por pantalla:

- Nombre de USUARIO
- Contraseña actual
- Contraseña nueva

Datos proporcionados por el sistema:

- El sistema no proporcionará datos en este proceso.

3.1.4.1.3 Proceso

Se mostrará por pantalla el formulario para introducir los datos del USUARIO.

Los datos necesarios a introducir serán:

- **Nombre de USUARIO:** Dato obligatorio. Se utilizará para buscar al USUARIO.
- **Contraseña actual:** Dato obligatorio. Se utilizará para buscar al USUARIO.
- **Contraseña nueva:** Dato obligatorio. Será la nueva clave de acceso.

Se comprobará que todos los datos sean introducidos. De lo contrario se indicará un mensaje de error. Con el nombre y contraseña actual, se buscará el registro en la base de datos para reemplazar la contraseña.

3.1.4.1.4 Salidas

Con los datos ingresados, se modificará el registro de USUARIO en la base de datos.

3.1.5 Conexión con el servidor

3.1.5.1 Especificación

3.1.5.1.1 Introducción

Este proceso deberá realizar la conexión con la base de datos del servidor SQL Server.

3.1.5.1.2 Entradas

Por pantalla:

- No se ingresarán datos en este proceso.

Datos proporcionados por el sistema:

- Nombre de instancias SQL Server en la red.

3.1.5.1.3 Proceso

Se mostrará por pantalla el formulario en el cual el USUARIO(Administrador/Operador) seleccionará el servidor SQL al que se desea conectar. Con el servidor seleccionado se pulsará el botón conectar.

3.1.5.1.4 Salidas

Se procederá a introducir el nombre de USUARIO y la contraseña si la operación tuvo éxito o un mensaje de error en caso contrario.

3.1.6 Ingresar FORMATO

3.1.6.1 Especificación

3.1.6.1.1 Introducción

Este proceso deberá realizar la captura de los datos de un FORMATO.

3.1.6.1.2 Entradas

Por pantalla:

- Nombre de FORMATO
- Precio
- Precio de estreno
- Multa por retraso

Datos proporcionados por el sistema:

- El sistema no proporcionará datos para este proceso

3.1.6.1.3 Proceso

Se mostrará por pantalla el formulario Control Video-rental. En la pestaña formatos el Administrador del sistema pulsará el botón nuevo para ingresar los datos.

Los datos necesarios a introducir serán:

- **Nombre de FORMATO:** Dato obligatorio. Nombre con el que el nuevo FORMATO será registrado en la base de datos.
- **Precio: Dato obligatorio.** Precio asignado a un FORMATO.
- **Precio de estreno:** Dato obligatorio. Precio asignado a un FORMATO en el periodo de estreno de la PELICULA.
- **Multa por retraso:** Dato obligatorio. Cantidad de dinero asignada a un FORMATO en concepto de multa por día de retraso.

Se verificará que todos los campos sean introducidos y luego se pulsará el botón guardar.

3.1.6.1.4 Salidas

Con todos los datos mencionados se almacenará el registro en la base de datos.

3.1.7 Borrar FORMATO

3.1.7.1 Especificación

3.1.7.1.1 Introducción

Este proceso eliminará un registro de FORMATO en la base de datos.

3.1.7.1.2 Entradas

Por pantalla:

- No se ingresarán datos para este proceso.

Datos proporcionados por el sistema:

- El sistema no proporcionará datos para este proceso

3.1.7.1.3 Proceso

Se mostrará por pantalla el formulario Control Video-rental. En la pestaña formatos el Administrador del sistema seleccionará de una lista el FORMATO. Luego pulsará el botón borrar. Se pedirá la confirmación de la acción.

3.1.7.1.4 Salidas

El registro que corresponde al FORMATO seleccionado será borrado de la base de datos.

3.1.8 Modificar datos de FORMATO

3.1.8.1 Especificación

3.1.8.1.1 Introducción

Este proceso modificará los datos de un FORMATO en la base de datos.

3.1.8.1.2 Entradas

Por pantalla:

- Nombre de FORMATO
- Precio
- Precio de estreno
- Multa por retraso

Datos proporcionados por el sistema:

- El sistema no proporcionará datos para este proceso

3.1.8.1.3 Proceso

Se mostrará por pantalla el formulario Control Video-rental. En la pestaña formatos el Administrador seleccionará el FORMATO. Pulsará el botón editar, para modificar los datos en las cajas de texto.

Los datos necesarios a introducir serán:

- **Nombre de FORMATO:** Dato opcional. Nuevo nombre de FORMATO.

- **Precio:** Dato opcional. Nuevo precio del FORMATO.
- **Precio de estreno:** Dato opcional. Nuevo precio de estreno.
- **Multa por retraso:** Dato opcional. Nueva multa por día de retraso.

Después de haber modificado cualquiera de los datos se pulsará el botón guardar.

3.1.8.1.4 Salidas

Con todos los datos mencionados se actualizará el registro de FORMATO en la base de datos.

3.1.9 Ingresar GÉNERO de PELÍCULA

3.1.9.1 Especificación

3.1.9.1.1 Introducción

Este proceso deberá realizar la captura de los datos de un GÉNERO de PELÍCULA.

3.1.9.1.2 Entradas

Por pantalla:

- Nombre del GÉNERO

Datos proporcionados por el sistema:

- El sistema no proporcionará datos para este proceso.

3.1.9.1.3 Proceso

Se mostrará el formulario Control Video-rental. En la pestaña Géneros el Administrador pulsará el botón nuevo.

Los datos necesarios a introducir serán:

- **Nombre del GÉNERO:** Dato obligatorio.

3.1.9.1.4 Salidas

Con los datos mencionados se almacenará el registro en la base de datos del servidor.

3.1.10 Borrar GÉNERO

3.1.10.1 Especificación

3.1.10.1.1 Introducción

Este proceso eliminará un registro de GÉNERO en la base de datos del servidor.

3.1.10.1.2 Entradas

Por pantalla:

- No se ingresarán datos para este proceso.

Datos proporcionados por el sistema:

- El sistema no proporcionará datos para este proceso.

3.1.10.1.3 Proceso

Se mostrará por pantalla el formulario Control Video-rental. En la pestaña Géneros el Administrador del sistema seleccionará de una lista el GÉNERO. Luego pulsará el botón borrar. Se pedirá la confirmación de la acción.

3.1.10.1.4 Salidas

El registro que corresponde al GÉNERO seleccionado será borrado de la base de datos del servidor.

3.1.11 Ingresar EVENTO

3.1.11.1 Especificación

3.1.11.1.1 Introducción

Este proceso deberá realizar la captura de los datos de un EVENTO.

3.1.11.1.2 Entradas

Por pantalla:

- Nombre del EVENTO.

Datos proporcionados por el sistema:

- El sistema no proporcionará datos para este proceso.

3.1.11.1.3 Proceso

Se mostrará por pantalla el formulario Control Video-rental. En la pestaña Eventos el Administrador pulsará el botón nuevo para ingresar el EVENTO.

Los datos necesarios a introducir serán:

- **Nombre del EVENTO:** Dato obligatorio.

3.1.11.1.4 Salidas

Con los datos mencionados se almacenará el registro en la base de datos.

3.1.12 Borrar EVENTO

3.1.12.1 Especificación

3.1.12.1.1 Introducción

Este proceso eliminará un registro de EVENTO en la base de datos.

3.1.12.1.2 Entradas

Por pantalla:

- No se ingresarán datos para este proceso.

Datos proporcionados por el sistema:

- El sistema no proporcionará datos para este proceso.

3.1.12.1.3 Proceso

Se mostrará por pantalla el formulario Control Video-rental. En la pestaña Eventos el Administrador del sistema seleccionará de una lista el EVENTO. Luego pulsará el botón borrar. Se pedirá la confirmación de la acción.

3.1.12.1.4 Salidas

El registro que corresponde al EVENTO seleccionado será borrado de la base de datos.

3.1.13 Registrar CLIENTE

3.1.13.1 Especificación

3.1.13.1.1 Introducción

Este proceso deberá realizar la captura de los datos de un CLIENTE la primera vez que alquila películas o contrata una FILMACIÓN.

3.1.13.1.2 Entradas

Por pantalla:

- CÉDULA del CLIENTE
- Nombre del CLIENTE
- Dirección del CLIENTE
- Teléfono del CLIENTE
- Edad del CLIENTE

Datos proporcionados por el sistema:

- Sexo del CLIENTE.

3.1.13.1.3 Proceso

Se mostrará por pantalla el formulario para introducir los datos del CLIENTE.

Los datos necesarios a introducir serán:

- **CÉDULA del CLIENTE:** Dato obligatorio.

- **Nombre del CLIENTE:** Dato obligatorio. Se corresponderá con el nombre que aparece en la CÉDULA.
- **Dirección del CLIENTE:** Dato obligatorio. Se corresponderá con la dirección que aparece en la CÉDULA.
- **Teléfono:** Dato opcional.
- **Edad:** Dato obligatorio.

Los datos proporcionados por el sistema serán:

- **Sexo:** Dato obligatorio.

Se comprobará que los datos obligatorios sean introducidos correctamente y hayan sido llenados. De lo contrario se indicará un mensaje de error.

3.1.13.1.4 Salidas

Con todos los datos mencionados se almacenará el registro en la base de datos.

3.1.14 Borrar CLIENTE

3.1.14.1 Especificación

3.1.14.1.1 Introducción

Este proceso deberá borrar los datos de un CLIENTE de la base de datos.

3.1.14.1.2 Entradas

Por pantalla:

- No se proporcionarán entradas en este proceso.

Datos proporcionados por el sistema:

- Los datos del CLIENTE.

3.1.14.1.3 Proceso

Se mostrará el formulario Clientes luego se pulsará el botón Buscar. Se mostrará el formulario Buscar Cliente que visualizará una lista con los datos de todos los clientes. Para borrar un CLIENTE se seleccionará de la lista el registro que desee eliminar, luego se dará click en el botón borrar. Deberá confirmar si desea eliminar dicho registro de la base de datos.

3.1.14.1.4 Salidas

Al realizar este proceso se borrará el registro del CLIENTE en la base de datos.

3.1.15 Modificar datos de CLIENTE

3.1.15.1 Especificación

3.1.15.1.1 Introducción

Este proceso deberá realizar la modificación de los datos de un CLIENTE en la base de datos.

3.1.15.1.2 Entradas

Por pantalla:

- CÉDULA del CLIENTE
- Nombre del CLIENTE
- Dirección del CLIENTE
- Teléfono del CLIENTE
- Edad del CLIENTE

Datos proporcionados por el sistema:

- Sexo del CLIENTE

3.1.15.1.3 Proceso

Se mostrará el formulario Clientes luego se pulsará el botón

Buscar. En el formulario Buscar Cliente se visualizará una lista con los datos de todos los clientes. Para modificar un cliente se seleccionará de la lista, luego se dará click en el botón Editar.

Los datos a modificar serán:

- **CÉDULA del CLIENTE:** Dato opcional.
- **Nombre del CLIENTE:** Dato opcional.
- **Dirección del CLIENTE:** Dato opcional.
- **Teléfono:** Dato opcional.
- **Edad:** Dato opcional.

Los datos proporcionados por el sistema serán:

- **Sexo:** Dato opcional.

3.1.15.1.4 Salidas

El registro será actualizado en la base de datos del servidor con las modificaciones hechas por el USUARIO(Administrador/Operador).

3.1.16 Buscar CLIENTE

3.1.16.1 Especificación

3.1.16.1.1 Introducción

Este proceso buscará un CLIENTE en la base de datos.

3.1.16.1.2 Entradas

Por pantalla: Opciones de búsqueda

- CÉDULA del CLIENTE
- Nombre del CLIENTE

Datos proporcionados por el sistema:

- El sistema no proporcionará datos para este proceso.

3.1.16.1.3 Proceso

Se mostrará por pantalla el formulario Buscar Cliente con la lista de los clientes registrados y las opciones de búsqueda.

Las opciones de búsqueda serán:

- **CÉDULA del CLIENTE:** Dato opcional.
- **Nombre del CLIENTE:** Dato opcional.

Se seleccionará el CLIENTE de la lista.

3.1.16.1.4 Salidas

Este proceso devolverá los datos del CLIENTE al formulario Clientes.

3.1.17 Buscar CLIENTE con películas alquiladas

3.1.17.1 Especificación

3.1.17.1.1 Introducción

Este proceso deberá realizar la búsqueda de clientes con películas alquiladas en el video-rental.

3.1.17.1.2 Entradas

Por pantalla: Opciones de búsqueda

- CÉDULA del CLIENTE
- Nombre del CLIENTE

Datos proporcionados por el sistema:

- El sistema no proporcionará datos en este proceso.

3.1.17.1.3 Proceso

Se mostrará el formulario Buscar Clientes con Películas Alquiladas, que visualizará una lista con los datos de los clientes que tienen alquileres realizados.

Las opciones de búsqueda serán:

- **CÉDULA del CLIENTE:** Dato opcional.
- **Nombre del CLIENTE:** Dato opcional.

Con cualquiera de los datos de búsqueda se irán mostrando las coincidencias en la lista. El USUARIO(Administrador/Operador) seleccionará al CLIENTE.

3.1.17.1.4 Salidas

Este proceso devolverá los datos del CLIENTE al formulario Devolución de Películas.

3.1.18 Registrar PELÍCULA

3.1.18.1 Especificación

3.1.18.1.1 Introducción

Este proceso deberá realizar la captura los datos de una nueva PELÍCULA adquirida y guardar automáticamente las existencias y disponibilidad de dicha PELÍCULA en cada FORMATO en la base de datos del servidor. Sólo el Administrador del sistema podrá tener acceso a este proceso.

3.1.18.1.2 Entradas

Por pantalla: datos para registrar la nueva PELÍCULA

- Título de la PELÍCULA
- Actor principal
- No. Copias por FORMATO
- Estreno

Datos proporcionados por el sistema:

- Código de PELÍCULA
- Formatos
- GÉNERO
- Carátula

3.1.18.1.3 Proceso

Se mostrará el formulario Películas. Se pulsará el botón Nuevo para introducir datos. Los campos obligatorios deberán ser llenados y las entradas por pantalla ingresadas correctamente y luego se pulsará el botón guardar.

Los datos necesarios a introducir serán:

- **Título de la PELÍCULA:** Dato obligatorio. Nombre que identifica a la PELÍCULA.
- **Actor principal:** Dato obligatorio. Nombre del actor principal.
- **No. de copias:** Dato obligatorio. Cantidad de existencias para cada FORMATO.

Los datos proporcionados por el sistema serán:

- **Código de PELÍCULA:** Dato obligatorio.
- **GÉNERO:** Dato obligatorio.
- **Formatos:** Dato Obligatorio. Se indicará a la aplicación el FORMATO en el que la PELÍCULA está disponible (DVD, VCD, etc.).
- **Carátula:** Dato opcional.

3.1.18.1.4 Salidas

Con todos los datos mencionados se almacenará un nuevo registro de PELÍCULA en la base de datos.

3.1.19 Borrar PELÍCULA

3.1.19.1 Especificación

3.1.19.1.1 Introducción

Este proceso deberá borrar los datos de una PELÍCULA de la base de datos. Sólo el Administrador del sistema podrá tener acceso a este proceso.

3.1.19.1.2 Entradas

Por pantalla:

- No se proporcionarán entradas en este proceso.

Datos proporcionados por el sistema:

- Los datos de PELÍCULA.

3.1.19.1.3 Proceso

Se mostrará el formulario Películas luego se pulsará el botón Buscar. En el formulario Buscar Película se visualizará una lista con los datos de todas las películas en existencia. Para borrar una PELÍCULA se seleccionará de la lista, luego se dará click en el botón borrar. Deberá confirmar si desea eliminar dicho registro de la base de datos.

3.1.19.1.4 Salidas

Al realizar este proceso se borrará el registro de la PELÍCULA en la base de datos.

3.1.20 Modificar datos de PELÍCULA

3.1.20.1 Especificación

3.1.20.1.1 Introducción

Este proceso deberá realizar la modificación de los datos de PELÍCULA en la base de datos del servidor. Sólo el Administrador del sistema podrá tener acceso a este proceso.

3.1.20.1.2 Entradas

Por pantalla: datos para modificar PELÍCULA

- Título de la PELÍCULA
- Actor principal
- No. Copias por FORMATO
- Estreno

Datos proporcionados por el sistema:

- Código de PELÍCULA
- Formatos
- GÉNERO
- Carátula

3.1.20.1.3 Proceso

Se mostrará el formulario Películas, luego se pulsará el botón Buscar. En el formulario Buscar Película se visualizará una lista con los datos de todas las películas en existencia. Después de haber elegido la PELÍCULA se pulsará el botón Editar. Luego se dará click en el botón Guardar para realizar las modificaciones.

3.1.20.1.4 Salidas

Al realizar este proceso se guardará el registro modificado de la PELÍCULA en la base de datos.

3.1.21 Buscar PELÍCULA

3.1.21.1 Especificación

3.1.21.1.1 Introducción

Este proceso deberá realizar la búsqueda de películas por código o nombre, en la base de datos del sistema. Con cualquiera de estos datos se buscará y se visualizará la información.

3.1.21.1.2 Entradas

Por pantalla: opciones de búsqueda.

- Código de PELÍCULA
- Nombre de PELÍCULA

Datos proporcionados por el sistema:

- El sistema no proporcionará datos para este proceso.

3.1.21.1.3 Proceso

Se mostrará el formulario Buscar Película en el cual el USUARIO(Administrador/Operador) podrá introducir ya sea el código o el nombre de la PELÍCULA.

Los datos necesarios a introducir serán:

- **Código de PELÍCULA:** Dato opcional. Se realizará la búsqueda en la base de datos tomando como parámetro el código introducido.
- **Nombre de PELÍCULA:** Dato opcional. Se realizará la búsqueda en la base de datos tomando como parámetro el nombre.

Una vez que el USUARIO indica el criterio que se usará para buscar la PELÍCULA, se ejecutará una consulta a la base de datos para mostrar la información solicitada.

3.1.21.1.4 Salidas

Al final del proceso se devolverá los datos de PELÍCULA al formulario Películas o Alquilar Película.

3.1.22 Registrar Alquiler de Películas

3.1.22.1 Especificación

3.1.22.1.1 Introducción

Este proceso deberá realizar la captura de los datos relacionados con el alquiler de películas y guardar un registro de alquiler en la base de datos.

3.1.22.1.2 Entradas

Por pantalla:

- No se ingresarán datos para este proceso.

Datos proporcionados por el sistema:

Referente al CLIENTE:

- CÉDULA
- Nombre
- Dirección

Referente a la PELÍCULA:

- Código
- Título
- FORMATO
- Costo

Referente al alquiler:

- Fecha de alquiler
- Fecha de devolución

3.1.22.1.3 Proceso

Se mostrará el formulario Alquiler de Películas. Al pulsar un botón se abrirá el diálogo Buscar Cliente donde se seleccionará al CLIENTE. Se presentará una lista vacía donde el USUARIO(Administrador/operador) podrá agregar películas, pulsando el botón de agregar, el cual abrirá el dialogo Buscar Película.

Después de seleccionar al CLIENTE se verificará que este no tenga películas alquiladas pendientes por devolver.

Luego de agregar a la lista las películas que desee el cliente, se calculará el importe del alquiler automáticamente, tomando en cuenta las promociones que tiene el video-rental. La fecha de devolución será calculada por el sistema tomando en cuenta la cantidad y estrenos de películas. Se pulsará el botón Alquilar para guardar la información.

3.1.22.1.4 Salidas

Con los datos mencionados se almacenarán los registros de las películas alquiladas en la base de datos.

3.1.23 Devolver películas

3.1.23.1 Especificación

3.1.23.1.1 Introducción

Este proceso deberá registrar la devolución de las películas alquiladas por los clientes.

