Universidad de las Ciencias Informáticas

"Facultad 6"

Título: "Personalización del Subsistema de Monitorización de Señales Digitales."

Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas

Autor(es): Jorge Alejandro Estrada Rodriguez.

Tutor(es): Ing. Eduardo Cepero Utra.

Co-tutor: Ing. Joel Macías Roque.

"Junio de 2013"

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis que tiene por título: Personalización del Subsistema de Monitorización de Señales Digitales 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 _____.

<Jorge Alejandro Estrada Rodriguez> **<Eduardo Cepero Utra>**

_____ _____

Firma del Autor Firma del Tutor

AGRADECIMIENTOS:

Agradecerle a mi mamá, mi papá y mi hermana que me han ayudado incondicionalmente en todo momento.

A mi tía Ana y a Suleidy que se han encargado de mí en estos años de universidad.

A mis abuelos, tíos y primos que siempre han estado pendientes de mí, sin importar la distancia.

A los muchachos que vienen a mi lado desde el primer año, especialmente a Breissy.

Al piquete de la redbull por darme ánimos cuando me veían acobardado.

A todos los que de una forma u otra me ayudaron en todos estos años.

A todos, gracias.

DEDICATORIA

A Haydeé Galindo, donde quiera que esté,

por representar tanto para tantas personas.

A ti va dedicado mi fruto.

Resumen:

El presente trabajo de diploma expone la realización de un Subsistema de Monitorización de Señales Digitales autónomo, mediante el cual se pueden monitorizar señales de radio y televisión, además de registrar incidencias que ocurran durante la transmisión de los canales de audio y/o video. Se presentan en el cuerpo del presente documento diagramas y artefactos generados por la metodología de software: Proceso Unificado de Software (RUP, *por sus siglas en inglés*) que guió la construcción de la aplicación. Es válido resaltar que se desarrolló la aplicación con tecnologías y herramientas en su mayoría libres. El software desarrollado permite la monitorización de canales de radio y televisión, así como la medición de los niveles de audio de dichos canales, brinda la posibilidad de añadir, eliminar y reproducir canales televisivos, visualiza en tiempo real el video que se monitoriza y además hace un registro de las incidencias que ocurran durante el período de transmisión. Los registros consisten en plasmar fallas que pudieran ocurrir en la transmisión, por ejemplo que no llegue imagen. El sistema al recibir flujo de video registra características específicas del material que se monitoriza, tales como el *bitrate* y *framerate* que son datos del video. Además registra la profundidad de colores y borrosidad de la imagen que se está recibiendo, esto se logra capturando y analizando un *frame* del video que se monitoriza.

Palabras Claves:

Canales, Monitorización, Señales digitales, Transmisión

ÍNDICE:

ÍNDICE DE TABLAS:

ÍNDICE DE ILUSTRACIONES:

INTRODUCCIÓN.

El desarrollo de los ordenadores personales ha hecho posible un mayor acercamiento entre la informática y el hombre. En la actualidad es difícil prescindir de los equipos de cómputo que, aún sin ser esenciales, priman en la cotidianidad de la mayoría de las personas. Las compañías dedicadas al negocio de la informática se han dado a la tarea de informatizar distintas esferas de la actividad humana, atendiendo a sus gustos y preferencias. Una buena estrategia fue mezclarse aún más con las telecomunicaciones, apostando esta vez por la digitalización de las señales de radio y televisión, puesto que ambos medios de comunicación son muy utilizados por la sociedad.

La radio y la televisión constituyen un fuerte movimiento dedicado a la información y el entretenimiento, actualmente una gran cantidad de personas acceden a estos medios de difusión. Es por ello que buscando calidad en estos servicios se procede a digitalizar las señales. La digitalización es un proceso tecnológico donde datos, gráficos, sonidos e imágenes se transforman en bits, codificando y comprimiendo las señales originales. (1)

Los primeros intentos de transmitir imágenes a distancia se realizan mediante la electricidad y sistemas mecánicos. La electricidad hacía de medio de unión entre los puntos y servía para realizar la captación y recepción de la imagen. Luego se comenzó a transmitir de forma analógica, que consiste en emitir un conjunto de ondas hertzianas[1] continuas en el tiempo, estas ondas viajan por el espacio y una vez traducidas se convierten en audio, gráficos e imágenes. La forma más reciente de transmitir imágenes es digitalmente, es la transmisión de variables eléctricas con dos niveles bien diferenciados que se alternan en el tiempo. Más explícitamente, son una combinación de ceros y unos que son interpretadas y convertidas en gráficos, imágenes y sonido por un dispositivo electrónico. (2)

La televisión digital se emite vía redes informáticas, a través de una red de computadoras. Con la forma de transmisión *broadcast* la información se emite desde un nodo y llega a todos los nodos pertenecientes a una red. Existen otras formas tales como *multicast*, donde la información llega a múltiples nodos pero no a todos. Las formas *unicast* y *anycast* consisten en enviar la información a un solo nodo y al nodo más cercano, respectivamente. También se puede lograr la transmisión de señales digitales por ondas terrestres, que se hace a través de una red de repetidores terrestres. Las señales viajan en el espacio y se

[1] **ondas hertzianas:** ondas electromagnéticas, cuya frecuencia se fija convencionalmente por debajo de 3000 GHz, que se propagan por el espacio sin guía artificial. (41)

reciben con una antena convencional y un receptor digital que puede ser un dispositivo externo conectado al radio o televisor, o bien estos últimos dispongan de un receptor digital integrado.

Actualmente existen países que emiten señales analógicas por lo costosas que resultan ser las digitales, característica que constituye la principal desventaja de este tipo de señal. Las señales digitales permiten incrementar la oferta de canales ya que se pueden transmitir 4 canales digitales en el mismo espacio que se necesita para emitir un canal analógico. Este tipo de señal no presenta degradación, es decir, la imagen y el sonido se reciben más claros, eliminando el ruido ambiental, las interferencias y la sensación de llovizna. Además es posible incluirle una serie de servicios extra tales como brindar una guía electrónica de programación con toda la oferta de canales digitales y la elección de idioma y subtítulos. (3)

Aunque las señales digitales presenten ventajas con respecto a las analógicas, principalmente desde el punto de vista de calidad, no están exentas de tener fallas y errores en la transmisión. Debido a esto se hace necesario crear sistemas que se encarguen de monitorizar las señales digitales. Un sistema de monitorización de señales digitales consiste en un equipo capaz de recibir y reproducir las señales digitales de radio y televisión, al igual que un cliente más. La diferencia radica en que estos sistemas son capaces de analizar el audio y el video que están recibiendo con el objetivo de detectar fallas en las transmisiones, para poder dar lugar a corregir la transmisión. Los sistemas de monitorización de radio y televisión han venido evolucionando conjuntamente con los equipos de transmisión ya que los primeros deben presentar la misma tecnología que los transmisores para poder recepcionar la información emitida.

La “fiebre” de la televisión digital se ha expandido por el mundo entero y si bien una década atrás era aceptable que algunos países dieran prórroga al cambio debido a la escasez de recursos financieros o capital humano, ahora este cambio urge. Cuba, aún siendo un país subdesarrollado, ha comenzado a dar sus primeros pasos con el objetivo de implantar la tecnología digital, guiados de la mano de la República Popular China que ha ayudado con equipamiento y material humano. El gobierno cubano le ha otorgado gran prioridad a la transición, tanto es así que se presagia que a principios de la década de 2020 se logre el apagón analógico[2]. (4)

Aunque países como China ayuden mediante donaciones, la implantación de las señales digitales en Cuba implica un alto costo monetario; por lo que se hace necesario y de vital importancia explotar al máximo instituciones que pueden ayudar en gran medida en esta nueva empresa. Una fuerte candidata

[2] **Apagón analógico:** es cuando no se emiten señales analógicas en un territorio.

para ayudar en la implantación de la nueva tecnología es la Universidad de las Ciencias Informáticas (UCI), que fue fundada con el objetivo de desarrollar la industria del software en Cuba. La UCI cuenta con centros productivos, dentro de los que se encuentra Geoinformática y Señales Digitales (GEYSED). El departamento Señales Digitales, perteneciente a GEYSED, atiende directamente el proyecto Sistema de Transmisión de Canales Virtuales (STCV), los especialistas pertenecientes al proyecto mencionado, trabajan directamente en la automatización de los procesos de transmisión de señales digitales de radio y televisión.

El Sistema de Transmisión de Canales Virtuales (STCV) para el proceso de transmisión de medias[3] cuenta con el sistema SIAV Transmisiones, conformado por tres subsistemas: Programación, Transmisión y Monitorización. Los subsistemas mencionados son dependientes uno de otro mediante información que se almacena en una base de datos común. El Subsistema Monitorización es el encargado de visualizar en tiempo real, así como monitorizar los canales que emite el transmisor; en caso que no se esté emitiendo señal alguna por el Subsistema Transmisión queda inútil el Subsistema Monitorización ya que este no brinda la posibilidad de monitorizar emisiones de otros transmisores. Si el subsistema fuera autónomo, es decir que funcione sin la dependencia del Transmisión, representaría una ventaja comercial ya que se podría comercializar como un producto más del departamento Señales Digitales.

A partir de que se encontrara la deficiencia mencionada anteriormente, se hizo un estudio del Subsistema Monitorización que arrojó problemas que se basan principalmente en la trivialidad del proceso de análisis de las señales. Los sistemas de monitorización velan principalmente que no ocurran anomalías en el proceso que se monitoriza y en caso de ocurrir algo que no esté planificado, emiten una señal o registran dicha anomalía en un reporte. El Subsistema Monitorización no registra las anomalías que puedan surgir, como por ejemplo la ausencia de audio o video, por lo que no se puede saber con posterioridad si ocurrió una falla. La falta de esta opción hace que el usuario tenga que velar con mayor atención las emisiones, para así darse cuenta por él mismo de los errores que puedan ocurrir durante la transmisión, aumentando considerablemente la no detección de estos errores. El Subsistema Monitorización no brinda la posibilidad de monitorizar señales de radio, opción que de existir, sería un gancho comercial ya que se amplían las prestaciones de la aplicación.

Por lo anteriormente expuesto se plantea el siguiente problema de investigación: ¿Cómo garantizar la independencia funcional del Subsistema de Monitorización, así como detectar y registrar deficiencias en el

[3] **Media:** archivo de audio o video.

proceso de transmisión del proyecto Sistema de Transmisión de Canales Virtuales (STCV)? Para solucionarlo es necesario proponer el objetivo general: Desarrollar una personalización del Subsistema de Monitorización de Señales Digitales para el Sistema de Transmisión de Canales Virtuales. El problema está enmarcado en el objeto de estudio: Procesos de monitorización de señales digitales, específicamente en el campo de acción: Proceso de monitorización de señales digitales en el Sistema de Transmisión de Canales Virtuales (STCV).

Como idea a defender se plantea que con la Personalización del Subsistema de Monitorización de Señales Digitales se garantizará una independencia funcional del Subsistema Transmisión, así como la detección y registro de deficiencias en las transmisiones.

Las tareas que se definieron en aras de cumplir el objetivo general son:

1. Describir la evolución histórico-lógica de los procedimientos y técnicas de la transmisión y monitorización de señales digitales.
2. Caracterizar los sistemas existentes para la monitorización de señales digitales.
3. Determinar las tendencias y tecnologías actuales más apropiadas para el desarrollo del Subsistema de Monitorización de Señales Digitales.
4. Desarrollar artefactos y documentación correspondiente al proceso de desarrollo del software según la metodología seleccionada.
5. Implementar las funcionalidades requeridas para el subsistema de Monitorización de Señales Digitales.
6. Diseñar las pruebas para la validación del subsistema desarrollado.
7. Aplicar las pruebas diseñadas para validar el subsistema desarrollado.

Métodos teóricos:

Análisis histórico – lógico: se utilizó para estudiar el desarrollo histórico de soluciones semejantes, las tendencias y tecnologías actuales convenientes para el desarrollo del Subsistema de Monitorización de Señales Digitales y también para hacer un estudio abarcador de todo lo referente a la monitorización de señales digitales.

Analítico – Sintético: fue utilizado con el fin de examinar elementos bibliográficos y definiciones que abordarán el tema de las señales digitales, específicamente la monitorización de las mismas, con el objetivo de caracterizarlas desde el punto de vista de sus rasgos distintivos.

Modelación: se empleó para representar gráficamente la solución propuesta, por ejemplo en modelos tales como el de dominio, modelo de despliegue y modelo de implementación, necesarios durante el ciclo de vida del software. La modelación se hace con el objetivo de crear abstracciones que ayuden a comprender el funcionamiento del sistema.

Métodos empíricos:

Entrevista: la entrevista se realizó con el fin de adquirir conocimientos acerca de los procesos de transmisión y monitorización de señales digitales en el proyecto Sistema de Transmisión de Canales Virtuales. Se les realizó a los profesores que integran el proyecto anteriormente mencionado, tomando una población de cinco individuos de los cuales se toma uno como muestra para ser entrevistado, utilizando una técnica de muestreo no probabilístico, específicamente muestreo intencional. (Ver Anexo 1).

Observación: se emplea este método con el objetivo de ver de una forma práctica el funcionamiento de la solución SIAV Transmisiones y así conocer a fondo las principales deficiencias que presenta. Se observan principalmente los procesos de transmisión y monitorización de señales digitales.

CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA.

En este capítulo se expondrán una serie de conceptos asociados a la investigación que contribuirán en gran medida al entendimiento de la misma. Estos van especialmente dirigidos a personas que no dominen el tema que se trata en el presente trabajo de diploma. Se presentará una descripción del objeto de estudio y se profundizará en cuanto al dominio del problema, con el propósito de hacerle llegar más información al lector y así entienda el basamento teórico de la investigación. También se analizarán algunas soluciones similares al software que se desea realizar, con el fin de estudiar sus ventajas y desventajas. Posteriormente, a partir de ese estudio, se decidirá cuáles de las ventajas que presentan las soluciones serán añadidas a la solución que se desea.

1.1. Conceptos asociados al dominio del problema.

A continuación se definen y explican los significados de las palabras claves en el trabajo, con el objetivo principal de comprender el contenido de la investigación lo más sencillo posible. Se expondrán varias definiciones escritas por especialistas o instituciones especializadas en el tema en cuestión para contar con sustento científico. En caso de quedar ambiguo algún concepto, se elaborará uno propio en aras de adaptarlo a los intereses específicos de la investigación en curso.

Transmisión: se entiende por transmisión como la acción y efecto de transmitir, que dicho propiamente de una emisora de radio o de televisión puede ser difundir noticias, programas de música y espectáculos. (5)

Es admitido además como el envío o intercambio de información en formato analógico o digital. Se dice que la televisión y la radio transmiten sus programas, ya que estos llegan al público a través de antenas, cables y otros dispositivos. (5)

Se puede concluir que transmisión es enviar o transmitir audio y/o video a través de antenas, cables u otros dispositivos, ya sea en formato digital o analógico.

