Universidad de las Ciencias Informáticas

FACULTAD 6

Título: Subsistema distribuido de transmisión de contenido audiovisual.

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE

INGENIERO EN CIENCIAS INFORMÁTICAS

Autores:

Raúl Cabrera Miranda.

Nathalie María Salas Torres**.**

Tutora:

Ing. Mónica Vigil Martínez.

La Habana, Junio de 2013

"Año 54 de la Revolución"

". . Si los jóvenes fallan, todo fallará. Es mi más profunda convicción que la juventud cubana luchará por impedirlo. Creo en ustedes."

Fidel Castro Ruz

Declaración de autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Raúl Cabrera Miranda

_____

Firma del Autor

Nathalie M. Salas Torres

_____

Firma del Autor

Ing. Mónica Vigil Martínez

_____

Firma del Tutor

Datos de contacto

Tutora: Ing. Mónica Vigil Martínez.

Institución: Universidad de las Ciencias Informáticas.

Dirección de la institución: Carretera a San Antonio de los Baños, Km. 2 ½, Reparto: Torrens.

Municipio: La Lisa.

Provincia: La Habana.

Correo electrónico: mvigil@uci.cu

Cargo del trabajador: Profesor.

Título de la especialidad de graduado: Ingeniería en Ciencias Informáticas.

Año de graduación: 2012.

Institución donde se graduó: Universidad de las Ciencias Informáticas.

Agradecimientos

Nathalie

A la Revolución por darme la oportunidad de estudiar y desarrollar todo mi intelecto.

A mi abuelo Jorge por confiar en mí y brindarme siempre su amor mientras la vida se lo permitió.

A mi mamá, que siempre estuvo ahí dándome aliento cada vez que lo necesitaba.

A mi abuelita querida Olguita por confiar en mí y apoyarme siempre.

A toda mi familia por el apoyo y amor que siempre me han dado, por ser mis mejores maestros. En especial a mis tíos Olga Lidia y Ariel.

A mi madrina Baby por apoyarme siempre, aún en los momentos más difíciles.

A Darian por ser el amor de mi vida, por brindarme su apoyo y amor y sobre todas las cosas por soportarme.

A mis suegros por ser mis segundos padres, por el cariño que siempre me han dado y apoyarme en todos los momentos difíciles por los que pasé.

A todas mis amistades, en especial a todos los que me ayudaron en la realización de este trabajo de diploma.

A todos los profesores del proyecto Sistema de Transmisión de Canales Virtuales por sus oportunas sugerencias.

A mi tutora Mónica por su ayuda.

Al tribunal por sus sugerencias.

A todas aquellas personas que de una forma u otra aportaron su granito de arena en la realización de este trabajo de diploma.

Raúl

A mis padres y en especial a mi madre por darme siempre su apoyo y confiar en mí.

A mi novia, hermana y demás familiares.

A mis profesores y en especial a mi tutora Mónica.

A todos aquellos que me ayudaron y apoyaron a lo largo de mi carrera y mi vida.

Dedicatoria

Nathalie

Dedico este trabajo de diploma a las personas más importantes en mi vida, que han creído en mí y han hecho todo lo posible por verme hecha una ingeniera.
A mi abuelo Jorge que aunque por cuestiones de la vida no pudo estar aquí conmigo compartiendo mi felicidad, sé que donde esté se siente muy orgulloso de mi.
A mi mamá, mi abuela, mis tíos, mi novio y mi familia en general.

Raúl

Todo el esfuerzo y las horas de empeño para lograr realizar este trabajo quiero dedicarlas a mi mamá, por hacer de mi lo que soy.

Resumen

El producto Sistema de Transmisión de Canales Virtuales (STCV), constituye una plataforma compuesta por módulos que deberán desplegarse integrados. Uno de estos módulos lo constituye el subsistema de transmisión, el cual se encarga de realizar la transmisión de los contenidos audiovisuales utilizando tecnología streaming. El presente trabajo tiene como objetivo desarrollar un subsistema distribuido para la transmisión de contenido audiovisual (SDTCA) que responda correctamente a la alta demanda concurrente de reproducción de medias en el producto antes mencionado, basado en los principales protocolos y formatos de video para realizar streaming. Este documento se encuentra estructurado en cuatro capítulos que recogen las diferentes etapas de desarrollo del subsistema, las cuales van desde la fundamentación teórica de la investigación, a través de la propuesta y el modelado de la solución hasta llegar a la implementación y validación de la aplicación.

Palabras claves

Audiovisual, distribuido, streaming, subsistema, transmisión.

Indice de contenido

Indice de tablas

Indice de figuras

Introducción

A partir del incesante avance de la ciencia y la tecnología, la comunicación dejó de ser exclusivamente oral para desarrollarse a través de otros medios, como la prensa, la radio, el cine y la televisión. Desde sus inicios la televisión se ha convertido en el medio de comunicación por excelencia y su importancia no ha dejado de crecer en todo el mundo (Corona, 2009). La auténtica revolución que supuso en sus orígenes no es comparable con ningún otro sistema de comunicación. Su influencia en los más variados aspectos de la vida es algo que se ha convertido en una realidad más que evidente. Este medio de comunicación ofrece un sin número de programas dirigidos a la sociedad, empleándose como medio de entretenimiento o como una vía para difundir información.

La televisión hasta tiempos recientes, se transmitía solamente de forma analógica y su modo de llegar a los televidentes era mediante el aire a través de ondas de radio. Como resultado del constante desarrollo de la misma y el presuroso impulso de las redes, las tecnologías y la necesidad de automatizar un conjunto de procesos, se convirtió en un enorme desafío el desarrollo computarizado de este medio de comunicación, surgiendo así la televisión digital. Transmitir programas televisivos en forma binaria permite consumir menor ancho de banda y recursos en relación con la televisión analógica, por lo que las estaciones de televisión proveen imágenes más claras y con mejor calidad de sonido. Con el aumento actual de las velocidades de conexión a Internet, el incremento del número total de internautas[1] y la disminución en gastos de conexión se ha hecho cada vez más común encontrar accesible en la web contenidos tradicionales de televisión.

La influencia de Internet en la televisión sigue una tendencia notablemente creciente. El porcentaje de personas que ven televisión utilizando esta vía se ha incrementado en los últimos años, ocupando el primer lugar con respecto a otros medios de comunicación como la prensa y la radio. En el mundo actual los tres países que poseen un elevado número de espectadores que observan video en Internet son: Brasil con un 89%, España con 86% y Estados Unidos con un 80% respectivamente (Eiken, 2011).

[1] Persona que navega en Internet visitando páginas web o cualquier persona que haciendo uso de una aplicación en un ordenador obtiene información de Internet o interactúa con otras personas.

Inicialmente los usuarios lograban descargar y reproducir contenidos multimedia a través de la red, resultando imprescindible obtener todo el archivo grabado en disco para luego hacer uso del mismo. Dichos archivos eran cada vez más ricos en cuanto a calidad y contenido, por lo que comenzaron a ocupar más espacio, provocando que los tiempos de espera para su descarga comenzaran a ser cada vez mayores. Es por dicha razón que nace la tecnología streaming, para permitir la distribución de contenido multimedia a través de una red (Novoa, 2007). Para difundir este tipo de contenido es necesario un servidor de streaming, que permita gestionar las conexiones y anchos de banda que los usuarios conectados estén solicitando, solucionando en gran parte el problema de difusión de contenido multimedia en Internet.

En los últimos tiempos ha surgido una tendencia que ha provocado el aumento del número de usuarios que descargan y reproducen enormes volúmenes de medias de la red, causando que muchos servidores no soporten esa cantidad de conexiones. Además, las aplicaciones que transmiten archivos multimedia están sujetas a variaciones en la red y a cambios en el rendimiento de los componentes que están implicados. Un servidor streaming consigue tratar las variaciones en la red con el uso de la técnica de buffering pero no controla el rendimiento de los componentes. Estos pueden ser tratados con un alto grado de eficiencia con la utilización de un sistema distribuido puesto que los mismos poseen una capacidad para crecer sin aumentar su complejidad ni disminuir su rendimiento. Este tipo de sistema ofrece una enorme confiabilidad: al distribuir el contenido multimedia, ya que incluye la posibilidad de seguir operando correctamente ante cualquier fallo que ocurra en alguno de sus componentes.

Cuba desde hace algunos años se encuentra inmersa en un proceso de informatización, ejemplo de ello lo constituye la Universidad de Ciencias Informáticas (UCI), en la cual en cada una de sus facultades se desarrollan productos que satisfacen las necesidades de disímiles clientes. El STCV es uno de los productos del Centro de Geoinformática y Señales Digitales (GEYSED) de la Facultad 6, específicamente del Departamento de Señales Digitales. El mismo es el encargado de automatizar los procesos de planificación y transmisión de canales virtuales que se ejecutan en una entidad dedicada a la gestión y transmisión de contenidos audiovisuales. En dicho producto, el subsistema de trasmisión televisiva está desarrollado de forma centralizada, por lo que se encuentra instalado en una única computadora, la cual es responsable de controlar la realización de todas las transmisiones utilizando el protocolo UDP[2].

[2] Proporciona una comunicación muy sencilla entre las aplicaciones de varios ordenadores.

Además debe tratar el flujo haciendo función de servidor y cliente a la vez por cada canal, lo que provoca que deba atender las recepciones y envío de muchos canales de forma simultánea. Esto trae como consecuencia que se sature el sistema en caso de existir una alta demanda de materiales audiovisuales de forma concurrente y de aumentar el número de canales a transmitir, provocando inestabilidad en el flujo de datos, así como mala visualización y audición de las medias transmitidas. Este subsistema no cuenta además con funcionalidades como la opción de insertar en los videos subtítulos o infocintas.

A partir de la **situación problémica** planteada, se formula el siguiente **problema a resolver:** ¿Cómo contribuir de manera concurrente y con alta demanda a la reproducción de materiales audiovisuales en el producto STCV para que el flujo de streaming encargado de la transmisión de un canal sea estable? Se define como **objeto de estudio:** Los sistemas distribuidos y el streaming de contenido audiovisual y como **campo de acción:** Subsistema distribuido de transmisión de contenido audiovisual para el producto STCV. Para dar solución a la problemática planteada se tiene como **objetivo general:** Desarrollar un subsistema distribuido para la transmisión de contenido audiovisual que responda correctamente a la alta demanda concurrente de reproducción de medias en el producto STCV, basado en los principales protocolos y formatos de video para realizar streaming.

Se tiene como **idea a defender**: La implementación de un subsistema distribuido para la transmisión televisiva del producto STCV, permitirá responder correctamente a la alta demanda de reproducción de contenido audiovisual de forma concurrente.

Para lograr los objetivos trazados se realizan las siguientes **tareas de investigación**:

- Elaborar el marco teórico sobre transmisión de contenido audiovisual.
- Seleccionar la metodología, herramientas y el lenguaje de programación a utilizar.
- Modelar el subsistema de transmisión televisiva.
- Implementar el subsistema de transmisión televisiva.
- Validar la solución propuesta.

Para posibilitar el cumplimiento del objetivo propuesto fueron usados los siguientes métodos científicos:

Métodos teóricos:

- **Analítico-Sintético:** Se sintetizan los aspectos más relevantes para la investigación, referente a temas relacionados con las características principales, particularidades y conceptos fundamentales referentes a los sistemas distribuidos y la tecnología streaming.

- **Análisis Histórico-Lógico:** Se realiza un profundo análisis referente al surgimiento y desarrollo de los sistemas distribuidos y la tecnología streaming, con el objetivo de adquirir un mayor conocimiento referente al funcionamiento de los mismos.

- Para la realización de los métodos teóricos **Analítico-Sintético** y **Análisis Histórico-Lógico** se utilizó la técnica análisis documental, con el objetivo de obtener información sobre el objeto de estudio y campo de acción en documentos normativos propios de la institución como manuales y actas.

- **Modelación:** Se utilizó para reflejar la estructura, relaciones internas y características del subsistema a través de diagramas, mediante el lenguaje unificado de modelado (UML).

Métodos empíricos:

- **Entrevista**: Se realizó una entrevista con el propósito de adquirir información acerca del tema de investigación y la problemática existente en el subsistema de trasmisión actual. Los aspectos específicos tratados durante la entrevista fueron: los formatos y códec[3] que pueden ser empleados para la transmisión de contenido audiovisual a través de la red. Se tomó como personal a entrevistar a profesores pertenecientes al proyecto STCV y al Departamento Señales Digitales. De un total de 8 personas, las cuales representan la población, se seleccionó para aplicar la entrevista una muestra de 4 profesores, lo cual representa el 50 % de la población seleccionada (Ver Anexo1).

[3]Abreviatura de codificador-decodificador. Describe una especificación desarrollada en software, hardware o una combinación de ambos, capaz de transformar un archivo con un flujo de datos (stream) o una señal. Los códecs pueden codificar el flujo o la señal (a menudo para la transmisión, el almacenaje o el cifrado) y recuperarlo o descifrarlo del mismo modo para la reproducción o la manipulación en un formato más apropiado para estas operaciones.

Capítulo 1: Fundamentación teórica del SDTCA

Introducción

En el presente capítulo se hace referencia a las demostraciones teóricas relacionadas con el proceso de transmisión de contenido audiovisual. Para un mejor entendimiento del contenido de la investigación, se realiza un estudio referente a los principales conceptos relacionados con el dominio del problema y la descripción del objeto de estudio, así como el análisis de las soluciones desarrolladas hasta el momento con características similares al subsistema que se desea desarrollar.

1.1 Conceptos asociados al dominio del problema

Un **servidor** es un tipo de software que realiza ciertas tareas, que tienen como máximo objetivo brindar servicios a los usuarios. Este término también se utiliza para referirse a un ordenador físico en el cual se encuentran instaladas aplicaciones, cuyo propósito es proveer datos de modo que los clientes puedan utilizarlos. Los **servidores streaming,** por su parte, se encargan de distribuir archivos multimedia por la red (Quintero, 2006) y según el tipo de contenido que gestionan podrán recepcionar peticiones que permiten adelantar, retrasar, pausar, detener y reproducir el flujo de información enviado. Esta tecnología permite recibir el contenido audiovisual de manera ininterrumpida, utilizando una capacidad de la memoria RAM en el ordenador cliente que almacena paulatinamente porciones de la media que se está visualizando. Este tipo de técnica se denomina **buffering**.

Para la distribución de archivos multimedia a través de la red, por cuestiones de seguridad y considerando la importancia que trae consigo la distribución de los mismos, resulta imprescindible la utilización de una regla o estándar que permita establecer correctamente la comunicación entre los ordenadores involucrados (emisor y receptor). Esta regla o patrón se denomina **protocolo** (Torres, 2009)**.**

Los datos multimedia que se envían por la red son codificados en un formato que pueda ser entendido tanto por el servidor como por la aplicación cliente, permitiendo comprimir el archivo de tal forma que no sobrecargue la red y afecte la comunicación entre los ordenadores que estén involucrados en el proceso de transmisión. El estado de la compresión es inversamente proporcional al estado de la calidad, por lo que se debe asegurar obtener un balance entre ambos aspectos para adquirir un archivo de reducido tamaño y al mismo tiempo con una calidad visual y auditiva aceptable. Es por ello que resulta imprescindible la utilización de los **códec** (Colectivo de autores de la Universidad de Boston, 2010). Para

la codificación se hace necesario tener en cuenta el tipo de dato a codificar, utilizando para cada uno un tipo de códec diferente. El **códec de audio** está diseñado para la compresión y descompresión de señales de sonido audible para el ser humano, por ejemplo, música o archivos de voz, este códec cumple fundamentalmente la función de reducir la cantidad de datos digitales necesarios para reproducir una señal auditiva (González, 2009). Mientras que los **códec de video** se encargan de comprimir y descomprimir video digital, con el objetivo de obtener un almacenamiento sustancialmente menor de la información de video (Saínz, 2007).