3.1.23.1.2 Entradas

Por pantalla:

- No se ingresarán datos en este proceso

Datos proporcionados por el sistema:

Referente al CLIENTE:

- CÉDULA
- Nombre

Referente a la PELÍCULA:

- Código
- Titulo
- Fecha de alquiler
- Fecha de devolución
- Multa por retraso

3.1.23.1.3 Proceso

Se mostrará por pantalla el formulario Devolver Películas para registrar la devolución. Al dar click al botón Buscar se llamará al proceso Buscar Clientes con películas alquiladas y con los datos devueltos, se presentará una lista con las películas alquiladas por el CLIENTE.

El USUARIO(Administrador/Operador) marcará una casilla por cada PELÍCULA a devolver. Si la fecha en la que el CLIENTE devuelve las películas, excede a la fecha de devolución, se le cobrará un RECARGO al CLIENTE.

3.1.23.1.4 Salidas

Con todos los datos mencionados se realizará la actualización de la base de datos del sistema, se guardarán los registros de devolución y se aumentará la disponibilidad de películas.

3.1.24 Registrar FILMACIÓN

3.1.24.1 Especificación

3.1.24.1.1 Introducción

Este proceso deberá realizar la captura de los datos de un contrato de FILMACIÓN y realizar automáticamente la reservación en la agenda de trabajo.

3.1.24.1.2 Entradas

Por pantalla: datos para codificar una FILMACIÓN.

- Fecha del EVENTO
- Dirección del EVENTO
- Hora del EVENTO
- Precio de FILMACIÓN
- Adelanto de dinero

Datos proporcionados por el sistema:

- Fecha de contrato
- Número de CEDULA del CLIENTE
- Nombre del CLIENTE
- Teléfono del CLIENTE
- Tipo de EVENTO
- SALDO

3.1.24.1.3 Proceso

Se mostrará el formulario para realizar la introducción de los datos de una FILMACION. Se deberá dar click al botón nuevo y seleccionar la fecha. Al pulsar el botón Buscar se presentará el diálogo Buscar Cliente.

Los datos necesarios a introducir serán:

- **Fecha del EVENTO:** Dato obligatorio.
- **Dirección del EVENTO:** Dato obligatorio.

- **Hora del EVENTO:** Dato obligatorio.
- **Precio de la FILMACIÓN:** Dato obligatorio.
- **Adelanto de dinero:** Dato opcional.

Se verificará que los datos obligatorios sean llenados e introducidos correctamente. El sistema calculará de forma automática el SALDO en caso que el CLEINTE de ADELANTO. Al dar click al botón Guardar almacenará el registro en la base de datos.

3.1.24.1.4 Salidas

Con todos los datos mencionados se almacenará la FILMACIÓN en la base de datos del servidor.

3.1.25 Cancelar FILMACIÓN

3.1.25.1 Especificación

3.1.25.1.1 Introducción

Este proceso deberá realizar la eliminación de un contrato de FILMACIÓN de la base de datos.

3.1.25.1.2 Entradas

Por pantalla:

- No se ingresarán datos para este proceso.

Datos proporcionados por el sistema:

Referente al CLIENTE:

- CÉDULA

Referente a la FILMACIÓN:

- Fecha del EVENTO
- Tipo de EVENTO

3.1.25.1.3 Proceso

Se mostrará el formulario Filmación que visualizará el calendario de trabajo. El USUARIO(Administrador/Operador) seleccionará del calendario la fecha y se mostrará en una lista las filmaciones programadas para ese día. Se deberá seleccionar la FILMACIÓN a eliminar y luego pulsar el botón Borrar. Se pedirá la confirmación antes de proceder a borrar el registro de la base de datos.

3.1.25.1.4 Salidas

El registro de FILMACIÓN especificado por el cliente será eliminado de la base de datos.

3.1.26 Modificar datos de FILMACIÓN

3.1.26.1 Especificación

3.1.26.1.1 Introducción

Este proceso deberá realizar la modificación de un contrato de FILMACIÓN de la base de datos.

3.1.26.1.2 Entradas

Por pantalla: datos para modificar la FILMACIÓN

- Fecha del EVENTO
- Dirección del EVENTO
- Hora del EVENTO
- Precio de FILMACIÓN
- Adelanto de dinero

Datos proporcionados por el sistema:

- Fecha de contrato
- Número de CEDULA del CLIENTE
- Nombre del CLIENTE
- Teléfono del CLIENTE
- Tipo de EVENTO
- SALDO

3.1.26.1.3 Proceso

El USUARIO seleccionará del calendario la fecha y se mostrará en una lista las filmaciones programadas para ese día. Se deberá seleccionar de la lista la FILMACIÓN a modificar y luego pulsar el botón Editar. Cuando los datos se hayan modificado se dará click al botón Guardar.

3.1.26.1.4 Salidas

El registro de FILMACIÓN será actualizado en la base de datos con las modificaciones hechas por el USUARIO(Administrador/Operador).

3.1.27 Generar reporte de alquileres

3.1.27.1 Especificación

3.1.27.1.1 Introducción

Este proceso mostrará los alquileres realizados en el video-rental en un rango de fechas.

3.1.27.1.2 Entradas

Por pantalla:

- No se ingresarán datos para este proceso

Datos proporcionados por el sistema:

- Fecha de Inicio
- Fecha Fin

3.1.27.1.3 Proceso

Se mostrará el formulario Reporte de Alquileres donde el USUARIO(Administrador/Operador) podrá seleccionar el rango de fechas. El sistema generará el informe con los alquileres realizados en el período especificado.

3.1.27.1.4 Salidas

Con los datos proporcionados, el sistema consultará la base de datos del servidor y presentará los resultados.

3.1.28 Generar informe de clientes

3.1.28.1 Especificación

3.1.28.1.1 Introducción

Este proceso mostrará los alquileres realizados por un CLIENTE.

3.1.28.1.2 Entradas

Por pantalla:

- Nombre del CLIENTE

Datos proporcionados por el sistema:

- El sistema no proporcionará datos para este proceso

3.1.28.1.3 Proceso

Se mostrará el formulario Reporte de clientes en donde el USUARIO(Administrador/Operador) ingresará el nombre del CLIENTE. El sistema generará un informe con los alquileres realizados por el CLIENTE.

3.1.28.1.4 Salidas

Con los datos proporcionados, el sistema consultará la base de datos y presentará los resultados.

3.1.29 Generar informe devoluciones

3.1.29.1 Especificación

3.1.29.1.1 Introducción

Este proceso mostrará las devoluciones de películas realizadas en el video-rental en un rango de fechas y las devoluciones pendientes para un día específico.

3.1.29.1.2 Entradas

Por pantalla:

- No se ingresarán datos para este proceso

Datos proporcionados por el sistema:

- Fecha de Inicio
- Fecha Fin

3.1.29.1.3 Proceso

Se mostrará el formulario Reporte devoluciones donde el USUARIO(Administrador/Operador) podrá seleccionar el rango de fechas. El sistema generará el informe con las devoluciones realizadas o pendientes en el período especificado.

3.1.29.1.4 Salidas

Con los datos proporcionados, el sistema consultará la base de datos y presentará los resultados.

3.1.30 Generar informe de filmaciones

3.1.30.1 Especificación

3.1.30.1.1 Introducción

Este proceso mostrará las filmaciones programadas en el video-rental en un rango de fechas.

3.1.30.1.2 Entradas

Por pantalla:

- No se ingresarán datos para este proceso

Datos proporcionados por el sistema:

- Fecha de Inicio
- Fecha Fin

3.1.30.1.3 Proceso

Se mostrará el formulario Reporte de Filmaciones donde el USUARIO(Administrador/Operador) podrá seleccionar el rango de fechas. El sistema generará el informe con las filmaciones programadas en el período especificado.

3.1.30.1.4 Salidas

Con los datos proporcionados, el sistema consultará la base de datos y presentará los resultados.

3.1.31 Generar informe de ingresos

3.1.31.1 Especificación

3.1.31.1.1 Introducción

Este proceso mostrará los ingresos obtenidos por los servicios del video-rental en un rango de fechas especificado por el USUARIO Administrador.

3.1.31.1.2 Entradas

Por pantalla:

- No se ingresarán datos para este proceso

Datos proporcionados por el sistema:

- Fecha de Inicio
- Fecha Fin

3.1.31.1.3 Proceso

Se mostrará el formulario Reporte de ingresos donde el USUARIO podrá seleccionar el rango de fechas. El sistema generará el informe con los ingresos percibidos en el período especificado por el USUARIO.

3.1.31.1.4 Salidas

Con los datos proporcionados, el sistema consultará la base de datos y presentará los resultados.

3.1.32 Generar informe de clientes morosos

3.1.32.1 Especificación

3.1.32.1.1 Introducción

Este proceso mostrará los clientes que se encuentran en mora en un rango de fechas especificado por el USUARIO(Administrador/Operador).

3.1.32.1.2 Entradas

Por pantalla:

- No se ingresarán datos para este proceso

Datos proporcionados por el sistema:

- Fecha de Inicio
- Fecha Fin

3.1.32.1.3 Proceso

Se mostrará el formulario Reporte de Clientes morosos, donde el USUARIO podrá seleccionar el rango de fechas. El sistema generará el informe con los nombres, días de retraso y el RECARGO para cada cliente moroso, en el período especificado por el USUARIO.

3.1.32.1.4 Salidas

Con los datos proporcionados, el sistema consultará la base de datos del servidor y presentará los resultados.

3.1.33 Generar informe de películas

3.1.33.1 Especificación

3.1.33.1.1 Introducción

Este proceso mostrará los datos de las películas registradas en la base de datos.

3.1.33.1.2 Entradas

Por pantalla:

- No se ingresarán datos para este proceso

Datos proporcionados por el sistema:

- El sistema no proporcionará datos para este proceso

3.1.33.1.3 Proceso

Se mostrará el formulario Reporte de Películas, en donde se visualizarán los datos de todas las películas registradas en el video-rental.

3.1.33.1.4 Salidas

Con los datos proporcionados, el sistema consultará la base de datos del servidor y presentará los resultados.

3.1.34 Generar informe de películas más alquiladas

3.1.34.1 Especificación

3.1.34.1.1 Introducción

Este proceso mostrará las películas más alquiladas en un rango de fechas especificado por el USUARIO(Administrador/Operador).

3.1.34.1.2 Entradas

Por pantalla:

- No se ingresarán datos para este proceso.

Datos proporcionados por el sistema:

- Fecha de Inicio
- Fecha Fin

3.1.34.1.3 Proceso

Se mostrará el formulario Reporte de películas más alquiladas, donde el USUARIO podrá seleccionar el rango de fechas. El sistema generará el

informe con las películas que han sido más alquiladas, en el período especificado por el USUARIO.

3.1.34.1.4 Salidas

Con los datos proporcionados, el sistema consultará la base de datos y presentará los resultados.

3.1.35 Respaldar base de datos

3.1.35.1 Especificación

3.1.35.1.1 Introducción

Este proceso deberá realizar el respaldo de la base de datos del servidor.

3.1.35.1.2 Entradas

Por pantalla:

- No se proporcionarán entradas en este proceso.

Datos proporcionados por el sistema:

- El sistema no proporcionará datos en este proceso.

3.1.35.1.3 Proceso

El sistema realizará automáticamente el respaldo e indicará el progreso del proceso.

3.1.35.1.4 Salidas

El sistema indicará con un mensaje el éxito de la operación.

3.1.36 Restaurar base de datos

3.1.36.1 Especificación

3.1.36.1.1 Introducción

Este proceso deberá realizar la restauración de un respaldo de la base de datos.

3.1.36.1.2 Entradas

Por pantalla:

- No se proporcionarán entradas en este proceso.

Datos proporcionados por el sistema:

- El sistema no proporcionará datos en este proceso.

3.1.36.1.3 Proceso

El sistema realizará automáticamente la restauración de un respaldo anterior de la base de datos.

3.1.36.1.4 Salidas

El sistema indicará con un mensaje el éxito de la operación.

3.2 Requisitos de funcionamiento

Requisitos dinámicos: Es importante que el tiempo de respuesta no aumente exponencialmente con el número de registros en la base de datos.

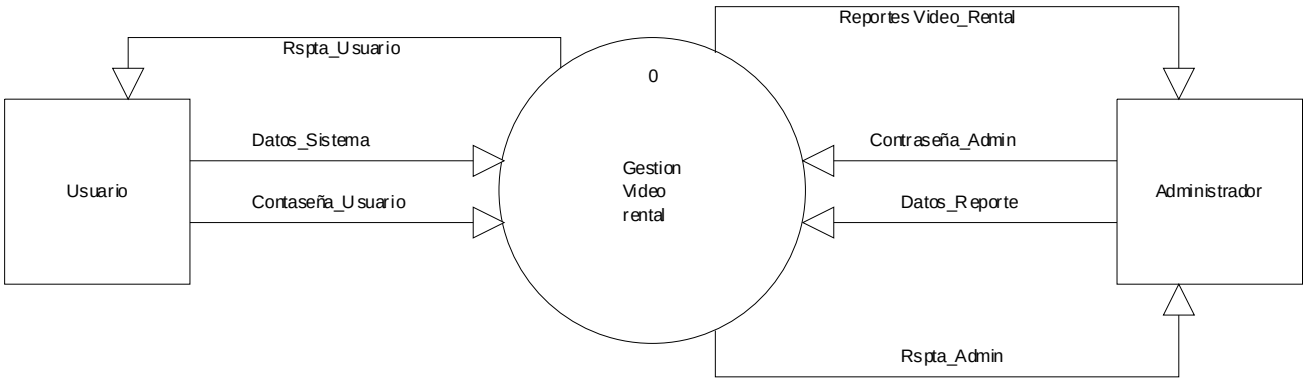
3.3 Otros requisitos

3.3.1 Base de datos

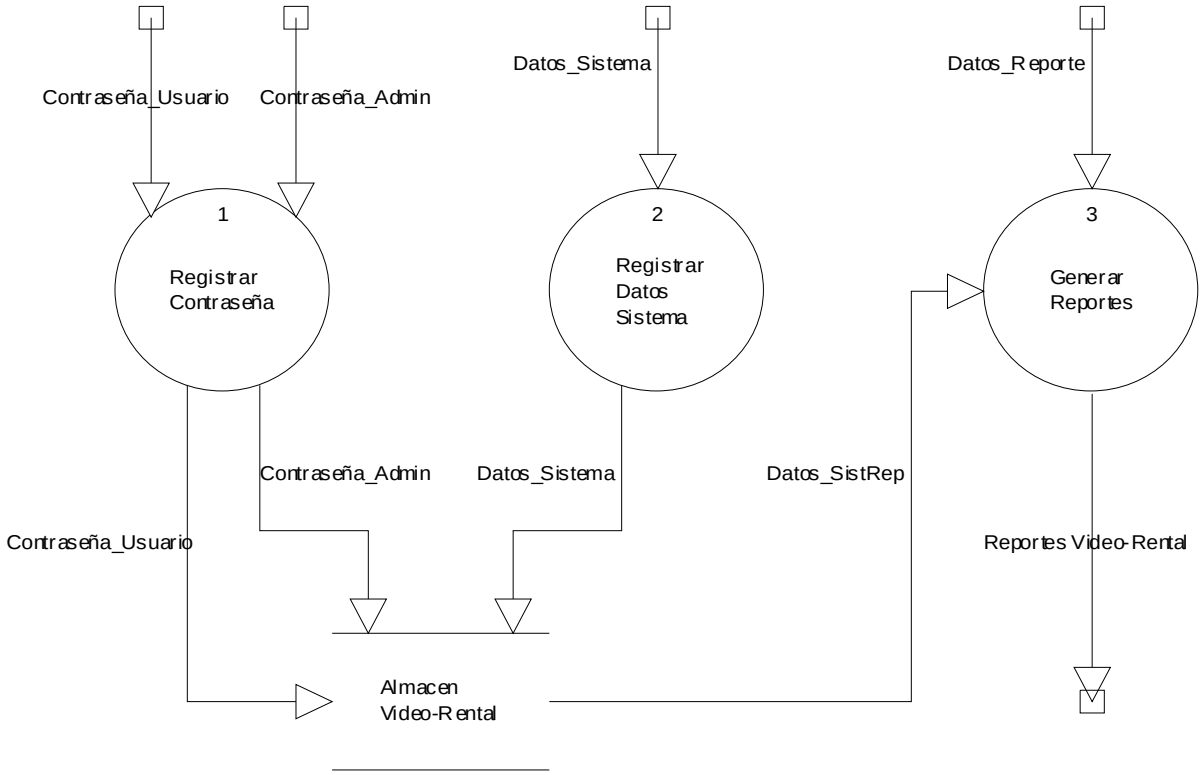
El almacenamiento de información se realizará por medio de una base de datos relacional que estará alojada en un servidor SQL Server.