Monitorización: según el Diccionario de la Real Academia Española (DRAE) se refiere a acción y efecto de monitorizar.

El diccionario Webster define la monitorización como: para ver, comprobar y observar con un propósito especial. Y el diccionario de Oxford lo define como: observar, supervisar o mantener en examen, a medida o prueba a intervalos, especialmente con el propósito de la regulación o control, o para controlar o regular

la calidad técnica de algo. Cualquier instrumento o dispositivo para el seguimiento de algún proceso o cantidad. (6)

Se puede concluir que monitorización es observar de modo continuo algún evento. Por lo que refiriéndose específicamente a las señales de radio y televisión, es observar de modo continuo o a intervalos de tiempo la transmisión de señales radiales o televisivas con el fin de conocer si se está realizando de forma correcta.

Señales: en Física es una manifestación ya sea visual o auditiva, emitida por una fuente de energía, o recibida por un sistema que la capta. (5)

Una señal es una onda electromagnética, contenedora de datos, emitida por un transmisor. Esta onda es recibida por un sistema electrónico capaz de entender dichos datos.

Señales analógicas: son variables eléctricas que evolucionan en el tiempo en forma análoga a alguna variable física. Varían en forma continua entre un límite inferior y un límite superior. (2)

Señales digitales: las señales digitales son aquellas que están representadas por funciones que pueden tomar un cierto número finito de valores en cualquier intervalo de tiempo. Son variables eléctricas con dos niveles bien diferenciados que se alternan en el tiempo transmitiendo información según un código previamente acordado. Cada nivel eléctrico representa uno de dos símbolos: 0 ó 1, V o F. (2)

Canal: vía estrecha por donde fluye una corriente hasta llegar a su destino. (5)

Un canal se define como la vía por la cual se envían las señales radiales o televisivas al receptor, dichas señales pueden ser digitales o analógicas.

1.2. Objeto de estudio.

Se le ha definido como objeto de estudio a la presente investigación los "procesos de monitorización de señales digitales".

1.2.1. Descripción del objeto de estudio.

En cualquier campo de la ciencia es de suma importancia revisar de forma continua el comportamiento de algún evento, a esta acción se le conoce como monitorizar. Antiguamente una persona con conocimientos básicos de la materia a monitorizar era la encargada de observar de forma ininterrumpida o bien continua

el fenómeno o actividad que se necesitara velar. En el caso que ocurriera alguna anomalía dicha persona tomaba nota con el fin de hacer un análisis posterior.

Con el avance de los equipos de cómputo se han confeccionado sistemas que de forma automática hacen observaciones, registran las incidencias ocurridas y en caso de ser graves generan una alarma. La monitorización se utiliza en la medicina, la seguridad de inmuebles, tiendas y en las redes informáticas, por solo mencionar algunos ejemplos. Cada una con especificidades distintas pero con un mismo objetivo, velar por el correcto funcionamiento de un evento.

La transmisión de señales de radio y televisión no queda exenta de la utilización de la monitorización, proceso vital para garantizar la calidad con que llegan estas señales a su destino. Con la digitalización de las señales de radio y televisión cambia un tanto la forma en que se transmite por lo que es necesario cambiar la manera de recepcionar y con ello se transforma el proceso de monitorización de señales.

Actualmente son empleadas nuevas tendencias en las señales de radio y televisión que potencian en gran medida un aumento en la calidad con que son transmitidas. La tecnología *streaming*[4] la cual utiliza un almacenamiento temporal y permite descargar audio y/o video en lo que se va reproduciendo, gracias a esta tecnología se pueden transmitir simultáneamente múltiples canales con el uso de un solo transmisor. Debido a ello las aplicaciones informáticas desarrolladas con el objetivo de monitorizar señales digitales deben ser capaces de velar una mayor cantidad de canales.

Un software destinado a la monitorización de señales digitales debe permitir la visualización en tiempo real de varios canales simultáneamente, es decir, que exista un seguimiento de los videos mientras estén en su estado de reproducción. De esta forma se garantiza que se observe el video para así detectar deficiencias que puedan ocurrir durante su transmisión. Se debe tener un control estricto además de una información visual del audio de la media que se está monitorizando. Además es importante identificar, notificar y registrar la existencia de alguna deficiencia en la transmisión.

---

[4] **streaming:** tecnología se basa en un sistema que permite acceder a un archivo situado en un servidor de Internet sin necesidad de descargarlo antes para reproducir su contenido. El archivo se descarga al ordenador, pero en forma de flujo de datos, y sólo permanece de forma temporal. (44)

1.3. Descripción actual del dominio del problema.

La UCI está conformada por siete facultades, las cuales a su vez cuentan con centros productivos que trabajan en aras de informatizar distintas esferas sociales. Tal es el caso del centro GEYSED, el cual radica en la Facultad 6 de dicha Universidad. Este centro se divide en dos departamentos: Geoinformática y Señales Digitales, este último contiene el proyecto Sistema de Transmisión de Canales Virtuales (STCV).

El STCV cuenta con una solución capaz de planificar, transmitir y monitorizar canales de televisión, la misma está compuesta por tres subsistemas, donde el Subsistema Planificación permite gestionar la parrilla de programación[5], de espacios televisivos y radiales, así como su exportación a formatos de lectura pública. El Subsistema Transmisión se guía por la planificación que le programa el Subsistema Planificación, el Transmisión gestiona los canales para su posterior transmisión, así como la administración y visualización en tiempo real de los mismos. Aunque este último visualiza en tiempo real los canales que se transmiten, no profundiza en el proceso de monitorización. El Subsistema Monitorización es el encargado de la visualización en tiempo real del video en emisión y comprueba los niveles de audio de los materiales que se monitorizan.

Los subsistemas aunque se vean por separado trabajan acoplados, ya que adquieren información de una base de datos común. Solo si fue planificada la hora de los canales, es que estos son transmitidos y monitorizados. Más detalladamente, el planificador programa la hora, medias y canales a transmitir, luego toda la información referente a la planificación realizada es guardada en una base de datos, la cual es consultada por el Subsistema Transmisión que accede a la información y emite a la hora, la media y por el canal que el planificador programó. El Subsistema Monitorización, de acuerdo a la información almacenada en la base de datos anteriormente mencionada, visualiza en tiempo real y monitoriza las señales emitidas por cada uno de los canales definidos.

1.4. Análisis de soluciones existentes.

Con el objetivo de indagar en cuanto a lo que se quiere lograr se analizarán algunas de las soluciones informáticas que existen para la monitorización de señales digitales. Así se podrá determinar si se pueden adaptar a las necesidades reales que urgen, con el fin de resolver la problemática planteada. Las ventajas

[5] **parrilla de programación:** es un término utilizado en el ámbito radial y televisivo para hacer alusión a la lista de programas que tiene un canal.

que presenten dichas soluciones serán analizadas y en caso de ser factibles se tendrán en cuenta para el desarrollo del software.

- ✓ **TV LOG:** es un sistema de registro de audio y video, para grabar, monitorear y supervisar varias señales de televisión simultáneamente. Está desarrollado bajo la plataforma LINUX por ser absolutamente estable para operaciones de grabación continua de 24 horas sin detención o corte alguno. (7)
- ✓ **SIAV Transmisiones:** es una suite de aplicaciones para la automatización de procesos de gestión, procesamiento, publicación, transmisión y monitorización de contenidos audiovisuales. Compuesta por 3 subsistemas con despliegue distribuido o independiente. Presenta un Subsistema de Monitorización que visualiza y monitoriza en tiempo real el video en emisión y comprueba los niveles de audio.

- ✓ **Videoma Broadcast Monitor (VBM):** es una plataforma estable de monitorización y gestión de contenido procedente de Televisión Digital Terrestre (TDT), Satélite, TV Analógica y Radio, que combina los procesos de programación manual o automatizada, grabación, extracción de fotogramas, catalogación, análisis de la información, archivo y recuperación del contenido. Es una aplicación web, modular y escalable que permite la grabación, monitorización y visualización de múltiples canales. El acceso es vía LAN / INTERNET y presenta como requisitos del sistema: Windows 2008 Server o Windows 7 e Internet Explorer o Firefox. (8)

Características	TV Log	SIAV Transmisiones	VBM
Registro de incidencias.	**NO**	**NO**	**NO**
Software Libre.	**SI**	**SI**	**NO**
Monitorización de radio y TV, conjuntamente.	**NO**	**NO**	**SI**
Autonomía.	**SI**	**NO**	**SI**

Tabla 1: Análisis de soluciones similares.

Se definieron las características que se necesitan para la realización de la solución propuesta, las cuales se plantearon en una tabla (Ver Tabla 1) en la que se especifica si los sistemas analizados presentan o no estas características. Autonomía que es la más ambigua o menos descriptiva de las características planteadas, significa que el software pueda monitorizar señales de radio o televisión de cualquier transmisor; es decir, que su funcionamiento no dependa de un transmisor específico.

Las soluciones planteadas anteriormente realizan los procesos básicos que se necesitan para desarrollar un software de monitorización de señales digitales, tales como monitorizar y visualizar varios canales simultáneamente. Sin embargo las mismas presentan inconvenientes, por ejemplo VBM es de software privativo, lo que va en contra de la política de software libre llevada a cabo por el país, además no registra incidencias que ocurran en los canales que se monitorizan. TV LOG por su parte no brinda la posibilidad de monitorizar transmisiones de radio y tampoco permite registrar reportes de incidencias. El Subsistema Monitorización que compone el sistema SIAV Transmisiones amén de no registrar las incidencias que ocurren en los canales que se monitorizan, no permite monitorizar señales de radio y solamente monitoriza las señales emitidas por el Subsistema Transmisión de dicho sistema (SIAV Transmisiones), en caso que no funcione el Transmisión no se puede monitorizar ninguna señal.

Después de analizar individualmente cada solución, se puede concluir que las herramientas no satisfacen los problemas planteados en la investigación, por lo que se decide desarrollar una personalización del Subsistema de Monitorización de Señales Digitales. No obstante se tomarán algunas de las características que presentan estas soluciones, por ejemplo la visualización y monitorización simultánea de múltiples canales, característica que es común en las tres soluciones analizadas. También se tendrá en cuenta la comprobación de los niveles de audio que permite el SIAV Transmisiones, así como la extracción de fotogramas que brinda VBM.

1.5. Conclusiones parciales.

El estudio de los principales conceptos asociados al dominio del problema, permitió profundizar los conocimientos acerca de los elementos teóricos que sustentan el trabajo de diploma. Se caracterizaron y analizaron soluciones existentes similares, lo que permitió enriquecer el software que se desea, ya que se le añadirán algunas de las características que presentan las soluciones analizadas. Se determinó desarrollar una personalización del Subsistema de Monitorización de Señales Digitales, con el cual se garantizará la solución del problema planteado en la investigación.

CAPÍTULO II. HERRAMIENTAS Y TECNOLOGÍAS.

La evolución de la informática se basa principalmente en el perfeccionamiento de las herramientas y tecnologías para el desarrollo de software, que constituyen en la actualidad, instrumentos fundamentales para garantizar la calidad de los procesos de desarrollo. En el capítulo en curso se estudian y definen las herramientas y tecnologías que se utilizarán en la investigación, teniendo en cuenta las más factibles para la construcción de aplicaciones vinculadas a la monitorización de señales digitales de radio y televisión. Se definirá la metodología de desarrollo, el lenguaje de modelado y la herramienta CASE (del inglés, *Computer Aided Software Engineering*) que permiten modelar el desarrollo de la aplicación. Además será elegido el lenguaje de programación y el Entorno Integrado de Desarrollo, los cuales serán utilizados para la implementación de la solución. Se elegirá un marco de trabajo que permita el uso de librerías que faciliten la implementación de la solución propuesta. El propósito del estudio que se brinda a continuación está basado en hacer llegar el porqué son factibles para el subsistema las tecnologías que se elijan.

2.1. Metodologías de desarrollo de Software.

Una metodología de desarrollo de software conforma un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos de software. Guían el desarrollo de una aplicación y dejan documentación referente a la misma, lo cual propicia un mejor entendimiento a terceros. (9)

Existen disímiles metodologías de software, las cuales se clasifican en dos grandes grupos:

Metodologías ágiles: es un método basado en el desarrollo iterativo e incremental, donde los requerimientos y soluciones evolucionan mediante la colaboración de grupos autos dirigidos y multidisciplinarios. (10)

Metodologías robustas: están guiadas por una fuerte planificación. Centran su atención en llevar una documentación exhaustiva de todo el proceso de desarrollo y en cumplir con un plan de proyecto, definido en la fase inicial del mismo. (10)

2.1.1. Proceso Unificado de Desarrollo (RUP).

RUP, por sus siglas en inglés, es una metodología que guía el desarrollo de software. Constituye una forma disciplinada de asignar tareas y responsabilidades en un proyecto de desarrollo, definiendo quién

hace qué, cómo y cuándo. Tiene como objetivo fundamental asegurar la producción de software de calidad dentro de plazos y presupuestos predecibles. (11)

Esta metodología además incorpora el concepto de "mejores prácticas" de la ingeniería de software, definido por cinco características fundamentales:

Dirigido por casos de uso: el desarrollo está dirigido a satisfacer las necesidades de los usuarios del sistema expresadas en casos de uso.

Centrado en la arquitectura: el desarrollo se centra en una arquitectura bien definida, con relaciones claras entre sus distintos componentes.

Iterativo: el problema y la solución se organizan en pequeñas piezas, de manera que cada iteración se dirige específicamente al desarrollo de un conjunto de ellas.

Incremental: cada iteración se construye sobre la base creada por las iteraciones anteriores, agregándole capacidades al sistema.

Controlado: el proceso se planifica y en cada momento está claro lo que debe hacerse. (11)

La metodología RUP, se escoge para guiar la construcción de la solución propuesta ya que es muy utilizada para la documentación y realización de software orientados a objetos. Es una metodología robusta y los requisitos, del software propuesto, son bien conocidos, así como la tecnología a utilizar por lo que estos no presentarán cambios de último momento. Genera una amplia documentación, lo cual es primordial para poder darle mantenimiento y soporte al software, ya que esto último lo harán personas que no estuvieron presentes en la construcción del mismo porque el autor de este trabajo, al terminarlo, abandona el centro de estudios. También se decide RUP como metodología de desarrollo porque se estudia en las asignaturas Ingeniería de Software I y II en la carrera Ingeniero en Ciencias Informáticas, por lo que se tiene un mayor conocimiento acerca de esta metodología.