La transmisión de los contenidos multimedia puede significar un reto en la forma de establecer la conexión con otra computadora, existiendo diferentes formas para ello. Una de ellas es la conexión punto a punto o **Unicast**, donde el cliente establece un enlace con el servidor y recibe o envía una cadena de datos independiente a las que se estén trabajando con otros clientes. Además de crear un canal exclusivo con el servidor, viéndose como un carril por separado, sumándose el consumo de ancho de banda en dependencia del momento en el que estos se conectan. Otra forma de conexión es la denominada **Multicast,** donde una sola copia de la información multimedia se envía a una dirección (dirección Multicast) que puede corresponder a un router con capacidad de replicar la cadena de datos, sin consumo superior del ancho de banda. Es recomendable utilizar esta vía de conexión cuando el ancho de banda es reducido o bien la cantidad de accesos esperados supera la capacidad de conexión a la red local o a Internet del o los servidores involucrados (Cicileo, 2010).

1.2 Streaming de contenido audiovisual

El acceso de los usuarios a Internet y la posibilidad de adquirir hardware cada vez más potentes son factores que impulsan el proceso de difusión de contenidos multimedia. Con el surgimiento y evolución de la red de redes, la transmisión de materiales audiovisuales en tiempo real y bajo demanda ha marcado un hito en la historia de los medios de difusión. Resulta común en la actualidad que los usuarios disfruten de la posibilidad de visualizar y escuchar archivos de audio y video desde cualquier parte del mundo con solo contar con un ordenador con acceso a Internet.

La tecnología streaming posibilita la comunicación entre diferentes usuarios de una forma cómoda. Esta técnica de transferencia de datos, permite procesar la información de forma rápida y constante, permitiendo publicar videos y mensajes audiovisuales en la red sin que el usuario espere que se cargue completamente el contenido. Posibilitando ver dicho contenido antes que termine la transmisión,

proporcionándole la posibilidad de avanzar hasta el punto que desee del mismo, ajustándose a las circunstancias de ese momento, como la velocidad de conexión o el tráfico en la red. Esta tecnología por su parte permite la interactividad y un control más preciso de la reproducción, desarrollando aplicaciones audiovisuales que permiten enlazar documentos de similares características creando hipervideo[4] (Bernal, 2010).

La filosofía del streaming consiste en dividir un archivo grande en pequeños paquetes que se pueden enviar de forma continua, para que el destinatario comience a reproducir el archivo en cuanto llena su buffer de entrada. Los paquetes son almacenados solamente el tiempo necesario para reproducirlos, por lo que estos no quedan almacenados en el disco duro. Por otra parte, el servidor debe ser capaz de informar al cliente de los formatos de audio y video, para que el mismo pueda descomprimir y reproducir el contenido correctamente. El contenido a reproducir no tiene que ser necesariamente un archivo almacenado localmente, sino que puede ser una fuente de captura en tiempo real (Novoa, 2007).

1.2.1 Tipos de streaming

El proceso de streaming se puede dividir en dos categorías en función de cómo se obtiene la información a difundir: **streaming en directo o en vivo** y **streaming bajo demanda**.

- El **streaming en directo o en vivo** es aquel que transmite eventos que están sucediendo justo en el momento de la difusión. Por ejemplo, la transmisión de conciertos o clases, son eventos que típicamente se difunden usando este tipo de streaming. La transmisión de radio y televisión por Internet también tiene estas características, aunque en ocasiones la información que se difunde, no parte de un evento en directo (por ejemplo, un programa que ha sido grabado previamente, pero que se va a difundir en un momento determinado) (Novoa, 2007).

- El **streaming bajo demanda**, permite obtener acceso a los contenidos audiovisuales de forma personalizada, ofreciendo la posibilidad de solicitar y visualizar un material concreto en el momento exacto que el telespectador lo desee, por lo que existe la posibilidad de observar dicho material en tiempo real (Torres, 2009). La transmisión del medio comienza desde el inicio del evento a ser

[4] Es una secuencia de video que permite la navegación entre el video y otros elementos hipermedia.

reproducido para cada uno de los clientes. El medio a transmitir puede estar ya preparado desde el comienzo del proceso en un fichero comprimido (Novoa, 2007).

1.2.2 Protocolos empleados para emitir streaming

Un protocolo no es más que un conjunto de normas o estándares que especifican el método para enviar y recibir datos entre varios ordenadores, controlando o permitiendo la conexión, comunicación y transferencia de datos entre estos.

- **Protocolo RTP**

RTP son las siglas de Real-time Transport Protocol. Es un protocolo de nivel de aplicación utilizado para la transmisión de información en tiempo real, especialmente creado para el manejo de datos streaming (Novoa, 2007). Consta de campos de datos extra, que hacen posible manejar información streaming. Cuenta con un control que hace que el servidor haga fluir los datos de video a una velocidad correcta para la proyección en tiempo real. Proporciona un mecanismo llamado Timestamp[5] que el receptor utiliza para reconstruir el tiempo original de los paquetes que le son enviados. Algunos formatos de video se dividen en distintos paquetes RTP, por tanto, todos ellos pueden tener el mismo Timestamp, por eso se necesita de la ayuda de los números de secuencia para ordenar los paquetes.

- **Protocolo RTSP**

El protocolo RTSP (Real Time Streaming Protocol) establece y controla uno o varios flujos sincronizados de datos, ya sean de audio o de video. Este protocolo de flujo de datos en tiempo real actúa como un mando a través de la red para servidores multimedia y se sitúa en el nivel de aplicación dentro del modelo OSI. Al ser un protocolo no orientado a conexión, en el transcurso de una sesión RTSP un cliente puede abrir y cerrar varias conexiones a nivel de transporte, sin que dicha sesión se vea afectada. RTSP es únicamente un protocolo de control, por lo que el envío de datos de streaming se deberá realizar mediante algún otro protocolo destinado a tal efecto (Novoa, 2007).

- **Protocolo SDP**

[5] Es una secuencia de caracteres, que denotan la hora y fecha (o alguna de ellas) en la cual ocurrió determinado evento.

El Session Description Protocol (SDP), es un protocolo para describir los parámetros de inicialización de los flujos multimedia. Es un protocolo textual, y se puede usar en conjunción con RTSP o SIP para informar al cliente de las características del flujo de streaming que se va a enviar, tanto en lo referente al contenido como a la manera en que se va a transmitir. Un SDP consiste en una cadena de texto donde se informa de los contenidos audiovisuales disponibles, en qué formato se encuentran y otros parámetros como por ejemplo la duración (Novoa, 2007).

- **Protocolo TCP**

TCP es un protocolo de transporte orientado a conexión entre estaciones de trabajo que establecen comunicación/conexión. Al establecerse la comunicación entre las dos estaciones de trabajo, se asegura que el flujo de datos entre ellas sea fiable, asegurándose de que los datos llegan correctamente del emisor al destinatario, en el orden estipulado y completo (Novoa, 2007).

- **Protocolo UDP**

El protocolo UDP (User Datagram Protocol) es un protocolo simple que provee las funciones básicas de la capa de transporte. Tiene una sobrecarga mucho menor que el protocolo TCP, ya que no está orientado a la conexión y no proporciona mecanismos sofisticados de retransmisión y flujo de control (Roldán, 2010). Dicho protocolo proporciona una comunicación muy sencilla entre las aplicaciones de dos ordenadores. Este protocolo de transporte suele ser usado en aplicaciones de streaming (video o audio), ya que en estas es más importante la recepción rápida de los datos que la verificación de los mismos (Novoa, 2007).

- **Protocolo SIP**

Dentro del campo de los protocolos para el control de streaming, uno de los que más se puede comparar con RTSP es SIP. El objetivo del mismo es la inicialización, modificación y finalización de sesiones interactivas de usuario donde intervienen elementos multimedia como el video, mensajería instantánea, juegos online y realidad virtual. SIP es un protocolo de control cuya funcionalidad es similar a RTSP. Este protocolo funciona en colaboración con otros muchos protocolos, pero solamente interviene en la parte de negociación al establecer, modificar y finalizar la sesión de comunicación (Novoa, 2007).

- **Protocolo HTTP**

HTTP (Hypertext Transport Protocol) es el protocolo usado en cada transacción de la web. Es el sistema mediante el cual se envían las peticiones de acceso a una página, la respuesta con el contenido y también información en ambos sentidos. Específicamente en la transmisión de video, este protocolo es usado para que un servidor web pueda repartir los datos a un medio visualizador, permitiéndole al cliente la descarga automática (Santos, 2007). Aunque muchos de los servidores streaming existentes utilizan HTTP, este protocolo no es el ideal para hacer streaming, ya que en caso de una pérdida de cierto paquete, este es retransmitido y deja de continuar la transmisión a la velocidad más rápida posible.

- **Protocolo RTMP**

Real Time Messaging Protocol (RTMP) proporciona un servicio de mensajes a través de un flujo de transporte fiable, destinado a transportar corrientes paralelas de video, audio y mensajes de datos, con información de temporización asociada. Stream Chunk RTMP[6] fue diseñado para trabajar con RTMP, ya que puede manejar cualquier protocolo que envía un flujo de mensajes. Cada mensaje contiene indicación de la hora y la identificación del tipo de carga útil. Stream Chunk RTMP y RTMP juntos son adecuados para una amplia variedad de aplicaciones de audio y video (MThornburgh, 2012).

1.2.3 Aplicaciones del streaming

Las aplicaciones del streaming se pueden encontrar en gran medida en Internet, siendo utilizadas para difundir noticias, arte, música, sin contar la utilidad que ha traído a grandes negocios que gestionan archivos multimedia. Este tipo de aplicaciones solamente no se encuentran en Internet, sino que se han expandido a las redes celulares, siempre y cuando estos cuenten con wireless. La mayoría de los servicios que ofrece el streaming dependen en gran medida de su tipo. En el caso del streaming en vivo, las transmisiones de contenidos audiovisuales son posibles debido a la capacidad del mismo de transmitir audio y video a través de la red en el instante que es emitido o reproducido. Por otra parte los clientes a través de la web pueden tener acceso a archivos de noticias, difusión de entrevistas, conferencias y presentaciones corporativas utilizando el streaming bajo demanda (Bernal, 2010).

1.2.4 Servidores streaming existentes

[6] Está diseñado para trabajar con el protocolo RTMP, pero es capaz de manejar cualquier protocolo que envía una corriente de mensajes, además de brindar servicios para empaquetar y multiplexar.

A continuación se hace un estudio preliminar sobre los servidores streaming más utilizados, que dará la posibilidad de conocer si algunos de ellos puede ser utilizado como propuesta de la solución a desarrollar.

- **Darwing Streaming Server**

Darwin Streaming Server (DSS) es la versión de código abierto de la tecnología de Apple QuickTime Streaming Server, proporcionando un alto nivel de personalización y ejecución en una variedad de plataformas tales como Linux, Solaris y Windows. Al ser un servidor de código abierto resulta altamente configurable y adaptable a las necesidades de los usuarios (Torres, 2009). Este servidor permite transmitir videos a diferentes tipos de clientes, mediante el uso de protocolos tales como: RTP y RTSP. Darwin Streaming Server permite hacer streaming de video y clips de audio en formatos MP3 y MPEG.

- **Quick Time Streaming Server**

Quick Time Streaming Server es el servidor de video bajo demanda comercial y propietario de Apple. Este servidor se distribuye conjuntamente con las versiones del servidor Mac OS X. Es un servidor multimedia basado en estándares altamente compatibles, además de contar con la capacidad de permitir añadir servicios multimedia a la web u ofrecer estos contenidos mediante cualquier otro mecanismo. Quick Time es uno de los más versátiles y rentables servidores para crear y reproducir streaming a través de Internet. Se encuentra escrito en función de diferentes formatos, tales como: H264, AAC, MP3, MPEG-4, 3GPP. Dicho servidor utiliza estándares compatibles con los reproductores multimedia garantizando una gran confiabilidad y escalabilidad.

- **Icecast**

Icecast es un servidor de streaming que actualmente soporta entre otros los formatos Ogg, Vorbis y MP3. Se puede utilizar para crear una estación de radio por Internet. Es versátil, ya que es posible agregar nuevos formatos de una forma relativamente fácil y soporta estándares abiertos para la comunicación y la interacción. En la actualidad este servidor es compatible con plataformas como Unix, Linux, Solaris, OpenBSD, FreeBSD (Childers, 2010). Este servidor es un proyecto de código libre mantenido por Xiph.org[7]. Es capaz de distribuir contenido tanto de audio como de video, aunque al principio fue

[7] Es una organización que desarrolla formatos de archivos multimedia abiertos y software libre. Está centrada principalmente en la familia de formatos Ogg.

exclusivamente de audio (Abadía, 2011). Este servidor es capaz de manejar los protocolos HTTP y SHOUTcast[8] (Torres, 2009).

- **Red5**

Red5 es un servidor Open Source para entregar contenido streaming en Flash, para ello utiliza el protocolo RTMP, con el cual se puede transmitir contenido en tiempo real. Este servidor tiene todas las cualidades del Flash Media Server de Adobe. Utiliza la sintaxis de Action Script Comunication[9] con la cual se pueden desarrollar aplicaciones de comunicación en tiempo real (Perugachi, 2011).

El servidor Red5 posee un conjunto de características propias, las cuales se detallan brevemente a continuación:

- Chats multiusuario en tiempo real.
- Transmisión de señal de televisión en tiempo real.
- Facilidad para implementar pantallas compartidas en la cual múltiples usuarios pueden interactuar en una misma pantalla.
- Análisis y reportes de datos en tiempo real.

- **Flash Media Server**

Flash Media Server (FMS) es un servidor propietario para información y contenido interactivo de Adobe System. Las aplicaciones Flash se conectan a FMS usando el protocolo RTMP. Este servidor puede enviar y recibir datos de los usuarios conectados. Los clientes conectados pueden realizar llamadas a procedimientos remotos (RPC) a la aplicación servidor y esta a su vez puede llamar métodos de los clientes. También puede usarse un SharedObject (objetos compartidos) para sincronizar estructuras de datos complejas y llamar métodos remotos en múltiples clientes, al tener a ellos suscriptos a un objeto compartido determinado. Este servidor permite a los usuarios realizar streaming de audio y video.

[8] Es un protocolo multiplataforma para streaming de audio.

[9] Es el lenguaje de programación de la Plataforma Adobe Flash. La programación con Action Script permite mucha más eficiencia en las aplicaciones de la plataforma Flash para construir animaciones de todo tipo, desde simples a complejas, ricas en datos e interfaces interactivas.

- **Windows Media Services**

Es una plataforma de altas prestaciones para la transmisión de contenido de audio y video en directo o a petición a través de Internet o una Intranet. Este servidor de contenidos multimedia en streaming permite al administrador generar contenidos en directo (audio/video) (Ribas-Corbera, 2003). Posee una licencia privativa. Este servidor es actualmente utilizado en la UCI por parte del equipo de trabajo encargado de la administración del sitio Inter-nos y utiliza el protocolo MMS para las transmisiones.