7.1. DIAGRAMA DE FLUJO DE DATOS

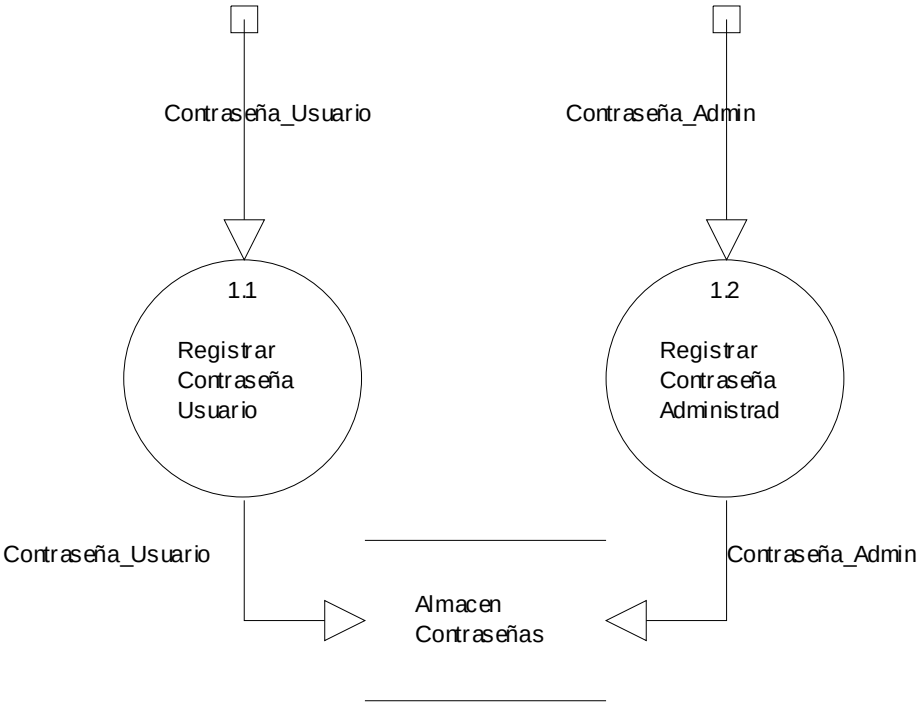
NIVEL 0



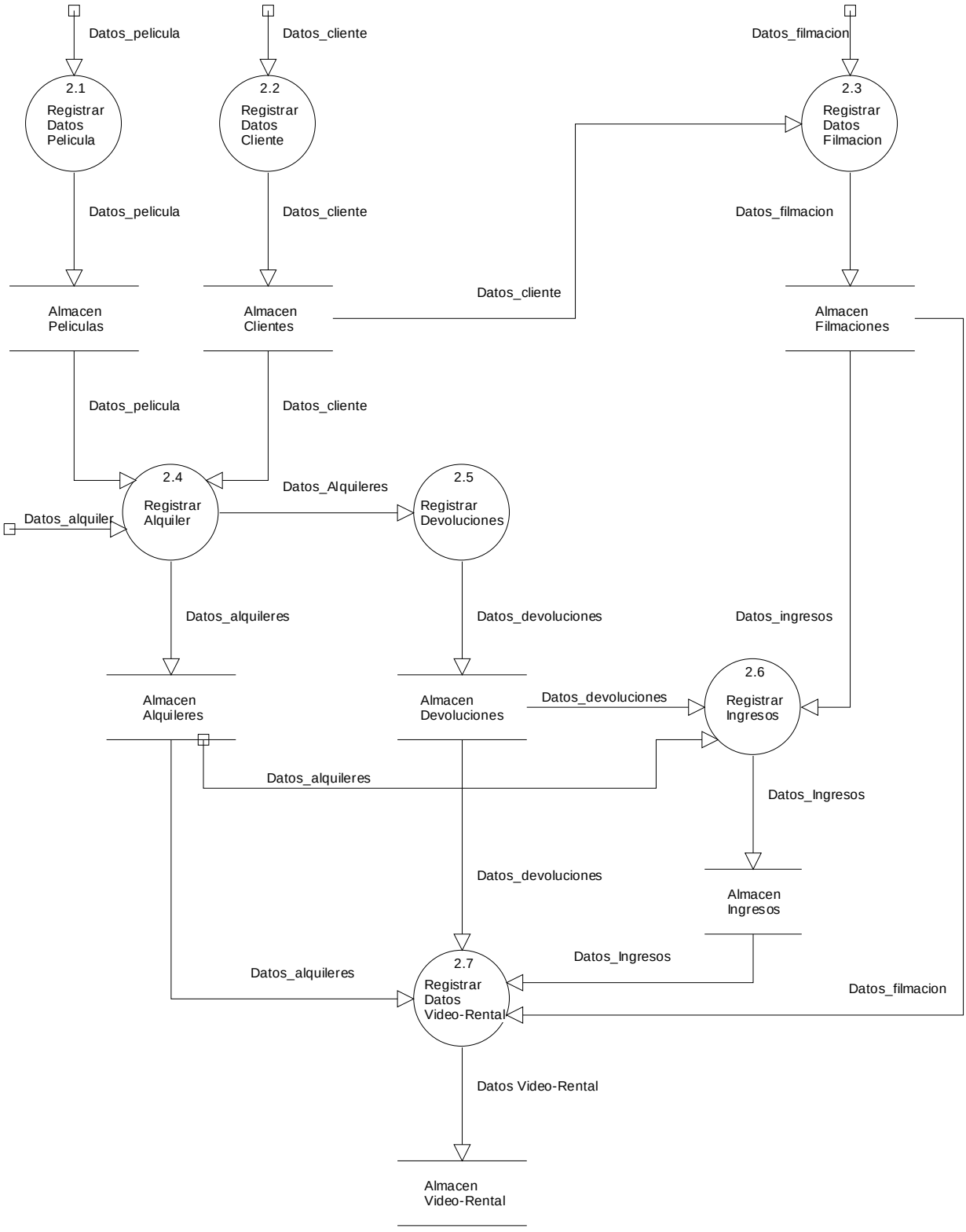
NIVEL 1 DIAGRAMA 1



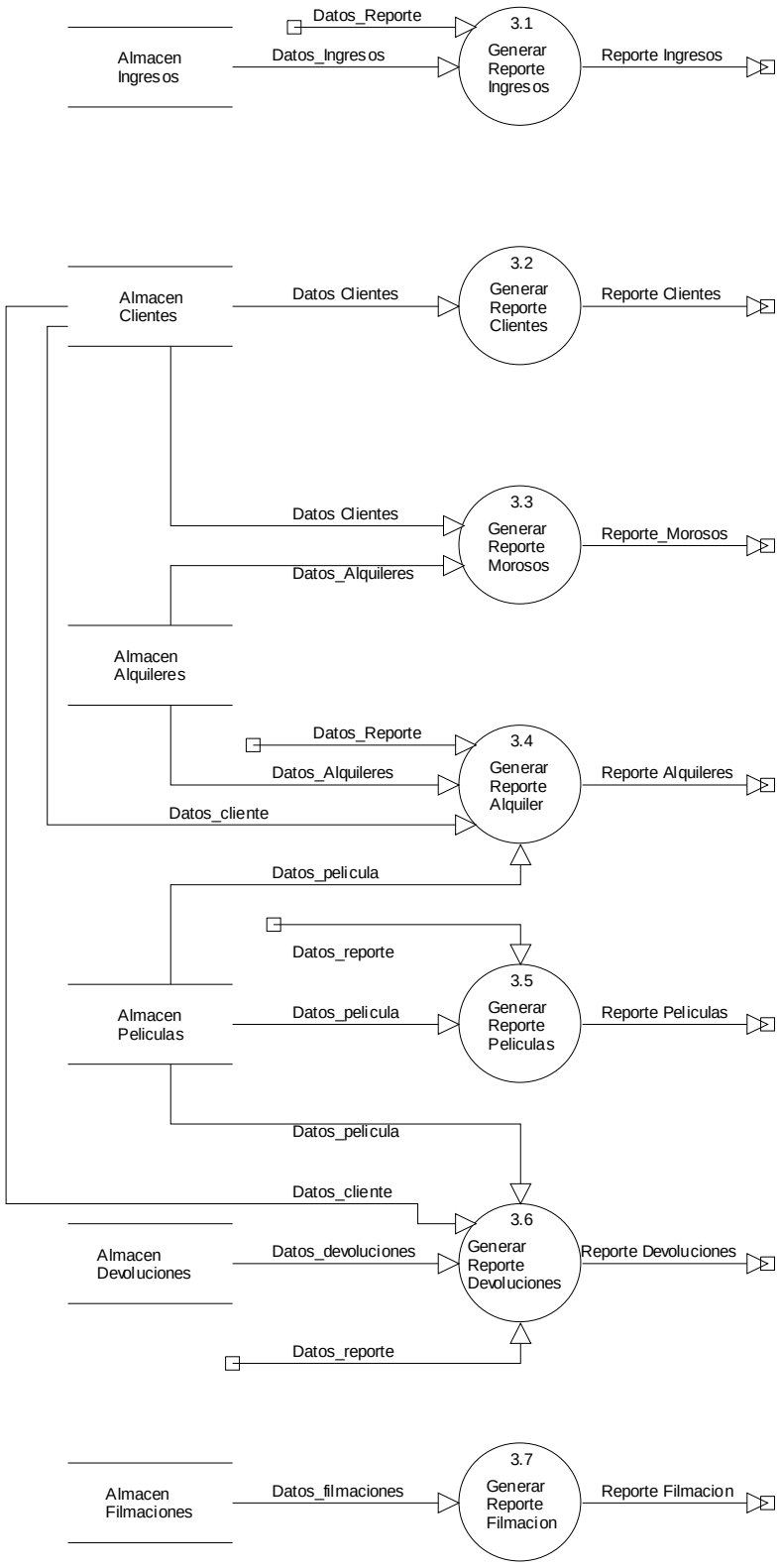
NIVEL 2 DIAGRAMA 1



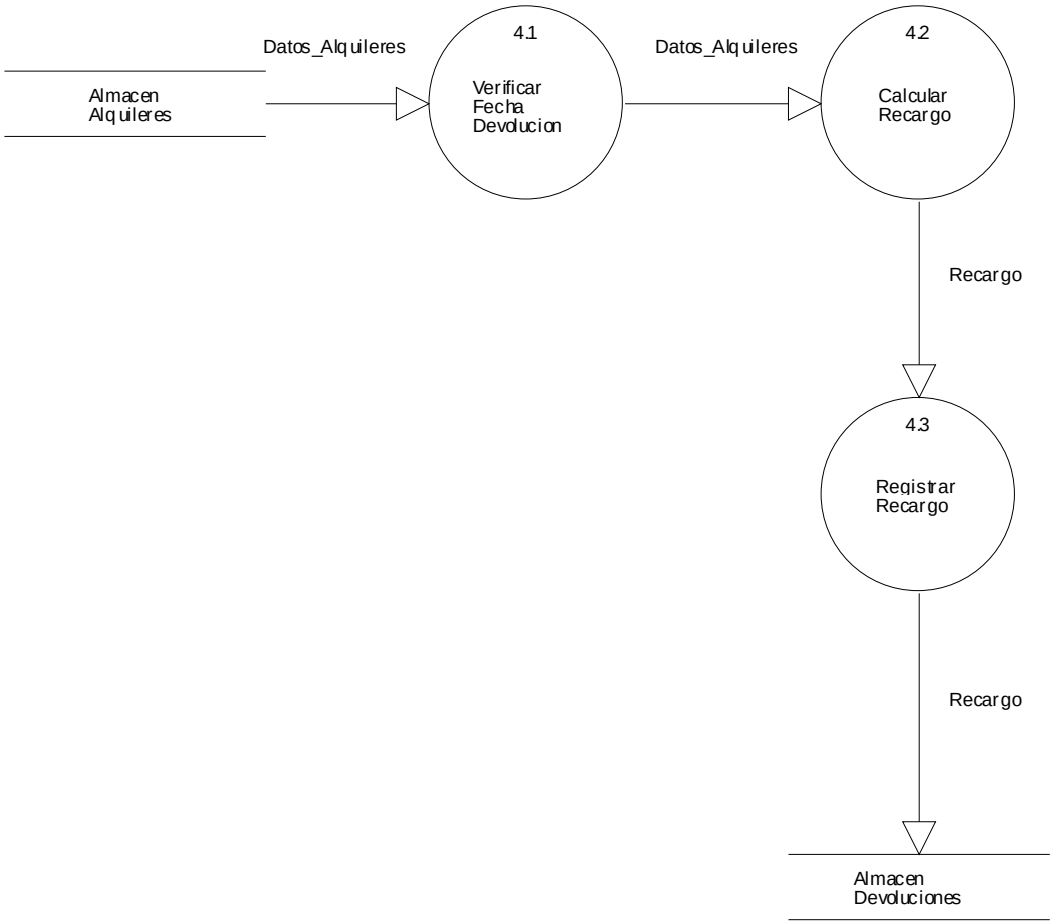
NIVEL 2 DIAGRAMA 2



NIVEL 2 DIAGRAMA 3



NIVEL 3 DIAGRAMA 1



7.2. DICCIONARIO DE DATOS

Datos_Sistema = Datos_Cliente + Datos_Pelicula + Datos_Filmacion + Datos_Alquiler + Datos_Reporte + Datos_Usuario

Datos_Alquileres = Datos_Pelicula + Datos_Cliente + Datos_alquiler

Datos_alquiler = FechaAlq + FechaDev

Datos_Filmacion = Datos_Cliente + FechaContrato + FechaEvento + DirEvento + HoraEvento + TipoEvento + Precio + Adelanto

Datos_Cliente = NoCedula + Nombre + Direccion + Telefono + Edad + Sexo

Datos_Pelicula = IdPelicula + Titulo + ActorPrincipal + Estreno + Datos_Genero + Datos_formato

Datos_Genero = IdGenero + Nombre

Datos_Formato = IdFormato + Nombre + Precio + PrecioEstreno + multaXretraso

Datos_Evento = IdEvento + TipoEvento

Datos_Devolucion = IdCliente + IdPelicula + Monto + FechaAlq + FechaDev + Genero + Titulo + Formato

Datos_Reporte = FechaIni + FechaFin + TipoReporte

Datos_Usuario = NombreUsuario + Contraseña + TipoAcceso

Almacen Datos_Sistema = Almacen Contraseña + Almacen Video-Rental

Almacen Video-Rental = Almacen Película + Almacen Cliente + Almacen Filmacion + Almacen Alquileres + Almacen Devoluciones
Almacen Contraseña = Contraseña_Usuario + Contraseña_Admin
Almacen Película = Datos_Pelicula
Almacen Cliente = Datos_Cliente

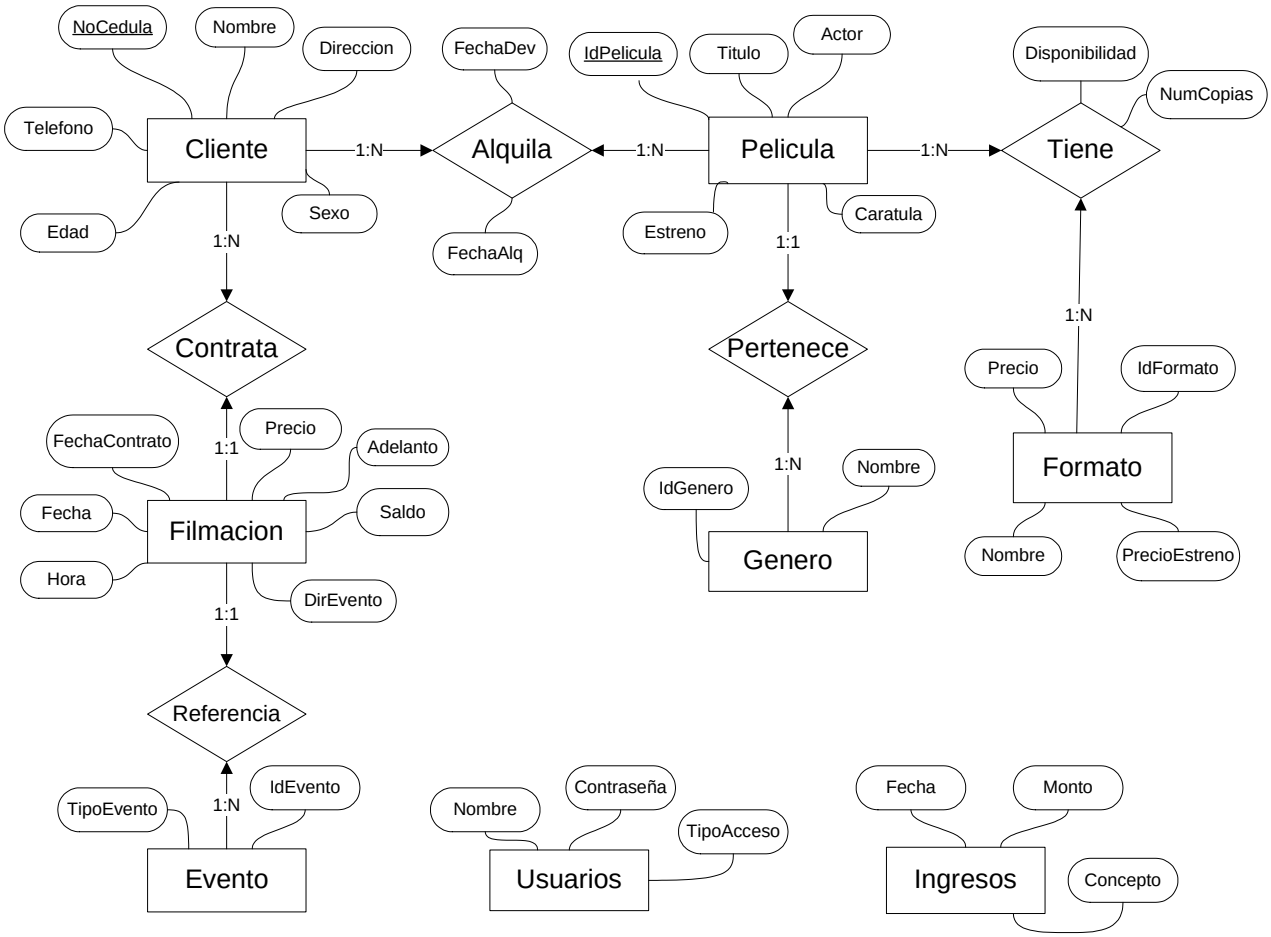
Almacen Filmacion = Datos_Filmacion

Almacen Alquileres = Datos_Alquiler

Almacen Devoluciones = Datos Devolución

Reportes Video_Rental = Reporte Ingresos + Reporte Alquileres + Reporte Morosos + Reporte Devoluciones + Reporte Clientes + Reporte Películas + Reporte Filmación

7.3. DIAGRAMA ENTIDAD-RELACIÓN



7.4. DISEÑO DE DATOS

DISEÑO DE DATOS

1. Tabla Cliente		
Nombre Atributo	**Tipo**	**Tamaño**
NoCedula	Char	20
Nombre	Varchar	30
Direccion	Varchar	40
Telefono	Varchar	15
Edad	Int	4
Sexo	Char	2

2. Tabla Pelicula		
Nombre Atributo	**Tipo**	**Tamaño**
IdPelicula	Varchar	5
Titulo	Varchar	60
ActorPrincipal	varchar	30
IdGenero	Smallint	2
Estreno	Bit	1
Caratula	varchar	500
Fecha	Date	

3. Tabla Alquila		
Nombre Atributo	**Tipo**	**Tamaño**
NoCedula	Char	20
IdPelicula	Varchar	5
FechaAlq	Datetime	8
FechaDev	Datetime	8
Monto	Float	8
Formato	Varchar	20

4. Tabla Devolucion		
Nombre Atributo	**Tipo**	**Tamaño**
NoCedula	Char	20
IdPelicula	Varchar	5
Formato	Varchar	20

FechaAlq	Datetime	8
FechaDev	Datetime	8
Monto	Float	8

5. Tabla Formato		
Nombre Atributo	**Tipo**	**Tamaño**
IdFormato	Smallint	2
Nombre	Varchar	20
Precio	Float	8
PrecioEstreno	Float	8
multaXretraso	Float	8

6. Tabla Genero		
Nombre Atributo	**Tipo**	**Tamaño**
IdGenero	Smallint	2
Nombre	Varchar	20

7. Tabla Tiene		
Nombre Atributo	**Tipo**	**Tamaño**
IdPelicula	Varchar	5
IdFormato	Smallint	2
Disponibilidad	Int	4
Numcopias	Int	4

8. Tabla Evento		
Nombre Atributo	**Tipo**	**Tamaño**
IdEvento	Smallint	2
TipoEvento	Varchar	20

9. Tabla Filmacion		
Nombre Atributo	**Tipo**	**Tamaño**
Fecha_contrato	Datetime	8
NoCedula	Char	20
IdEvento	Smallint	2
Precio	Float	8
Dir_Evento	Varchar	50
Fecha	Datetime	8
Hora	Varchar	10
Adelanto	Float	8
Saldo	Float	8
Estado	Bit	1

10. Tabla Ingresos		
Nombre Atributo	**Tipo**	**Tamaño**
Fecha	Datetime	8
Concepto	Varchar	50
Monto	Float	8

11. Tabla Usuario		
Nombre Atributo	**Tipo**	**Tamaño**
NombreUsuario	Varchar	25
Contraseña	Varchar	20
TipoAcceso	Varchar	10

VIII.CONCLUSIONES

Apoyándonos en las herramientas, métodos y procedimientos que propone la metodología de la Ingeniería del Software, concluimos que hemos logrado cumplir con los objetivos planteados.

Al utilizar la metodología de la Ingeniería del Software alcanzamos desarrollar un sistema de calidad, seguro y confiable, además aplicando el modelo cliente/servidor pudimos automatizar los procesos de gestión y control del video-rental, creando una interfaz amigable para el usuario del sistema.

El sistema es una herramienta útil y fácil de usar debido a la combinación de las características del lenguaje de Programación C#, el Sistema Gestor de Bases de Datos SQL Server 2000.

IX. RECOMENDACIONES

- Que los estudiantes utilicen las herramientas, métodos y procedimientos de la metodología de la Ingeniería del Software en el desarrollo de sus proyectos para lograr cumplir sus objetivos propuestos.

- Motivar en el Departamento de Computación que los estudiantes desarrollen sistemas para empresas e instituciones como práctica para la vida laboral.