2.2. Lenguaje Unificado de Modelado (UML v2.0).

UML es ante todo un lenguaje y un lenguaje proporciona un vocabulario y unas reglas para permitir una comunicación. En este caso, este lenguaje se centra en la representación gráfica de un sistema, indica cómo crear y leer los modelos, pero no dice cómo. UML permite entender, diseñar, configurar, mantener y controlar la información generada en el desarrollo de software. El lenguaje de modelado pretende unificar

la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar. (12)

El uso de este lenguaje de modelado permitirá:

- ✓ **Visualizar:** permite expresar de una manera gráfica un sistema de forma que este se pueda entender.
- ✓ **Especificar:** detalla cuáles son las características que va a tener cierto sistema antes de ser construido.
- ✓ **Construir:** una vez que estén especificados los modelos se pueden construir los sistemas diseñados.
- ✓ **Documentar:** se pueden utilizar como documentación los elementos gráficos desarrollados y estos pueden servir para una futura versión. (12)

Se elige como lenguaje de modelado UML porque permite crear diseños de una forma convencional, fácil de comprender para hacérselas llegar a otros. Es un lenguaje que permite modelar proyectos y que estos puedan ser comprensibles tanto para los desarrolladores como los clientes, ya que permite analizarlo y explicarlo de una forma sencilla. Puede aplicarse en una gran variedad de formas para dar soporte a metodologías de desarrollo de software, por ejemplo RUP, que es utilizada para guiar la construcción de la solución propuesta en el presente trabajo de diploma.

2.3. Herramientas CASE.

Las herramientas CASE representan una forma que permite Modelar los Procesos de Negocios de las empresas y desarrollar los Sistemas de Información Gerenciales. Esencialmente es una herramienta que ayuda al ingeniero de software a desarrollar y mantener software. (13)

2.3.1. Visual Paradigmv8.0.

Visual Paradigm es una herramienta CASE que soporta el ciclo de vida completo del proceso de desarrollo de software a través de la representación de todo tipo de diagramas. Permite la representación de los modelos en todas las dimensiones que UML abarca. Ayuda a una rápida construcción de aplicaciones de calidad y a un menor costo. Permite representar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. (14)

Se escoge la herramienta anteriormente mencionada ya que propicia un conjunto de ayudas para el desarrollo de programas informáticos, transitando por el diseño hasta la generación de documentación y código fuente. Se tiene más experiencia de trabajo con Visual Paradigm que con otras herramientas de modelado. La herramienta utiliza como lenguaje de modelado UML y este fue escogido para modelar la solución propuesta. Además soporta el ciclo de vida de la metodología de software escogida.

2.4. Entorno de Desarrollo Integrado.

Un entorno de desarrollo integrado (IDE, *por sus siglas en inglés*), es una aplicación de software que proporciona servicios integrales a los programadores de computadoras para el desarrollo de software. El mismo puede estar pensado para su utilización con un único lenguaje de programación o bien puede dar cabida a varios de estos. Un IDE normalmente se compone de:

- ✓ Un editor de código fuente.
- ✓ Un compilador y/o un intérprete.
- ✓ Automatización de generación de herramientas.
- ✓ Un depurador. (15)

2.4.1. QtCreatorv2.4.1.

QT Creator es un IDE de desarrollo que permite crear aplicaciones de escritorio. Puede ser usado para desarrollar interfaces gráficas de usuario y también para el desarrollo de programas sin interfaz como herramientas de la consola y servidores. Es distribuido bajo los términos de GNU *Lesser General Public License* (LGPL[6]). Tiene entre sus principales características un avanzado editor de código, provee soporte para edición de C++ y completamiento de código. (16)

Es un IDE muy cómodo para trabajar ya que posee una ayuda documentada que resulta ser un importante apoyo. Es de software libre. En el STCV se utiliza este IDE, por lo que hay un buen dominio del mismo, reduciendo así el tiempo en la implementación. Además permite el uso del *framework* Qt el cual presenta características que serán utilizadas en la implementación, dichas características serán explicadas más adelante.

---

[6] **GNU Lesser General Public License:** es una licencia de software creada por la Free Software Foundation. Los contratos de licencia de la mayor parte del software están diseñados para jugar con su libertad de compartir y modificar dicho software.

2.5. Marco de trabajo.

Un Marco de trabajo o *framework*, en el desarrollo de software, es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto. (17)

2.5.1. Qtv4.8.0.

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, tiene métodos para acceder a bases de datos mediante SQL, es uno de los más utilizados en la actualidad por lo que presenta una amplia bibliografía para su estudio. Contiene además módulos para el trabajo con protocolos de red. Es producido por la división de software Qt de Nokia, desarrollado bajo licencia LGPL y liberado como software libre y de código abierto. (18)

Se decide hacer uso del marco de trabajo Qt pues permite la utilización del paradigma de programación orientada a objetos. Cuenta con una serie de herramientas y clases que permiten el fácil acceso a bibliotecas para el procesamiento de imágenes, audio y video. Esto último será de gran utilidad para el desarrollo del subsistema ya que con ello se garantiza la recepción y reproducción de medias, requisito básico para monitorizar señales digitales de radio y televisión.

2.6. Lenguaje de Programación.

Un lenguaje de programación es aquel elemento dentro de la informática que permite crear programas mediante un conjunto de instrucciones, operadores y reglas de sintaxis; que pone a disposición del programador para que este pueda comunicarse con los dispositivos hardware y software existentes. (19)

Los lenguajes de programación se pueden clasificar en dos categorías, lenguajes de bajo nivel y lenguajes de alto nivel. Un lenguaje de bajo nivel se ocupa más de la interacción directa por hardware, por lo que es más adecuado para programas como el dispositivo de código de los drivers que necesita realmente el acceso al hardware. Desde un lenguaje de bajo nivel está sujeta a todos los matices del hardware que está accediendo, sin embargo, un programa escrito en un lenguaje de bajo nivel es generalmente difícil de portar a otras plataformas. Estas lenguas son casi siempre compiladas. (20)

Un lenguaje de alto nivel se centra más en los conceptos que son fáciles de entender para la mente humana, tales como objetos o funciones matemáticas. Un lenguaje de alto nivel por lo general es más fácil de comprender que un lenguaje de bajo nivel, y por lo general toma menos tiempo para desarrollar un programa en un lenguaje de alto nivel que en un lenguaje de bajo nivel. No es, sin embargo, imposible de mezclar alto y bajo nivel de funcionalidad en un idioma. (20)

2.6.1. C++.

C++ es un lenguaje que permite programar con el estilo orientado a objeto. Además, también se puede emplear mediante programación basada en eventos para crear programas que usen interfaz gráfica de usuario.

Las principales ventajas que presenta el lenguaje C++ son:

- ✓ **Versatilidad:** es un lenguaje de propósito general, por lo que se puede emplear para resolver varios tipos de problemas.
- ✓ **Portabilidad:** el lenguaje está estandarizado y un mismo código fuente se puede compilar en diversas plataformas.
- ✓ **Eficiencia:** es uno de los lenguajes más rápidos en cuanto a ejecución.
- ✓ **Herramientas:** existe una gran cantidad de compiladores, depuradores y bibliotecas. (21)

Se selecciona C++ como lenguaje de programación para el desarrollo del subsistema ya que es compatible con disímiles plataformas incluida GNU/Linux. Es el lenguaje nativo del Entorno de Desarrollo Integrado "Qt Creator" el cual fue elegido para la confección de la aplicación.

2.7. Sistema Gestor de Base de Datos (SGBD).

Un SGBD es un programa que facilita una serie de herramientas para manejar bases de datos (BD) y obtener resultados de ellas. Además de almacenar la información, se le pueden hacer preguntas sobre esos datos, obtener listados, generar pequeños programas de mantenimiento de la BD, o ser utilizado como servidor de datos para programas más complejos realizados en cualquier lenguaje de programación. (22)

2.7.1. PostgreSQL v9.1.

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional. Utiliza un modelo cliente/servidor y usa multiprocesos para garantizar la estabilidad del sistema. Un fallo en uno de los

procesos no afectará el resto y el sistema continuará funcionando. Estabilidad, potencia, robustez, facilidad de administración e implementación de estándares han sido las características que más se han tenido en cuenta durante su desarrollo. PostgreSQL funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema. (23)

Para el desarrollo de este trabajo se escoge PostgreSQL por ser un gestor de base de datos de software libre, además de su gran robustez y el nivel de escalabilidad que presenta. Soporta grandes volúmenes de datos, limitados solo por el sistema de almacenamiento donde se aloje la base de datos. Características muy importantes para la realización del Subsistema de Monitorización de Señales Digitales.

2.8. Bibliotecas.

- ✓ ***GStreamer.***

GStreamer es una biblioteca para la construcción de gráficos de componentes de manipulación de medias. Las aplicaciones que soporta van desde la reproducción del simple formato Ogg/Vorbis a audio complejo (mezcla) y el procesamiento de vídeo. La biblioteca *gstreamer* se utiliza para el trabajo con audio en el Subsistema de Monitorización de Señales Digitales, específicamente en la visualización de los niveles de audio.

- ✓ ***OpenCV.***

OpenCV es una biblioteca informática de código abierto la cual es gratuita para uso comercial. Se centra principalmente hacia el procesamiento imagen en tiempo real. Incluye tanto la interfaz de C tradicionales, así como una nueva interfaz C++. (24)

Se utiliza esta biblioteca por las facilidades que brinda para el procesamiento de imágenes, útil para el análisis de un *frame*[7] de un video para determinar la borrosidad del video. También por la posibilidad que brinda de trabajar con C++ que es el lenguaje de programación escogido para la realización del subsistema. Una razón de peso por la que se escoge esta biblioteca es por ser de software libre, lo cual no determina pero influye en gran medida en la elección.

- ✓ ***FFmpeg.***

[7] **frame:** se denomina *frame* en inglés a una imagen específica dentro de una sucesión que componen una animación. (43)

FFmpeg es una colección de software libre que puede grabar, convertir y hacer *streaming* de audio y video. Está desarrollado en GNU/Linux, pero puede ser compilado en la mayoría de los sistemas operativos, incluyendo Microsoft Windows. (25)

Esta biblioteca es usada con el fin de capturar un *frame* para su posterior análisis con *OpenCV* como se explicó anteriormente. Además brinda información acerca del video que se está analizando tales como el *bitrate*[8], *framerate*[9] y dimensiones, los cuales son registrados para analizarlos con posterioridad.

2.9. Conclusiones parciales.

En el capítulo se plasma un resumen, basado en los resultados del estudio, análisis y caracterización de las tendencias y tecnologías actuales que se escogieron para el desarrollo del Subsistema de Monitorización de Señales Digitales. Fue de vital importancia este proceso pues quedaron definidas las herramientas y tecnologías, las cuales garantizan el cumplimiento del objetivo general de este trabajo de diploma, con una robusta calidad. Finalmente, a modo de recapitulación, queda definida para guiar la construcción del subsistema la metodología de desarrollo Proceso Unificado de Desarrollo (RUP), UML como lenguaje de modelado y la herramienta CASE Visual Paradigm. Como IDE queda establecido Qt Creator y atendiendo a esto lenguaje de programación escogido es C++, ya que este es su lenguaje nativo. Quedó definido el *framework* Qt ya que presenta bibliotecas que permiten el trabajo con imágenes, audio y video. Como sistema gestor de base de datos se definió PostgreSQL.

[8] **bitrate:** es la cantidad de bits que tiene un archivo de audio o video en un segundo.
[9] **framerate:** la cantidad de *frames* que tiene un segundo de una animación específica.

CAPITULO III. PRESENTACIÓN DE LA SOLUCIÓN PROPUESTA.

Con el objetivo de explicar los conceptos asociados al entorno del subsistema se representará el modelo de dominio. En este capítulo se especifican los requisitos funcionales y no funcionales que debe presentar el Subsistema Monitorización de Señales Digitales, los cuales serán agrupados en casos de uso del sistema. Los casos de uso luego de ser definidos serán descritos para detallar las acciones que debe realizar el software propuesto.

3.1. Modelo de dominio.

El Modelo de Dominio se utiliza como base para analizar los requisitos del usuario en el flujo de la ingeniería de software. Es una actividad del desarrollo orientado a objetos que permite establecer cuáles son los objetos que conforman el dominio de la aplicación. Además presenta la relación de los conceptos más importantes. (26)

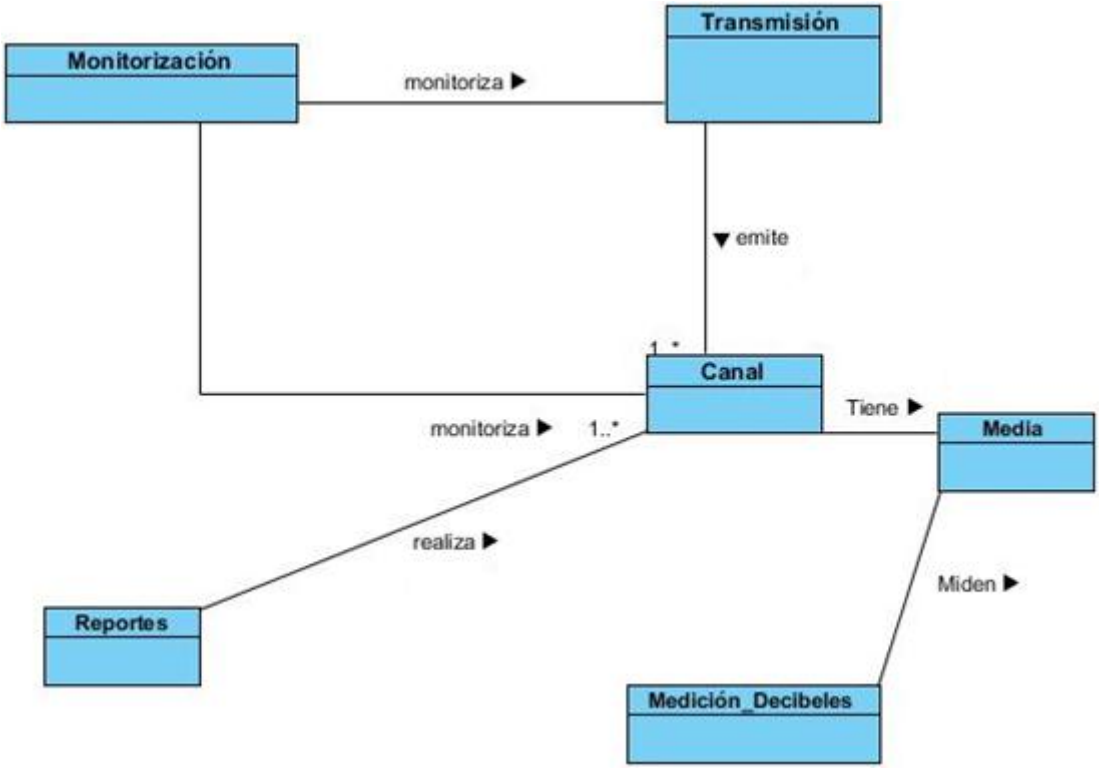


Ilustración 1. Modelo de dominio.