- **VLC**

VideoLAN es una solución completa de software multiplataforma para video streaming, la cual se encuentra bajo licencia GPL. Esta solución está diseñada para servir videos en redes que soporten grandes anchos de banda en diferentes protocolos tales como RTSP y HTTP. Incluye para su funcionamiento VLS (VideoLAN Server) y VLC (VideoLAN Client), el primero con el objetivo de servir archivos de audio y video, canales satelitales, televisión digital y videos en vivo sobre una red ya sea Unicast o Multicast, mientras que el segundo es utilizado como cliente para recibir, decodificar y mostrar streaming de diversos formatos.

- **Flumotion**

Flumotion es un software de streaming basado en el marco de GStreamer, que utiliza su capacidad de codificación de streaming y añade funciones más potentes, como el streaming a través del protocolo HTTP. Flumotion tiene como objetivo construir una solución de software de extremo a extremo que integra adquisición, codificación, multiformato de transcodificación y streaming de contenidos. También permite que el procesamiento sea repartido en varios equipos, por lo que la plataforma puede escalar para manejar más espectadores en formatos más corrientes. Todo esto se hace utilizando un enfoque modular (Barreiros, 2012). Este servidor posee una licencia comercial y otra licencia GPL versión 2.2, siendo esta la que se utiliza en los proyectos del Departamento de Señales Digitales.

1.3 Sistemas Distribuidos

Como se había referido en la introducción, los sistemas distribuidos pueden ser eficientes en aplicaciones streaming, debido a su capacidad de reponerse a cualquier fallo asegurando la buena calidad de los servicios. Estos sistemas no son más que un conjunto de computadoras interconectadas que comparten un estado, ofreciendo una visión de sistema único (Hall, 1995). En dichos sistemas, los recursos de

diferentes máquinas en red se integran de forma que desaparece la dualidad local/remoto (Lafuente, 2011).

1.3.1 Propiedades de los sistemas distribuidos

Un objetivo importante de un sistema distribuido es la **transparencia,** la cual se basa en ocultar el hecho de que sus procesos y los recursos están físicamente distribuidos en varios equipos. Estos hechos y procesos pueden estar involucrados con la representación de los datos, la forma de acceder a un recurso, su movimiento y ubicación, los recursos replicados y los recursos compartidos. Otra de las características de los sistemas distribuidos es su **escalabilidad**, definida como la capacidad del sistema para crecer, con el aumento de usuarios y los recursos compartidos de alguna manera (Neuman, 1994).

Por otra parte existen propiedades vinculadas estrechamente a las necesidades básicas de los clientes, una de ellas es la **fiabilidad** que puede definirse como la capacidad del sistema para realizar correctamente y en todo momento las funciones para las que se ha diseñado, la misma se divide en dos aspectos fundamentales: **disponibilidad**[10] y **tolerancia a fallos**[11]. Aunque un sistema distribuido se puede considerar como múltiples computadoras interconectadas entre sí, resulta necesario que el mismo se comporte según lo desean los usuarios. Esto requiere de ciertas características como la **consistencia**, ya que sin ella resulta imposible el funcionamiento del mismo. Esta propiedad de los sistemas distribuidos depende del estado global del sistema en sí. Este estado estriba de los datos que son compartidos entre los ordenadores involucrados. Si alguno de estos datos tendiera a cambiar solamente para algunas máquinas, por problemas de actualización, réplica, copias anticuadas en la caché o problemas de sincronía en el tiempo de los equipos, la consistencia se vería afectada notablemente, dada la diferencia de información contenida en los recursos compartidos para todos los ordenadores.

1.3.2 Ventajas de los sistemas distribuidos

- **Ventajas de los sistemas distribuidos con respecto a los centralizados:**

[10] Fracción de tiempo que el sistema está operando.

[11] Capacidad del sistema para seguir operando correctamente ante el fallo de alguno de sus componentes, enmascarando el fallo al usuario o a la aplicación.

Un sistema distribuido no es más que una colección de computadoras separadas físicamente y conectadas entre sí por una red de comunicaciones distribuida. Cada máquina posee sus componentes de hardware y software que el usuario percibe como un solo sistema. Mientras que los sistemas centralizados son aquellos que se ejecutan en un único sistema informático sin interactuar con ninguna otra computadora. La diferencia más importante entre un sistema distribuido y uno centralizado es precisamente la forma de realizar la comunicación entre procesos. En un sistema centralizado, las interacciones entre procesos se hacen sobre memoria compartida, mientras que en un sistema distribuido se necesita intercambio de mensajes. Los sistemas distribuidos poseen un conjunto de ventajas con respecto a los sistemas centralizados, entre las que se pueden nombrar (Hurtado, 2008):

- **Economía:**

Estos sistemas tienen en potencia una proporción precio/desempeño mucho mejor que la de un sistema centralizado, ya que el costo de servidores se retribuye en la maximización de tiempos de trabajo y evita los tiempos muertos. Es decir que se hace más en menos tiempo y resulta mucho más barato añadir servidores y clientes cuando se requiere aumentar la potencia de procesamiento.

- **Velocidad:**

Un sistema distribuido puede tener mayor poder de cómputo que una mainframe[12].

- **Confiabilidad:**

Un sistema distribuido ofrece mayor confiabilidad: al distribuir la carga de trabajo en muchas máquinas. La falla de un circuito descompondrá a lo más una máquina y el resto seguirá intacto.

- **Crecimiento por incrementos:**

Si se necesita añadir poder de cómputo, con un sistema distribuido, podrían añadirse más procesadores al sistema, lo que permite un desarrollo gradual conforme surjan las necesidades.

1.3.3 Protocolos usados en los sistemas distribuidos

[12] Computadora grande, potente y costosa usada principalmente para el procesamiento de una gran cantidad de datos.

Un protocolo en sistemas distribuidos permite que componentes heterogéneos puedan desarrollarse independientemente y por medio de módulos de software que componen el protocolo. Son capaces de hacer que la comunicación sea transparente entre ambos componentes, siempre y cuando estos se encuentren tanto en el receptor como en el emisor (Hurtado, 2008).

Existen protocolos usados tanto por los sistemas distribuidos como por la tecnología streaming, como suelen ser: TCP (Protocolo de Control de Transmisión) y HTTP (Protocolo de Transferencia de Hipertexto), los cuales fueron explicados anteriormente. Los sistemas distribuidos trabajan además con los protocolos:

- **IP (Protocolo de Internet)**

Protocolo de la capa de red, que permite definir la unidad básica de transferencia de datos y se encarga del direccionamiento de la información, para que llegue a su destino en la red.

- **SMTP (Protocolo de Transferencia de Correo Simple)**

Protocolo de la capa de aplicación, que permite el envío de correo electrónico por la red.

- **XML-RPC**

Es un protocolo muy simple ya que solamente define algunos tipos de datos y comandos útiles, además de una descripción completa de corta extensión. La simplicidad de XML-RPC contrasta con la mayoría de protocolos RPC que tiene una documentación extensa y requiere considerable soporte de software para su uso (Laurent, 2001).

1.3.4 Aplicaciones de los sistemas distribuidos

Las aplicaciones de los sistemas distribuidos abarcan prácticamente todas las aplicaciones comerciales y técnicas de los ordenadores. Debido a la facilidad, fiabilidad y comodidad que proporcionan los sistemas distribuidos, pueden ser utilizados en **sistemas comerciales**, por lo que resulta necesario que dichas aplicaciones posean un grado alto de fiabilidad. Otro ámbito donde se destaca el uso de este tipo de sistemas son las **redes WAN,** ejemplo de ello lo constituyen los servicios comunes que brinda Internet: correo electrónico, servicio de noticias, transferencia de archivos, etc. Las **aplicaciones multimedia** a pesar de ser las últimas incorporaciones a los sistemas distribuidos, ofrecen un gran panorama y un enorme potencial para nuevas aplicaciones basadas en los sistemas multimedia distribuidos. Estos

incluyen los sistemas de colaboración y conferencia, televisión de alta resolución y la enseñanza a distancia (Hurtado, 2008).

1.4 Soluciones existentes

Luego de analizar los servidores streaming existentes, con el objetivo de conocer si algunos de estos puede ser instalado de manera distribuida se concluyó, que solamente Flumotion puede ser implementado de esta forma.

1.4.1 Flumotion

El servidor Flumotion posee una licencia comercial y otra licencia GPL versión 2.2. Este servidor en su versión libre soporta estándares como Ogg, Vorbis y Theora. Dicho servidor cuenta con herramientas de administración gráficas, con objetivos de crear y manipular flujos de audio y video sencillo. Flumotion en su arquitectura incluye entre sus componentes un Manager encargado de proporcionar el control de la conexión y de los servicios streaming y un conjunto de Worker, dedicados a controlar la codificación y envío de la información de audio y video. Cada uno de estos worker puede atender un número de peticiones predeterminada, por lo que puede considerarse la implementación de este servidor de forma distribuida. Dicho servidor presenta dificultad en cuanto al balanceo de carga, lo que trae como consecuencia la incapacidad de servir de forma correcta todas las peticiones que se le realizan. Para ello en la UCI se realizó un trabajo de diploma encargado de presentar un clúster de balanceo de carga que permite que un conjunto de servidores compartan la carga de trabajo y de tráfico de sus clientes, dividiendo equitativamente las peticiones de servicios que reciba el clúster entre los nodos que lo componen.

Flumotion brinda algunas ventajas que no poseen otros servidores de streaming, dada su condición de permitir instalarse en varias máquinas y funcionar como un único componente. La licencia libre del mismo solamente permite hacer streaming utilizando el protocolo HTTP. Aunque este posibilita insertar texto sobre la imagen en tiempo real, no permite modificarlo ni moverlo una vez iniciada la transmisión, por lo que no se pueden agregar subtítulos, ni infocintas. Por los elementos antes mencionados y la situación que presenta este servidor con respecto el balanceo de carga, no resulta óptimo para el uso en la solución del producto.

1.5 Conclusiones parciales

- El estudio de los protocolos que pueden ser utilizados para emitir streaming permitió determinar la utilización del protocolo UDP, ya que este tiene en cuenta el envío continuo de información, permitiendo así una transmisión de forma rápida.
- El análisis de los servidores streaming existentes arrojó como resultado que solamente Flumotion puede implementarse de forma distribuida, sin embargo, este servidor no resulta factible para el desarrollo de la solución a implementar ya que utiliza el protocolo HTTP para realizar streaming, el cual no es el protocolo ideal para el objetivo de la presente investigación.

Capítulo 2: Tecnologías y herramientas utilizadas en el SDTCA

Introducción

En el presente capítulo se realiza un análisis de las tecnologías y herramientas que serán empleadas para dar solución al problema de investigación planteado, con el objetivo de modelar, implementar y garantizar el correcto funcionamiento de la solución propuesta.

2.1 Metodología de desarrollo de software

Para llevar a cabo la realización de un software, resulta imprescindible contar con una guía que indique las actividades que deben realizarse, con el propósito de lograr un producto con una elevada calidad. Es por ello la importancia de la utilización de las metodologías de desarrollo de software, siendo estas un conjunto de procedimientos y técnicas que establecen una forma eficiente de desarrollar y documentar un producto de software.

2.1.1 Proceso Unificado de Desarrollo de Software (RUP)

El Proceso Unificado de Desarrollo de Software (RUP) es una de las metodologías de desarrollo de software robustas más utilizadas, definiendo quién, cómo, cuándo y qué debe realizarse.

A continuación se muestran las principales características de esta metodología (Ivar Jacobson, 2000):

- **Dirigido por casos de uso**: En RUP los casos de uso no son sólo una herramienta para especificar los requisitos del sistema, estos también guían su diseño, implementación y prueba. Los casos de uso inician el proceso de desarrollo y posibilitan establecer trazabilidad entre los artefactos que son generados en las diferentes actividades del proceso de desarrollo.

- **Iterativo e Incremental:** Plantea la implementación del producto a realizar en iteraciones, por lo que se pueden definir diferentes objetivos en cada iteración y así completar el producto iteración por iteración.

- **Centrado en la arquitectura**: RUP presta especial atención al establecimiento temprano de una buena arquitectura, con el objetivo de evitar un fuerte impacto ante cambios posteriores durante la construcción y mantenimiento. La arquitectura muestra la visión común del sistema completo en la que tanto el equipo de proyecto como los usuarios deben estar de acuerdo (Jacobson, 2000). La

arquitectura se desarrolla mediante iteraciones, fundamentalmente en la fase de elaboracion, centrándose en casos de uso relevantes desde el punto de vista arquitectónico. Es por esta razón que la arquitectura y los casos de uso mantienen una estrecha relación, ya que estos últimos conducen al desarrollo de dicha arquitectura, mientras que esta debe ser capaz de guiar estos casos de uso durante el proceso de desarrollo de softwrare.

Dicha metodología divide el proceso en 4 fases, dentro de las cuales se realizan varias iteraciones, como se muestra en la siguiente figura:

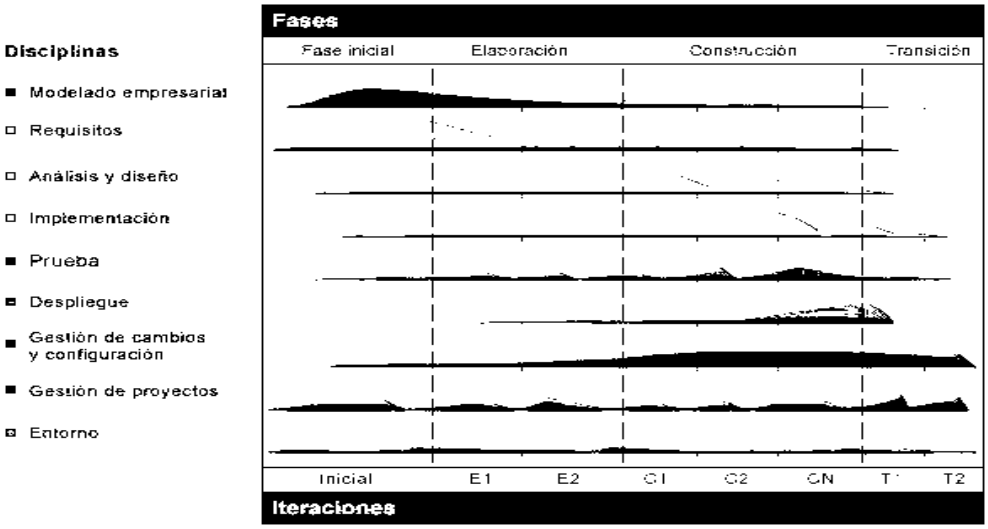


Figura 1. Fases y disciplinas de RUP. Tomado de (Jacobson, 2000).

Se decidió utilizar el Proceso Unificado de Desarrollo de Software (RUP) como metodología de desarrollo, debido a su capacidad de adaptabilidad a cualquier entorno, posibilitando obtener una documentación detallada de todo el proceso y definiendo claramente los objetivos pertenecientes a cada fase. Esta metodología propone el proceso de desarrollo en iteraciones, posibilitando a medida que avanza el desarrollo del software la corrección de errores y mejoras del mismo. El subsistema a implementar tiene bien definidos los requisitos funcionales ya que la mayoría se obtuvieron del subsistema de transmisión anterior y el uso de esta metodología es recomendable para productos donde los requisitos se encuentran bien definidos. Otro aspecto que influyó en la decisión del uso de dicha metodología fue el objetivo de mantener la línea base de la arquitectura y lograr un mayor acoplamiento entre la solución propuesta y los restantes módulos que forman parte del producto STCV.