X. REFERENCIAS BIBLIOGRÁFICAS

1. Blanco, Luis Miguel. *Crystal Reports para Visual Studio.Net.(Visual Basic.NET). (2003). Grupo EIDOS.*
2. Ibarra, Martin. *Apuntes de Ingeniería del software.* (2007).
3. Mendoza, María Esperanza, Narváez, Griselda Yolanda, Roque, Álvaro Ramón. *Control De Subsidios Del "Hospital Ingenio San Antonio".* (2003).
4. Morales Maradiaga, María, Morales Osejo, Silvia. *Sistema de automatización para la elaboración e impresión de recibos y matriculas del Instituto Socorro Santana Solís (ISSS).* (2007).
5. Pressman, Roger. *Ingeniería del Software. Un Enfoque Práctico, Quinta Edición.* (2002).
6. Chiavenato, Idalberto. Introducción a la Informática, (1992). http: //www.monografias.com/Introducción a la informática.
7. http://es.wikipedia.org/wiki/Bases_de_datos.
8. http://www.getec.etsit.upm.es/docencia/gproyectos/planificacion/cvida.htm.
9. Soto, Lauro. Tutorial de C#, 2005.http://www.programacion.net/tutorial/csharp/3/
10. Elmoo, Características de SQL Server 2000, (2000). http://www.asiutn.org.ar/sysacad/InstructivoSQLDesktop.htm.
11. Instalación de SQL-Server Personal Edition, (2009). http://www.monografias.com. Características de SQL Server 2000
12. http://es.wikipedia.org/wiki/Cliente-servidor, (2005).

ANEXOS

DIAGRAMA CONCEPTUAL USUARIO ADMINISTRADOR

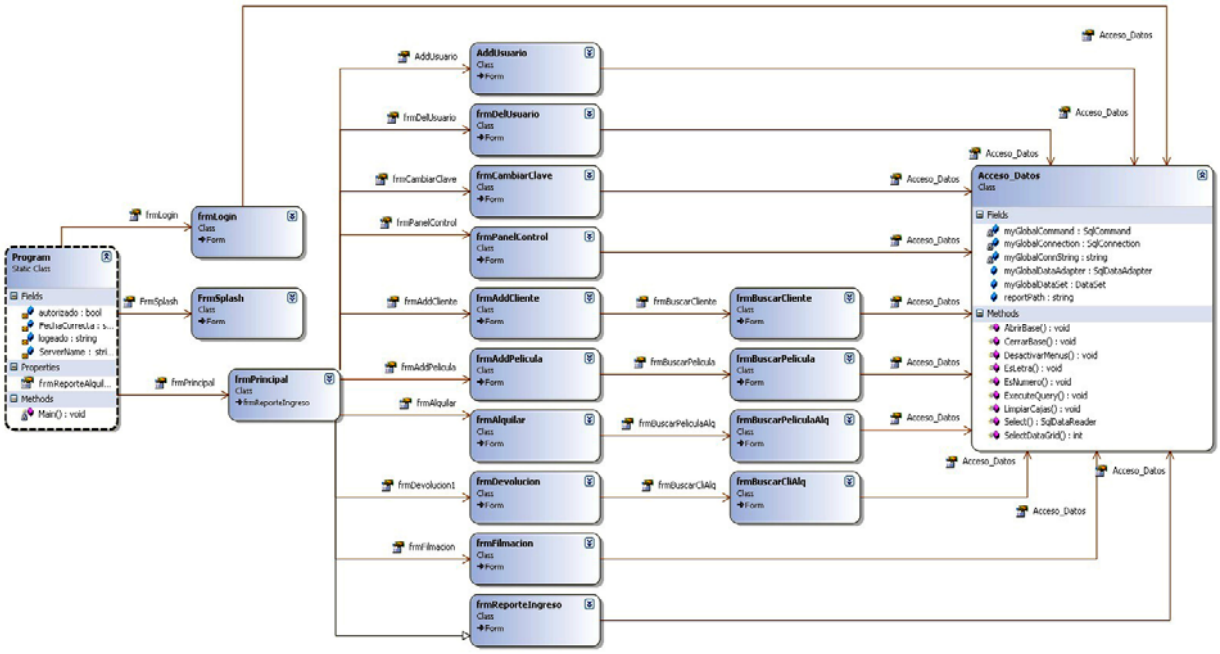
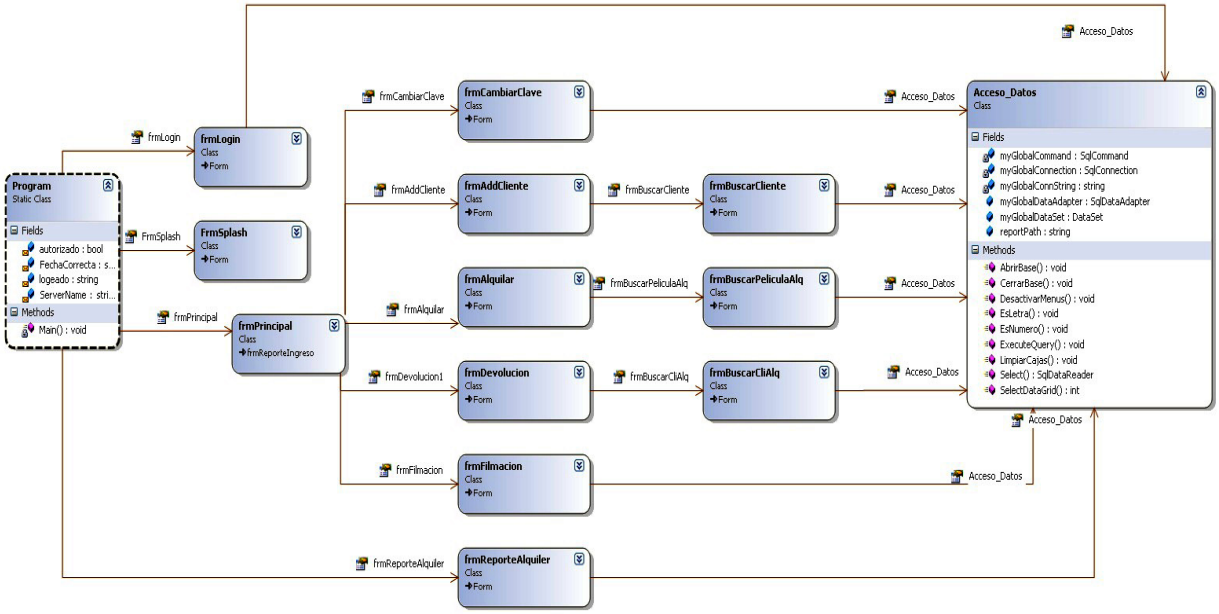


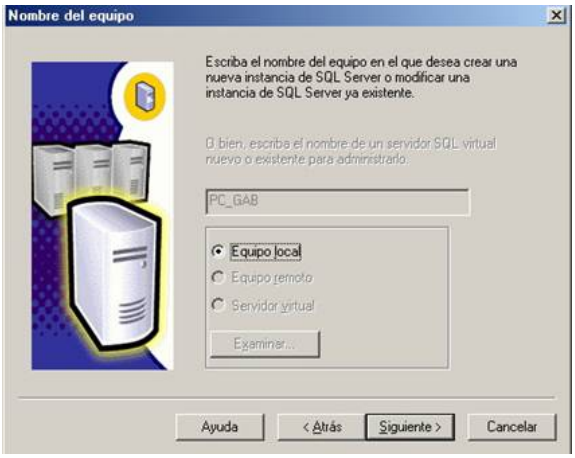
DIAGRAMA CONCEPTUAL USUARIO OPERADOR



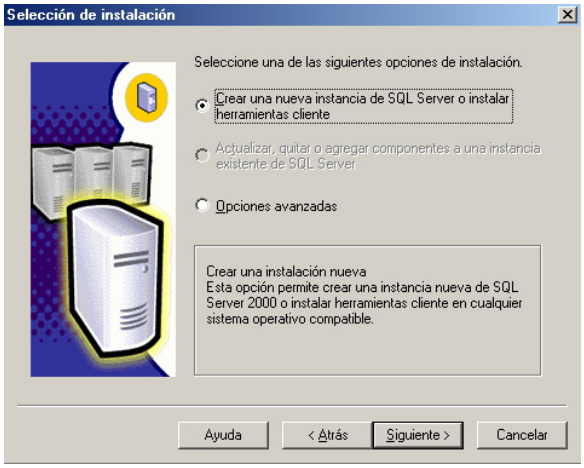
INSTALACION Y CONFIGURACION DE SQL SERVER 2000

A continuación describimos los pasos para realizar la instalación de SQL-Server

1) Una vez insertado el CD, si este no activa la ventana de instalación ejecutar el archivo autorun.exe ubicado en el directorio raíz del CD.
2) Seleccionar 'Componentes de SQL Server 2000' y luego "Instalar servidor de Base de datos".
3) En la ventana de instalación seleccionar 'Equipo Local'

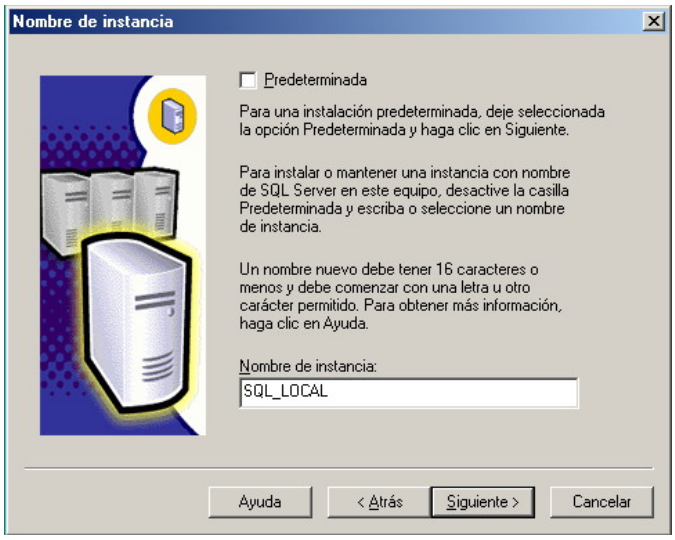


4) Crear una nueva instancia de SQL Server.

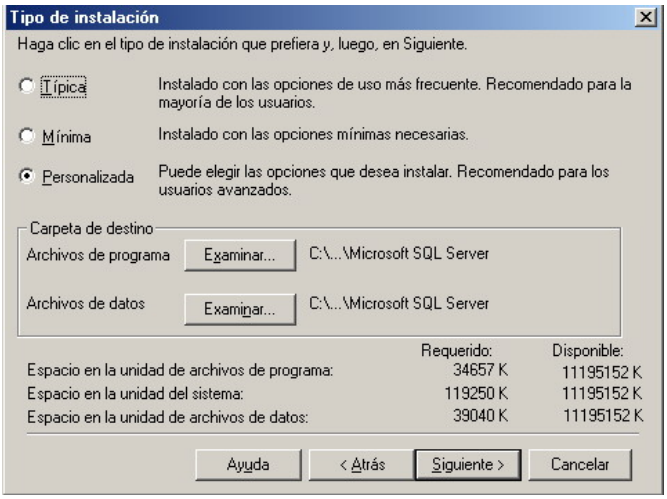


5) En la ventana de **Definición de Instalación** seleccionar "Herramientas cliente y servidor"

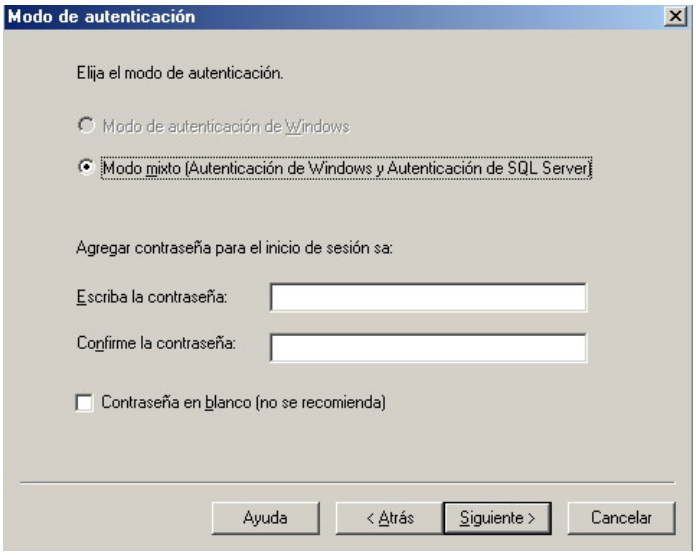
6) En la ventana **Nombre de la Instancia** desmarcar la opción Predeterminada, y en la casilla de Nombre de la instancia, Ingresar un nombre



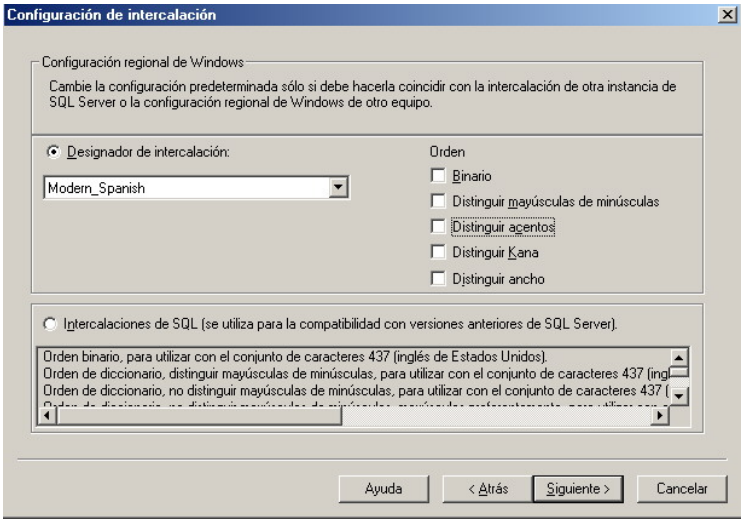
7) En la ventana **Tipo de instalación** seleccionar 'Personalizada'



8) En la ventana **Modo de autenticación** seleccionar modo mixto e ingresar una clave para el inicio de sesión del System Administrator (sa).



9) En la ventana **Configuración de Intercalación** desmarcar la casilla 'Distinguir acentos'



10) Continuar con la instalación hasta finalizar la misma.

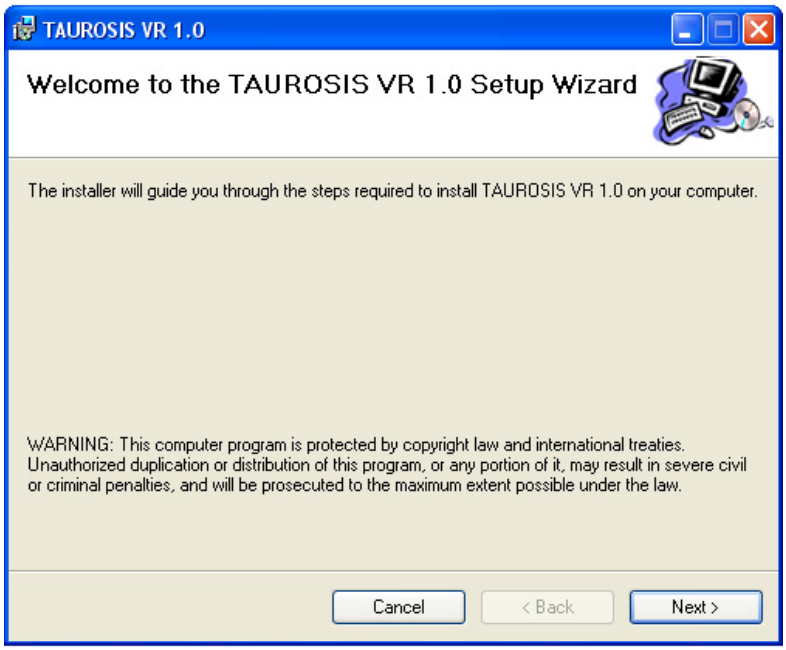
REQUISITOS E INSTALACION DE TAUROSIS VR 1.0

Requisitos de instalación de TAUROSIS VR 1.0

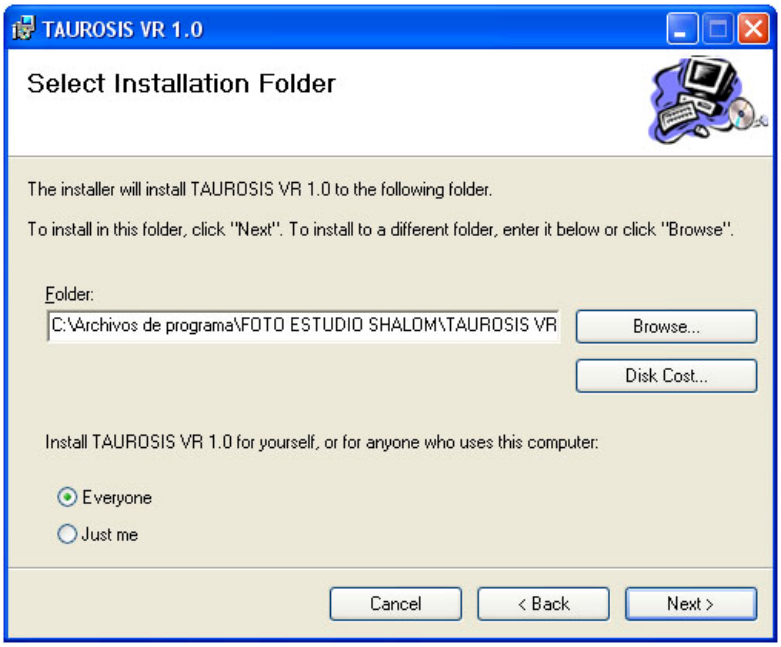
- Sistema operativo Windows XP SP2 o SP3
- Memoria RAM 256 MB o superior
- Disco duro 90 MB libres

A continuación describimos los pasos para realizar la instalación de TAUROSIS VR 1.0

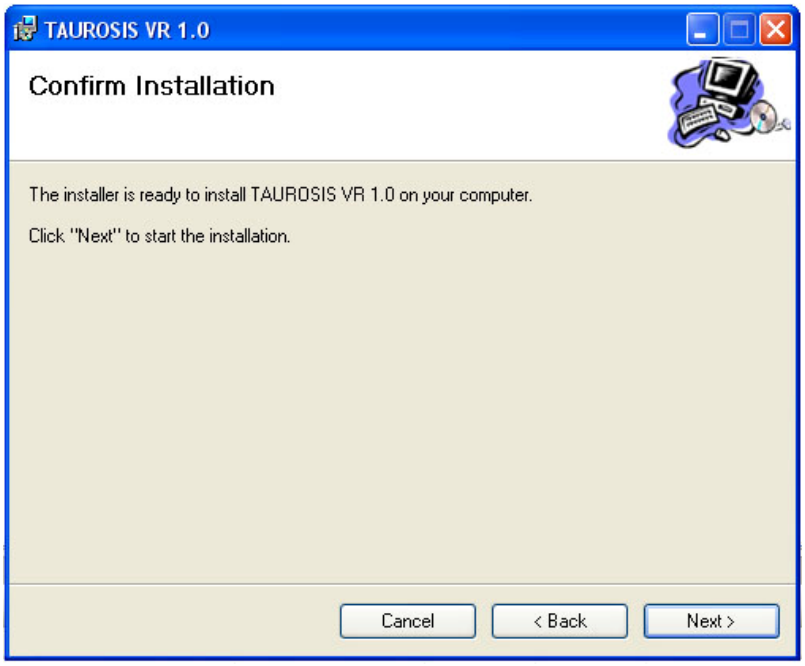
1) Una vez insertado el CD, si este no activa la ventana de instalación ejecutar el archivo setup.exe o TAUROSIS VR 1.0 ubicado en el directorio raíz del CD.
2) En la ventana Setup dar click en next.