Clase Transmisión: es quien emite los canales de audio y/o video.

Clase Monitorización: es quien recibe las transmisiones con el fin de monitorizarlas.

Clase Canal: es la vía por la que se emiten las señales digitales.

Clase Media: son los archivos de audio y/o video que se transmiten y monitorizan.

Clase Reportes: son informes de fallas que sucedan durante la transmisión.

Clase Medición_Decibeles: mide los niveles de audio de las medias transmitidas a través de cada uno de los canales.

3.2. Requisitos Funcionales.

Uno de los desafíos principales de los ingenieros de software lo constituye tener la seguridad de haber especificado un sistema que recoge las necesidades del cliente y satisfaga sus expectativas. Actualmente la solución a este desafío no tiene una vía definida, por lo que garantizar un sólido proceso de ingeniería de requisitos viene siendo la mejor solución que se dispone actualmente.

La ingeniería de requisitos facilita el mecanismo apropiado para comprender lo que quiere el cliente, analizando necesidades, confirmando su viabilidad, negociando una solución razonable, especificando la solución sin ambigüedad, validando la especificación y gestionando los requisitos para que se transformen en un sistema operacional. (26)

Los requisitos funcionales son las acciones que debe cumplir el sistema que se implementará, por lo que requieren de un trabajo serio y preciso para definirlos. A continuación se presentan los requisitos funcionales del Subsistema de Monitorización de Señales Digitales, acompañados de una breve descripción.

RF_1: Autenticar usuario:

El subsistema permite la autenticación del usuario en el sistema con los permisos correspondientes.

RF_2: Visualizar programación televisiva del día.

El subsistema debe permitir seleccionar uno de los canales de televisión en transmisión, se ejecuta mostrando la planificación que tienen dichos canales.

RF_3: Visualizar media seleccionada.

El subsistema debe permitir seleccionar uno de los canales de televisión que se están transmitiendo y este se visualiza a mayor resolución.

RF_4: Mostrar datos del canal de televisión.

El subsistema debe permitir seleccionar uno de los canales que están en transmisión y ver los datos del mismo.

RF_5: Actualizar transmisión televisiva.

El subsistema debe permitir actualizar el flujo de los canales de televisión así como la programación del día y los datos asociados a los canales.

RF_6: Insertar datos de configuración para establecer conexión a la base de datos.

El subsistema debe permitir insertar datos para conectarse a la base de datos correcta.

RF_7: Editar datos de configuración de conexión a la base de datos.

El subsistema debe permitir a los usuarios modificar los parámetros establecidos en la configuración para la comunicación con la base de datos.

RF_8: Guardar datos de configuración de conexión a la base de datos.

El módulo debe permitir a los usuarios guardar los datos de la configuración realizada.

RF_9: Monitorizar canal de audio.

Este requisito permite la monitorización de los canales de audio que están en transmisión.

RF_10: Mostrar niveles de los canales de audio.

El subsistema permite mostrar los niveles de la señal de audio correspondiente al canal de audio seleccionado.

RF_11: Listar canales de audio en transmisión.

El subsistema permite mostrar la lista de canales de audio en transmisión.

RF_12: Mostrar datos del canal de audio.

El subsistema permite mostrar los datos de un canal de audio seleccionado.

RF_13: Mostrar programación del día del canal de audio.

El subsistema permite visualizar la programación para el día, de un canal de audio seleccionado.

RF_14: Notificar error.

El subsistema permite notificar los errores que se produzcan durante la transmisión.

RF_15: Actualizar transmisión radial.

El subsistema permite actualizar la lista de canales de audio que inician o concluyen la transmisión.

RF_16: Generar alarma visual de error en la monitorización.

El subsistema debe permitir generar una alarma visual cuando se produzca algún fallo en la transmisión de los canales.

RF_17: Registrar reporte de incidencias.

El subsistema debe registrar un reporte de incidencias en un archivo.

RF_18: Adicionar canal independiente.

El subsistema debe permitir que se le añadan canales que no estén definidos en la programación.

RF_19: Reproducir canal independiente.

El subsistema debe permitir visualizar el/los canal/canales que desee el usuario.

RF_20: Eliminar canal independiente

El subsistema debe permitir eliminar un canal de los que fueron adicionados con antelación.

RF_21: Cerrar sesión

El subsistema debe permitir cerrar la sesión en la que se encuentra el usuario.

3.3. Requisitos No Funcionales.

Los requerimientos no funcionales hacen relación a las características del sistema que aplican de manera general como un todo, más que a rasgos particulares del mismo. Estos requerimientos son adicionales a los requerimientos funcionales que debe cumplir el sistema, y corresponden a aspectos tales como la disponibilidad, mantenibilidad, flexibilidad, seguridad y facilidad de uso. (27)

Los requisitos no funcionales son primordiales para el éxito de los sistemas. Son aplicados más bien al diseño de la aplicación y establecen restricciones en el producto que se está desarrollando. Atendiendo a que los errores que se deben a una mala aplicación de los requisitos no funcionales son difíciles y

costosos de resolver, se plantean los pertenecientes al Subsistema de Monitorización de Señales Digitales.

Usabilidad:

- ✓ El sistema debe poder ser usado por cualquier persona que tenga conocimientos básicos de computación. Todas sus funcionalidades deben ser consistentes y bien organizadas. Debe tener una interfaz gráfica uniforme, incluyendo pantallas, menú y opciones.

Eficiencia:

- ✓ El tiempo de respuesta promedio para la reproducción debe ser de 3 segundos. El tiempo de respuesta máximo debe ser de 6 segundos.

Seguridad:

- ✓ El sistema debe exigir a los usuarios autenticarse antes de poder usar sus funcionalidades. Podrá ser utilizado sólo por usuarios autorizados por el administrador.

Software:

- ✓ Sistema Operativo: Ubuntu 10.04 o superior.
- ✓ Sistema Gestor de Base de Datos: PostgreSQL 9.1.

Hardware:

- ✓ Procesador: Dual Core 3.0 GHz
- ✓ Memoria RAM: 2 GB.
- ✓ Tarjeta de Red: Ethernet a 100 Mbps.
- ✓ Capacidad de almacenamiento en disco duro: 40 GB.

Interfaz:

- ✓ La aplicación debe tener una interfaz de usuario sencilla, amigable y fácil de entender. La información se presentará de forma clara, con una estructura bien organizada.

3.4. Descripción del sistema propuesto.

Después de realizar el modelo de dominio e identificados los requisitos funcionales y no funcionales es posible modelar y proponer el sistema que se desea desarrollar. Para ello es necesario hacer una

descripción previa de los actores del sistema, representando de forma secuencial las acciones más significativas que realiza un actor en el intercambio con el sistema.

3.4.1. Descripción de los actores del sistema.

Un actor del sistema es aquel que interactúa con el sistema que se pretende desarrollar, puede ser una persona u otro sistema. A continuación se muestran los actores del Subsistema de Monitorización de Señales Digitales:

Actor	***Objetivo***
Operador de Monitorización	Es el encargado de monitorizar las señales que se están transmitiendo.
Usuario	Persona que interactúa con la interfaz gráfica de la aplicación para autenticarse en el sistema.

Tabla 2. Descripción de los actores del sistema.

3.4.2. Diagrama de casos de uso del sistema.

Los casos de uso del sistema representan una funcionalidad del sistema en cuestión y especifican las acciones que tienen lugar durante la interacción actor-sistema. Se forman basándose en los requisitos funcionales. Se representa a continuación el diagrama de casos de uso del sistema del Subsistema de Monitorización de Señales Digitales (Ver Ilustración 2).

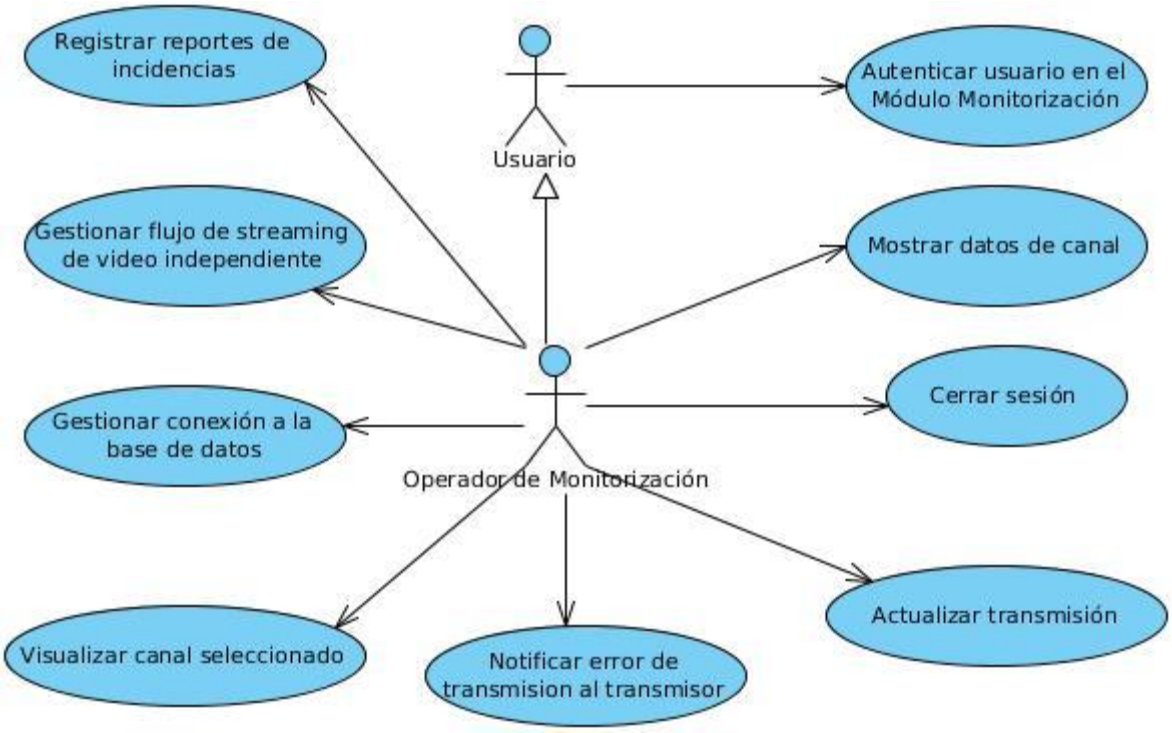


Ilustración 2. Diagrama de Casos de Uso del Sistema.

3.4.3. Descripción de los casos de uso.

En este epígrafe se hace una descripción detallada de lo que realiza el actor y el sistema en el caso de uso Gestionar flujo de *streaming* de video independiente. Las demás descripciones se pueden ver en el anexo 2.

Objetivo	Adicionar, eliminar y/o reproducir canales de televisión para monitorizar.
Actores	Operador de Monitorización: (Inicia) Inserta, elimina y monitoriza canales de televisión.
Resumen	El CU inicia cuando el usuario oprime el botón “cargar canal televisivo”, introduce los datos que se solicitan y finaliza cuando se reproducen los canales de televisión listados.
Complejidad	Media.
Prioridad	Secundario.
Precondiciones	El usuario con rol Operador de Monitorización existe en la BD.
Postcondiciones	Se adicionó el canal televisivo a monitorizar. Se eliminó un canal televisivo.

	Se seleccionó un canal televisivo para ser monitorizado.
Flujo de eventos	
Flujo básico < Gestionar flujo de streaming de video independiente>	

	Actor	**Sistema**
1.	Pulsa en el botón “cargar canales televisivos”.	
2.		Muestra un formulario para introducir los datos del canal televisivo y una lista donde se muestran los canales televisivos adicionados.
3.	Selecciona una de las siguientes opciones. - Adicionar canales televisivos a monitorizar. - Eliminar canales televisivos (Ir a la Sección 1 “Eliminar”)	
4.	Introduce los datos pertenecientes a los canales de televisión que desea monitorizar. Nombre del canal. URL. Descripción.	
5.	Pulsa el botón “Adicionar”.	
6.		Valida los datos introducidos.
7.		Guarda los datos adicionados en la base de datos.
8.		Actualiza la lista de canales televisivos adicionados. Termina el caso de uso.

Flujos alternos		
5a Evento:<Cancelar>		
	Actor	**Sistema**
5a	Pulsa el botón “Cancelar”.	
6a		Cancela la operación y regresa a la interfaz de monitorización. Termina el caso de uso.
7aEvento:<Campos vacíos>		

	Actor	Sistema
7 a		Muestra un mensaje indicando que existen campos vacíos. Regresa al paso 4 del flujo básico.
5c Evento:<OK>		
	Actor	**Sistema**
5c	Presiona el botón “OK”.	
6c		Se reproducen los canales que están en la lista. Termina el caso de uso
Sección 1: “Eliminar”		
Flujo básico < gestionar flujo de *streaming* de video independiente >		
	Actor	**Sistema**
4.	Selecciona un canal televisivo en la lista que se muestra.	
5.	Presiona el botón eliminar.	
6.		Elimina de la base de datos el canal televisivo.
7.		Actualiza la lista de canales televisivos. Termina el caso de uso.
Flujosalternos		
5a Evento<Cancelar>		
	Actor	**Sistema**
5a	Pulsa el botón “Cancelar”.	
6a		Cancela la operación y regresa a la interfaz de monitorización. Termina el caso de uso.
5b Evento<No seleccionado>		
	Actor	**Sistema**
5b	No selecciona un canal televisivo en la lista que se muestra.	
6b		No elimina de la base de datos el canal televisivo.
5aEvento<OK>		
5 a	Presiona el botón “OK”.	
6 a		Se reproducen los canales que están en la lista. Termina el caso de uso

Relaciones	**CU Incluidos**	NA
	CU Extendidos	NA

Requisitos no funcionales	NA

Tabla 3.Descripción del Caso de Uso Gestionar flujo de *streaming* de video independiente.

3.5. Conclusiones parciales.

Al no estar definidos con claridad los procesos del negocio se hizo el modelo de dominio que permite esclarecer los principales conceptos ligados al Subsistema de Monitorización de Señales Digitales. Se definieron los requisitos funcionales y no funcionales, los cuales, de ser implementados correctamente, garantizan las condiciones y capacidades que debe cumplir el software para que se puedan monitorizar señales digitales. Se modelaron y describieron los casos de uso del sistema que detalla cómo será la interacción actor-componente, estos facilitan el entendimiento de las funcionalidades que debe realizar la aplicación.

CAPÍTULO IV: CONSTRUCCIÓN Y VALIDACIÓN DE LA SOLUCIÓN PROPUESTA.