2.2 Lenguaje de modelado

El creciente desarrollo de los productos informáticos, la complejidad y elaboración de los mismos, ha traído consigo la necesidad de crear lenguajes de modelado que sirvan como ejemplos a seguir por el equipo de desarrollo. Estos lenguajes tienen como objetivo lograr un modelo común, comprensible tanto para probadores, desarrolladores y analistas.

2.2.1 Lenguaje Unificado de Modelado (UML 2.0)

El Lenguaje Unificado de Modelado (UML) se utiliza para especificar, visualizar, construir y documentar artefactos de un software, capturando decisiones y conocimientos sobre lo que se deben desarrollar. Este lenguaje puede ser utilizado en conjunto con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios. UML permite su utilización en herramientas interactivas de modelado visual que tengan generadores de informes así como generadores de código para una gran variedad de lenguajes de programación. La especificación de este lenguaje de modelado no define un proceso estándar, pero puede ser útil en un proceso de desarrollo iterativo (Jacobson, 2005).

Dicho lenguaje de modelado define los tipos de objetos importantes para un sistema y para su implementación, así como las relaciones entre los mismos, especificando la historia de los objetos en el tiempo a través del comportamiento dinámico. UML 2.0 permite agrupar los modelos en paquetes, posibilitándole al equipo de desarrollo dividir grandes sistemas en piezas de trabajo (Jacobson, 2005). Esta versión de UML posee cierta compatibilidad con versiones anteriores, además de estar significativamente enriquecida en relación con las actividades de dichas versiones. Incluye una semántica compartida entre las acciones y actividades más flexibles, con el objetivo de realizar un mayor número de modelos.

Se decidió la utilización de este lenguaje de modelado para la realización de la solución a implementar, debido a la capacidad que posee de permitir modelar el subsistema a desarrollar mediante la representación visual y la combinación de diversos elementos gráficos para crear diagramas. Además es el lenguaje utilizado por RUP para llevar la documentación del sistema, así como facilitar la etapa de diseño y desarrollo del mismo.

2.3 Herramienta CASE

Las herramientas CASE son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software, reduciendo el costo de las mismas en términos de tiempo y dinero.

2.3.1 Visual Paradigm 8.0

Visual Paradigm es una herramienta CASE que propicia un conjunto de ayudas para el desarrollo de programas informáticos, permitiendo modelar todos los artefactos que se obtienen a partir del análisis del negocio y el sistema. Es una herramienta multiplataforma, capaz de soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas. Existe una alternativa gratuita de este software. Fue diseñado para una amplia gama de usuarios interesados en la construcción de sistemas de software de forma fiable a través de la utilización de un enfoque orientado a objetos. Esta herramienta tiene la capacidad de generar código y realizar ingeniería inversa con diferentes lenguajes de programación. Facilita la integración con disímiles entornos de desarrollo integrados y otras herramientas CASE, brindando un conjunto de opciones que permiten la exportación e importación de componentes (Sierra, 2007).

Se decidió utilizar Visual Paradigm como herramienta CASE, ya que posee una licencia gratuita, utiliza como lenguaje de modelado UML, además de ser multiplataforma. Mientras que otras herramientas CASE como Rational Rose Enterprise Edition no presentan estas caracteristicas, puesto que esta solamente permite ser ejecutada en Windows y posee una licencia propietaria.

2.4 Lenguaje de programación

Como resultado de la dificultad de comunicación insalvable entre la computadora y el programador, surgen los lenguajes de programación, los cuales hacen posible la comunicación con el microprocesador, utilizando términos y símbolos relacionados con el tipo de problema a resolver, mediante el empleo de herramientas que brinda la informática.

2.4.1 C++

C++ es un lenguaje de programación basado en C, al cual se le añadieron funcionalidades de las que este último carecía, como la programación orientada a objetos. Es un lenguaje multipropósito que se adapta a múltiples situaciones y puede ser usado tanto para la programación a bajo como a alto nivel (Mora, 2009). Actualmente es un lenguaje potente que mantiene las ventajas de C en cuanto a riqueza de operadores,

expresiones, flexibilidad y eficiencia. El código es portable y puede ejecutarse en cualquier máquina y bajo cualquier sistema operativo (González, 2008).

Las principales ventajas que presenta este lenguaje de programación son (Mora, 2009):

- **Difusión:** Posee un gran número de usuarios y existe una gran cantidad de recursos dedicados a su estudio.
- **Versatilidad:** C++ puede emplearse para resolver cualquier tipo de problema.
- **Portabilidad:** Está estandarizado y un mismo código fuente se puede compilar en diversas plataformas.
- **Eficiencia:** Es uno de los lenguajes más rápidos en cuanto a tiempo de ejecución, ya que permite la separación de un programa en módulos que admiten compilación independiente. Además de incluir un uso extensivo de apuntadores para la memoria, arreglos, estructuras y funciones, al mismo tiempo que es capaz de manejar actividades de bajo nivel y generar programas eficientes.
- **Herramientas**: Existe una gran cantidad de compiladores, depuradores, librerías.

Luego del análisis previamente realizado en correspondencia al lenguaje de programación C++, se decidió utilizar dicho lenguaje en la implementación de la aplicación a realizar, debido a que el mismo es compatible en varias plataformas incluyendo Linux. Además de incorporar una mayor velocidad de ejecución en comparación con otros lenguajes como Java, lo cual resulta de suma importancia para una transmisión sin retardos.

2.5 Marco de trabajo

Los marcos de trabajo definen en términos generales, un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular, que sirve como referencia para enfrentar y resolver nuevos problemas de índole similar. Típicamente puede incluir soporte de programas, bibliotecas y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un producto.

2.5.1 Qt (versión 4.7.3)

Qt es un marco de trabajo multiplataforma para el desarrollo de aplicaciones e interfaces de usuario. Utiliza el lenguaje de programación C++ de forma nativa. Cuenta con un gran número de bibliotecas

que pueden ser empleadas para el desarrollo de software. Es producido por la división de software Qt de Nokia, desarrollado bajo licencia LGPL y liberado como software libre y de código abierto.

Presenta un gran número de características entre las que se destacan las siguientes:

- Garantiza la portabilidad entre sistemas operativos.
- Posibilita el trabajo con hilos independientemente del sistema operativo.
- Contiene un módulo para el trabajo con protocolos de red, soporte para aplicaciones orientadas a componentes y trabajo con los gestores de bases de datos más conocidos (Arévalo, 2011).

Se decide hacer uso del marco de trabajo Qt, ya que utiliza como lenguaje de programación C++ de forma nativa y cuenta con una serie de herramientas y clases que permiten el fácil acceso a bibliotecas para el procesamiento de audio y video. Además es un marco de trabajo multiplataforma, por lo que puede ser usado tanto en Windows como en Linux. Permite desarrollar interfaces gráficas de usuario, integrándolo con el IDE QtCreator (Maldonado, 2011).

2.6 Entorno de Desarrollo Integrado (IDE)

Con el objetivo de contar con un entorno de programación empaquetado como un programa de aplicación; es decir, un editor de código, un compilador, un depurador y un constructor de interfaz gráfica surgen los entornos de desarrollo integrado, llamado también IDE. Estos no son más que un programa informático compuesto por un conjunto de herramientas de programación, que puede dedicarse en exclusiva a un solo lenguaje de programación o bien puede utilizarse para varios. Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes.

2.6.1 QtCreator (versión 2.2.1)

QtCreator es un IDE creado por Trolltech para el desarrollo de aplicaciones con las bibliotecas Qt. Está disponible para los sistemas operativos Linux, Mac y Windows. Diseñado para hacer que el desarrollo en C++ de la aplicación Qt sea más rápido y fácil. Permite a los desarrolladores crear aplicaciones de escritorio y plataformas de dispositivos móviles (Albán, 2011). Este IDE presenta un gran número de características entre las que se destacan las siguientes (Nokia, 2011):

- **Editor de código sofisticado:** Avanzado editor de código de QtCreator, provee soporte para edición de C++, sensible al contexto de ayuda y completamiento de código.

- **El control de versiones:** QtCreator se integra con la mayoría de los sistemas de control de versiones populares, incluyendo Git, Subversion, Perforce, CVS y Mercurial.
- **Los diseñadores de interfaz de usuario:** Este IDE ofrece el editor visual integrado Qt Diseñador, muy intuitivo para el desarrollo de interfaces.
- **Proyecto, Construcción y Gestión:** Si se importa un proyecto existente o se crea uno desde cero, QtCreator genera todos los archivos necesarios.
- **Escritorio y objetivos móviles**: QtCreator ofrece soporte para crear y ejecutar aplicaciones para equipos de escritorio y dispositivos móviles.

Para la implementación de la solución a realizar se decidió la utilización de este IDE, debido a la biblioteca que el mismo permite incluir, denominada Phonon, la cual posibilita reproducir videos de una fuente determinada, conocer algunas características de archivos locales tales como la duración, además de poseer una gran capacidad de captar streaming de video utilizando protocolos como HTTP y UDP. Por otra parte este IDE permite la inclusión del sistema de señal / ranura, posibilitando así la notificación de eventos, asociación de tiempo de espera de los mismos a instancias de objetos, métodos o funciones. QtCreator, además, posee herramientas de navegación rápidas, así como resaltado de sintaxis y completamiento de código, lo cual resulta muy útil para el producto a desarrollar.

2.7 Biblioteca

Las bibliotecas o library (en inglés) contienen datos y código que proporcionan servicios a programas independientes pasando a formar parte de estos. Son consideradas como conjunto de subprogramas utilizados para desarrollar software.

2.7.1 GStreamer

GStreamer es una biblioteca para la creación de aplicaciones de streaming media. La misma permite realizar cualquier tipo de streaming de aplicaciones multimedia. Está diseñada para hacer más fácil la realización de aplicaciones de audio, video o ambos, además de poseer la capacidad de procesar cualquier tipo de flujo de datos (Taymans, 2012). Esta biblioteca posee una licencia GPL, además de soportar múltiples lenguajes de programación, tales como C, Python, Java, Ruby, Perl, entre otros (Smith, 2008).

- **Características de GStreamer** (Wim Taymans, 2012)**:**

- Incluye una API para aplicaciones multimedia.
- Permite mezclar diferentes tipos de medios como pueden ser audio, video, imágenes y subtítulos.
- Posibilita aplicar diferentes tipos de filtros, tanto de audio como de video.
- Facilita la transmisión y recepción de datos en tiempo real.
- Proporciona codificación y decodificación en diferentes formatos.
- Posibilita la reproducción de archivos multimedia.

Se decidió la utilización de esta biblioteca para la implementación del subsistema a desarrollar debido a las posibilidades que brinda, de ser capaz de procesar cualquier tipo de flujo de datos, además de mezclar audio, video y subtítulos. Siendo estas algunas de las ventajas que lo sitúan en una mejor posición con respecto a otros como LibVLC, el cual por sí solo no es capaz de realizar edición, por lo que debe auxiliarse de la librería OpenCV.

2.8 Sistema Gestor de Bases de Datos (SGBD)

En el mundo actual existe cada vez más, una mayor demanda de datos, por ello las bases de datos se reconocen como una de las principales aplicaciones de la informática. Los SGBD, específicamente, permiten la definición de bases de datos, así como la elección de las estructuras de datos necesarias para el almacenamiento y búsqueda de los mismos.

2.8.1 PostgreSQL (versión 9.1)

El SGBD PostgreSQL se ha ganado la reputación de ser confiable y mantener la integridad de los datos. Posee la capacidad de ejecutarse en la mayoría de los sistemas operativos más utilizados en el mundo incluyendo Windows, Linux y varias versiones de UNIX. Es un sistema de gestión de bases de datos que posee características significativas del motor de datos, entre las que se pueden incluir las subconsultas, los valores por defecto, las restricciones a valores en los campos (constraints) y los disparadores (triggers).

Posee una integridad referencial e interfaces nativas para lenguajes como C++, Perl, Python y Ruby. Funciona en todos los sistemas operativos Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64), y Windows. Debido a la liberación de la licencia, PostgreSQL se puede modificar y distribuir de forma gratuita para cualquier fin.

Tras el análisis realizado referente al gestor de base de datos PostgreSQL, se decidió la utilización del mismo en su versión 9.1, ya que al ser la última versión de este SGBD no incluye los errores existentes en versiones anteriores, además es capaz de crear un clúster de alta disponibilidad. Este gestor posee un carácter orientado a objetos y las prestaciones que brinda, cumplen con las necesidades de la solución a implementar.

2.9 Conclusiones parciales

- Con la utilización de la metodología de desarrollo de software RUP y el lenguaje de modelado UML el equipo de trabajo obtendrá un mejor control de los requerimientos y cambios en el proceso de desarrollo.
- El uso del IDE QtCreator posibilita incluir en la solución a desarrollar componentes capaces de captar streaming de video y realizar procesos automáticos mediante señales y slots.
- Con la utilización de la biblioteca GStreamer se logrará que el subsistema a implementar pueda multiplexar audio, video y subtítulos.
- La utilización del SGBD PostgreSQL le proporcionará al subsistema la posibilidad de adquirir las carteleras a transmitir y gestionar datos persistentes comunes entre los restantes subsistemas del producto STCV.

Capítulo 3: Presentación y construcción del SDTCA

Introducción

En el presente capítulo con el objetivo de lograr una mejor visión del subsistema a desarrollar, se elabora el modelo de dominio, se definen los requerimientos funcionales y no funcionales de la solución propuesta, se especifican los actores y casos de uso más significativos. Además se lleva a cabo el diseño y construcción del subsistema.

3.1 Modelo de Dominio

El modelo de dominio es una representación visual de las clases conceptuales más significativas. Utilizando la notación UML, un modelo de dominio se representa a través de un conjunto de diagramas de clases en los que no se define ninguna operación, mostrándose los objetos de dominio o clases conceptuales con sus determinados atributos y las asociaciones que se establecen entre ellos (Larman, 2003).

3.1.1 Eventos principales del entorno

El subsistema de transmisión del producto STCV es el encargado de realizar el proceso de transmisión de materiales audiovisuales. El mismo es capaz de iniciar la emisión de un canal de forma automática o manual. Una vez puesta en marcha de forma manual una emisión, el operador de transmisión puede realizar operaciones tales como: parar, pausar, atrasar y adelantar las medias pertenecientes al canal que se está emitiendo. En ambas formas de transmisión la programación de los canales debe tenerse en cuenta para la transmisión de las medias. En la interfaz de usuario se visualiza la información referente a los canales con transmisión para ese día, por ejemplo el nombre del canal, la lista de medias correspondiente al mismo, así como los datos de dichas medias.

3.1.2 Diagrama de clases del modelo de dominio

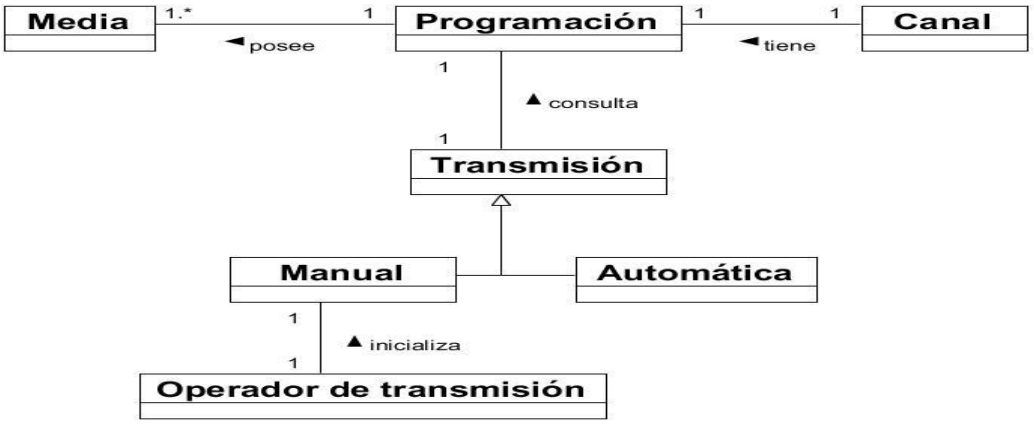


Figura 2. Modelo de dominio.