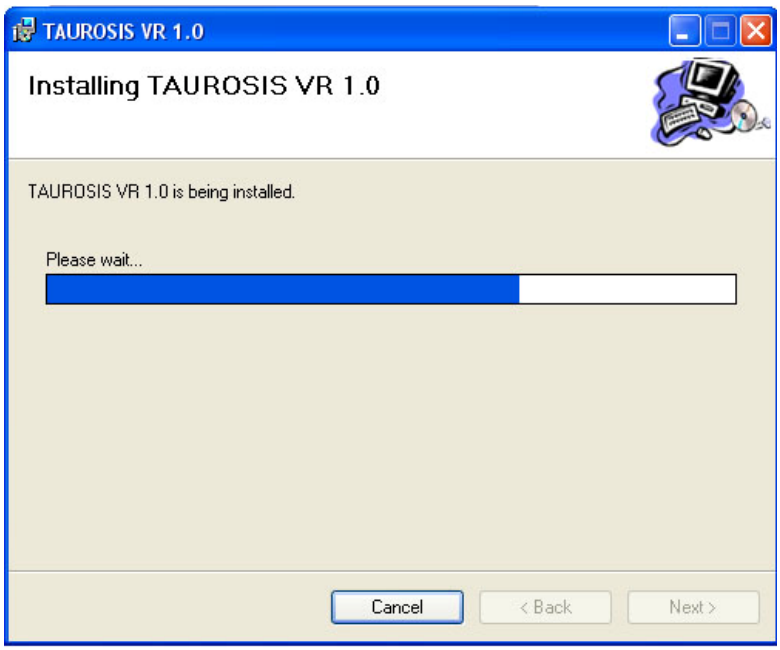
3) En la ventana Select Installation Folder escoger la ruta de instalación y dejar Everyone seleccionado.

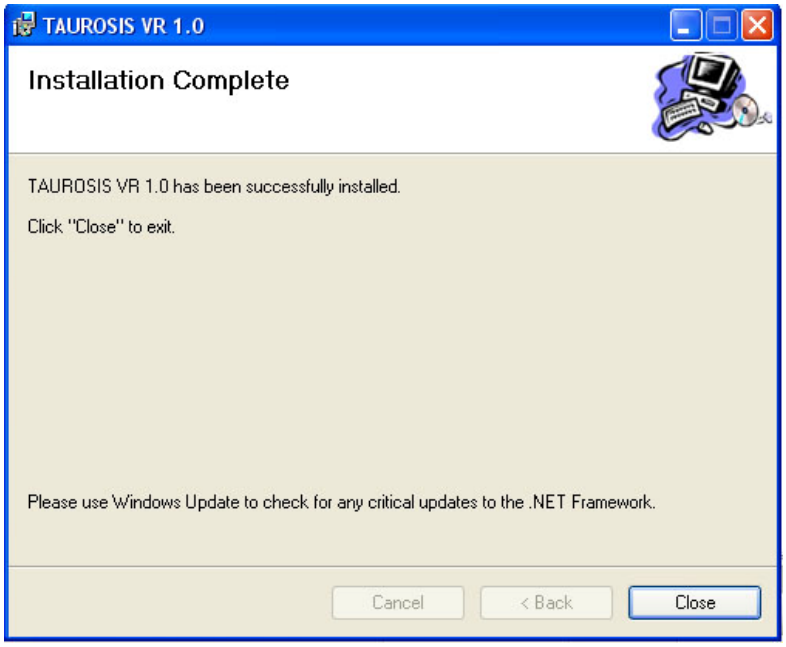


4) En la siguiente ventana confirmar la instalación dar click en next.



5) Esperar hasta que la instalación finalice y dar click en Close.





6) Por último buscar el archivo ejecutable en la ruta que especificó en el paso 3, en la carpeta donde instaló la aplicación y darle doble click.

FORMULARIOS, CODIGOS DE FUNCIONES Y EVENTOS PRINCIPALES

//Código de la clase Acceso_Datos.cs

```
public class Acceso_Datos
    {
        static SqlCommand myGlobalCommand;
        static string myGlobalConnString = "Data Source=" + Program.ServerName + ";
        User ID=SHALOM;Initial Catalog=VideoClub;Trusted_Connection=Yes";
        static public string reportPath =
        @"C:\ProyectoVideoClub(Nuevo)Trabajndo en red\prjVideoClub\prjVideoClub";
        static public SqlDataAdapter myGlobalDataAdapter;
        static public DataSet myGlobalDataSet;
        static SqlConnection myGlobalConnection;

        public static void AbrirBase()
        {
            //Abre la conexion con la base de datos
            try
            {
                myGlobalConnection = new SqlConnection(myGlobalConnString);
                myGlobalConnection.Open();
            }
            catch (Exception error)
            {
                MessageBox.Show(error.ToString());
            }
        }

        public static void ExecuteQuery(string myQuery)
        {
            //Esta funcion puede usarse para insert o update
            myGlobalCommand = new SqlCommand(myQuery, myGlobalConnection);
            myGlobalCommand.ExecuteNonQuery();
            myGlobalCommand.Dispose();
        }

        public static SqlDataReader Select(string mySelectQuery)
        {
            //Esta funcion  realiza selecciones, solo se le pasa la cadena de seleccion
            SqlCommand mySelectCommand;
            SqlDataReader myDataReader;
            mySelectCommand = new SqlCommand(mySelectQuery, myGlobalConnection);
            myDataReader = mySelectCommand.ExecuteReader();
            mySelectCommand.Dispose();
            return myDataReader;
        }
```

```
public static int SelectDataGrid(string mySelectQuery, string table, DataGridView DataGrid)
{
    //Esta funcion se usa para enlazar los DataGridviews automaticamente
    myGlobalDataAdapter = new SqlDataAdapter(mySelectQuery, myGlobalConnection);
    myGlobalDataSet = new DataSet();
    myGlobalDataAdapter.Fill(myGlobalDataSet, table);
    DataGrid.DataSource = myGlobalDataSet.Tables[table];
    return myGlobalDataSet.Tables[table].Rows.Count;
}

public static void CerrarBase()
{
    //Cierra la conexion con la base de datos
    myGlobalConnection.Close();

}


public static void DesactivarMenus(MenuStrip ms, ToolStrip ts, bool activo)
{
    ms.Enabled = activo;
    ts.Enabled = activo;

}

public static void EsNumero(KeyPressEventArgs e)
{
    //Validar los campos numericos, no introducir letras
    if (e.KeyChar == 8)
    {
        e.Handled = false;
        return;
    }

    if (e.KeyChar >= 48 && e.KeyChar <= 57)
        e.Handled = false;
    else
        e.Handled = true;
}



    public static void EsLetra(KeyPressEventArgs e)
    {
        //Validar las cajas de texto, no introducir números
        if (e.KeyChar == 8)
        {
            e.Handled = false;
            return;
        }
        if (e.KeyChar == 32)
        {
            e.Handled = false;
            return;
        }

        if (e.KeyChar >= 58)
            e.Handled = false;

        else
            e.Handled = true;

    }

}
```

Formulario Login

//Código del evento Load

```
private void frmLogin_Load(object sender, EventArgs e)
{
    ActivarCajas(activo);
    //Capturar los nombres de instancias SQL en la red
    SqlDataSourceEnumerator instance = SqlDataSourceEnumerator.Instance;
    DataTable table = instance.GetDataSources();

    //Insertar los nombres de servidores SQL en el combo
    foreach(DataRow row in table.Rows)
        cmbServidor.Items.Add(row["ServerName"]);

}
```

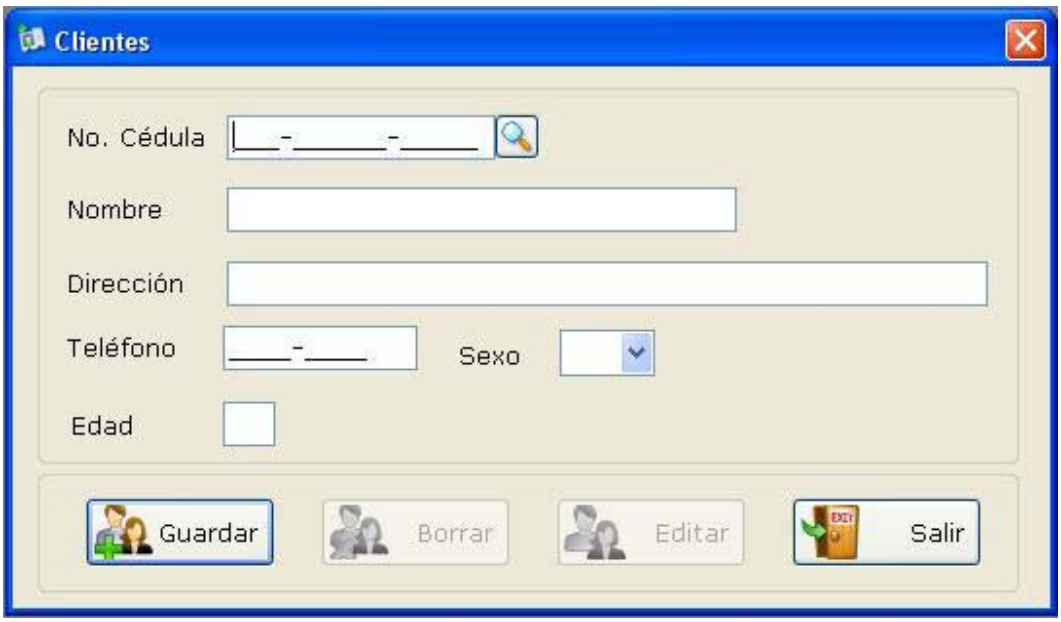
//Código del botón conectar

```
private void btnConectar_Click(object sender, EventArgs e)
{
    if (cmbServidor.SelectedIndex != -1)
    {
        Program.ServerName = cmbServidor.Text;
        Acceso_Datos.AbrirBase();
        activo = true;
        ActivarCajas(activo);
        txtUsr.Focus();
    }
    else
        MessageBox.Show("Debe seleccionar el servidor SQL","Información",
        MessageBoxButtons.OK, MessageBoxIcon.Information);
}
```

//Código del botón Aceptar

```
private void btnAceptar_Click(object sender, EventArgs e)
{
    if (txtUsr.Text.Length != 0 && txtClave.Text.Length != 0)
    {
        if (intentos < 2)//numero de intentos menor que 3
        {
            mySelectQuery = "SELECT Usuarios.* FROM Usuarios
                        WHERE (((Usuarios.NombreUsuario)='" + txtUsr.Text + "'));";
            myDataReader = Acceso_Datos.Select(mySelectQuery);

            if (myDataReader.Read() && myDataReader.GetString(1).CompareTo(txtClave.Text) == 0)
            {
                Program.autorizado = true;
                if (myDataReader.GetString(2) == "ADMIN")
                    frmPrincipal.admin = true;

                Program.logeado = myDataReader.GetString(0);
                myDataReader.Close();

                Close();
            }
            else{
                MessageBox.Show("Usuario o contraseña invalidos", "Error",
                MessageBoxButtons.OK, MessageBoxIcon.Warning);
                txtUsr.Focus();
                txtUsr.SelectAll();
                intentos = intentos + 1;
                myDataReader.Close();

        }
        else{
            MessageBox.Show("Intentos Completos, cerrando aplicación...", "Acceso Denegado",
                        MessageBoxButtons.OK, MessageBoxIcon.Warning);
            Application.Exit();
        }
    }
    else{
        MessageBox.Show("Debe escribir un usuario y una contraseña", "Error",
        MessageBoxButtons.OK, MessageBoxIcon.Stop);
        txtUsr.Focus();
    }

}
```

Formulario Clientes



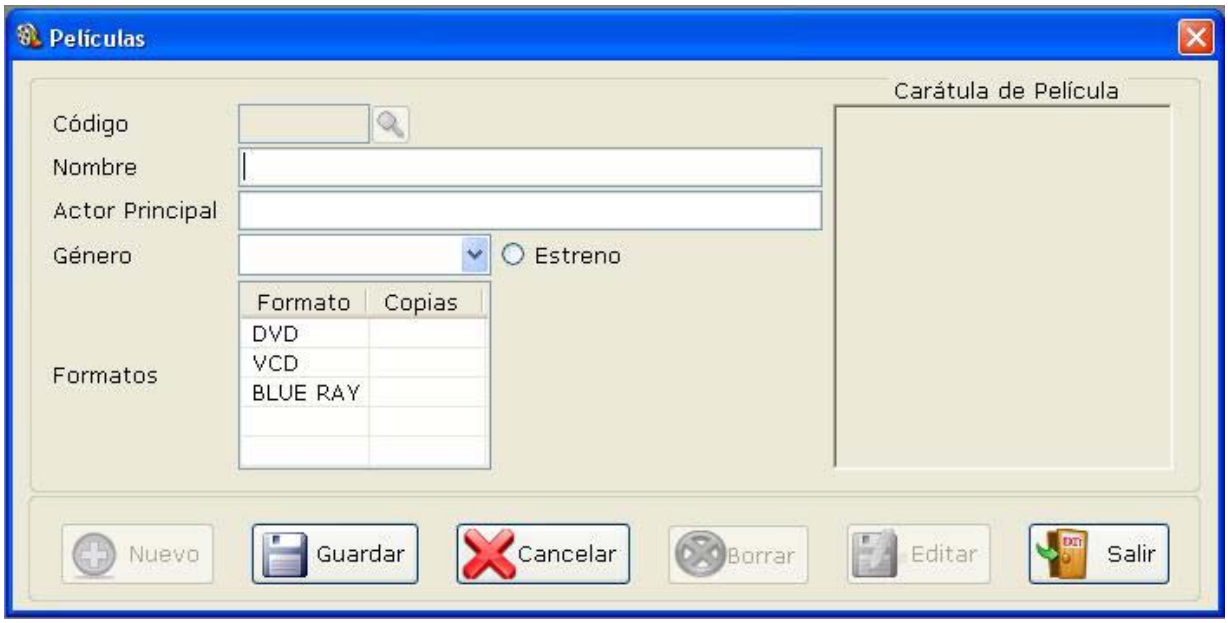
//Código del botón buscar

```
private void btnBuscar_Click(object sender, EventArgs e)
{
    //Llamar al formulario Buscar Cliente
    BuscarCliente = new frmBuscarCliente();
    BuscarCliente.ShowDialog();
    if (BuscarCliente.GetSalir() == false)
    {
        DesHabCajas(true);
        cedula = BuscarCliente.GetNoCedula();
        mskdTxtBxNoCedula.Text = cedula;

        txtNombre.Text = BuscarCliente.GetNombre();
        txtDireccion.Text = BuscarCliente.GetDireccion();
        mskTelefono.Text = BuscarCliente.GetTelefono();
        txtEdad.Text = BuscarCliente.GetEdad().ToString();
        cmbSexo.Text = BuscarCliente.GetSexo();
        DesacBotones(false);

    }
    else
    {
        DesHabCajas(false);
        DesacBotones(true);
    }

    mskdTxtBxNoCedula.Focus();
    Buscar = true;
}
```

//Código del botón Guardar

```
private void btnAgregar_Click(object sender, EventArgs e)
{
    if (ValidarDatos())
    {
        if ((MessageBox.Show("Está seguro que desea guardar esta información",
                "VideoClub", MessageBoxButtons.YesNo)) == DialogResult.Yes)
        {

            mskTelefono.TextMaskFormat = MaskFormat.ExcludePromptAndLiterals;
            if (mskTelefono.Text.Length > 0)
                telefono = mskTelefono.Text.Trim();
            else
                telefono = "N/A";

            if (Editar)//si se esta editando
            {
                mskdTxtBxNoCedula.TextMaskFormat = MaskFormat.IncludePromptAndLiterals;
                //Actualizar el registro en la base de datos
                strQuery = "UPDATE Cliente SET NoCedula = '" + mskdTxtBxNoCedula.Text + "',
                Nombre = '" + txtNombre.Text.Trim().ToUpper() + "',
                Direccion = '" + txtDireccion.Text.Trim().ToUpper() + "',
                Telefono = '" + telefono + "', Edad = '" + txtEdad.Text.Trim() + "',
                Sexo = '" + cmbSexo.Text.Trim() + "'
                WHERE NoCedula = '" + NoCedulaAnterior + "'";
                Acceso_Datos.ExecuteQuery(strQuery);
                MessageBox.Show("Registro Editado con éxito", "VideoClub");

                Editar = false;
                LimpiarCajas();
                DesacBotones(true);

            }
            else
            {
                //Verificar que el cliente no este registrado en la base de datos
                if (!ExisteCliente())
                {
                    mfMaskFormat = mskdTxtBxNoCedula.TextMaskFormat;

                    mskdTxtBxNoCedula.TextMaskFormat = MaskFormat.ExcludePromptAndLiterals;

                    // si no hay valor en el control, salir
                    if (mskdTxtBxNoCedula.Text.Length != 0)

                       mskdTxtBxNoCedula.TextMaskFormat = mfMaskFormat;

                   //Guardar el cliente en la base de datos
                   strQuery = "INSERT INTO Cliente(NoCedula,Nombre,Direccion,Telefono,Edad,Sexo)
                               Values( '" + mskdTxtBxNoCedula.Text + "',
                                       '" + txtNombre.Text.Trim().ToUpper() + "',
                                       '" + txtDireccion.Text.Trim().ToUpper() + "','" + telefono + "',
                                       " + txtEdad.Text.Trim() + ",'" + cmbSexo.Text.Trim() + "')";
                   NombreCliente = txtNombre.Text.Trim();
                   Acceso_Datos.ExecuteQuery(strQuery);
                   MessageBox.Show("Cliente guardado con éxito", "VideoClub");
                   LimpiarCajas();
                }
                else
                    MessageBox.Show("Ya existe un cliente con este número de cédula", "VideoClub");
            }
        }

        mskdTxtBxNoCedula.Focus();
    }
    Buscar = false;
}
```

Formulario Películas



//Código del Botón Guardar

```
private void btnGuardar_Click(object sender, EventArgs e)
{
    int idFormato = 0;
    nuevo = false;
    if (Editar)
    {
        idGenero = cmbGenero.SelectedIndex + 1;
        if (rdbEstreno.Checked)
            estreno = 1;
        else
            estreno = 0;

        if (Editar)//Editar los datos de pelicula
        {
            //Actualizar datos de formato
            for (int i = 0; i < lvFormatos.Items.Count; i++)
            {
                if (lvFormatos.Items[i].SubItems[1].Text != "")
                {
                    strSelect = "SELECT IdFormato FROM Formato
                    WHERE (Nombre = '" + lvFormatos.Items[i].SubItems[0].Text + "')";
                    myDataReader = Acceso_Datos.Select(strSelect);
                    myDataReader.Read();
                    idFormato = int.Parse(myDataReader.GetValue(0).ToString());
                    myDataReader.Close();

                    strSelect = "SELECT Numcopias,Disponibilidad FROM
                    Tiene WHERE IdPelicula =
                    '" + CodPelAnterior + "' AND IdFormato = " + idFormato;
                    myDataReader = Acceso_Datos.Select(strSelect);
                    myDataReader.Read();

                    //La disponibilidad será la nueva disponibilidad menos el numero de copias alquiladas
                    int Ncopias = myDataReader.GetInt32(0);
                    int Ndisp = myDataReader.GetInt32(1);
                    int Pelalq = Ncopias - Ndisp;
                    int disponibilidad = int.Parse(lvFormatos.Items[i].SubItems[1].Text) - Pelalq;
                    myDataReader.Close();
```