En el presente capítulo se tratarán temas fundamentales con respecto a la construcción de la solución propuesta, será definida la arquitectura del sistema y se describirán las características de los patrones de diseño que serán utilizados. Además se desarrollará la etapa del diseño y luego la de implementación. Es importante aclarar que para lograr la transición al Diseño, normalmente se utiliza el flujo de trabajo análisis debido a que, según lo propuesto por RUP, este le antecede durante el proceso de desarrollo de software. No obstante una variante válida es no utilizar en absoluto el modelo del análisis para describir los resultados del Análisis, producto a que los requisitos se analizan como parte integrada de la captura de requisitos, lo que trae como exigencia durante el proceso un mayor formalismo en la realización del modelo de casos de uso. Esto último es posible hacerlo pues el equipo de trabajo es capaz de comprender los resultados que los casos de uso pueden proporcionar y los requisitos son simples, conocidos y se cuenta con cierta comprensión de los mismos. Por tanto se decide prescindir de la realización del modelo del análisis en la investigación, garantizando con ello un ahorro considerable de tiempo en la realización de la misma.

4.1. Arquitectura de software.

JimArlow e IlaNeustad en su libro "*UML and the Unified Process Practical Object Oriented Analysis and Design"* definen la arquitectura de software de un programa o sistema como: la estructura o estructuras del sistema, las cuales comprometen elementos de software, las propiedades externamente visibles de esos elementos y las relaciones entre los mismos. (28)

El proceso de arquitectura de software toma los requisitos de los clientes, los analiza y produce un diseño para obtener un software que satisfaga sus necesidades. La arquitectura de software puede considerarse como un mapeo entre lo que un software debe lograr y los detalles de la implementación como código. Al obtener la arquitectura correcta se garantizará la coincidencia óptima entre requisitos y resultados. El software con buena arquitectura llevará a cabo las tareas especificadas dentro de los parámetros de los requisitos originales. Si no se cuenta con una arquitectura adecuada, puede que sea difícil o incluso imposible implementar, operar, mantener e integrar el software correctamente con otros sistemas. (29)

En la aplicación se utilizará la arquitectura 3 capas, variante de la arquitectura *N* capas que consiste en separar un proyecto en Capa de Presentación, Capa de Negocio y Capa de Datos. La misma permite distribuir el trabajo de creación de una aplicación por niveles, garantizando así una relación acoplamiento-

cohesión de buena calidad. La cohesión es un parámetro que valora la facilidad con la que en una aplicación es posible reunir componentes. El acoplamiento evalúa las relaciones que se establecen entre partes o componentes de un software, logrando un nivel bajo de acoplamiento y alto de cohesión se puede reutilizar código existente en otro contexto o reutilizar funcionalidades ya definidas.

Capa de Presentación: esta es la parte que ve el usuario, las pantallas que se le muestran para que interactúe con el programa. En esta capa se realizan validaciones para comprobar que no hay errores de formato. Esta capa se comunica únicamente con la capa de negocio llevando y trayendo los datos o registros necesarios.

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 todos los procesos que deben realizarse en el subsistema.

Capa de Datos: es donde se realiza todo el almacenamiento de datos. Recibe solicitudes de almacenamiento o recuperación de información desde la capa de negocio. (30)

Se define la arquitectura 3 capas pues es más sencillo el mantenimiento y soporte de la aplicación, en caso de que sobrevenga algún cambio, sólo se ataca al nivel requerido sin tener que revisar entre código mezclado. Tiene mayor flexibilidad ya que se pueden añadir nuevos módulos para dotar al subsistema de nuevas funcionalidades, en caso de necesitarlas en un futuro. Además es una arquitectura altamente escalable y se logra con la misma una mejor organización en la estructura de la aplicación.

4.2. Principios del diseño.

El diseño de un sistema de software es la búsqueda de soluciones que se ajusten a los requisitos del usuario, actividad necesaria para conseguir un software bien acabado. Es el proceso de definición de la arquitectura de software, en el cual se hace una invención y selección de programas que cumplan los objetivos de un sistema. Es una combinación de creatividad, intuición y experiencia por parte del ingeniero de software y este aplicando las guías, métodos y heurísticas que conforman el proceso es capaz de crear un diseño de excelente calidad. (31)

Con el objetivo de hacer entender mejor la temática abordada en el epígrafe se describen a continuación un conjunto de principios para el diseño del software:

- ✓ **En el proceso de diseño no deberá utilizarse orejeras:** un buen diseñador deberá tener en

cuenta enfoques alternativos, juzgando todos los que se basan en los requisitos del problema y los recursos disponibles para realizar el trabajo.

- ✓ **El diseño deberá poderse rastrear hasta el modelo de análisis:** dado que un solo elemento del modelo de diseño suele hacer un seguimiento de los múltiples requisitos, es necesario tener un medio de rastrear como se han satisfecho los requisitos por el modelo de diseño.
- ✓ **El diseño no deberá inventar nada que ya esté inventado:** los sistemas se construyen utilizando un conjunto de patrones de diseño, muchos de los cuales probablemente ya se han encontrado antes. Estos patrones deberán elegirse siempre como una alternativa para reinventar. Hay poco tiempo y los recursos son limitados. El tiempo de diseño se deberá invertir en la representación verdadera de ideas nuevas y en la integración de esos patrones que ya existen.
- ✓ **El diseño deberá minimizar la distancia intelectual entre el software y el problema como si de la misma vida real se tratara:** es decir, la estructura del diseño del software (siempre que sea posible) imita la estructura del dominio del problema.
- ✓ **El diseño deberá presentar uniformidad e integración:** un diseño es uniforme si parece que fue una persona la que lo desarrolló por completo. Las reglas de estilo y de formato deberán definirse para un equipo de diseño antes de comenzar el trabajo sobre el diseño. Un diseño se integra si se tiene cuidado a la hora de definir interfaces entre los componentes del diseño.
- ✓ **El diseño deberá estructurarse para degradarse poco a poco, incluso cuando se enfrenta con datos, sucesos o condiciones de operación aberrantes:** un software bien diseñado no deberá nunca explotar como una bomba Deberá diseñarse para adaptarse a circunstancias inusuales, y si debe terminar de funcionar, que lo haga de forma suave.
- ✓ **El diseño no es escribir código y escribir código no es diseñar:** incluso cuando se crean diseños procedimentales para componentes de programas, el nivel de abstracción del modelo de diseño es mayor que el código fuente. Las únicas decisiones de diseño realizadas a nivel de codificación se enfrentan con pequeños datos de implementación que posibilitan codificar el diseño procedimental.
- ✓ **El diseño deberá revisarse para minimizar los errores conceptuales (semánticos):** A veces existe la tendencia de centrarse en minucias cuando se revisa el diseño, olvidándose del bosque por culpa de los árboles. Un equipo de diseñadores deberá asegurarse de haber afrontado los elementos conceptuales principales antes de preocuparse por la sintaxis del modelo del diseño. (26)

4.3. Patrones de diseño.

Cada patrón describe un problema que ocurre una y otra vez en el entorno, y describe la esencia de la solución a ese problema, de tal modo que pueda utilizarse esta solución un millón de veces más, sin siquiera hacerlo de la misma manera dos veces. Los patrones de diseño son una solución probada, son la experiencia documentada de cómo resolver un problema, lo cual facilita el trabajo a los programadores. Un sistema informático es más comprensible, siempre y cuando esté documentado con los patrones que usa. Los patrones de diseño soportan la reutilización de arquitecturas de software. Estos describen una estructura común y recurrente de comunicación entre componentes, resuelven un problema general de diseño dentro de un contexto particular. (32)

Erich Gamma, Richard Helm, Ralph Jhonson y JhonVlissidis formaron un equipo denominado “pandilla de los cuatro” (GoF, *por sus siglas en inglés*), ellos recopilaron y documentaron 23 patrones de diseño aplicados por expertos diseñadores del software. Los patrones GoF, como se les conoce, están clasificados en tres grupos, según sus propósitos: creacionales, estructurales y de comportamiento. (33)

Los patrones de Asignación de Responsabilidades (GRASP, *por sus siglas en inglés*) describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones (34). En el diseño de la aplicación se utilizaron los patrones Experto, Creador, Bajo Acoplamiento y Alta Cohesión pertenecientes al grupo de patrones GRASP. Mientras que dentro de los patrones GoF se utilizó Observador.

Patrones GRASP:

Experto: es un patrón que se utiliza para asignar responsabilidades a las clases que contienen la información necesaria para cumplir con las mismas. Si esta asignación mencionada anteriormente se hace de forma correcta se garantizan definiciones de clases sencillas y más cohesivas que son más fáciles de comprender y mantener. (35)

Este patrón fue utilizado en el desarrollo de la aplicación ya que las clases que contienen una información específica son las más idóneas para implementar acciones que dependan de dicha información. Se utilizó, por citar algunos ejemplos, en las clases qvumeter y pantalla.

Creador: el patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos. Se aplica en todos los casos donde una clase necesite instanciar a otra. Su correcta utilización supone menos dependencias respecto al mantenimiento y mejores oportunidades de reutilización. (35)

La utilización de este patrón se puede evidenciar en la clase principal, donde se crea una instancia de la clase pantalla con el fin de reproducir un flujo de *streaming* de video, así como otras acciones relacionadas con el mismo.

Bajo Acoplamiento: es un patrón que marca una buena práctica de la programación y se debe tener siempre en cuenta durante las decisiones de diseño. Soporta el diseño de clases más independientes, que reducen el impacto de los cambios, haciéndolas también más reutilizables. Con el uso de este patrón los componentes no se afectan por cambios de otros componentes. (35)

Se evidencia el patrón Bajo Acoplamiento en la clase defocuseddetection, la cual se encuentra en el *plugin* de calidad. Esta clase es la encargada de detectar el grado de borrosidad de una imagen, la misma es independiente ya que no presenta cambios locales al crearse cambios en las clases afines.

Alta Cohesión: este patrón, como el anterior, ha de tenerse en cuenta en cada momento del diseño. Consiste en una clase con una importante funcionalidad y poco trabajo que hacer. Colabora con otros objetos para compartir el esfuerzo si la tarea es grande. Gracias al patrón es más fácil el entendimiento del diseño, además se simplifican el mantenimiento y las mejoras en funcionalidad. (35)

El patrón Alta Cohesión es utilizado en la clase *Logger* perteneciente al *plugin* de calidad, la cual se encarga únicamente de registrar datos en *logs*. La clase presenta una alta capacidad de reutilización.

Patrones Gof:

Observador: el patrón observador define una dependencia del tipo “uno a muchos” entre objetos, se definen objetos observadores y un objeto observado, con el fin de detectar algún cambio en el objeto observado y reproducirlo en los observadores (35). Se evidencia el uso de este patrón en la aplicación cuando son utilizados los *SIGNAL* y *SLOT*, que son los mecanismos de manejos de eventos en el *framework Qt.*

4.4. Diagrama de clases del diseño.

El Diagrama de clases permite especificar la estructura de clases del sistema, con relaciones entre clases y estructuras de herencia. Durante el diseño, el Diagrama de Clases se elabora para tener en cuenta los detalles concretos de la implementación del sistema. El mismo está compuesto por relaciones y clases y una vez terminado muestra cómo debe quedar la aplicación, desde un punto de vista lógico. (36)

Se muestra una representación del diagrama de clases del diseño (Ilustración 3) del Subsistema de Monitorización de Señales Digitales (para ver diagrama completo, ir a anexo 3). En el mismo se representan las clases pertenecientes al subsistema con sus métodos más importantes y las relaciones que existen entre dichas clases. El diagrama está estructurado en tres grupos, cada uno representa una capa de la arquitectura de software definida con anterioridad en el epígrafe 4.1 del presente capítulo. Cada grupo a su vez está conformado por las clases pertenecientes a la capa que representa dicho grupo.

Las clases por su parte interactúan unas con otras y se representan estas interacciones mediante relaciones de asociación, agregación y composición. Existen varios tipos de relaciones además de las mencionadas, tales como las relaciones de dependencia y las relaciones de herencia, pero estas no son utilizadas en el diagrama que se está presentando.

Las relaciones de asociación se representan específicamente con una línea continua entre las clases que presenten este tipo de relación, valga la redundancia. Tal es el caso de la clase Principal y la clase UrlWindow las cuales tienen una relación de asociación, la cual se tiene en caso de que exista entre las clases una dependencia que sea permanente en el tiempo. En concreto y de una forma más clara, se está en presencia de una relación de asociación siempre y cuando una clase X tiene en su código fuente, una vez se implemente, un objeto de tipo Y. En este caso y volviendo a los ejemplos citados con anterioridad, la clase Principal tiene un objeto de la clase UrlWindow por lo que presentan una relación de asociación.

En el diagrama que se describe hay relaciones de agregación, las cuales se representan visualmente con una línea continua y un rombo sin relleno en uno de los extremos. Esta última es un tipo de relación en la que se puede identificar un todo y partes que conforman ese todo. Por ejemplo una computadora está formada por una placa base, cables y circuitos por lo que la relación computadora-cable es igual que todo-partes y sin las partes el todo no funciona, como la computadora no funciona sin cables. En el diagrama que se evidencia este tipo de relación, por solo citar un ejemplo, entre las clases Autenticarse y

C_Monitorización, ya que la clase Autenticarse está formada por un concepto de la clase C_Monitorización, por lo que el rombo va en el extremo de la relación que se corresponde con el todo.

La tercera y última de las relaciones que se encuentran en el diagrama de clases del diseño que se está explicando es la de composición. La misma se representa de igual forma que la de agregación pero el rombo estará relleno, además tiene la misma filosofía de todo-partes. La diferencia radica en que las partes que componen un todo no pueden ser partes de otro, es decir son única y exclusivamente partes de un solo todo. Es válido aclarar que una vez deje de existir el todo dejará de existir la parte también. En este diagrama específicamente hay varias relaciones de composición, una de ellas es entre las clases C_Monitorización y Pantalla, donde Pantalla conforma a C_Monitorización y al no relacionarse con más ninguna otra clase entonces presenta una relación de composición, por lo que una vez destruida la clase C_Monitorización, su homóloga Pantalla dejará de existir también.