3.1.3 Descripción de las clases

- **Media:** Son todas las medias planificadas para ese día.
- **Canal:** Representan cada uno de los canales televisivos que se desea transmitir en el día.
- **Programación:** Indica los canales planificados para el día y las medias que se transmitirán para cada uno.
- **Transmisión automática:** El subsistema de acuerdo a la programación de cada canal, realiza la transmisión del mismo.
- **Transmisión manual:** El operador de transmisión realiza acciones sobre las medias.
- **Operador de transmisión:** Es el responsable de parar, pausar, atrasar y adelantar las medias pertenecientes a un canal.

3.2 Requisitos

Los requisitos son la descripción de los servicios proporcionados por el sistema y sus restricciones operativas, reflejando las necesidades de los clientes de dicho sistema, con el objetivo de ayudarlos a resolver una situación específica (Sommerville, 2005).

3.2.1 Requisitos funcionales

Ian Sommerville en su libro Ingeniería de software (página 109, editado el año 2005) señala que los requisitos funcionales: ‘’Son declaraciones de los servicios que debe proporcionar el sistema, de la

manera en que este debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares''.

A continuación se exponen los requisitos funcionales para el SDTCA:

RF 1: Autenticar usuario.

Descripción: El subsistema debe controlar el acceso a las funcionalidades de acuerdo a los permisos que posean los usuarios registrados.

Entradas:

- Usuario (Cadena de caracteres).
- Contraseña (Cadena de caracteres).

Salida:

- Token de seguridad (Cadena de caracteres).

RF 2: Modificar parámetros de configuración para establecer conexión a la base de datos.

Descripción: El subsistema debe permitir al usuario modificar los parámetros establecidos en la configuración para la comunicación con la base de datos.

Entradas:

- Servidor (Cadena de caracteres).
- Puerto (Número entero).
- Base de datos (Cadena de caracteres).
- Usuario (Cadena de caracteres).
- Contraseña (Cadena de caracteres).

Salida:

- Fichero de configuración (xml).

RF 3: Listar programación del canal televisivo.

Descripción: El subsistema debe permitir listar la programación de un canal determinado.

Entrada:

- Id del canal (Número entero).

Salida:

- Lista de programaciones asociadas al canal (Lista).

RF 4: Listar canales con transmisión televisiva.

Descripción: El subsistema debe permitir mostrar la lista de canales que tienen transmisión para ese día

Salida:

- Lista de canales en transmisión (Lista).

RF 5: Mostrar datos del canal televisivo.

Descripción: El subsistema debe permitir mostrar los datos de un canal en trasmisión.

Entrada:

- Id del Canal (Número entero).

Salida:

- Datos del canal (Cadena de caracteres).

RF 6: Mostrar datos de la media en transmisión.

Descripción: El subsistema debe permitir mostrar los datos del archivo audiovisual que se está transmitiendo en ese instante.

Entrada:

- Id del canal en transmisión (Número entero).

Salida:

- Datos de la media (Cadena de caracteres).

RF 7: Visualizar transmisión del canal televisivo.

Descripción: El subsistema debe permitir visualizar la transmisión del canal seleccionado.

Entrada:

- Id del canal (Número entero).

Salida:

- Flujo de streaming (Video).

RF 8: Iniciar transmisión de forma automática.

Descripción: Al iniciar la aplicación el subsistema debe seleccionar para transmitir una media cuya hora planificada sea la más cercana a la hora actual.

Entrada:

- Lista de programación (Lista).

Salida:

- Id de la media (Número entero).

RF 9: Seleccionar automáticamente una media a transmitir.

Descripción: Cuando una media culmine su transmisión, el subsistema debe seleccionar para transmitir la media siguiente.

Entrada:

- Notificación del Gestorestor (Cadena de caracteres).

Salida:

- Id de la media (Número entero).

RF 10: **Transmitir media.**

Descripción: El subsistema debe permitir la comunicación con el Gestor de procesos para transmitir una media.

Entrada:

- Id de la media (Número entero).
- Id del canal (Número entero).

Salida:

- Flujo de streaming (Video).

RF 11: Reproducir media.

Descripción: El subsistema debe permitir al operador de transmisión reproducir una media.

Entrada:

- Id de la media (Número entero).
- Id del canal (Número entero).

Salida:

- Flujo de streaming (Video).

RF 12: Pausar transmisión.

Descripción: El módulo debe permitir al operador de transmisión pausar la transmisión.

Entrada:

- Id del canal en transmisión (Número entero).

- Id del programa en transmisión (Número entero).

Salida:

- Flujo de streaming pausado (Video).

RF 13: Adelantar hacia la próxima media la transmisión televisiva.

Descripción: El subsistema debe permitir al operador de transmisión adelantar hacia la próxima media la transmisión televisiva.

Entrada:

- Id de la media siguiente a la media que se está transmitiendo (Número entero).
- Id del canal en transmisión (Número entero).

Salida:

- Flujo de streaming (Video).

RF 14: Detener transmisión televisiva.

Descripción: El subsistema debe permitir al operador de transmisión detener la transmisión.

Entrada:

- Id del canal en transmisión (Número entero).

Salida:

- Flujo de streaming detenido (Video).

RF 15: Atrasar la transmisión hacia la media anterior.

Descripción: El módulo debe permitir al operador de transmisión ir hacia la media anterior de la transmisión.

Entrada:

- Id de la media anterior a la media que se está transmitiendo (Número entero).
- Id del canal en transmisión (Número entero).

Salida:

- Flujo de streaming anterior (Video).

RF 16: Insertar material externo en transmisiones televisivas.

Descripción: El subsistema debe permitir al operador de transmisión insertar un material externo en la programación, se debe obtener este material desde un dispositivo externo o desde el servidor.

Entrada:

- Dirección de la media externa (Cadena de caracteres).
- Posición de la media en la lista de medias (Número entero).

Salida:

- Lista de programación modificada (Lista).

RF 17: Listar infocinta.

Descripción: El subsistema debe permitir listar las infocintas que han sido creadas.

Salida:

- Lista de infocintas (Lista).

RF 18: Insertar infocinta.

Descripción: El subsistema debe permitir adicionar infocintas.

Entrada:

- Id del canal (Número entero).
- Id del programa (Número entero).
- Posición (Cadena de caracteres).
- Texto (Cadena de caracteres).
- Cantidad de veces a repetir (Número entero).

Salida:

- Emisión y recepción de la Infocinta (Video).

RF 19: Eliminar infocinta.

Descripción: El subsistema debe permitir al operador de transmisión eliminar las infocintas que han sido creadas.

Entrada:

- Texto (Cadena de caracteres).

Salida:

- Confirmación de eliminación (Cadena de caracteres).

RF 20: Insertar logo en transmisión televisiva.

Descripción: El subsistema debe permitir insertar un determinado logo en la transmisión.

Entrada:

- Dirección del logo (Cadena de caracteres).
- Id del canal (Número entero).
- Id del programa (Número entero).
- Posición del logo (Cadena de caracteres).

Salida:

- Flujo de streaming con logo (Video).

RF 21: Mostrar porciento de transmisión.

Descripción: El subsistema debe mostrar el porciento de transmisión de la media que se encuentra reproduciéndose.

Entrada

- Id del programa (Número entero).
- Id del canal (Número entero).

Salida

- Porciento de transmisión (Número entero).

3.2.2 Requisitos no funcionales

Según refleja Ian Sommerville en su libro Ingeniería del software (página 109, editado el año 2005): ‘’Los requisitos no funcionales son restricciones de los servicios o funciones ofrecidos por el sistema. Los requerimientos no funcionales a menudo se aplican al sistema en su totalidad’’.

A continuación se muestran los requisitos no funcionales pertenecientes al SDTCA:

Usabilidad:

- El subsistema debe poseer un ambiente sencillo y de navegación fácil para los usuarios.
- Todas sus funcionalidades deben estar organizadas y ser eficientes.
- Debe tener una interfaz gráfica uniforme, que cumpla con las normas de diseño establecidas para los subsistemas del producto STCV.

Eficiencia:

- El subsistema de transmisión televisiva debe chequear el estado de disponibilidad del servidor de base de datos para con los clientes.
- El tiempo entre una media y otra debe ser inferior a 3 segundos.
- Al reanudarse la energía en caso de alguna falla en las estaciones servidoras, el operador de transmisión realizará de forma manual la transmisión televisiva.

Software:

- Bibliotecas:
  - Para el plugin de transmisión:

gstreamer-tools

gstreamer0.10-doc

gstreamer0.10-FFmpeg

gstreamer0.10-Fluendo-mp3

gstreamer0.10-plugins-base

gstreamer0.10-plugins-ugly

gstreamer0.10-plugins-bad

libqxmlrpc

  - Para la aplicación cliente:

Vlc 1.1.9 o superior
libqt4-sql-psql 4.7.4 o superior
libvlc-dev 1.1.9 o superior
libnotify-bin
Phonon
libqxmlrpc

  - Para el Gestor de procesos:

libqxmlrpc

- Instalar la fuente Segoe UIL.

- Mapear las carpetas donde estarán los videos y los logos de los canales.

Hardware:

Cada nodo debe contar con las siguientes características:

- Dual Core.
- 1 GB RAM.
- Tarjeta de red 100 mbps.

Restricciones de diseño e implementación:

- Se utilizará el lenguaje UML y la herramienta case Visual Paradigm para el modelado del subsistema.
- El lenguaje de programación a utilizar en la implementación del subsistema será C++, con Entorno de Desarrollo Integrado QtCreator y biblioteca GStreamer.

Seguridad:

- El subsistema será utilizado sólo por el personal autorizado.
- Debe exigir a los usuarios autenticarse antes de utilizar sus funcionalidades.
- La contraseña se encuentra codificada.

Interfaz:

- Los colores de las interfaces gráficas serán claros en la mayor parte de la aplicación logrando una vista agradable a los usuarios y resaltando con otras tonalidades los mensajes de interacción de los que dependen las funcionalidades críticas.
- Los calendarios de las programaciones deben poder mostrarse de forma horizontal.
- Los canales deben mostrarse agrupados en un panel inferior de la interfaz principal.
- El subsistema requiere la comunicación a través del protocolo XMLRPC con el Gestor de procesos.
- El subsistema requiere de comunicación a través de la librería QPSQL con la Base de Datos.

3.3 Modelo de casos de uso

Un modelo de casos de uso sirve como acuerdo entre clientes y desarrolladores, además de proporcionar la entrada para el análisis, el diseño y las pruebas. Es un modelo del sistema que contiene actores, casos de uso y sus relaciones (Jacobson, 2000).

3.3.1 Definición de los actores del sistema

Un actor no es parte del sistema en desarrollo, es un agente externo que intercambia con el mismo para lograr un resultado esperado (Jacobson, 2000).

Tabla 1. Definición de los actores del sistema.

Actor	Descripción
Operador de transmisión.	Es el encargado de transmitir la programación.

3.3.2 Diagrama de casos de uso del sistema

Los casos de uso son fragmentos de funcionalidades que el sistema ofrece con el propósito de aportar un resultado para sus actores. Para ello un caso de uso especifica una secuencia de acciones que el sistema debe llevar a cabo interactuando con cada uno de sus actores, incluyendo alternativas dentro de esta secuencia (Jacobson, 2000).

Los 21 requisitos funcionales correspondientes al subsistema propuesto, se agrupan en 8 casos de uso, los cuales se muestran a continuación:

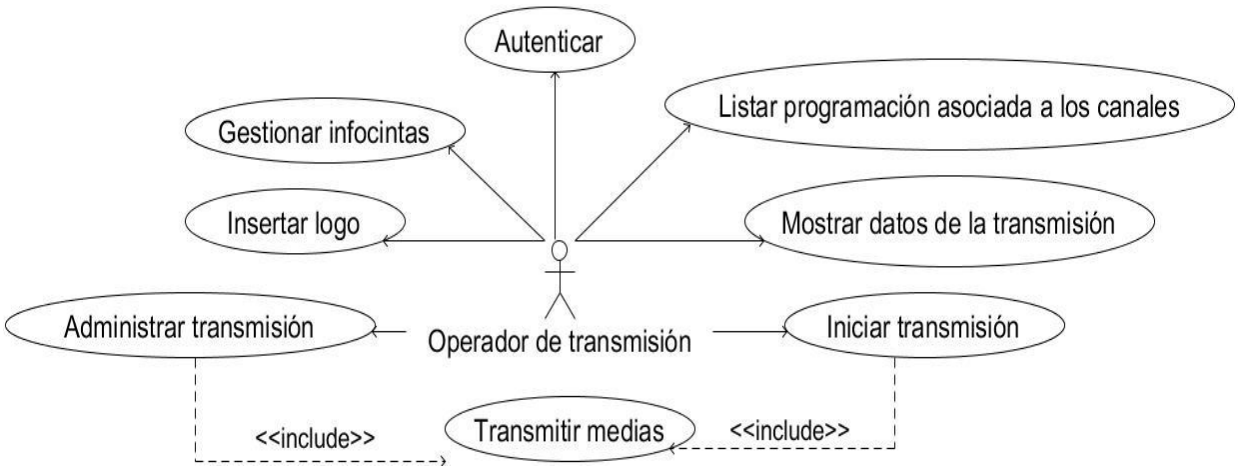


Figura 3. Diagrama de casos de uso del sistema.

3.3.3 Descripción de los principales casos de uso

A continuación se muestran la descripción de los casos de uso críticos. (Para ver el resto dirigirse al Anexo 2).

Tabla 2. Descripción del caso de uso ''Iniciar transmisión''.

Objetivo	Iniciar la transmisión de las programaciones planificadas de forma automática.	
Actores	Operador de transmisión (Inicia) Transmisión de canales.	
Resumen	El caso de uso inicia cuando una vez autenticado el operador de transmisión, se lista la programación de medias previamente realizada y se transmite automáticamente en cada canal de acuerdo a la hora planificada la media con la hora de inicio posterior más cercana a la hora actual . Termina este caso de uso cuando se transmiten todas las medias.	
Complejidad	Alta.	
Prioridad	Crítico.	
Precondiciones	El usuario con rol operador de transmisión está autenticado. Se listaron los canales con su programación planificada.	
Postcondiciones	Se transmitió la programación de todos los canales.	
Flujo de eventos		
Flujo básico Realizar transmisión automática		
	Actor	**Sistema**
1	Inicia la aplicación.	
2		Valida las opciones para realizar una transmisión. 2.1 Verifica que exista al menos una programación para el día. 2.2 Verifica que se pueda acceder a las medias planificadas.

3		Cuando la hora actual coincide con la hora de inicio planificada de una media dirigirse al caso de uso Transmitir media.
4		Cuando recibe la notificación del Gestor de la culminación de la transmisión selecciona la siguiente media y se dirige al caso de uso Transmitir media, hasta que culmine la lista de programación. Termina el caso de uso.
Requisitos Funcionales	Iniciar transmisión de forma automática. Seleccionar automáticamente media a transmitir.	
Relaciones	**CU Incluidos**	Transmitir medias.
	CU Extendidos	No existen.