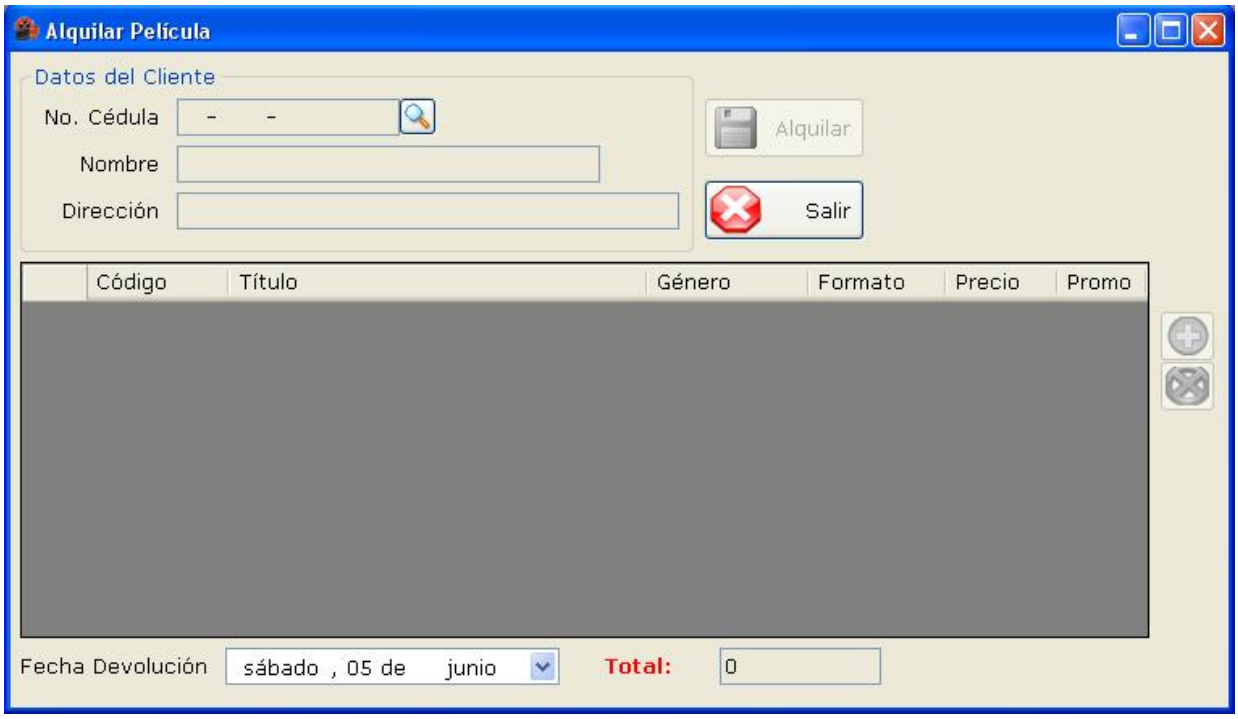
```
                //Actualizar la disponibilidad y el numero de copias
                strQuery = "UPDATE Tiene SET " +
                           "Numcopias = " + lvFormatos.Items[i].SubItems[1].Text + "" +
                           ", Disponibilidad = " + disponibilidad +
                           " WHERE IdPelicula = '" + CodPelAnterior + "' AND IdFormato =
                           " + idFormato;

                Acceso_Datos.ExecuteQuery(strQuery);
            }
        }



        //Actualizar Datos de la Pelicula
        strQuery = "UPDATE Pelicula SET " +
                   "IdPelicula = '" + mskCodPelicula.Text.Trim().ToUpper() + "'" +
                   ", Titulo = '" + txtNombre.Text.Trim().ToUpper() + "'" +
                   ", ActorPrincipal = '" + txtActor.Text.Trim().ToUpper() + "'" +
                   ", IdGenero = " + idGenero +
                   ", Caratula = '" + strPathPicture + "'" +
                   ", Estreno = " + estreno +
                   " WHERE IdPelicula = '" + CodPelAnterior + "'";
        Acceso_Datos.ExecuteQuery(strQuery);


        MessageBox.Show("Registro Editado con éxito", "VideoClub");
        LimpiarCajas();
        Editar = false;
        DesHabCajas(true);
        btnEditar.Enabled = false;
        btnBorrar.Enabled = false;

    }
  }
else //Guardar una nueva pelicula
{
    if (ValidarDatos())
    {
        idGenero = cmbGenero.SelectedIndex + 1;
        if (rdbEstreno.Checked)
            estreno = 1;
        else
            estreno = 0;
        //Ingresar una nueva película
        strQuery = "INSERT INTO Pelicula(IdPelicula, Titulo, ActorPrincipal,
        IdGenero, Caratula, Estreno) " +
        "VALUES ('" + mskCodPelicula.Text.Trim().ToUpper() + "','" +
        txtNombre.Text.Trim().ToUpper() + "','" + txtActor.Text.Trim().ToUpper() +
        "'," + idGenero + ",'" + strPathPicture + "'," + estreno + ")";

        Acceso_Datos.ExecuteQuery(strQuery);

        //Ingresar el numero de copias y disponibilidad para cada formato
        for (int i = 0; i < lvFormatos.Items.Count; i++)
        {
            strSelect = "SELECT IdFormato FROM Formato WHERE (Nombre = '" +
            lvFormatos.Items[i].SubItems[0].Text + "')";
            myDataReader = Acceso_Datos.Select(strSelect);
            myDataReader.Read();
            idFormato = int.Parse(myDataReader.GetValue(0).ToString());
            myDataReader.Close();
```

```
        //si no se ingresan datos para un formato el numero de copias y disponibilidad es = 0
        if (lvFormatos.Items[i].SubItems[1].Text != "")
        {

            strQuery = "INSERT INTO Tiene(IdPelicula,IdFormato,Disponibilidad,Numcopias) " +
                       "VALUES('" + mskCodPelicula.Text.Trim().ToUpper() + "'," + idFormato +
                       "," + lvFormatos.Items[i].SubItems[1].Text + "," +
                       lvFormatos.Items[i].SubItems[1].Text + ")";
            Acceso_Datos.ExecuteQuery(strQuery);
        }

        else
        {

            strQuery = "INSERT INTO Tiene(IdPelicula,IdFormato,Disponibilidad,Numcopias) " +
                       "VALUES('" + mskCodPelicula.Text.Trim().ToUpper() + "'," + idFormato +
                       "," + 0 + "," + 0 + ")";
            Acceso_Datos.ExecuteQuery(strQuery);

        }
    }
    MessageBox.Show("La Película "+ txtNombre.Text.ToUpper() +" ha sido guardada ", "VideoClub");
    LimpiarCajas();
    DesHabCajas(true);
    btnEditar.Enabled = false;
    btnBorrar.Enabled = false;
    btnBuscar.Enabled = true;

  }
 }
}
```

//Código del picturebox Caratula

```
    private void pctbCaratula_Click(object sender, EventArgs e)
    {
        //Tipos de formatos que puede cargar el picturebox
        flCaratula.Filter = "Imágenes JPG (*.jpg)|*.jpg|Mapas de bits (*.bmp)|
        *.bmp|Imágenes PNG (*.png)|*.png|Imágenes GIF (*.gif)|*.gif";
        flCaratula.ShowDialog();
        if (flCaratula.FileName != "")
            pctbCaratula.Load(flCaratula.FileName);
        strPathPicture = flCaratula.FileName;
    }

    private void lvFormatos_MouseDown(object sender, MouseEventArgs e)
    {
        lvi = lvFormatos.GetItemAt(e.X, e.Y);
        X = e.X;
        Y = e.Y;
    }

    private void LimpiarCajas()
    {
        //Despues de guardar limpiar las cajas de texto
        txtActor.Clear();
        mskCodPelicula.Clear();
        txtNombre.Clear();
        cmbGenero.SelectedIndex = -1;

        for (int i = 0; i < lvFormatos.Items.Count; i++)
        {
            lvFormatos.Items[i].SubItems[1].Text = "";
        }

        //Queda pendiente limpiar el picturebox
        pctbCaratula.Image = null;
        rdbEstreno.Checked = false;
    }
```

Formulario Alquilar Película



//Código del botón Alquilar

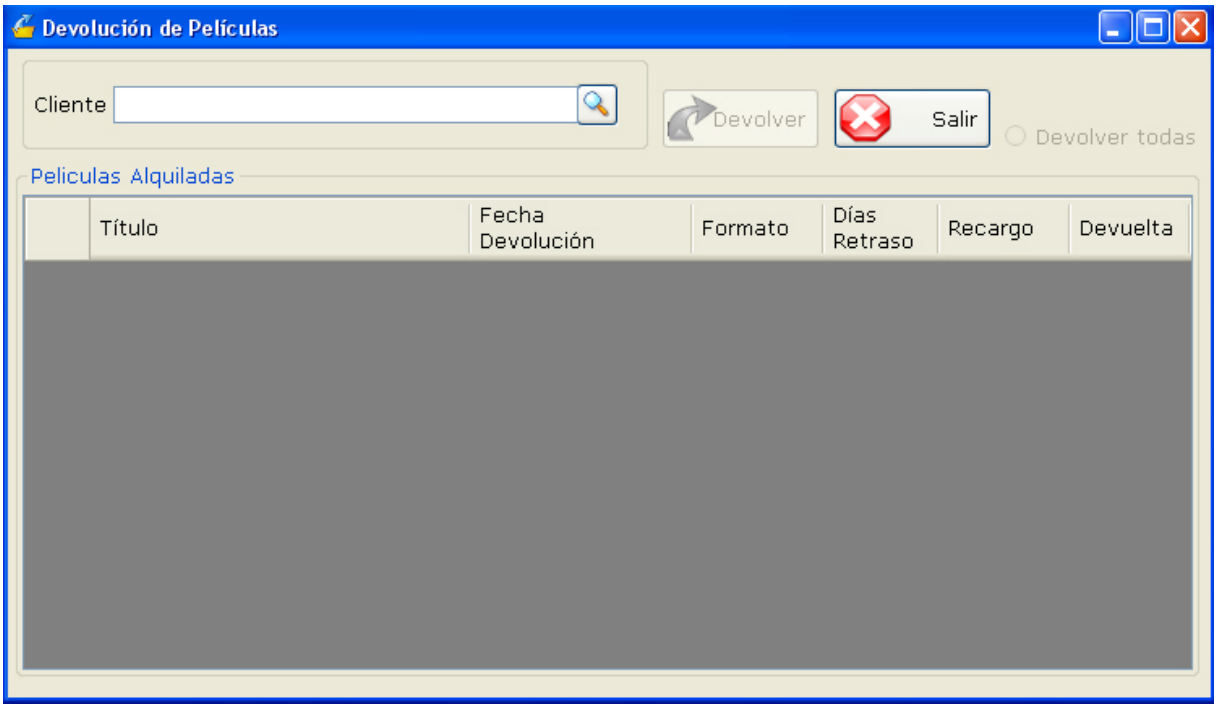
```
private void btnAlquilar_Click(object sender, EventArgs e)
{
    j = 0;
    int state;
    dtpFechaDev.Value = diasDev;
    //Instrucciones para dar formato a las fechas
    fechaAlq = String.Format("{0}-{1}-{2}", DateTime.Today.Year,
    DateTime.Today.Month, DateTime.Today.Day);
    fechaFormato = String.Format("{0:d}", fechaAlq);

    fechaDev = String.Format("{0}-{1}-{2}", dtpFechaDev.Value.Year,
    dtpFechaDev.Value.Month, dtpFechaDev.Value.Day);
    fechaFormato = String.Format("{0:d}", fechaDev);

    fechaDevEs = String.Format("{0}-{1}-{2}", fechaEstreno.Year,
    fechaEstreno.Month, fechaEstreno.Day);
    fechaFormatoEs = String.Format("{0:d}", fechaDevEs);

    //verificar si se ha cambiado la fecha del sistema
    if (DateTime.Now.ToShortDateString() == Program.FechaCorrecta)
    {
        //Verificar estrenos retorna true cuando hay uno o mas estrenos
        if (VerificarEstrenos())
        {
            while (j < dtgPeliculas.RowCount)
            {
                estAlq = 0;
                state = int.Parse(dtgPeliculas.Rows[j].Cells[5].Value.ToString());
     //Si state es 0 la pelicula no es pelicula de promocion, si es 1 la pelicula es de promocion

                //Obtener el monto
                monto = double.Parse(dtgPeliculas.Rows[j].Cells[4].Value.ToString());

                if (dtgPeliculas.Rows[j].Cells[6].Value.ToString() == "True")
                {
                    estAlq = 1;
                    //LLamar funcion Alquilar cuando la pelicula es estreno
                    AlquilarPeliculas(fechaDevEs, monto);
                }

            else
            {
                if (state == 0)
                {
                    //LLamar funcion Alquilar para una pelicula no estreno
                    AlquilarPeliculas(fechaDev, monto);
                }
                else
                {
                    //LLamar función Alquilar para una pelicula promocional el monto es 0
                    AlquilarPeliculas(fechaDev, 0);
                }
            }
            //Actualizar la disponibilidad en la base de datos
            ActualizarDisponibilidad();
            j++;
        }
    }
    else
    {
        estAlq = 0;
```

```
    //Guardar el alquiler
    while (j < dtgPeliculas.RowCount)
    {
      state = int.Parse(dtgPeliculas.Rows[j].Cells[5].Value.ToString());
//Si state es 0 la pelicula no es pelicula de promocion, si es 1 la pelicula es de promocion

        //Obtener el monto
        monto = double.Parse(dtgPeliculas.Rows[j].Cells[4].Value.ToString());
        if (state == 0)
        {
            //LLamar funcion Alquilar cuando no es promocional
            AlquilarPeliculas(fechaDev, monto);
        }
        else
        {
          //LLamar función Alquilar para una pelicula promocional el monto es 0
          AlquilarPeliculas(fechaDev, 0);
        }

        //Actualizar la disponibilidad en la base de datos
        ActualizarDisponibilidad();
        j++;
    }



              //Actualizar la disponibilidad en la base de datos
              ActualizarDisponibilidad();
              j++;
           }

           //Actualizar la fecha de devolución para las peliculas que no son estrenos
           strQuery = "UPDATE Alquila SET FechaDev = '" + fechaDev + "' WHERE NoCedula =
                    '" + cedula + "' AND Estreno = " + 0 + " ";
           Acceso_Datos.ExecuteQuery(strQuery);
        }
        MessageBox.Show("Alquiler Registrado ", "VideoClub", MessageBoxButtons.OK,
        MessageBoxIcon.Information);
        LimpiarCajas();
        HabDesBotones(false);
    }

        //La fecha no es correcta
    else
    {
        MessageBox.Show("Ha cambiado la fecha del sistema debido a esto no se puede guardar el Alquiler",
        "VideoClub",MessageBoxButtons.OK,MessageBoxIcon.Error);
        Close();
    }
}
```

//Código de la función AlquilarPeliculas()

```
private void AlquilarPeliculas(string dateDevol, double monto)
{
    strQuery = "INSERT INTO Alquila(NoCedula, IdPelicula, FechaAlq,
               FechaDev, Monto, Formato, Estreno)" +
               "VALUES ('" + cedula + "','" + dtgPeliculas.Rows[j].Cells[0].Value.ToString()
               + "','" + fechaAlq + "','" + dateDevol + "'," + monto + ",'" +
               dtgPeliculas.Rows[j].Cells[3].Value + "', " + estAlq + " )";
    Acceso_Datos.ExecuteQuery(strQuery);
    if (monto > 0)
    {
        strQuery = "INSERT INTO Ingresos (Fecha, Concepto, Monto)" +
                   "VALUES ('" + fechaAlq + "', 'Alquiler' ," + monto + ")";
        Acceso_Datos.ExecuteQuery(strQuery);
    }

}
```