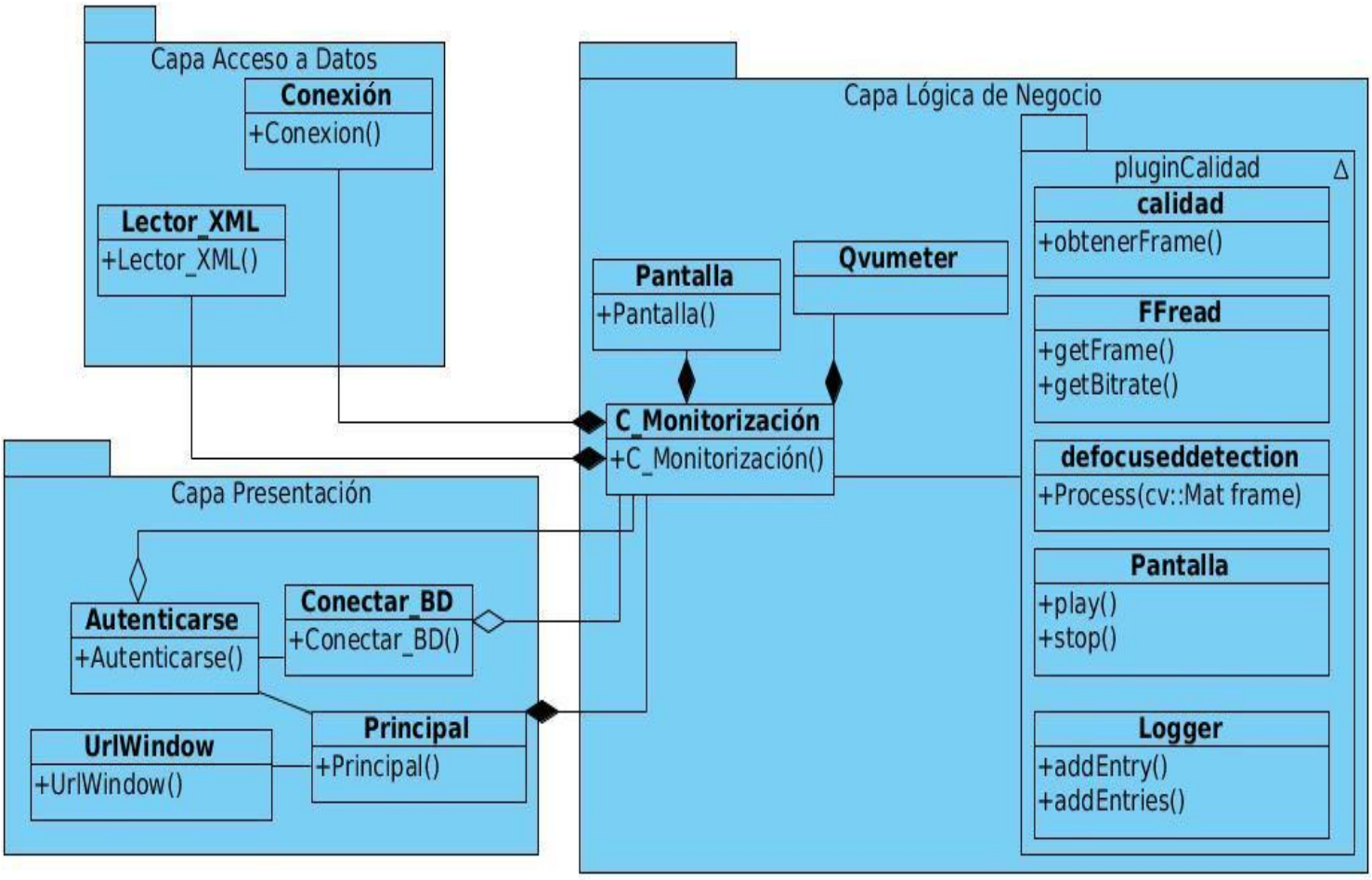


Ilustración 3. Diagrama de Clases del Diseño.

4.5. Diagrama de despliegue.

Un diagrama de despliegue es un grafo de nodos unidos por conexiones de comunicación. Un nodo es un recurso de ejecución tal como un computador, un dispositivo o memoria. El diagrama de despliegue permite la representación delas relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. En la vista de despliegue se representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación. (37)

El diagrama de despliegue perteneciente al Subsistema de Monitorización de Señales Digitales, que a continuación se muestra (Ilustración 4) está compuesto por tres nodos. Uno de los nodos es una PC Cliente que será donde se monitorizarán la señales digitales de radio y televisión, además ahí también se registrarán las incidencias. Por otro lado se encuentra un servidor de *streaming*, en el cual estarán las medias en transmisión y además un servidor de Base de Datos que contendrá información referente a dichas medias. La información de las medias consiste en el nombre del canal, la *url* por donde se está transmitiendo y la programación de los canales. La PC Cliente se conecta con el servidor de *streaming* mediante el protocolo UDP y al servidor de Base de Datos por el protocolo TCP/IP.

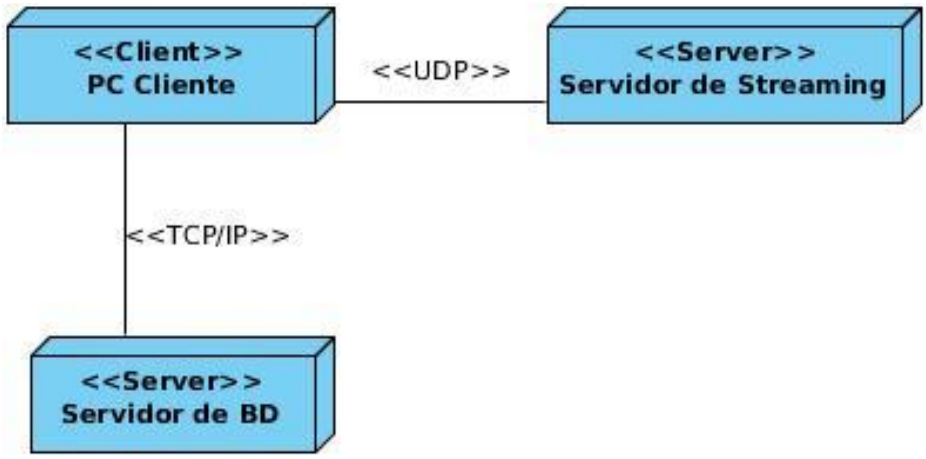


Ilustración 4. Diagrama de Despliegue.

4.6. Modelo de implementación.

El Modelo de Implementación es comprendido por un conjunto de componentes y subsistemas que constituyen la composición física de la implementación del sistema. Entre los componentes se pueden encontrar datos, archivos, ejecutables, código fuente y los directorios. Este artefacto describe cómo se implementan los componentes, congregándolos en subsistemas. Para representar los diagramas del Modelo de Implementación se puede emplear el diagrama de UML de Componentes. (38)

4.6.1. Diagrama de componentes.

Un diagrama de componentes muestra las dependencias lógicas entre componentes de software. Se representa como un grafo de componentes de software unidos por medio de relaciones de dependencia que indican que un componente utiliza los servicios de otro. (39)

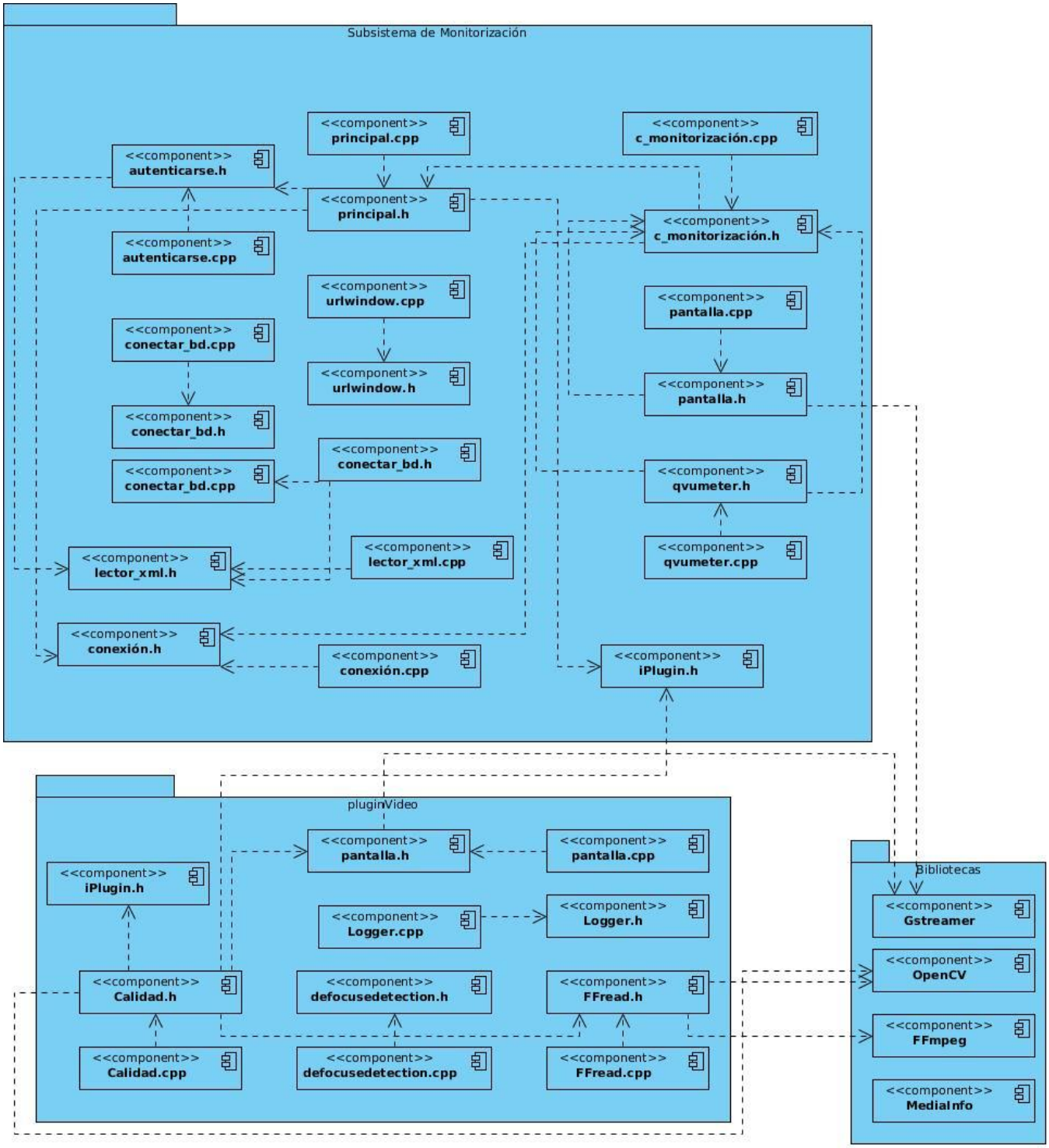


Ilustración 5. Diagrama de Componentes.

4.7. Pruebas de software.

Las pruebas del software son un elemento crítico para la garantía de calidad del software y representa una revisión final de las especificaciones, del diseño y de la codificación. Una vez generado el código fuente se debe probar el software para que el usuario final detecte la menor cantidad de errores posibles. (26)

Existen varios tipos de prueba cada una con características diferentes y orientadas a probar partes específicas del software. A continuación se explican los objetivos que persiguen los distintos tipos de prueba.

- ✓ **Pruebas de Unidad:** las pruebas unitarias tienen como objetivo verificar la funcionalidad y estructura de cada componente, individualmente, una vez que ha sido codificado. La prueba de unidad es un proceso para probar los subprogramas, las subrutinas, los procedimientos individuales o las clases en un programa. Es decir, es mejor probar primero los bloques desarrollados más pequeños del programa, que inicialmente probar el software en su totalidad. Las motivación es para hacer esto son tres. Primera, las pruebas de unidad son una manera de manejar los elementos de prueba combinados, puesto que se centra la atención inicialmente en unidades más pequeñas del programa. (40)
- ✓ **Pruebas de Integración:** el objetivo de las pruebas de integración es verificar el correcto ensamblaje entre los distintos componentes una vez que han sido probados unitariamente con el fin de comprobar que interactúan correctamente a través de sus interfaces, tanto internas como externas, cubren la funcionalidad establecida y se ajustan a los requisitos no funcionales especificados en las especificaciones necesarias. (40)
- ✓ **Pruebas de Aceptación:** el objetivo de las pruebas de aceptación es validar que un sistema cumple con el funcionamiento esperado y permitir al usuario de dicho sistema que determine su aceptación, desde el punto de vista de su funcionalidad y rendimiento. Las pruebas de aceptación son definidas por el usuario del sistema y preparadas por el equipo de desarrollo, aunque la ejecución y aprobación final corresponden al usuario. (40)
- ✓ **Pruebas de Sistema:** las pruebas de sistema buscan discrepancias entre el programa y sus objetivos o requerimientos, enfocándose en los errores hechos durante la transición del proceso al diseñar la especificación funcional. Esto hace a las pruebas de sistema un proceso vital de pruebas, ya que en términos del producto, número de errores hechos, y severidad de esos errores,

es un paso en el ciclo de desarrollo generalmente propenso a la mayoría de los errores. Las pruebas de sistema no son procesos para probar las funciones del sistema o del programa completo, porque ésta sería redundante con el proceso de las pruebas funcionales. Las pruebas del sistema tienen un propósito particular: para comparar el sistema o el programa con sus objetivos originales (requerimientos funcionales y no funcionales). (40)

4.7.1. Prueba de caja negra.

La prueba de caja negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del software. O sea, los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa (por ejemplo, archivos de datos) se mantiene. Una prueba de caja negra examina algunos aspectos del modelo fundamentales del sistema. (26)

La partición equivalente es un método de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre de forma inmediata una clase de errores (por ejemplo, proceso incorrecto de todos los datos de carácter) que, de otro modo, requerirán la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar. (26)

Se decidió realizarle al Subsistema de Monitorización de Señales Digitales pruebas de unidad, específicamente de caja negra, por el método de partición equivalente. Para la realización de este tipo de pruebas se definen condiciones de entrada válida y no válida, con las que se probarán los requisitos funcionales del subsistema.

Se expone a continuación el diseño de casos de prueba (DCP) del caso de uso del sistema (CUS) Gestionar flujo de *streaming* independiente. Los casos de prueba de los demás casos de uso del sistema se pueden ver en el anexo 4.

DCP CUS: Gestionar flujo de *streaming* de video independiente.

Descripción general:

El CU inicia cuando el usuario oprime el botón cargar canal televisivo, introduce los datos que se solicitan y finaliza cuando reproduce los canales de televisión seleccionados.

Condiciones de ejecución:

Usuario con rol Operador de Monitorización tiene que existir en la base de datos

Escenarios a probar en el Caso de Uso:

Escenario	Descripción	Nombre del canal	URL	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Adicionar canal televisivo correctamente.	El Operador de Monitorización introduce Nombre del canal, URL y Descripción de los canales de televisión que desea monitorizar.	V	V	NA	Actualiza la lista de canales televisivos a monitorizar.	Botón "cargar canales televisivos"/ Botón "Adicionar"
		V1	udp://@224,0,0,7:1234			
EC 1.2 Adicionar canal televisivo incorrectamente.	Si la URL del canal está vacía no permite adicionar el canal y muestra un mensaje de error al usuario.	V	I	NA	Muestra un mensaje de error Ejemplo: "Debe especificar la URL del canal".	Botón "cargar canales televisivos"/ Botón "Adicionar"
		I	V	NA		
EC 1.3 Cancelar.	Se cancela la operación.	NA	NA	NA	Cancela la operación y regresa a la interfaz monitorizar.	Botón "cargar canales televisivos"/ Botón "Cancelar"
EC 1.4 OK	Se reproducen los canales que están en la lista.	NA	NA	NA	Se reproducen los canales que están en la lista y regresa a la interfaz monitorizar.	Botón "cargar canales televisivos"/ Botón "OK"

Tabla 4. SC1: Adicionar canales televisivos a monitorizar.