Tabla 3. Descripción del caso de uso ‘’Transmitir medias’’.

Objetivo	Realizar la transmisión de la media seleccionada.
Actores	Caso de uso Administrar transmisión y caso de uso Iniciar transmisión (Inicia). Transmisión de las medias.
Resumen	El caso de uso se inicia cuando el subsistema envía los datos de la transmisión y de la media seleccionada al Gestor y se reproduce dicha media. Termina el caso de uso.
Complejidad	Alta.
Prioridad	Crítico.
Precondiciones	El usuario con rol operador de transmisión está autenticado. Se listaron los canales con su programación planificada. Debe estar seleccionado un canal. Debe estar seleccionada una media.
Postcondiciones	Se transmitió la media deseada.

<table>
<tr><td colspan="4">Flujo de eventos</td></tr>
<tr><td colspan="4">Flujo básico Transmitir medias</td></tr>
<tr><td></td><td colspan="2">Actor</td><td>Sistema</td></tr>
<tr><td>1</td><td colspan="2"></td><td>Detiene la media que se está transmitiendo si esta pertenece al canal seleccionado.</td></tr>
<tr><td>2</td><td colspan="2"></td><td>Crea el identificador de la transmisión y busca los restantes datos de la media seleccionada y la transmisión (IP y Puerto del canal seleccionado, dirección física de la media seleccionada, subtítulo, logo, gama y puerto destino).</td></tr>
<tr><td>3</td><td colspan="2"></td><td>Se conecta con el Gestor y le envía los datos antes mencionados.</td></tr>
<tr><td>4</td><td colspan="2"></td><td>Recibe notificación de la transmisión (nombre del proceso y el flujo en el que se encuentra).</td></tr>
<tr><td>5</td><td colspan="2"></td><td>Visualiza la media seleccionada. Termina el caso de uso.</td></tr>
<tr><td colspan="2">Requisitos Funcionales</td><td colspan="2">Transmitir media.
Visualizar transmisión del canal televisivo.</td></tr>
<tr><td colspan="2" rowspan="2">Relaciones</td><td>CU Incluidos</td><td>No existen.</td></tr>
<tr><td>CU Extendidos</td><td>No existen.</td></tr>
</table>

3.4 Modelo de análisis

El modelo de análisis estructura los requisitos de un modo que facilita su comprensión, preparación, modificación y en general su mantenimiento. Puede considerarse como la primera aproximación al modelo de diseño (Jacobson, 2000).

El análisis es una etapa de la metodología RUP, de la cual se puede prescindir de acuerdo a la manera en que se haya decidido implementar el sistema, pasando a analizar los requisitos como parte del diseño o como parte integrada de la captura de los requisitos, lo que demuestra que la manera de emplear el análisis es consecuente con la construcción del sistema (Jacobson, 2000).

Por lo antes expuesto, se decide pasar del flujo de requisitos directamente al diseño, sin la realización del análisis. Además, los requisitos a implementar en el subsistema a desarrollar son bien conocidos y se cuenta con una comprensión de los mismos. En general, las tecnologías sobre las que se decidió desarrollar dicho subsistema son conocidas, lo que permite prescindir del refinamiento de los requisitos e ir directo al diseño.

3.5 Definición de la arquitectura de software

La arquitectura de software es la estructura u organización de los componentes del programa, la manera en que estos componentes interactúan, y la estructura de datos que utilizan los mismos. En un sentido más amplio, los componentes pueden generalizarse para representar elementos importantes del sistema y sus interacciones (Pressman, 2005).

La selección de un estilo y un patrón arquitectónico es una decisión elemental del diseño en el desarrollo de un sistema. Para la construcción del subsistema se utiliza el estilo arquitectónico de llamada-retorno y dentro de este, el patrón arquitectónico en capas, específicamente en tres capas. Este estilo arquitectónico permite que se obtenga una estructura del programa que resulte relativamente fácil modificar y cambiar de tamaño, mientras que el patrón arquitectónico seleccionado se utiliza con el objetivo de separar la capa de presentación, la capa de negocio y la capa de datos.

Según Pressman en su libro Ingeniería de Software un enfoque práctico: la arquitectura de tres capas permite aislar a la tecnología que implementa la base de datos, de forma que sea fácil cambiar esta tecnología. Utiliza mucho código lejos del cliente y los cambios de mantenimiento ocurren de forma centralizada. La idea de las tres capas encaja con las prácticas orientadas a objetos de hoy en día: todo el procesamiento tiene lugar por medio de los mensajes que se envían a los objetos y no mediante trozos de código asociados a cada objeto.

La siguiente figura representa el patrón arquitectónico en tres capas, donde las clases del diseño fueron agrupadas correspondientemente en cada una de ellas: presentación, lógica del negocio y acceso a datos.

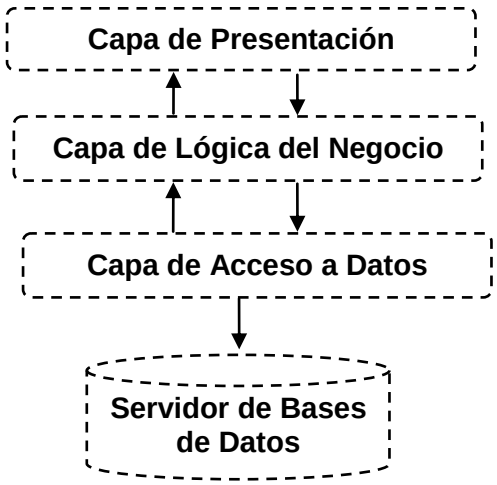


Figura 4. Patrón arquitectónico 3 capas.

La **capa de presentación** es la que presenta el sistema al usuario, le comunica la información y captura la información del usuario en un mínimo de proceso. Esta capa se comunica únicamente con la capa de negocio. También es conocida como interfaz gráfica y generalmente se presentan como formularios.

La **capa de negocio** es donde se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados y con la capa de datos, para solicitar al gestor de base de datos almacenar o recuperar datos de él.

La **capa de datos** es donde residen los datos y es la encargada de acceder a los mismos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

3.6 Patrones de diseño

Los patrones de diseño, constituyen soluciones estándar que brindan respuesta a un problema común en materia de diseño de sistemas e implementación. Estos permiten establecer una gran comunicación entre los diseñadores, facilitando además el aprendizaje del programador inexperto y logrando crear parejas problema-solución. Cada patrón sugiere además numerosas ventajas, dentro de ellas la reutilización de código y el permitir la realización de un diseño apto para el cambio. Los patrones de diseño se aplican a un elemento específico de diseño como un agregado de componentes para resolver algún problema de diseño, relaciones entre los componentes o los mecanismos para efectuar las comunicaciones de componente a componente.

En la construcción del modelo de diseño fueron utilizados los patrones GRASP: Experto, Creador, Bajo acoplamiento, Alta cohesión y Controlador. También fueron utilizados los patrones GoF: Observador y Singleton.

3.6.1 Patrones GRASP

Los patrones de diseño GRASP (General Responsibility Asignment Software Patterns) o patrones de asignación de responsabilidades describen los principios fundamentales de la asignación de responsabilidades a objetos, expresado en forma de patrones (Larman, 1999).

- *Experto*: El patrón Experto propone la asignación de la responsabilidad de experto a la clase que cuenta con la información necesaria para cumplir con la misma. Si esta asignación se hace de manera correcta se puede entender el subsistema de una mejor manera siendo más fácil de mantener y ampliar, brindando la oportunidad de reutilizar sus componentes en futuras aplicaciones (Larman, 2003). Este patrón se evidencia en la clase Programación, ya que la misma contiene un objeto de canal y el listado de medias. Además es la clase que puede iniciar y realizar cualquier operación sobre una transmisión.

- *Creador*: El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos. Se aplicará en todos los casos donde una clase tiene la responsabilidad de crear una nueva instancia de la otra. La correcta asignación permite que la aplicación pueda soportar un bajo acoplamiento, una mayor claridad, el encapsulamiento y reutilización (Larman, 1999). Este patrón fue aplicado en la clase Televisión, la cual es la encargada a partir de los datos de la base de datos crear todas las programaciones para ese día.

- *Bajo acoplamiento***:** Este patrón estimula a asignar responsabilidades de modo tal que no influya en que las dependencias de una clase sean de muchas otras. Tener un bajo acoplamiento soporta el diseño de clases más independientes que reducen el impacto de los cambios y permite una mayor reutilización de código (Larman, 1999). Este patrón fue utilizado en todas las clases de la solución propuesta.

- *Alta cohesión***:** Se ocupa de que las clases del diseño realicen las funcionalidades necesarias para cumplir con las tareas que tienen definidas. Plantea la contribución entre clases para realizar tareas de elevada complejidad (Larman, 1999). Este patrón fue utilizado en todas las clases de la solución propuesta.

- *Controlador***:** Define quién deberá encargarse de atender un evento del sistema. Es una clase que para el diseñador representa de alguna manera al sistema global (Larman, 2003). Este patrón se evidencia a través de la clase Programación.

3.6.2 Patrones GoF

Los patrones GoF (Gang of Four) o pandilla de los cuatro como se le conoce son agrupados en tres grandes categorías basadas en su propósito: creacionales, estructurales y de comportamiento (Larman, 2003). Los estructurales tratan la combinación de clases, su relación y la formación de estructura de alta complejidad, mientras que los creacionales tratan la creación de instancias y los de comportamientos tratan la interacción y la cooperación entre clases.

- *Observador***:** Define una dependencia “uno a muchos” entre objetos, para que cuando uno de ellos cambie su estado, todos los que dependan de él sean avisados y puedan actualizarse convenientemente. Este patrón se evidencia en la clase Tabla_item y Tabla_pausada_item para que los elementos pausados posean una referencia exacta de los elementos a quienes correspondan en la lista de programación.

- *Singleton (Instancia única):* Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. Este patón es utilizado en la clase Comunicación, la cual constituye una instancia común para todas las programaciones.

3.7 Modelo de diseño

El modelo del diseño es un modelo de objetos que describe la realización física de los casos de usos centrándose en cómo los requisitos funcionales y requisitos no funcionales junto con otras restricciones relacionadas con el entorno de la implementación tienen impacto en el sistema a considerar. Sirve de abstracción de la implementación del sistema y es de ese modo utilizada como una entrada fundamental de las actividades de implementación (Jacobson, 2000).

3.7.1 Diagrama de clases del diseño

El diagrama de clases del diseño muestra cómo quedaría implementada toda la aplicación en términos lógicos. Representa las clases que serán utilizadas dentro del sistema y las relaciones que existen entre ellas. Los diagramas de clases del diseño constituyen un elemento fundamental en la concepción de la aplicación que se propone, ya que servirán de guía a los desarrolladores al constituir una aproximación del sistema que se desea implementar, contribuyendo de esta forma a la calidad del producto final.

A continuación se muestran los diagramas de clases del diseño pertenecientes a los casos de uso críticos del subsistema. En estos diagramas se reflejan cada una de las capas, clases y métodos que serán utilizados para la realización de estas funcionalidades. (Para ver el resto dirigirse al Anexo 3).

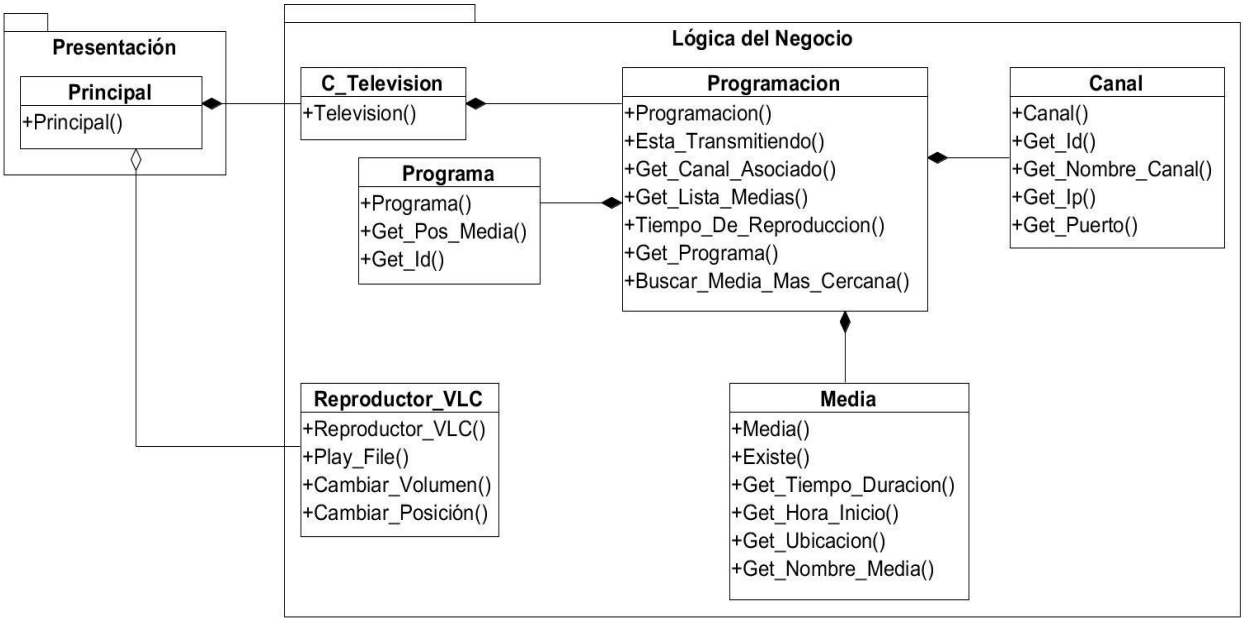


Figura 5. Diagrama de clase del diseño del caso de uso “Iniciar transmisión”.