Formulario Devolución de Películas



//Código Buscar cliente

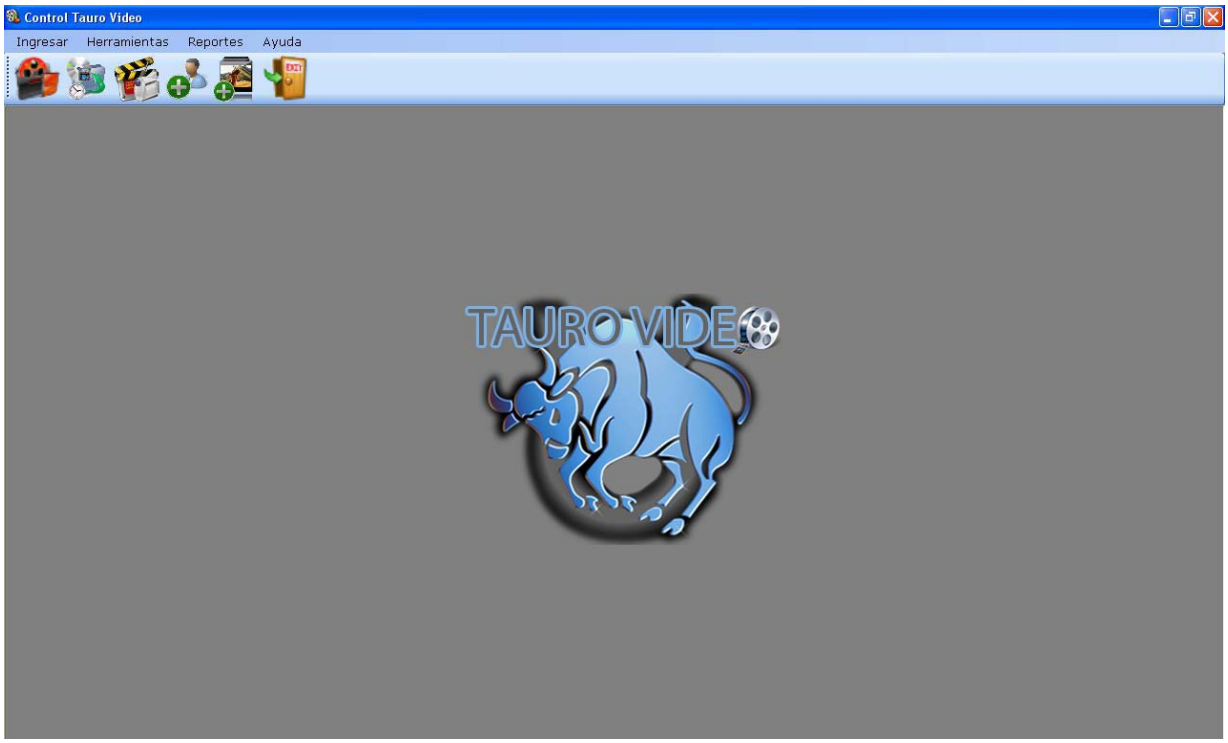
```
 private void btnBuscar_Click(object sender, EventArgs e)
        {
            //Llamar al formulario Clientes con peliculas alquiladas
            ClientePelAlq = new frmBuscarCliAlq();
            ClientePelAlq.ShowDialog();
            if (ClientePelAlq.GetSalir() == false)
            {
                //Obtener los datos
                dtgDevolucion.Rows.Clear();
                Ncedula = ClientePelAlq.GetCedula();
                txtCliente.Text = ClientePelAlq.GetNombre();
                LlenarDataGridView();

            }
            rdbtodas.Enabled = true;
        }
```

```
private void btnDevolver_Click(object sender, EventArgs e)
{
    fechaDev = String.Format("{0}-{1}-{2}", DateTime.Now.Year, DateTime.Now.Month,
             | DateTime.Now.Day);
    nPeliculas = dtgDevolucion.RowCount;
    bool devuelto = false;
    float multa;

    if (dtgDevolucion.RowCount > 0)
    {
        //devolución por cada una de las peliculas
        for (int i = 0; i < nPeliculas; i++)
        {
            if (dtgDevolucion.Rows[i].Cells[8].Value.ToString() == true.ToString())
            {
                Alquiler = DateTime.Parse(dtgDevolucion.Rows[i].Cells[2].Value.ToString());

                //Instrucciones para dar formato a las fechas
                fechaAlq = String.Format("{0}-{1}-{2}", Alquiler.Year, Alquiler.Month, Alquiler.Day);
                fechaFormato = String.Format("{0:d}", fechaAlq);
                fechaFormato = String.Format("{0:d}", fechaDev);
                float monto = float.Parse(dtgDevolucion.Rows[i].Cells[6].Value.ToString());
                IdPel = dtgDevolucion.Rows[i].Cells[0].Value.ToString();
                nombre_formato = dtgDevolucion.Rows[i].Cells[4].Value.ToString();

    //Guardar la devolucion
    strQuery = "INSERT INTO Devolucion(NoCedula, IdPelicula, Formato,FechaAlq,
                FechaDev, Monto) " +
                "VALUES ('" + Ncedula + "', '" + IdPel + "','" + nombre_formato
                + "','" + fechaAlq +
                "','" + fechaDev + "'," + monto + ");";

    Acceso_Datos.ExecuteQuery(strQuery);

    multa = float.Parse(dtgDevolucion.Rows[i].Cells[7].Value.ToString());
    //Si hay multa guardarla
    if (multa > 0)
    {
        strQuery = "INSERT INTO Ingresos (Fecha, Concepto, Monto)" +
                    "VALUES ('" + fechaDev + "', 'Multa por retraso' ," + multa + ")";
        Acceso_Datos.ExecuteQuery(strQuery);
    }
```

```
                //Borrar el registro de alquiler de la tabla Alquila
                strQuery = "DELETE FROM Alquila WHERE ( Alquila.IdPelicula = '" + IdPel + "'
                AND Alquila.NoCedula = '"
                + Ncedula + "' AND Alquila.Formato = '" + nombre_formato + "')";

                Acceso_Datos.ExecuteQuery(strQuery);
                ActualizarDisponibilidad(i);
                devuelto = true;
            }

        }
        if(!devuelto)
        {
            MessageBox.Show("Marque la Casilla de Devolución de al menos una pelicula",
                            "Devolver", MessageBoxButtons.OK);
            return;
        }
        dtgDevolucion.Rows.Clear();

        LlenarDataGridView();
        if (dtgDevolucion.RowCount < 1)
        {
            btnDevolver.Enabled = false;
            txtCliente.Text = "";
        }
    }
    rdbtodas.Checked = false;
}


private void btnSalir_Click(object sender, EventArgs e)
{

    Close();
}

private void LlenarDataGridView()
{
    float multa=0;

    strSelect = "SELECT  Pelicula.IdPelicula, Pelicula.Titulo, Alquila.FechaAlq,
                Alquila.FechaDev, Alquila.Formato,
                Alquila.Monto, Formato.multaXretraso, Alquila.NoCedula " +
                "FROM    Alquila INNER JOIN Pelicula ON Alquila.IdPelicula = Pelicula.IdPelicula
                INNER JOIN Formato ON Alquila.Formato = Formato.Nombre COLLATE
                Modern_Spanish_CI_AI " + "WHERE   (Alquila.NoCedula = '" + Ncedula + "')
                ORDER BY Alquila.FechaDev";

    myDataReader = Acceso_Datos.Select(strSelect);

    while (myDataReader.Read())
    {
        Devolucion = DateTime.Parse( myDataReader.GetValue(3).ToString());
        dias = DateTime.Now.Subtract(Devolucion);
        days = dias.Days;
```

```
            days = dias.Days;

            if (days < 0)
            {
                days = 0;
                multa = 0;
            }
            else
            {
                multa = days * float.Parse(myDataReader.GetDouble(6).ToString());
            }

            RowDev = new DataGridViewRow();
            RowDev.CreateCells(dtgDevolucion);
            RowDev.SetValues(myDataReader.GetString(0), myDataReader.GetString(1),
            DateTime.Parse(myDataReader.GetValue(2).ToString()).ToShortDateString(),
            Devolucion.ToShortDateString(), myDataReader.GetString(4),
            days,myDataReader.GetValue(5).ToString(), multa,0);

            dtgDevolucion.Rows.Add(RowDev);
        }
        btnDevolver.Enabled = true;
        myDataReader.Close();

    }

private void ActualizarDisponibilidad(int j)
{
    //Actualizar la disponibilidad en la base de datos de las peliculas devueltas
    strQuery = "SELECT IdFormato FROM Formato WHERE Formato.Nombre = '" +
    dtgDevolucion.Rows[j].Cells[4].Value + "'";
    myDataReader = Acceso_Datos.Select(strQuery);
    myDataReader.Read();
    int idFormat = int.Parse(myDataReader.GetValue(0).ToString());
    myDataReader.Close();
    strQuery = "SELECT Disponibilidad FROM Tiene WHERE (Tiene.IdPelicula = '" +
    dtgDevolucion.Rows[j].Cells[0].Value.ToString() + "' and Tiene.IdFormato = " + idFormat + ")";
    myDataReader = Acceso_Datos.Select(strQuery);
    myDataReader.Read();
    int disp = int.Parse(myDataReader.GetValue(0).ToString());
    myDataReader.Close();
    disp = disp + 1;
    strQuery = "UPDATE Tiene SET Disponibilidad = " + disp + " WHERE (Tiene.IdPelicula = '" +
    dtgDevolucion.Rows[j].Cells[0].Value.ToString() + "' and Tiene.IdFormato = " + idFormat + ")";
    Acceso_Datos.ExecuteQuery(strQuery);
}
        private void frmDevolucion_Load(object sender, EventArgs e)
        {
            btnDevolver.Enabled = false;
        }

        private void rdbtodas_Click(object sender, EventArgs e)
        {
            int i = 0;
            devolver = !devolver;
            rdbtodas.Checked = devolver;
            while (i < dtgDevolucion.RowCount)
            {
                dtgDevolucion.Rows[i].Cells[8].Value = rdbtodas.Checked;
                dtgDevolucion.Rows[i].Cells[8].ReadOnly = rdbtodas.Checked;
                i++;
            }

        }


    }
```

Formulario Principal



//Código del menú principal y barra de herramientas

```
public partial class frmPrincipal : Form
    {
        internal static bool admin = false;
        string strQuery;
        bool activar = false;
        public frmPrincipal()
        {
            InitializeComponent();
        }

        private void clienteToolStripMenuItem_Click(object sender, EventArgs e)
        {
            MostrarCliente();
        }

        private void películaToolStripMenuItem_Click(object sender, EventArgs e)
        {
            MostrarPelicula();
        }

        private void alquilarToolStripMenuItem_Click(object sender, EventArgs e)
        {
            MostrarAlquilar();
        }
```

```
private void devolverToolStripMenuItem_Click(object sender, EventArgs e)
{
    MostrarDevolver();
}

private void filmaciónToolStripMenuItem_Click(object sender, EventArgs e)
{
    MostrarFilmacion();
}

private void panelDeConrtrolToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmPanelControl control;
    control = new frmPanelControl();
    control.MdiParent = this;
    Acceso_Datos.DesactivarMenus(menuStrip1,toolStrip1, false);
    control.Show();

}

private void Alquilar_Click(object sender, EventArgs e)
{
    MostrarAlquilar();
}

private void Devolver_Click(object sender, EventArgs e)
{
    MostrarDevolver();
}

private void Filmacion_Click(object sender, EventArgs e)
{
    MostrarFilmacion();
}

private void AgregarCliente_Click(object sender, EventArgs e)
{
    MostrarCliente();
}

private void AgregarPelicula_Click(object sender, EventArgs e)
{
    MostrarPelicula();
}

private void Salir_Click(object sender, EventArgs e)
{
    Acceso_Datos.CerrarBase();
    Application.Exit();
}

private void MostrarCliente()
{
    frmAddCliente Cliente;

    Cliente = new frmAddCliente();
    Cliente.MdiParent = this;
    Acceso_Datos.DesactivarMenus(menuStrip1, toolStrip1, false);
    Cliente.Show();
}
```

```
    private void MostrarPelicula()
    {
        frmAddPelicula Pelicula;

        Pelicula = new frmAddPelicula();
        Pelicula.MdiParent = this;
        Acceso_Datos.DesactivarMenus(menuStrip1, toolStrip1, false);
        Pelicula.Show();
    }

    private void MostrarFilmacion()
    {
        frmFilmacion Filmacion;
        Filmacion = new frmFilmacion();
        Filmacion.MdiParent = this;
        Acceso_Datos.DesactivarMenus(menuStrip1, toolStrip1, false);
        Filmacion.Show();
    }



    private void MostrarAlquilar()
    {
        frmAlquilar Alquilar;

        Alquilar = new frmAlquilar();
        Alquilar.MdiParent = this;
        Acceso_Datos.DesactivarMenus(menuStrip1, toolStrip1, false);
        Alquilar.Show();
    }

    private void MostrarDevolver()
    {
        frmDevolucion Devolver;

        Devolver = new frmDevolucion();
        Devolver.MdiParent = this;
        Acceso_Datos.DesactivarMenus(menuStrip1, toolStrip1, false);
        Devolver.Show();
    }


//Codigo para respaldar la base de datos
private void respaldarToolStripMenuItem_Click(object sender, EventArgs e)
{
    bool desea_respaldar = true;
    //poner cursor de relojito mientras respalda
    Cursor.Current = Cursors.WaitCursor;
    if (Program.ServerName == System.Environment.MachineName)
    {
        if (Directory.Exists(@"c:\ Respaldo"))
        {
            if (File.Exists(@"c:\ Respaldo\resp.bak"))
            {
                if (MessageBox.Show(@"Ya habia un respaldo anteriormente ¿desea remplazarlo?", "Respaldo",
                    MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)
                {
                    File.Delete(@"c:\ Respaldo\resp.bak");
                }
                else
                    desea_respaldar = false;

            }

        }
```

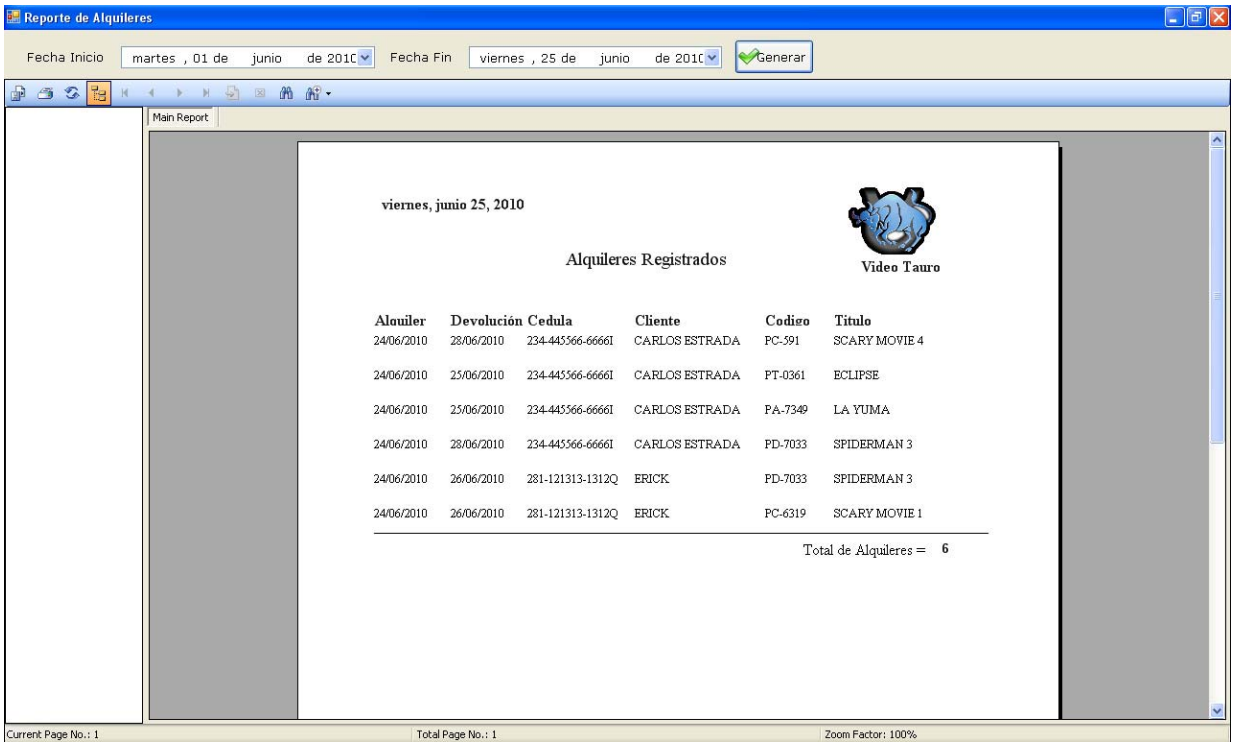
```
    else
        Directory.CreateDirectory(@"c:\ Respaldo");

    if (desea_respaldar)
    {
        //Respaldar la base de Datos usando la funcion ExecuteQuery ---------------
        strQuery = @"backup database VideoClub to disk ='c:\ Respaldo\resp.bak' with init,stats=10";
        Acceso_Datos.ExecuteQuery(strQuery);

        MessageBox.Show("Respaldo de la Base de Datos realizado con exito", "Respaldo",
                        MessageBoxButtons.OK, MessageBoxIcon.Information);

    }
}
else
{
    if (Directory.Exists(@"\\Shalom-pc\Respaldo"))
    {
        if (File.Exists(@"\\Shalom-pc\Respaldo\resp.bak"))
        {
            if (MessageBox.Show(@"Ya habia un respaldo anteriormente ¿desea remplazarlo?", "Respaldo",
                MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)
            {
                File.Delete(@"\\Shalom-pc\Respaldo\resp.bak");
            }
            else
                desea_respaldar = false;

        }

    }

    if (desea_respaldar)
    {
        //Respaldar la base de Datos usando la funcion ExecuteQuery ---------------
        strQuery = @"backup database VideoClub to disk ='\\Shalom-pc\Respaldo\resp.bak' with init,stats=10";
        Acceso_Datos.ExecuteQuery(strQuery);
        MessageBox.Show("Respaldo de la Base de Datos realizado con exito", "Respaldo",
        MessageBoxButtons.OK, MessageBoxIcon.Information);

    }
}

  private void frmPrincipal_MdiChildActivate(object sender, EventArgs e)
  {
      if (activar)
          Acceso_Datos.DesactivarMenus(menuStrip1, toolStrip1, true);
      activar = !activar;


  }

  //Código para restaurar la base de datos
  private void restaurarToolStripMenuItem_Click(object sender, EventArgs e)
  {
      Cursor.Current = Cursors.WaitCursor;
      string myRestoreConnString;
      SqlConnection myRestoreConnection;
      SqlCommand myCommand;
      SqlCommand myRestoreCommand;
      if (Program.ServerName == System.Environment.MachineName)
      {

          myRestoreConnString = "Data Source= (LOCAL);Initial Catalog=master;Integrated Security=True";
          myRestoreConnection = new SqlConnection(myRestoreConnString);
          myRestoreConnection.Open();

          strQuery = "ALTER DATABASE VideoClub SET SINGLE_USER WITH ROLLBACK IMMEDIATE";
          myCommand = new SqlCommand(strQuery, myRestoreConnection);
          myCommand.ExecuteNonQuery();
          myCommand.Dispose();
```

```
    try
    {
        if (File.Exists(@"c:\ Respaldo\resp.bak"))
        {
            if (MessageBox.Show("¿Está seguro de restaurar?", "Respaldo", MessageBoxButtons.YesNo,
                MessageBoxIcon.Question) == DialogResult.Yes)
            {
                strQuery = @"restore database VideoClub from disk = 'c:\ Respaldo\resp.bak' with replace";

                myRestoreCommand = new SqlCommand(strQuery, myRestoreConnection);
                myRestoreCommand.ExecuteNonQuery();

                MessageBox.Show("Se ha restaurado la base de datos", "Restauración",
                MessageBoxButtons.OK, MessageBoxIcon.Information);

            }
        }

        else
            MessageBox.Show(@"No haz hecho ningun respaldo anteriormente (o no está en la ruta correcta)",
            "Restauracion", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }

    catch (Exception exp)
    {
        MessageBox.Show(exp.Message);
    }

}
else
{

    myRestoreConnString = "Data Source=" + Program.ServerName + ";
    User ID=SHALOM;Initial Catalog=master;Trusted_Connection=Yes";
    myRestoreConnection = new SqlConnection(myRestoreConnString);

    myRestoreConnection.Open();

    strQuery = "ALTER DATABASE VideoClub SET SINGLE_USER WITH ROLLBACK IMMEDIATE";
    myCommand = new SqlCommand(strQuery, myRestoreConnection);
    myCommand.ExecuteNonQuery();
    myCommand.Dispose();

  try
  {
      if (File.Exists(@"\\Shalom-pc\Respaldo\resp.bak"))
      {
          if (MessageBox.Show("¿Está seguro de restaurar?", "Respaldo",
              MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)
          {

              strQuery = @"restore database VideoClub from disk = '\\Shalom-pc\Respaldo\resp.bak' with replace";
              myRestoreCommand = new SqlCommand(strQuery, myRestoreConnection);
              myRestoreCommand.ExecuteNonQuery();


              MessageBox.Show("Se ha restaurado la base de datos", "Restauración",
              MessageBoxButtons.OK, MessageBoxIcon.Information);

          }
      }

      else
          MessageBox.Show(@"No haz hecho ningun respaldo anteriormente (o no está en la ruta correcta)",
          "Restauracion", MessageBoxButtons.OK, MessageBoxIcon.Information);
  }

  catch (Exception exp)
  {
      MessageBox.Show(exp.Message);
  }
}
```

```
        strQuery = "ALTER DATABASE VideoClub SET MULTI_USER";
        SqlCommand myCommandrest = new SqlCommand(strQuery, myRestoreConnection);
        myCommandrest.ExecuteNonQuery();
        myCommandrest.Dispose();
        myRestoreConnection.Close();
        Acceso_Datos.AbrirBase();
    }

    private void nuevoUsuarioToolStripMenuItem_Click(object sender, EventArgs e)
    {
        AddUsuario addNewUser = new AddUsuario();
        addNewUser.MdiParent = this;
        addNewUser.Show();
    }

    private void modificarContraseñaToolStripMenuItem_Click(object sender, EventArgs e)
    {
        frmCambiarClave formCambiar = new frmCambiarClave();
        formCambiar.MdiParent = this;
        formCambiar.Show();
    }

    private void frmPrincipal_Load(object sender, EventArgs e)
    {
        if (!admin)
        {
            panelDeConrtrolToolStripMenuItem.Enabled = false;
            nuevoUsuarioToolStripMenuItem.Enabled = false;
            peliculaToolStripMenuItem.Enabled = false;
            AgregarPelicula.Enabled = false;
            borrarUsuarioToolStripMenuItem.Enabled = false;
        }

    }

    private void informeIngresosToolStripMenuItem_Click(object sender, EventArgs e)
    {
        frmReporteIngreso rptIngreso = new frmReporteIngreso();
        rptIngreso.Show();
    }


      private void informeMorososToolStripMenuItem_Click(object sender, EventArgs e)
      {
          frmReporteMorosos rptMorosos = new frmReporteMorosos();
          rptMorosos.Show();
      }

      private void inormeClientesToolStripMenuItem_Click(object sender, EventArgs e)
      {
          frmReporteClientes rptClientes = new frmReporteClientes();
          rptClientes.Show();
      }

      private void informeAlquileresToolStripMenuItem_Click(object sender, EventArgs e)
      {
          frmReporteAlquiler rptAlquiler = new frmReporteAlquiler();
          rptAlquiler.Show();
      }

      private void informeToolStripMenuItem_Click(object sender, EventArgs e)
      {
          frmReporteDevolucion rptDevolucion = new frmReporteDevolucion();
          rptDevolucion.Show();
      }

      private void informePeliculasToolStripMenuItem_Click(object sender, EventArgs e)
      {
          frmReportePeliculas rptPel = new frmReportePeliculas();
          rptPel.Show();
      }
```

```
    private void informeFilmacionesToolStripMenuItem_Click(object sender, EventArgs e)
    {
        frmReporteFilmacion rptFilm = new frmReporteFilmacion();
        rptFilm.Show();
    }



    private void acercaDeToolStripMenuItem1_Click(object sender, EventArgs e)
    {
        frmAcercaDe frmAcerca = new frmAcercaDe();
        frmAcerca.Show();
    }

    private void contenidoDeAyudaToolStripMenuItem_Click(object sender, EventArgs e)
    {
        Help.ShowHelp(this, "C:/ProyectoVideoClub(Nuevo)Trabajndo en red/prjVideoClub/
                            prjVideoClub/ayudavideotauro_tmphhp/ayudavideotauro.chm");
    }

    private void borrarUsuarioToolStripMenuItem_Click(object sender, EventArgs e)
    {
        frmDelUsuario frmBorrarUsuario = new frmDelUsuario();
        frmBorrarUsuario.MdiParent = this;
        frmBorrarUsuario.Show();
    }

    private void informePeliculaMasAlquiladaToolStripMenuItem_Click(object sender, EventArgs e)
    {
        frmReporteMasAlquilada frmMasAlquilada = new frmReporteMasAlquilada();
        frmMasAlquilada.Show();
    }
}
```