Escenario	Descripción	Lista canales	Respuesta del sistema	Flujo central
EC 2.1 Eliminar canal televisivo.	El Operador de Monitorización elimina el canal.	V	Elimina los canales seleccionados y actualiza la lista de canales.	Botón "cargar canales televisivos"/Botón "Eliminar"
EC 2.2 Eliminar canal televisivo. Incorrectamente	El Operador de Monitorización elimina el canal.	I	No elimina ningún canal	Botón "cargar canales televisivos"/Botón "Eliminar"
EC 2.3 Cancelar.	Cancela la operación y regresa a la interfaz monitorizar.	NA	Cancela la operación y regresa a la interfaz monitorizar.	Botón "cargar canales televisivos"/ Botón "Cancelar"
EC 2.4 OK	Reproduce los canales de la lista y regresa a la interfaz monitorizar.	NA	Reproduce los canales de la lista y regresa a la interfaz monitorizar.	Botón "cargar canales televisivos"/ Botón "OK"

Tabla 5. SC2: Eliminar canal televisivo.

Después de diseñados los casos de prueba estos fueron aplicados al software, detectando en una primera iteración veintitrés No Conformidades. Las mismas consistían en problemas de validación tales como no mostrar mensajes cuando se dejaban campos vacíos y permitir la inserción de datos incorrectos. En el caso de uso Generar alarma visual de error al transmisor se permitía enviar la notificación al transmisor sin seleccionar algunos datos de importancia, tales como el tipo de error o la *url* del canal que presenta problemas, haciendo que colapsara el software. El software funciona a plenitud siempre que reciba la información por el protocolo UDP, no siendo así con la información recibida mediante el protocolo *mms* ya que el flujo es reproducido pero no se miden los niveles de audio, ni se registran las incidencias que ocurran. Todos estos errores fueron corregidos y se procedió a realizarle una segunda iteración de pruebas a la solución creada, iteración que arrojó resultados satisfactorios, es decir, no se encontraron No Conformidades y se dio por terminada la etapa de pruebas. Los resultados de las pruebas explicados anteriormente se representan gráficamente (Ver Ilustración 6) para hacerles llegar, de una forma visual, un resumen de las pruebas realizadas.

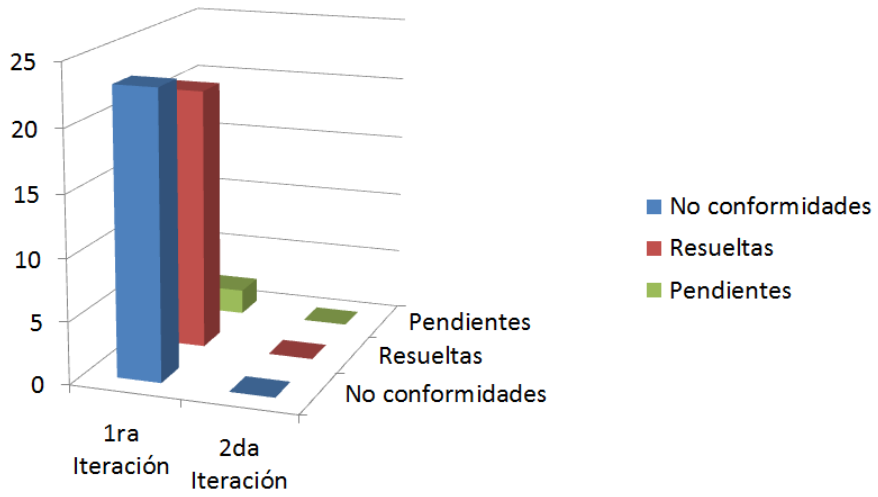


Ilustración 6. Resultado de las pruebas.

4.8. Conclusiones parciales.

En este capítulo se realizaron los flujos de trabajo: Diseño, Implementación y Prueba que arrojaron los artefactos correspondientes a cada uno de ellos, siendo estos claves para documentar el proceso de construcción y validación del subsistema. Con el diagrama de clases del diseño fue posible representar la estructura del Subsistema de Monitorización de Señales Digitales, garantizando con ello un mejor entendimiento de la información que se utilizará en la aplicación. Se expuso el Diagrama de Componentes, donde se hizo alusión a los distintos componentes que se utilizan en el desarrollo de la aplicación, así como la relación entre ellos, permitiendo mostrar la organización y las dependencias entre los componentes de subsistema de Monitorización de Señales Digitales. Se presentó también el diagrama de despliegue que muestra la distribución física del sistema, ayudando a tener una idea concisa de cómo se despliega la aplicación. Finalmente se le realizaron pruebas al software que permitieron encontrar un conjunto de No Conformidades que, después de resolverlas, ayudaron a que el usuario final tenga un software con la menor cantidad de fallas posibles.

CONCLUSIONES GENERALES.

La personalización del Subsistema de Monitorización de Señales Digitales posibilitó la puesta en práctica de los conocimientos adquiridos en la carrera Ingeniería en Ciencias Informáticas. Realizadas todas las tareas de la investigación científica, es posible afirmar que los objetivos fueron cumplidos satisfactoriamente por lo que se puede concluir que:

- ✓ Se describió la evolución histórico-lógica de los procedimientos y técnicas de la transmisión y monitorización de señales digitales, que permitió adquirir conocimientos y así comprender mejor la situación problemática de la investigación.
- ✓ Se caracterizaron y analizaron los sistemas existentes para la monitorización de señales digitales, lo que posibilitó ampliar los conocimientos en cuanto al funcionamiento de estos sistemas.
- ✓ Se definieron las herramientas y tecnologías actuales, con lo que se garantizó la calidad de la solución propuesta ya que, gracias a estas, el software cumple los requerimientos definidos.
- ✓ Se generaron los artefactos y documentación correspondiente al proceso de desarrollo del software, los cuales permitieron guiar la construcción de la aplicación, además de un fácil entendimiento del software a personas que no estuvieron presentes en el desarrollo del mismo.
- ✓ Se desarrolló una personalización del Subsistema de Monitorización de Señales Digitales que permite monitorizar tanto radio como televisión y registrar incidencias que ocurran en los canales televisivos, además de ser autónomo, por lo que se cumple el objetivo general planteado en la investigación y queda comprobada la idea a defender.
- ✓ Las pruebas aplicadas al subsistema arrojaron resultados satisfactorios por lo tanto se cuenta con un software capaz de monitorizar señales digitales y registrar incidencias ocurridas en los canales. Además es un producto autónomo, lo que hace que sea uno más en la gama del departamento Señales Digitales.

RECOMENDACIONES.

- ✓ Internacionalizar el software, es decir, que esté disponible en distintos idiomas.
- ✓ Profundizar la investigación en cuanto a la calidad perceptual de un video, para así ampliar la cantidad de tipos de errores a detectar.

BIBLIOGRAFÍA CONSULTADA Y REFERENCIADA.

1. **Gómez, Gustavo.** Asociación Mundial de Radios Comunitarias de América Latina y el Caribe. [En línea] 2007. [Citado el: 26 de 11 de 2012.] http://www.amarcalc.org/caraysenial/0.php?cys=7&s=5&n=1.

2. **Miyara, Federico.** *Conversores D/A y A/D.* [PDF] Rosario : Universidad Nacional de Rosario, 2004.

3. Televisión Digital. [En línea] Gobierno de España. Ministerio de Industria, Energía y Turismo. [Citado el: 1 de junio de 2013.] http://www.televisiondigital.es/TERRESTRE/INFORMACIONGENERAL/Paginas/TDTInfo.aspx.

4. **Avendaño, Bárbara y Rosabal, Heriberto.** Bohemia. [En línea] [Citado el: 01 de 05 de 2013.] http://www.bohemia.cu/2013/04/01/encuba/informatica2.html.

5. Real Academia de Lengua Española. [En línea] [Citado el: 22 de 11 de 2012.] http://www.rae.es/rae.html.

6. *Monitoring For Performance Based Timber Management Workshop.* [PDF] Richmond, U.S.A : Richmond, BC, 2002.

7. Tecnología hecha palabra. [En línea] Auvisión, 02 de 02 de 2008. [Citado el: 22 de 11 de 2012.] http://www.tecnologiahechapalabra.com/tecnologia/comunicados/telecom/articulo.asp?i=2142.

8. **Soluciones Videoma.** *Monitorización y Análisis Multicanal.* [PDF] Madrid, España : ISID, 2012.

9. Metodologías de Desarrollo de Software. [En línea] Universidad de Murcia. [Citado el: 15 de 01 de 2013.] http://www.um.es/docencia/barzana/IAGP/Iagp2.html#BM4.

10. **Figueroa, Roberto G., Solís, Camilo J. y Cabrera, Armando A.** *Metodologías Tradicionales vs. Metodologías Ágiles.* [DOC] Loja, Ecuador : Universidad Técnica Particular de Loja, Escuela de Ciencias en Computación, 2012.

11. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El Proceso Unificado de Desarrollo de Software.* s.l. : Pearson Educación, S.A, 2000. ISBN-84-7829-036-2.

12. **Hernández Orallo, Enrique.** *El Lenguaje Unificado de Modelado.* [PDF] Valencia, España : Universidad Politécnica de Valencia, 2012.

13. **Departamento de Sistemas Informáticos y Computación.** *Introducción a Herramientas CASE y Sistem Architect.* [PDF] Valencia, España : Universidad Politécnica de Valencia., 2002.

14. **Bustos, Guillermo.** *Guía de Uso de la Herramienta CASE Visual Paradigm Standard Edition Versión 8.0.* [PDF] Valparaiso : Pontificia Universidad Católica de Valparaiso, 2010.

15. **Blanco, Carlos.** CarlosBlanco.pro. [En línea] 08 de 04 de 2012. [Citado el: 21 de 05 de 2013.] http://carlosblanco.pro/2012/04/entornos-desarrollo-integrado-introduccion/.

16. **Nokia Corporation.** Qt. [En línea] Nokia. [Citado el: 05 de 12 de 2012.] http://qt.nokia.com/.

17. **Programadores CodeBox.Inc.** CodeBox. [En línea] 2010. [Citado el: 10 de 12 de 2012.] www.codebox.es/glosario.

18. **Mínguez Pérez, Eduardo y Garcicuño Enríquez, David.** *Qt.* [PDF] Salamanca, España : Departamento de Informática y Automática, Universidad de Salamanca, 2005.

19. [En línea] [Citado el: 12 de 02 de 2013.] www.definicion.org/lenguaje-de-programacion.

20. **S., Amaya.** Cplusplus. [En línea] 2010-2013. [Citado el: 21 de 05 de 2013.] http://www.cplusplus.com/info/description.

21. **Luján Mora, Sergio.** *C++ paso a paso.* [PDF] 2005.

22. **Gómez Ballester, Eva, y otros.** *Base de Datos 1.* 2006.

23. PostgreSQL. [En línea] [Citado el: 15 de 01 de 2013.] http://www.postgresql.org.es/sobre_postgresql..

24. [En línea] [Citado el: 10 de 12 de 2012.] http://sourceforge.net/projects/opencv/.

25. Sitio Oficial de FFmpeg. [En línea] [Citado el: 10 de 12 de 2012.] http://ffmpeg.or/.

26. **Pressman, Roger S.** *Ingeniería del Software. Un enfoque práctico.* Sexta Edición. 2005. ISBN:9701054733.

27. Procuraduría General de la Nación - República de Colombia. [En línea] [Citado el: 25 de 05 de 2013.] www.procuraduria.gov.co/infosim/media/file/Etapa4-ReqNoFunc.pdf.

28. **Arlow, Jim y Neustadt, Ila.** *UML and the Unified Process Practical Object Oriented Analysis and Design.* 2005.

29. Microsoft Developer Network. [En línea] Microsoft, 2013. [Citado el: 20 de 05 de 2013.] http://msdn.microsoft.com/es-ar/hh144976.aspx.

30. KernelError. [En línea] 06 de 02 de 2009. [Citado el: 02 de 05 de 2013.] http://kernelerror.net/programacion/php/arquitectura-3-capas/.

31. **García Peñalvo, Dr. Francisco José, Conde González, Miguel Ángel y Bravo Martín, Sergio.** [En línea] Universidad de Salamanca, 16 de 10 de 2008. [Citado el: 02 de 05 de 2013.] ocw.usal.es/ensenanzas-tecnicas/ingenieria-del-software/contenidos/Tema5-Principiosdeldisenodelsoftware-1pp.pdf.

32. **Mestras, Juan Pavón.** [En línea] Dep. de Ingeniería del Software e Inteligencia Artificial de la Universidad Complutense Madrid, 2004. [Citado el: 02 de 05 de 2013.] www.fdi.ucm.es/profesor/jpavon/poo/2.14PDOO.pdf.

33. Geek the Planet. [En línea] [Citado el: 02 de 05 de 2013.] http://geektheplanet.net/5462/patrones-gof.xhtml.

34. Prácticas de Software. [En línea] [Citado el: 02 de 05 de 2013.] http://www.practicasdesoftware.com.ar/2011/03/patrones-grasp/.

35. **Larman, Craig.** UML y patrones. 1999.

36. Grupo de Geofísica Computacional. [En línea] [Citado el: 03 de 05 de 2013.] http://mmc.geofisica.unam.mx/LuCAS/Tutoriales/doc-modelado-sistemas-UML/multiple-html/x219.html.

37. **Marca Huallpara, Hugo Michael y Quisbert Limachi, Nancy Susana.** *Diagrama de Despliegue.* [doc] Bolivia : Universidad Salesiana de Bolivia, 2008.

38. MeRinde. [En línea] [Citado el: 04 de 05 de 2013.] http://merinde.net/index.php?option=com_content&task=view&id=495&Itemid=291.

39. **Figueroa, Pablo.** [En línea] Universidad de Alberta, Canadá. [Citado el: 04 de 05 de 2013.] http://webdocs.cs.ualberta.ca/~pfiguero/soo/uml/implementacion01.html.

40. **Herrera Gonzáles, Carlos Arturo.** *Estrategias de Aplicación de Prueba de Unidad ,Integración, Sistema, y de Aceptación.* [Documento] Estado Puebla, México : Instituto Tecnológico Superior de Libres, 2012.

41. Sitio Oficial de la Radio y la Televisión pública de Colombia. [En línea] [Citado el: 21 de 11 de 2012.] http://www.rtvc.gov.co/index.php/atencion-al-ciudadano/glosarios/Glosario-RTVC-1/O/Ondas-radioel%C3%A9ctricas-u-ondas-hertzianas-9/..