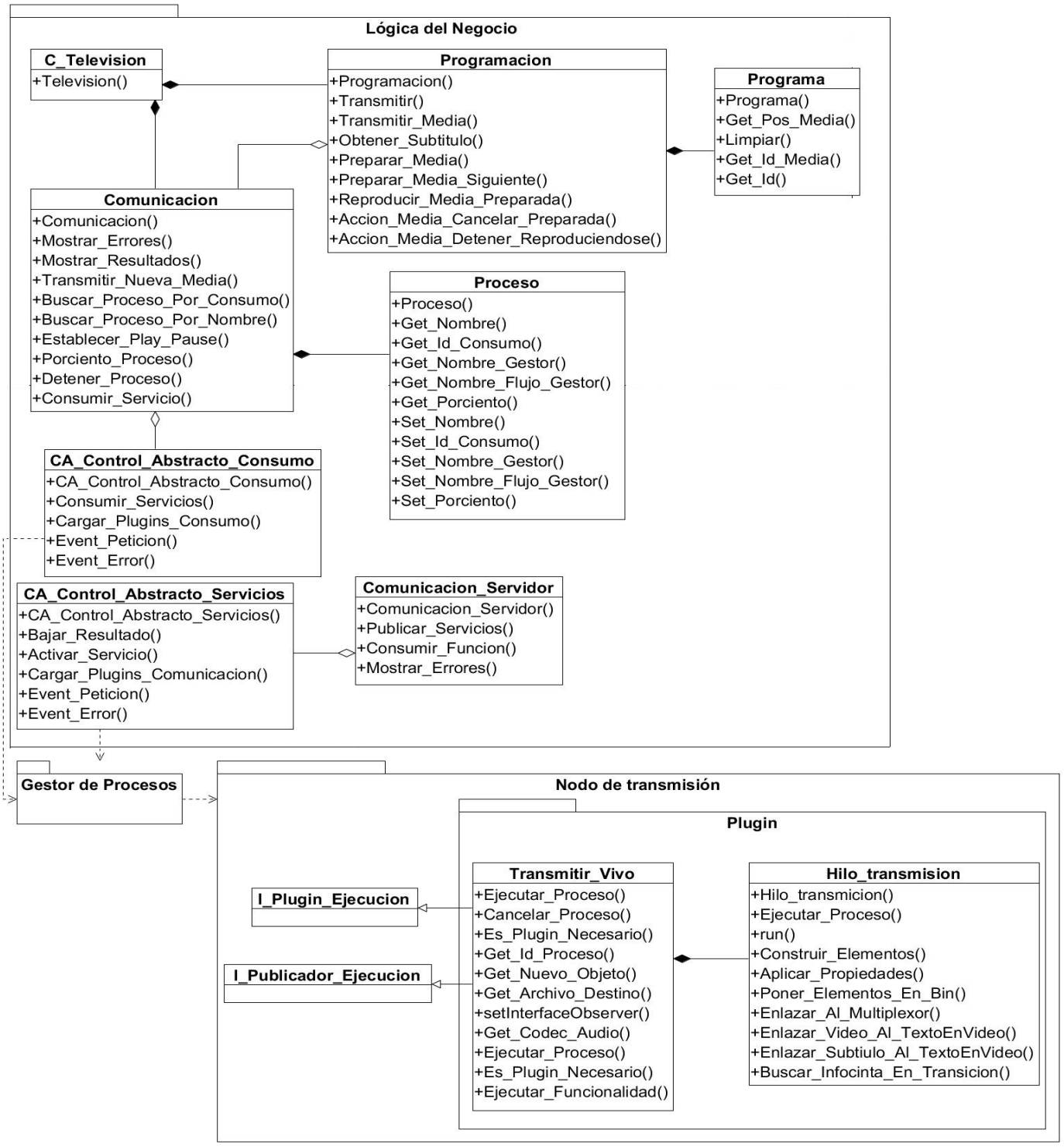


Figura 6. Diagrama de clase del diseño del caso de uso “Transmitir medias”.

3.8 Modelo de despliegue

El modelo de despliegue es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo. Se utiliza como entrada fundamental en las actividades de diseño e implementación debido a que la distribución del sistema tiene una influencia principal en su diseño (Jacobson, 2000).

3.8.1 Diagrama de despliegue

Un diagrama de despliegue es un grafo de nodos unidos por conexiones de comunicación que muestra las relaciones físicas entre los componentes hardware y software.

A continuación se muestra el diagrama de despliegue del subsistema, el cual está compuesto por cinco nodos. Al nodo **Servidor de Base Datos** se conectará la aplicación para consultar la información necesaria para realizar el proceso de transmisión. En el nodo **Almacén de Medias** se encuentran las medias que se van a transmitir. El nodo **Cliente de transmisión** es el que contiene la aplicación cliente, la cual es la encargada de llevar el control de las medias a transmitir, así como las operaciones realizadas sobre las mismas durante el proceso de transmisión. El nodo **Gestor de procesos** tiene como objetivo lograr un balanceo de carga entre los nodos de transmisión asociados. El **Nodo de transmisión** es el responsable de cargar los plugins tanto de lógica como de ejecución para la transmisión de una media determinada. Para establecer la conexión con el servidor de base datos se utiliza el librería QtSQL, para conectarse al Gestor de procesos y al Nodo de transmisión se hace uso del protocolo XML/RPC y para comunicarse con el Almacén de Medias es utilizado el protocolo NFS.

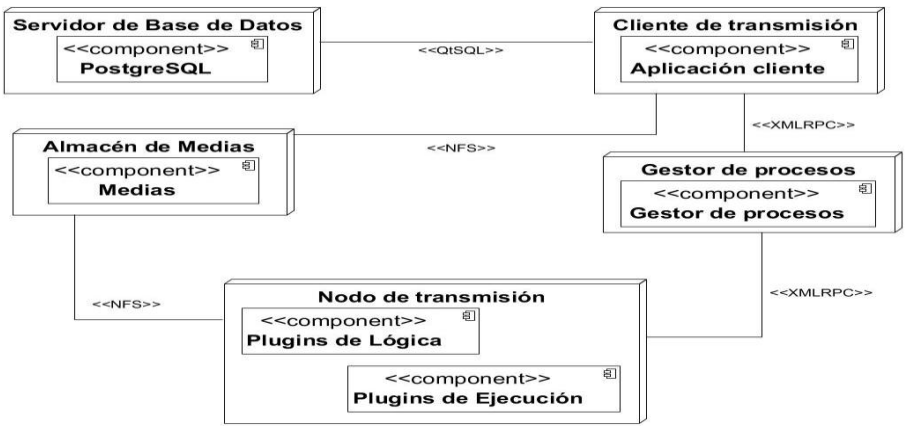


Figura 7. Diagrama de despliegue.

3.9 Modelo de implementación

El modelo de implementación está compuesto por un conjunto de sistemas y componentes que establece la composición física de la implementación del sistema (Jacobson, 2000).

3.9.1 Diagrama de componentes

Los diagramas de componentes son la estructura del modelo de implementación. Este diagrama presenta paquetes donde se agrupan los elementos físicos de un sistema, así como las relaciones entre ellos.

A continuación se muestra el diagrama de componentes perteneciente al subsistema distribuido de transmisión de contenido audiovisual. En el mismo se representan cada uno de los elementos físicos resultantes de la implementación de dicho subsistema, agrupados en 3 capas, de acuerdo con el patrón arquitectónico escogido. En la capa Presentación se agrupan las clases interfaces, con las cuales el operador de transmisión interactúa para realizar determinadas operaciones sobre el subsistema. En la capa Lógica del Negocio se encuentran las clases que reciben y dan respuesta a las peticiones del operador de transmisión, además de poseer la capacidad de comunicarse en caso de ser necesario con las clases ubicadas en la capa de Acceso a Datos, las cuales tienen el objetivo de almacenar o recuperar datos. Además se evidencia en dicho diagrama la relación entre las clases encargadas de la comunicación con el Gestor de procesos, así como las clases pertenecientes al Plugin, las cuales tienen como principal objetivo lograr una transmisión correcta de las medias. Se representan también todas las bibliotecas utilizadas en la implementación del subsistema, las cuales se encuentran agrupadas en el paquete de igual nombre.

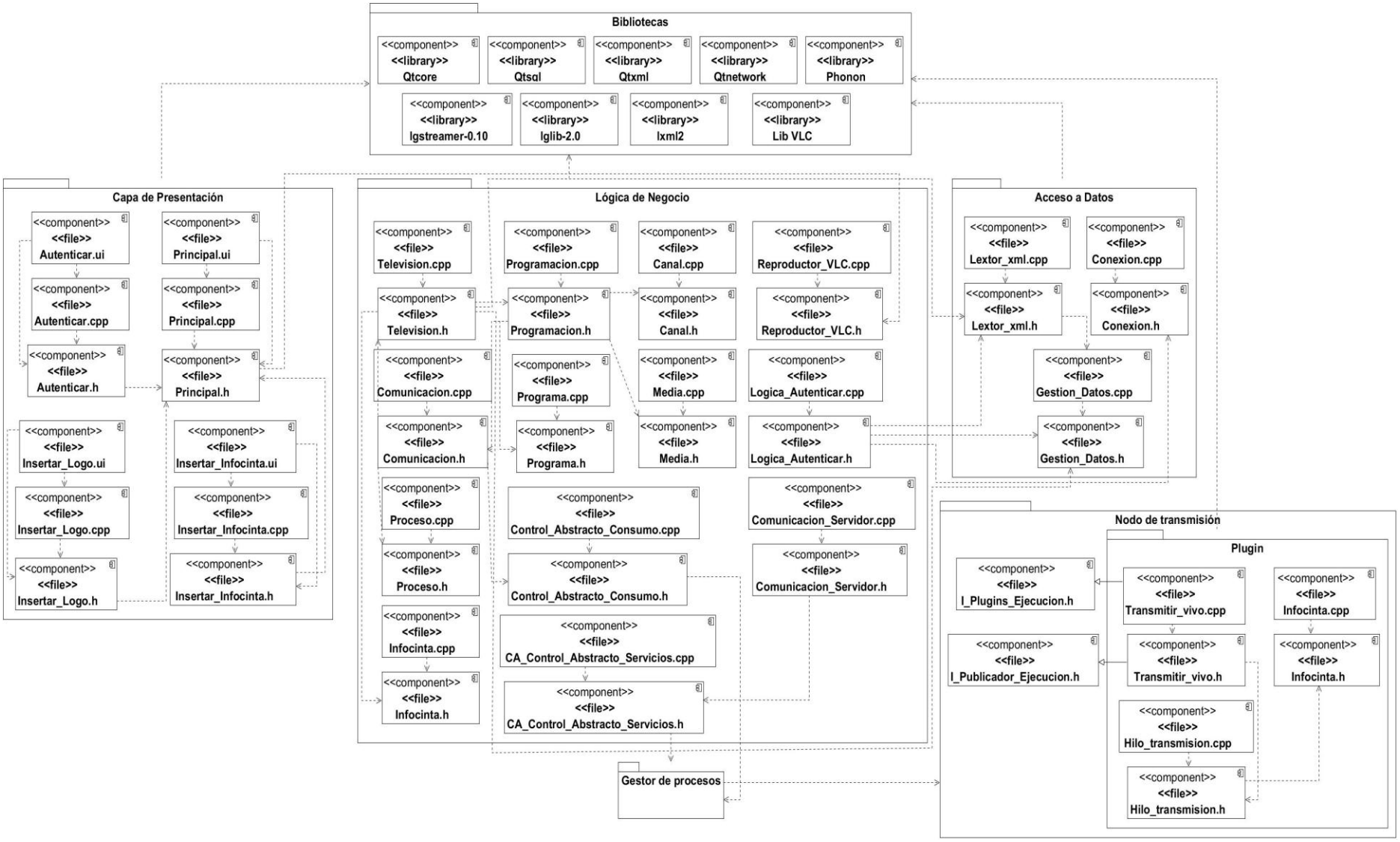


Figura 8. Diagrama de componentes.

3.10 Estándar de codificación

Las convenciones o estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código. A continuación se reflejan las convenciones utilizados para la implementación del subsistema:

- **Identificadores de variables**
  - Comenzarán siempre con la primera letra minúscula.
  - Para distinguir palabras dentro del nombre deberá emplearse un guión bajo.
- **Identificadores de punteros (apuntadores)**
  - Su nombre deberá comenzar con la letra p.
- **Identificadores de funciones**
  - La primera letra de cada palabra deberá ser mayúscula.
  - Para distinguir palabras dentro del nombre deberá emplearse un guión bajo.
- **Organización visual**
  - No manejar en los programas más de una instrucción por línea.
  - Declarar las variables en líneas separadas.

3.11 Conclusiones parciales

- La captura de los requisitos ayudó a precisar las prestaciones y características del subsistema.
- La elaboración del diagrama de casos de uso y la especificación de los mismos, permitió desarrollar las funcionalidades teniendo en cuenta el flujo de interacción del usuario y el subsistema y guiar las pruebas de software.
- La utilización de la arquitectura en tres capas propició una correcta estructuración del código fuente de la solución propuesta, posibilitando mayor independencia entre las clases.
- La definición de las clases del diseño y el uso de patrones del diseño permitió suprimir problemas de interacción y responsabilidad en el código fuente.

Capítulo 4: Validación del SDTCA

Introducción

En este capítulo se validará la solución propuesta mediante la realización de pruebas con el objetivo de medir la calidad del subsistema desarrollado.

4.1 Pruebas de software

Las pruebas de software permiten verificar y revelar la calidad de un producto de software. Las mismas tienen como objetivo probar el funcionamiento del software en su máxima capacidad y tratar de corregir todos los errores existentes en el mismo. Son un elemento crítico para la garantía de la calidad del software y representa una revisión final de las especificaciones del diseño y la codificación.

A continuación se exponen los niveles de pruebas existentes, los cuales se aplican a diferentes tipos de objetivos, en diferentes escenarios o niveles de trabajo.

- *Prueba de desarrollador:* Indica los aspectos de diseño e implementación de las pruebas más adecuadas que debe llevar a cabo el equipo de desarrolladores. En la mayoría de los casos, la ejecución de la prueba se produce inicialmente con el grupo de pruebas de desarrollador que la diseñó e implementó; aunque es recomendable que los desarrolladores creen las pruebas de forma que estén disponibles para que las ejecuten grupos de pruebas independientes.

- *Prueba independiente*: Indica el diseño y la implementación de la prueba realizada adecuadamente por alguien ajeno al equipo de desarrolladores. Puede considerar esta distinción un conjunto, que incluye validación y verificación independientes.

- *Prueba de unidad:* Se centra en la verificación de los elementos más pequeños del software que se puedan probar. Normalmente, las pruebas de unidad se aplican a componentes representados en el modelo de implementación para verificar que se cubren los flujos de control y los flujos de datos y que funcionan como se esperaba. El implementador realiza la prueba de unidad mientras se desarrolla la unidad.

- *Prueba de integración:* Las pruebas de integración se realizan para garantizar que los componentes del modelo de implementación funcionan correctamente cuando se combinan para

ejecutar un guión de uso. El destino de la prueba es un paquete o un conjunto de paquetes del modelo de implementación. Las pruebas de integración exponen el estado incompleto o los errores de las especificaciones de la interfaz del paquete.

- *Prueba del sistema*: Normalmente, la prueba del sistema se realiza cuando el software funciona en su totalidad. Un ciclo vital repetitivo permite que las pruebas del sistema se realicen mucho antes, en cuanto se hayan implementado subconjuntos bien formados del comportamiento de guiones de uso. Normalmente, el destino son los elementos en funcionamiento de extremo a extremo del sistema.

- *Prueba de aceptación:* La prueba de aceptación del usuario es la última acción de prueba antes de desplegar el software. El objetivo de la prueba de aceptación es comprobar si el software está preparado y lo pueden utilizar los usuarios para realizar las funciones y tareas para las que se diseñó.

Existen además diferentes tipos de pruebas, los cuales son:

Dimensión de calidad	**Tipos de prueba**
Funcionalidad	Función, Seguridad, Volumen.
Usabilidad	Usabilidad.
Fiabilidad	Integridad, Estructura, Stress.
Rendimiento	Contención, Carga.
Soporte	Configuración, Instalación.

Lugo de analizar los niveles y tipos de pruebas antes expuestos, se decidió aplicar pruebas funcionales a nivel de sistema utilizando el método de caja negra y pruebas estructurales a nivel de unidad utilizando el método de caja blanca.

4.2 Método de pruebas de caja blanca

La prueba de caja blanca, en ocasiones llamada prueba de cristal, es un método que usa la estructura de control descrita como parte del diseño al nivel de componentes para derivar los casos de prueba (Pressman, 2007). En estas pruebas se comprueban los caminos lógicos del software proponiendo casos de prueba que examinen que están correctas todas las condiciones y/o bucles para determinar si el estado real coincide con el esperado.

Mediante el método de prueba de caja blanca, se obtiene casos de prueba que:

- Garantizan que se ejercita por lo menos una vez todos los caminos independientes de cada módulo.
- Se ejerciten todas las decisiones lógicas en sus vertientes verdadera y falsa.
- Se ejecuten todos los bucles en sus límites y con sus límites operacionales.
- Se ejerciten las estructuras internas de datos para asegurar su validez.