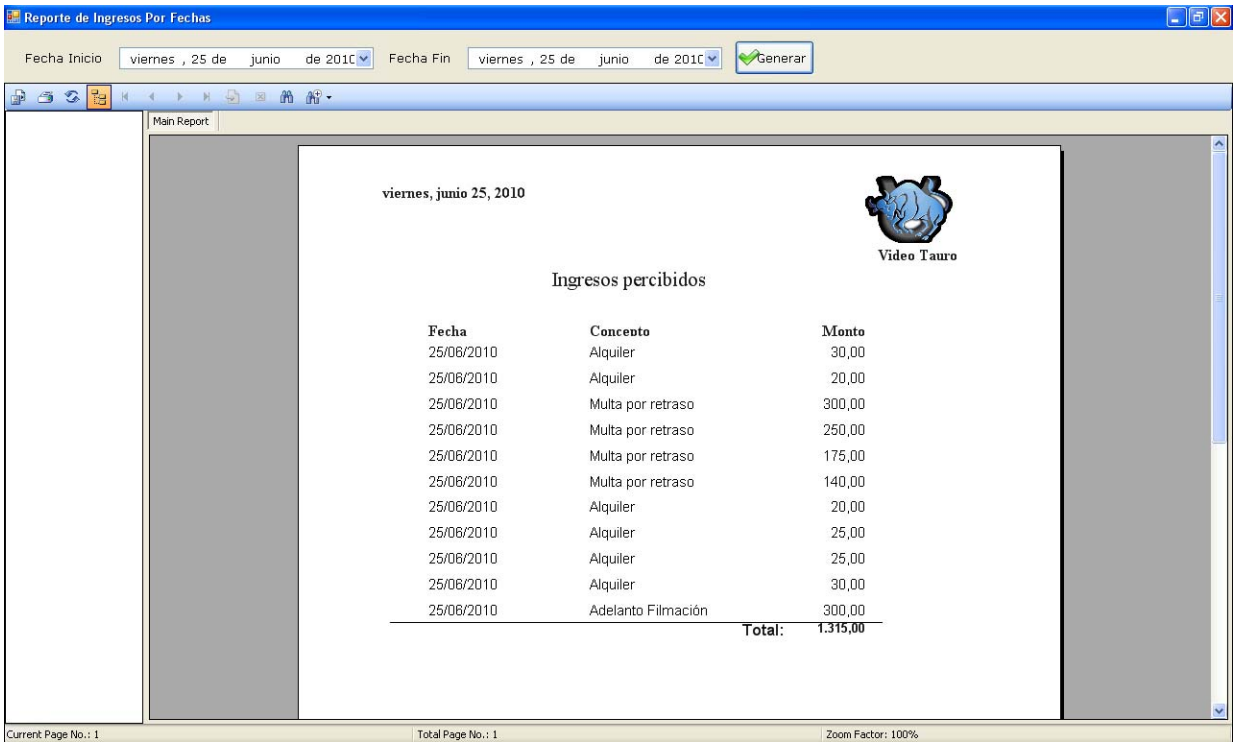
Reporte de Alquileres




```
public partial class frmReporteAlquiler : Form
{
    CRAlquilerPorFechas rptAlquiler = new CRAlquilerPorFechas();
    ParameterValues parametrosFechas = new ParameterValues();
    ParameterDiscreteValue fechaI = new ParameterDiscreteValue();
    ParameterDiscreteValue fechaF = new ParameterDiscreteValue();


    public frmReporteAlquiler()
    {
        InitializeComponent();
    }

    private void btnGenerar_Click(object sender, EventArgs e)
    {
        if (panel2.Controls.Contains(myReportViewer))
        {
            panel2.Controls.Remove(myReportViewer);
            CreateReportViewer();
        }
        else
        {
            CreateReportViewer();
        }
    }

    private void CreateReportViewer()
    {
        myReportViewer.Refresh();
        myReportViewer.Height = 680;
        myReportViewer.Width = 1200;
        Createreport();
        panel2.Controls.Add(myReportViewer);

    }
```

```
private void Createreport()
{

    myReportViewer.ReportSource = Acceso_Datos.reportPath + "\\" + "CRAlquilerPorFechas.rpt";

    CrystalDecisions.CrystalReports.Engine.ReportDocument objReport
   = new CrystalDecisions.CrystalReports.Engine.ReportDocument();

    try
    {
        objReport.Load(Acceso_Datos.reportPath + "\\" + "CRAlquilerPorFechas.rpt");

        fechaI.Value = dtpFechaIni.Value.ToShortDateString();
        fechaF.Value = dtpFechaFin.Value.ToShortDateString();

        parametrosFechas.Add(fechaI);
        rptAlquiler.DataDefinition.ParameterFields["@DateIni"].ApplyCurrentValues(parametrosFechas);
        parametrosFechas.Add(fechaF);

        rptAlquiler.DataDefinition.ParameterFields["@DateFin"].ApplyCurrentValues(parametrosFechas);
        this.myReportViewer.ReportSource = rptAlquiler;

    }

    catch
    {
        myReportViewer.Refresh();
        Createreport();
    }
    finally
    {

        objReport.Dispose();
        objReport.Close();
    }
}
```

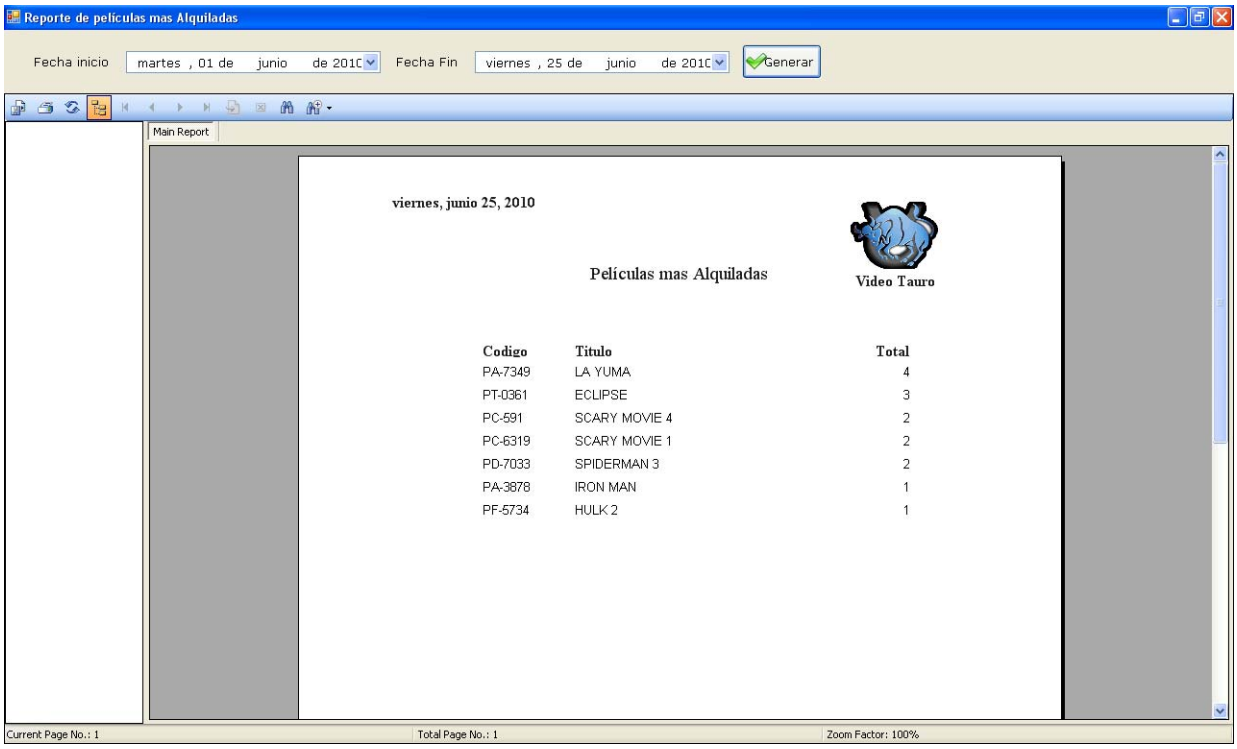
Reporte de Ingresos




```
public partial class frmReporteIngreso : Form
{
    CRIngresoPorFecha rptIngreso = new CRIngresoPorFecha();
    ParameterValues parametrosFechas = new ParameterValues();
    ParameterDiscreteValue fechaI = new ParameterDiscreteValue();
    ParameterDiscreteValue fechaF = new ParameterDiscreteValue();

    public frmReporteIngreso()
    {
        InitializeComponent();
    }

    private void btnGenerar_Click(object sender, EventArgs e)
    {
        if (dockPanel1.Controls.Contains(myReportViewer))
        {
            dockPanel1.Controls.Remove(myReportViewer);
            CreateReportViewer();
        }
        else
            CreateReportViewer();
    }

    private void CreateReportViewer()
    {
        myReportViewer.Refresh();
        myReportViewer.Height = 680;
        myReportViewer.Width = 1200;
        Createreport();
        dockPanel1.Controls.Add(myReportViewer);

    }
```

```
private void Createreport()
{

    //agregar la ruta del cristal report
    myReportViewer.ReportSource = Acceso_Datos.reportPath + "\\" + "CRIngresoPorFecha.rpt";
    CrystalDecisions.CrystalReports.Engine.ReportDocument objReport
    = new CrystalDecisions.CrystalReports.Engine.ReportDocument();

    try
    {
        //agregar la ruta del cristal report
        objReport.Load(Acceso_Datos.reportPath + "\\" + "CRIngresoPorFecha.rpt");


        fechaI.Value = dtpFechaIni.Value.ToShortDateString();
        fechaF.Value = dtpFechaFin.Value.ToShortDateString();

        parametrosFechas.Add(fechaI);
        rptIngreso.DataDefinition.ParameterFields["@fechaIn"].ApplyCurrentValues(parametrosFechas);
        parametrosFechas.Add(fechaF);

        rptIngreso.DataDefinition.ParameterFields["@fechaFin"].ApplyCurrentValues(parametrosFechas);
        this.myReportViewer.ReportSource = rptIngreso;
    }
    catch
    {
        myReportViewer.Refresh();
        Createreport();
    }
    finally
    {

        objReport.Dispose();
        objReport.Close();
    }
}
```

Reporte de Películas más alquiladas




```
public partial class frmReporteMasAlquilada : Form
{
    CRMasAlquilada rptMasAlquilada = new CRMasAlquilada();
    ParameterValues parametrosFechas = new ParameterValues();
    ParameterDiscreteValue fechaI = new ParameterDiscreteValue();
    ParameterDiscreteValue fechaF = new ParameterDiscreteValue();
    public frmReporteMasAlquilada()
    {
        InitializeComponent();
    }

    private void btnGenerar_Click(object sender, EventArgs e)
    {
        if (panel2.Controls.Contains(myReportViewer))
        {
            panel2.Controls.Remove(myReportViewer);
            CreateReportViewer();
        }
        else
            CreateReportViewer();
    }

    private void CreateReportViewer()
    {
        myReportViewer.Refresh();
        myReportViewer.Height = 680;
        myReportViewer.Width = 1200;
        Createreport();
        panel2.Controls.Add(myReportViewer);

    }
```

```
private void Createreport()
{

    //Add your Crystalreport file path.

    myReportViewer.ReportSource = Acceso_Datos.reportPath + "\\" + "CRMasAlquilada.rpt";
    CrystalDecisions.CrystalReports.Engine.ReportDocument objReport
    = new CrystalDecisions.CrystalReports.Engine.ReportDocument();

    try
    {
        //Add your Crystalreport file path.
        objReport.Load(Acceso_Datos.reportPath + "\\" + "CRMasAlquilada.rpt");


        fechaI.Value = dtpFechaInicio.Value.ToShortDateString();
        fechaF.Value = dtpFechaFin.Value.ToShortDateString();

        parametrosFechas.Add(fechaI);
        rptMasAlquilada.DataDefinition.ParameterFields["@fechaIn"].ApplyCurrentValues(parametrosFechas);
        parametrosFechas.Add(fechaF);

        rptMasAlquilada.DataDefinition.ParameterFields["@fechaFin"].ApplyCurrentValues(parametrosFechas);
        this.myReportViewer.ReportSource = rptMasAlquilada;
    }
    catch
    {
        myReportViewer.Refresh();
        Createreport();
    }
    finally
    {

        objReport.Dispose();
        objReport.Close();
    }
}
```

//Código de la función Respaldar Base de Datos

```
//Codigo para respaldar la base de datos
private void respaldarToolStripMenuItem_Click(object sender, EventArgs e)
{
    bool desea_respaldar = true;
    //poner cursor de relojito mientras respalda
    Cursor.Current = Cursors.WaitCursor;
    //si es un servidor local
    if (Program.ServerName == System.Environment.MachineName)
    {
        if (Directory.Exists(@"c:\ Respaldo"))
        {
            if (File.Exists(@"c:\ Respaldo\resp.bak"))
            {
                if (MessageBox.Show(@"Ya habia un respaldo anteriormente ¿desea remplazarlo?",
                "Respaldo", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)
                {
                    File.Delete(@"c:\ Respaldo\resp.bak");
                }
                else
                    desea_respaldar = false;

            }

        }
        else
            Directory.CreateDirectory(@"c:\ Respaldo");

        if (desea_respaldar)
        {
            //Respaldar la base de Datos usando la funcion ExecuteQuery ----------------
            strQuery = @"backup database VideoClub to disk =
                        'c:\ Respaldo\resp.bak' with init,stats=10";
            Acceso_Datos.ExecuteQuery(strQuery);

            MessageBox.Show("Respaldo de la Base de Datos realizado con exito",
                        "Respaldo", MessageBoxButtons.OK, MessageBoxIcon.Information);

        }
    }

    //cuando es un servidor remoto
  else
  {
      if (Directory.Exists(@"\\Shalom-pc\Respaldo"))
      {
          if (File.Exists(@"\\Shalom-pc\Respaldo\resp.bak"))
          {
              if (MessageBox.Show(@"Ya habia un respaldo anteriormente ¿desea remplazarlo?",
              "Respaldo", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)
              {
                  File.Delete(@"\\Shalom-pc\Respaldo\resp.bak");
              }
              else
                  desea_respaldar = false;

          }

      }

      if (desea_respaldar)
      {
          //Respaldar la base de Datos usando la funcion ExecuteQuery ----------------
          strQuery = @"backup database VideoClub to disk ='\\Shalom-pc\Respaldo\resp.bak' with init,stats=10";
          Acceso_Datos.ExecuteQuery(strQuery);
          MessageBox.Show("Respaldo de la Base de Datos realizado con exito",
                      "Respaldo", MessageBoxButtons.OK, MessageBoxIcon.Information);

      }
  }
}
```

//Código de la función Restaurar Base de datos

```
//Código para restaurar la base de datos
private void restaurarToolStripMenuItem_Click(object sender, EventArgs e)
{
    Cursor.Current = Cursors.WaitCursor;
    string myRestoreConnString;
    SqlConnection myRestoreConnection;
    SqlCommand myCommand;
    SqlCommand myRestoreCommand;
    if (Program.ServerName == System.Environment.MachineName)
    {
        //abrir una conexión temporal con la base de datos master
        myRestoreConnString = "Data Source= (LOCAL);Initial Catalog=master;Integrated Security=True";
        myRestoreConnection = new SqlConnection(myRestoreConnString);
        myRestoreConnection.Open();

        //establecer a la base de datos VideoClub un unico usuario
        strQuery = "ALTER DATABASE VideoClub SET SINGLE_USER WITH ROLLBACK IMMEDIATE";
        myCommand = new SqlCommand(strQuery, myRestoreConnection);
        myCommand.ExecuteNonQuery();
        myCommand.Dispose();

        try
        {
            if (File.Exists(@"c:\ Respaldo\resp.bak"))
            {
                if (MessageBox.Show("¿Está seguro de restaurar?", "Respaldo",
                MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)
                {
                    strQuery = @"restore database VideoClub from disk = 'c:\ Respaldo\resp.bak' with replace";

                    myRestoreCommand = new SqlCommand(strQuery, myRestoreConnection);
                    myRestoreCommand.ExecuteNonQuery();

                    MessageBox.Show("Se ha restaurado la base de datos", "Restauración",
                        MessageBoxButtons.OK, MessageBoxIcon.Information);

                }
            }

            else
                MessageBox.Show(@"No haz hecho ningun respaldo anteriormente (o no está en la ruta correcta)",
                "Restauracion", MessageBoxButtons.OK, MessageBoxIcon.Information);
            }

            catch (Exception exp)
            {
                MessageBox.Show(exp.Message);
            }

        }
        else
        {

            myRestoreConnString = "Data Source=" + Program.ServerName + ";
            User ID=SHALOM;Initial Catalog=master;Trusted_Connection=Yes";
            myRestoreConnection = new SqlConnection(myRestoreConnString);

            myRestoreConnection.Open();

            strQuery = "ALTER DATABASE VideoClub SET SINGLE_USER WITH ROLLBACK IMMEDIATE";
            myCommand = new SqlCommand(strQuery, myRestoreConnection);
            myCommand.ExecuteNonQuery();
            myCommand.Dispose();
```

```
    try
    {
        if (File.Exists(@"\\Shalom-pc\Respaldo\resp.bak"))
        {
            if (MessageBox.Show("¿Está seguro de restaurar?", "Respaldo",
            MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)
            {

                strQuery = @"restore database VideoClub from disk =
                            '\\Shalom-pc\Respaldo\resp.bak' with replace";
                myRestoreCommand = new SqlCommand(strQuery, myRestoreConnection);
                myRestoreCommand.ExecuteNonQuery();


                MessageBox.Show("Se ha restaurado la base de datos", "Restauración",
                    MessageBoxButtons.OK, MessageBoxIcon.Information);

            }
        }

        else
            MessageBox.Show(@"No haz hecho ningun respaldo anteriormente (o no está en la ruta correcta)",
            "Restauracion", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }

    catch (Exception exp)
    {
        MessageBox.Show(exp.Message);
    }


}

//establecer la base de datos VideoClub como multi usuario
strQuery = "ALTER DATABASE VideoClub SET MULTI_USER";
SqlCommand myCommandrest = new SqlCommand(strQuery, myRestoreConnection);
myCommandrest.ExecuteNonQuery();
myCommandrest.Dispose();
myRestoreConnection.Close();
Acceso_Datos.AbrirBase();
```