Es por ello que se considera a la prueba de caja blanca como uno de los tipos de pruebas más importantes que se le aplican a un software, logrando como resultado que disminuya en un gran porciento el número de errores existentes en el sistema y por ende una mayor calidad y confiabilidad del mismo (Pressman, 2005).

En la aplicación de esta prueba se utilizó la técnica del camino básico. Esta técnica permite obtener una medida de la complejidad lógica de un diseño y usar esta medida como guía para la definición de un conjunto básico de rutas de ejecución.

Los pasos para la realización de esta técnica son los siguientes:

- **Paso 1**: Generar el grafo de flujo de datos a partir del código/diseño procedimental (Nodos, Aristas, Regiones).
- **Paso 2**: Calcular la complejidad ciclomática V(G): métrica del software que da una medición de la complejidad de un programa.
    - V(G) = NA (Número de Aristas) - NN(Número de Nodos) + 2.
    - V(G) = P(Nodos predicados) + 1.

- V(G) = Número de regiones.

- **Paso 3**: Definir un conjunto básicos de caminos de ejecución a partir de la complejidad ciclomática.
- **Paso 4**: Generar un caso de prueba para cada camino de ejecución.

4.2.1 Diseño de casos de pruebas de caja blanca

A continuación se muestra el diseño de caso de prueba de caja blanca para la función más crítica implementada en la solución a probar.

- **Prueba al método Transmitir_Medias de la clase Programación.**

```
void Programacion::Transmitir_Media(int p_pos_media)
{
 if(transmitiendo) {                                              1
 Accion_Media_Detener_Reproduciendose();                          2
 }
 if(media_para_reproducir->Disponible()) {                        3
 Accion_Media_Cancelar_Preparada();                               4
 }
 if (automatica)                                                  5
 {
 disconnect(reloj,SIGNAL(timeout()),this,SLOT(Transmitir()));     6
 reloj->stop();
 automatica = false;
 }
 if (p_pos_media < plista_medias_copia->count())                  7
 {
  media_para_reproducir->Limpiar();                               8
  pivote_play = p_pos_media;
  if (plista_medias_copia->at(p_pos_media)->Existe()&& plista_medias_copia->at(p_pos_media) - >
Si_se_quiere_reproducir()==true)                                  9
 {
```

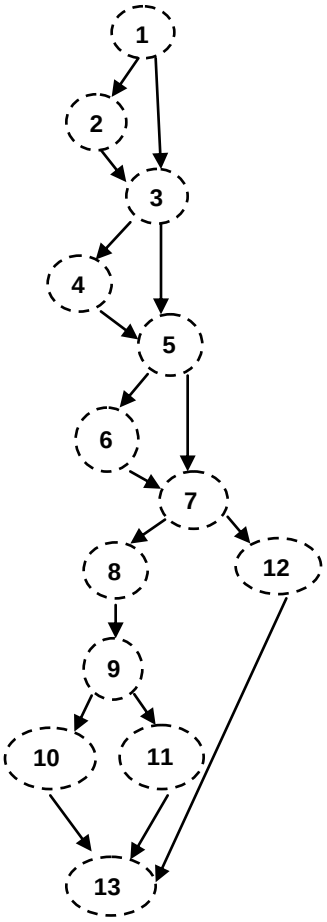
```
 int id_transmision=Costruir_Id();
 media_reproduciendose= new Programa(id_transmision,p_pos_media,
 plista_medias_copia->at(p_pos_media)->Get_Id(),plista_medias_copia->
 at(p_pos_media)->Get_Id_Media_Externa());
 QString sid_proceso = QString::number(id_transmision);
 QString dir_origen = "Transmitir::" + plista_medias_copia->at(p_pos_media)
 Get_Ubicacion_Fisica();
 QString host_destino = pcanal->Get_Ip();
 QString subtitulo = Obtener_Subtitulo();
 QString logo = this->logo; //Logo
 int gamma = 100;
 int puerto_destino = pcanal->Get_puerto();
 pcomunicacion- Transmitir_Nueva_Media(sid_proceso,dir_origen,host_destino,
 subtitulo,logo,gamma, puerto_destino);
 ptimerporciento->start(500);
 porciento = 0;
 transmitiendo=true;
}
else
{
 Transmitir_Media(p_pos_media+1);
}
}else
{
emit Canal_Finalizado(id_canal_para_transmision);
}
}
```

10

11

12

13



- **Complejidad Ciclomática:**

V(G)= A – N + 2.

V(G) =17 – 13 + 2.

V(G) = 6.

- **Caminos Independientes:**

Camino 1: 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 10 – 13.

Camino 2: 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 11 – 13.

Camino 3: 1 - 2 – 3 – 4 – 5 – 6 – 7 – 12 – 13.

Camino 4: 1 – 3 – 5 – 7 – 8 – 9 – 10 – 13.

Camino 5: 1 – 3 – 5 – 7 – 8 - 9 – 11 – 13.

Camino 6: 1 – 3 – 5 – 7 – 12 – 13.

A continuación se muestra el caso de prueba perteneciente al camino [1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 10 – 13]. (Para ver el resto dirigirse al Anexo 4).

Tabla 4. Caso de prueba para el camino [1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 10 – 13].

Función	Transmitir_Medias
Descripción	Si se desea transmitir una media y se está transmitiendo otra, detiene la transmisión de la misma y si existe una media próxima a reproducirse, deja de prepararla para su emisión. En el caso que no haya culminado la lista de medias, si la media que se desea transmitir existe físicamente y puede ser transmitida, construye y obtiene los datos necesarios para transmitirla.
Entrada	La función se invoca cuando se desea transmitir una media.
Resultado esperado	Se transmite una media.
Resultado de la prueba	Satisfactorio.

El resultado de la prueba de caja blanca aplicada al método anteriormente expuesto arrojó un resultado satisfactorio, ya que todos los nodos son visitados al menos una vez en alguno de los caminos básicos y todas las operaciones son válidas. Después de realizado un análisis detallado de este resultado se llega a la conclusión de que el subsistema no presenta ningún problema en cuanto a la funcionalidad del código fuente para transmitir una media.

4.3 Método de prueba de caja negra

Las pruebas de caja negra, también denominadas pruebas de comportamiento, se encuentran centradas en los requisitos funcionales del software. Este método no es una opción frente al método de caja blanca, sino un enfoque complementario que tiene probabilidades de descubrir una clase diferente de errores de los que se descubrirían con los métodos de caja blanca. Este método de prueba es realizado a nivel de sistema, es decir no se tiene ninguna relación con el código del producto (Pressman, 2007).

Las pruebas de caja negra se llevan a cabo sobre la interfaz del software, por lo que los casos de prueba pretenden demostrar que:

- Las funciones del software son operativas.
- La entrada se acepta de forma adecuada y que se produce una salida correcta.
- La integridad de la información externa se mantiene.

En la realización de esta prueba se utilizó la técnica de partición equivalente. Esta es una técnica que divide el dominio de entrada de un programa en clases de datos, a partir de las cuales pueden derivarse casos de prueba. La partición equivalente se esfuerza por definir un caso de prueba que descubra ciertas clases de errores, reduciendo así el número total de casos de prueba que deben desarrollarse (Pressman, 2007).

4.3.1 Resultado de los casos de prueba basados en casos de uso

A las funcionalidades Autenticar usuario, Administrar transmisión, Insertar logo y Gestionar infocintas, durante la fase de prueba del proceso de desarrollo de software se les aplicó el método de prueba de caja negra. (Para ver el diseño de casos de prueba de estas funcionalidades dirigirse al Anexo 5).

En esta fase del proceso de desarrollo de software se realizaron dos iteraciones a las funcionalidades antes mencionadas. En la primera iteración se encontraron dos no conformidades, ya que las funcionalidades "Insertar logo" y "Gestionar infocintas" no funcionaban correctamente. En el caso de la funcionalidad "Insertar logo" la posición donde sería ubicado el mismo en el video presentaba problemas en dependencia del tamaño de este. Mientras que la funcionalidad "Gestionar infocintas" presentaba dificultad en cuanto al número de veces que sería repetida la misma en el material que se estaba visualizando.

Posteriormente se realizó una segunda iteración de prueba para verificar que se habían eliminado los problemas de estas funcionalidades o en busca de nuevos problemas en el funcionamiento del subsistema. Esta segunda fase concluyó satisfactoriamente ya que no se encontró ningún problema en el subsistema implementado.

4.4 Conclusiones parciales

- La realización de las pruebas funcionales (caja negra) permitió validar las funcionalidades de la aplicación en correspondencia con los requisitos previamente capturados.

- La ejecución de pruebas unitarias (caja blanca) al subsistema, permitió validar el correcto funcionamiento del código de la aplicación, garantizando que no existan ciclos infinitos o código sin ejecutar.

Conclusiones generales

Luego de haberse realizado todas las tareas de la investigación se puede afirmar que se cumplieron satisfactoriamente las mismas, arribando a las siguientes conclusiones:

- Con la utilización del protocolo UDP para la emisión de streaming se logró una recepción rápida de los datos.
- El análisis del servidor Flumotion, demostró que el mismo no resultaba factible para el desarrollo de la solución a implementar.
- La utilización de la biblioteca GStreamer posibilitó a la aplicación el procesamiento de diferentes tipos de flujo de datos y multiplexar audio, video, imágenes y subtítulos.
- El proceso de desarrollo permitió generar todos los artefactos y documentación correspondientes al mismo.
- Con la realización de las pruebas de caja negra y caja blanca se logró probar el buen funcionamiento de la aplicación obteniendo un subsistema capaz de transmitir contenido audiovisual.

Recomendaciones

Luego de haber concluido el subsistema propuesto y cumplido los objetivos trazados, se plantean las siguientes recomendaciones:

- Desplegar el SDTCA en el producto STCV.
- Continuar el desarrollo de la aplicación llevando a cabo la implementación de nuevas funcionalidades tales como: el perfeccionamiento para la muestra de contenido alternativo y la visualización de una transmisión que proceda de una fuente origen en vivo.
- Implementar un módulo que posibilite la transmisión de contenido audiovisual bajo demanda.
- Divulgar el presente documento, permitiendo que sea utilizado como guía para futuras actualizaciones del Subsistema de Transmisión.

Bibliografía y Referencias bibliográficas

1. **Abadía, Alwjandro Romero.** *Proyecto de streaming de video.* 2011.
2. **Albán, Oscar Andrés Vivas.** *Introducción a Qt y QtCreator.* 2011.
3. **Arévalo, Jeans Manuel Arias.** *Procedimientos y técnicas de transmisión de señales televisivas para la plataforma PTARTV.* 2011.
4. **Barreiros, Federico Enmanuel.** *Video-conference system based on open source software.* 2012.
5. **Bernal, Hugo Javier.** *Análisis e implementación de un sistema video.* 2010.
6. **Childers, David.** *Icecast Streaming Handbook.* 2010.
7. **Cicileo, Guillermo.** *Multicast.* 2010.
8. **Colectivo de autores de la Universidad de Boston.** *La compresión de video y adobe media Encoder.* 2010.
9. **Corona, Melecio.** La TV y su influencia en el mundo moderno. [En línea] 2009. [Citado el: 21 de octubre de 2013].
10. **Cristian Perugachi, Julio Tamayo.** *Servidor de streaming con redes en Ubuntu 11.10.* 2011.
11. **Eiken, Observatorio Estratégico de Clúster.** *Consumo audiovisual a través de la red.* 2011.
12. **Giraldo Alvaro Roldán, Carlos Andrés, Cristian Benitez.** *Protocolo UDP.* 2010.
13. **González, Carlos Daniel.** Usabilidad web. Introducción a C++ y a la Resolución de problemas. [En línea] 2008. [Citado el: 6 de febrero de 2013].
14. **González, José.** *SVD para la transmisión progresiva de imágenes y la codificación de video digital.* 2009.
15. **Hall, Prentice.** *Distributed Operating Systems.* 1995.
16. **Hurtado, Omar.** *Nuevos paradigmas de los sistemas de información.* 2008.
17. **Ivar Jacobson, Grady Booch, James. Rumbaugh.** *El Proceso Unificado de Desarrollo de Software.* 2000.
18. **Ivar Jacobson, Grady Booch, James. Rumbaugh.** *El Lenguaje Unificado de Modelado.* 2005.

19. **Lafuente., Alberto.** *Introducción a los sistemas distribuidos.* 2011.
20. **Larman, Craig.** *Modelo del Dominio.* 2003.
21. **Larman, Craig.** *UML y Patrones Introducción al análisis y diseño orientado a objetos.* 1999.
22. **Laurent, Simon St.** *Programming Web Services with XML-RPC.* 2001.
23. **M Thornburgh, H Parmar.** *Adobe's Real Time Messaging Protocol.* 2012.
24. **Maldonado, Daniel.** El codigo K. [En línea] 2011. [Citado el: 7 de marzo de 2013].
25. **Mora, Sergio Luján.** Grupo de Investigación en Procesamiento(GPLSI). [En línea] 2009. [Citado el: 4 de febrero de 2013].
26. **Neuman.** *Proxy-Based Autorización y Contabilidad de Sistemas Distribuidos.* 1994.
27. **Nokia.** Qt. [En línea] 2011. [Citado el: 6 de febrero de 2013].
28. **Novoa, Pablo Montero.** *Servidor modular de streaming.* 2007.
29. **Pressman, Roger.** *Ingenería del Software. Un enfoque práctico.* 2005.
30. **Pressman, Roger.** *Ingeniería de software.* 2007.
31. **Quintero, Juan Pablo.** *Evaluación de servidores de streaming de video.* 2006.
32. **Reinosos, Carlos Billy.** *Introducción a la arquitectura de software.* 2004.
33. **Ribas-Corbera, Jordi.** *Windows Media Series.* 2003.
34. **Saínz, Javier.** *Diseño e implementación FPGA de un Estimador de Movimiento de Tamaño Variable para el H.264.* 2007.
35. **Santos, Jorge Chávez.** *Protocolos.* 2007.
36. **Scribd.** Herramientas CASE. [En línea] 2010. [Citado el: 2 de febrero de 2013].
37. **Sierra, Daniel.** Visual Paradigm For Uml. [En línea] 2007. [Citado el: 4 de febrero de 2013].
38. **Smith, Michael.** *GStreamer.* 2008.
39. **Sommerville, Ian.** *Ingeniería del Software.* 2005.
40. **Torres, Daniel Oscar.** *Evolución y tendencia de la tecnología streaming en Internet.* 2009.

41. **Wim Taymans, Steve Baker, Andy Wingo, Ronald S. Bultje, Stefan Kost.** *GStreamer Application Development.* 2012.

Glosario de términos

- **Plugins**: Aplicación diseñada para adicionarle funciones nuevas a un software en específico, sin afectar ni comprometer el funcionamiento de este.

- **Subsistema:** Conjunto de elementos interrelacionados entre sí que tienen un propósito determinado. En términos de gestión, cuenta con varios procesos básicos: las entradas, el procesamiento, el almacenamiento y las salidas.

- **Streaming:** Consiste en la distribución de audio o video sin interrupción, donde el usuario puede escuchar música o ver videos sin necesidad de ser descargados previamente. Este tipo de tecnología permite que se vaya almacenando en un búfer lo que se va escuchando, para que el usuario lo escuche en el momento que así lo desee.

- **Televisión:** Medio de comunicación que combina los mensajes de imágenes fijas y en movimiento con voz, música, efectos sonoros y especiales.

- **Transmisión:** Emitir la señal de radio o televisión desde una estación emisora a otra que actúa como receptora de la misma.