42. OpenCV. [En línea] OpenCV.org. [Citado el: 10 de 12 de 2012.] www.opencv.org.

43. **Departamento de Informática y Sistemas.** Grupo de Inteligencia Artificial y Sistema. [En línea] Universidad las Palmas, Gran Canaria. [Citado el: 25 de 01 de 2013.] http://gias720.dis.ulpgc.es/Gias/Cursos/Tutorial_html/frames/frames.htm.

44. **Delgado, Antonio.** Eroski Consumer. [En línea] 2009. [Citado el: 20 de 02 de 2013.] http://www.consumer.es/web/es/tecnologia/internet/2009/09/18/187866.php..

ANEXOS

Anexo 1. Entrevista:

Se presenta a continuación la entrevista que se le realizó al Ing. Yoandrys Pacheco, líder del proyecto Sistema de Transmisiones de Canales Virtuales.

Preguntas:

- ✓ ¿Cómo funciona el sistema SIAV Transmisiones?
- ✓ ¿Qué tecnologías se utilizan para la transmisión de señales digitales?
- ✓ ¿Cuáles son las deficiencias del sistema SIAV Transmisiones?
- ✓ ¿El subsistema Monitorización, cómo funciona?
- ✓ ¿En qué consiste la tecnología *streaming*?
- ✓ En caso de no existir una planificación ¿Qué es capaz de hacer el Subsistema Monitorización?

Anexo 2. Descripciones de los casos de uso del sistema:

CU 1. Autenticar usuario

Objetivo	Autenticarse en el sistema.	
Actores	Usuario: (Inicia) Autenticar usuario.	
Resumen	El usuario introduce los datos que se solicitan para acceder a la aplicación, estos se verifican y el mismo finaliza dándole los permisos, en caso de poseer los privilegios necesarios.	
Complejidad	Baja.	
Prioridad	Secundario.	
Precondiciones	El usuario con rol Operador de Monitorización existe en la BD.	
Postcondiciones	El usuario quedó autenticado.	
Flujo de eventos		
Flujobásico<Autenticarusuario>		
	Actor	**Sistema**
9.	Introduce su usuario y contraseña en los campos de autenticación.	
10.	Presiona el botón autenticar.	
11.		Valida que los campos introducidos

		estén correctos.
12.		Se muestra la interfaz correspondiente al módulo Monitorización Televisiva. Termina el caso de uso.
Flujosalternos		
3a Evento<Credencialesincorrectas>		
	Actor	**Sistema**
3a		Muestra un mensaje de error indicando que los datos insertados son incorrectos. Termina el caso de uso.

Relaciones	**CU Incluidos**	NA
	CU Extendidos	NA
Requisitos no funcionales	NA	

CU 2. Gestionar conexión a la BD del módulo Monitorización

Objetivo	Insertar, editar y guardar datos para conectarse a la base de datos del sistema.
Actores	Operador de Monitorización: (Inicia) Insertar, editar, guardar datos de configuración.
Resumen	El usuario ejecuta la aplicación y la configuración de conexión a la base de datos es incorrecta. Termina cuando se hayan realizado correctamente todas las configuraciones de la base de datos.
Complejidad	Media.
Prioridad	Crítico.
Precondiciones	La conexión a la Base de Datos está incorrecta.
Postcondiciones	La configuración de la conexión a la BD quedó correctamente establecida.
Flujo de eventos	
Flujo básico < Gestionar conexión a la BD del módulo Monitorización >	

	Actor	Sistema
1.	Ejecuta la aplicación.	
2.		Muestra un mensaje de error indicando que la configuración de la conexión a la BD es incorrecta.
3.	Pulsa la opción “OK” del mensaje mostrado.	
4.		Muestra una interfaz con los campos a modificar.
5.	Introduce los parámetros de los campos.	
6.	Pulsa en el botón “Guardar”.	
7.		Valida los parámetros ingresados. 12.1 Verifica que no existan campos vacíos. 12.2 Verifica que el formato de la dirección IP sea correcta. 12.3 Verifica que el formato del puerto sea correcto. 12.4 Verifica las credenciales del usuario (usuario, contraseña).
Flujosalternos		
6a. Evento<Campos vacíos>		
	Actor	**Sistema**
6a.		Muestra un mensaje de alerta indicando que hay campos vacíos. Regresa al paso 4 del flujo básico.
6b. Evento <Formato de IP >		
	Actor	**Sistema**
6b.		Muestra un mensaje de alerta indicando que el formato de la dirección IP es incorrecto. Regresa al paso 4 del flujo básico.
6c. Evento <Formato del Puerto >		
	Actor	**Sistema**
6c.		Muestra un mensaje de alerta indicando que el formato del Puerto es incorrecto. Regresa al paso 4 del flujobásico.

6d. Evento<Credenciales incorrectas>			
	Actor		**Sistema**
6d.			Muestra un mensaje de alerta indicando que el usuario o la contraseña son incorrectos. Regresa al paso 4 del flujo básico.
Relaciones	**CU Incluidos**	NA	
	CU Extendidos	NA	
Requisitos no funcionales	NA		

CU 3. Mostrar datos de canales

Objetivo	Mostrar datos de los canales en transmisión como; programación del día, propiedades del canal, canales en transmisión, niveles de audio de los canales en transmisión.
Actores	Operador de Monitorización: (Inicia) Selecciona canal y se muestran los datos de canales, muestra la programación asociada a los canales y niveles de audio asociado a los canales.
Resumen	El caso de uso inicia cuando el operador de monitorización selecciona uno de los canales que se están monitorizando, se muestran todos los datos del mismo. Termina el caso de uso.
Complejidad	Media.
Prioridad	Crítico.
Precondiciones	Existe al menos un canal en transmisión.
Postcondiciones	Los canales planificados tienen medias asociadas. Existe conexión con la base de datos.
Flujo de eventos	
Flujo básico <Mostrar datos de canales >	

	Actor	**Sistema**
1.	Selecciona una de las siguientes opciones. - Listarlos canales en transmisión. - Listar datos del canal en transmisión (Ir a la Sección 1 Mostrar datos del canal). - Listar la programación del canal (Ir a	

	la Sección 2 Mostrar programación).	
2.		Muestra la lista de canales en transmisión. Termina el caso de uso.
Sección 1: "Mostrar datos de canales"		
Flujo básico <Mostrar datos del canal >		
	Actor	**Sistema**
2.	Selecciona de la lista de canales en transmisión el canal deseado.	
3.		Muestra los datos del canal. Termina el caso de uso.
Sección 2: "Mostrar programación"		
Flujobásico<Mostrar programación>		
	Actor	**Sistema**
1.	Selecciona de la lista de canales en transmisión el canal deseado.	
2.		Muestra la programación asociada al canal. Termina el caso de uso.

Relaciones	**CU Incluidos**	NA
	CU Extendidos	NA
Requisitos no funcionales	NA	

CU 4.Visualizar canal seleccionado

Objetivo	Visualizar canal seleccionado con mayor resolución.
Actores	Operador de Monitorización: (Inicia) Visualizar canal.
Resumen	El Operador de Monitorización desea visualizar a mayor escala el canal que se está transmitiendo. Termina este caso de uso cuando se visualiza dicho canal en el visor, a mayor resolución conjuntamente con la medición de los decibeles de audio de dicho canal.
Complejidad	Media.
Prioridad	Secundario.
Precondiciones	Existe al menos un canal en transmisión.

Postcondiciones	Se visualizó a mayor escala la transmisión del canal seleccionado conjuntamente la medición de decibles de audio de dicho canal.		
Flujo de eventos			
Flujo básico <Visualizar canal seleccionado >			
	Actor		**Sistema**
1.	Selecciona el canal a visualizar.		
2.			Muestra en el visor el flujo del canal seleccionado a mayor escala y se muestra la medición de decibeles de audio. Termina el caso de uso.
Relaciones	**CU Incluidos**	NA	
	CU Extendidos	NA	
Requisitos no funcionales	NA		

CU 6.Actualizar transmisión

Objetivo	Actualizar transmission automáticamente.		
Actores	Operador de Monitorización: (Inicia) Actualización de transmisión.		
Resumen	El Operador de Monitorización comienza a monitorizar al actualizarse la base de datos, el sistema actualiza el flujo de los canales y la programación del día así como los datos de dichos canales, termina este caso de uso cuando se reproducen los canales actualizados.		
Complejidad	Baja.		
Prioridad	Crítico.		
Precondiciones	Existe conexión con la BD.		
Postcondiciones	El sistema quedó actualizado.		
Flujo de eventos			
Flujo básico <Nombre del flujo básico>			
	Actor		**Sistema**
1.	Se autentica correctamente.		
2.			Actualiza el flujo de streaming de los canales en transmisión así como los datos asociados a estos. Termina el CU.

Relaciones	**CU Incluidos**	NA
	CU Extendidos	NA
Requisitos no funcionales	NA	

CU 7. Registrar reportes de incidencias

Objetivo	Registrar incidencia que ocurran durante la transmisión.		
Actores	Operador de Monitorización: (Inicia) Registrar reportes de incidencias.		
Resumen	El Operador de Transmisión se autentica en el sistema y se comienza a recibir el flujo de los canales.		
Complejidad	Baja.		
Prioridad	Auxiliar.		
Precondiciones	El usuario con rol Operador de Monitorización existe en la Base de Datos. Existe al menos un canal en transmisión.		
Postcondiciones	Se registraron las incidencias.		
Flujo de eventos			
Flujo básico <Nombre del flujo básico>			
	Actor		**Sistema**
1.	Se autentica en el Módulo Monitorización.		
2.			Registra reportes de incidencias en logs. Termina el caso de uso
Relaciones	**CU Incluidos**	NA	
	CU Extendidos	NA	
Requisitos no funcionales	NA		

CU 8. Notificar error de transmisión al transmisor

Objetivo	Enviar un mensaje de error al transmisor.
Actores	Operador de Monitorización: (Inicia) Actualización de transmisión.
Resumen	El CU se inicia cuando el Operador de Monitorización presiona en la opción enviar mensaje. Termina este caso de uso cuando aparece un mensaje de confirmación.
Complejidad	Baja.
Prioridad	Secundario.
Precondiciones	El sistema esté funcional.

Postcondiciones	Envió un mensaje de error al transmisor.	
Flujo de eventos		
Flujo básico <Notificar error de transmisión al transmisor >		
	Actor	**Sistema**
1.	Pulsa el botón “enviar mensaje de error”.	
2.		Muestra un formulario para introducir el IP en el que se encuentra el transmisor.
3.	Inserta el IP.	
4.	Pulsa el botón “OK”.	
5.		Envía una notificación al transmisor. Muestra un mensaje de confirmación. Termina el caso de uso.
Flujosalternos		
Nº Evento <Condición que dio lugar a la extensión>		
	Actor	**Sistema**
1.	Inserta IP inválido.	
1.		No se envía el mensaje de notificación al transmisor. Termina el caso de uso.
Relaciones	**CU Incluidos**	NA
	CU Extendidos	NA
Requisitos no funcionales	NA	

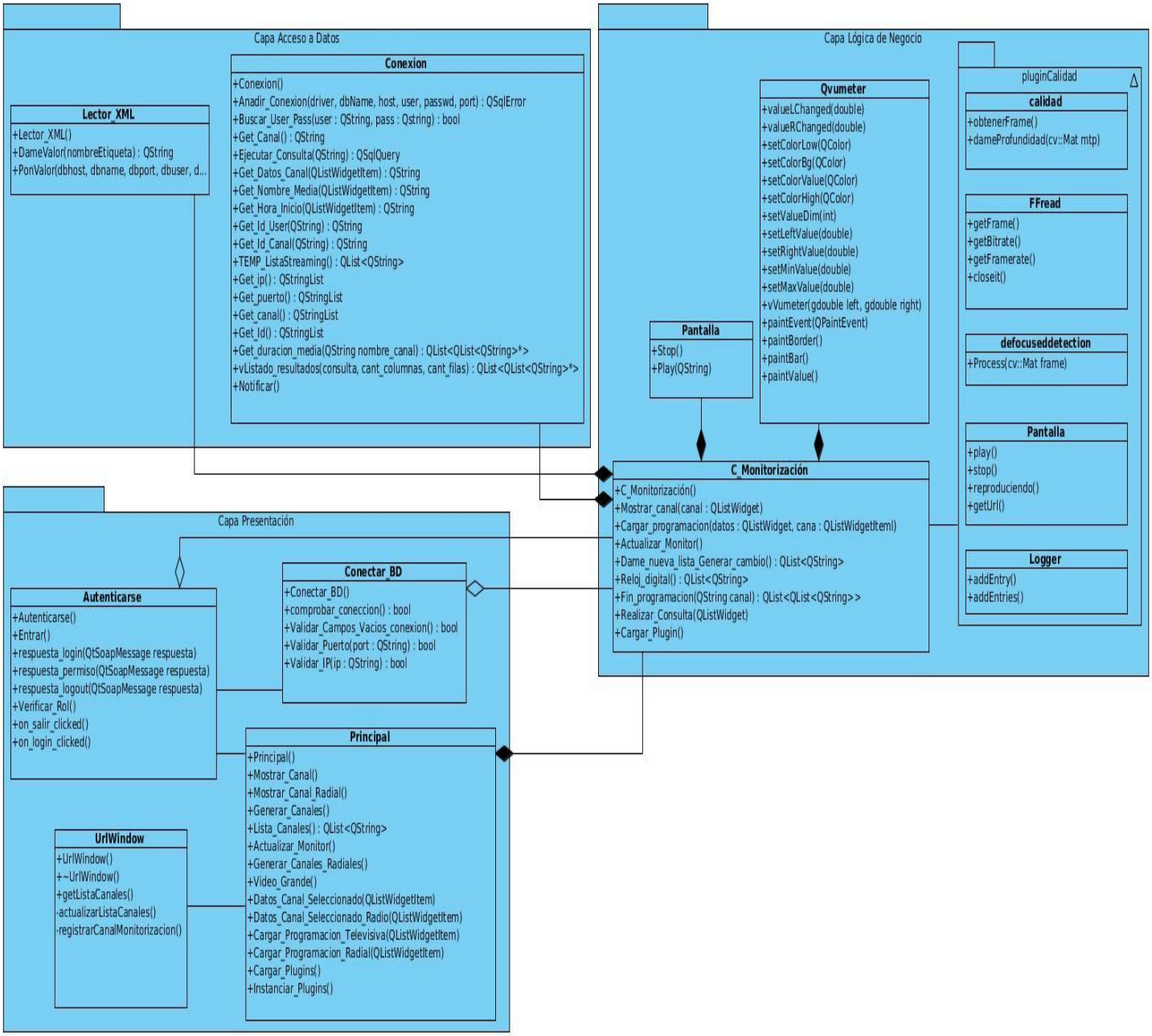
CU 9.Cerrarsesión

Objetivo	Cerrar sesión en el módulo Monitorización.
Actores	Operador de Monitorización: (Inicia) Cierra sesión.
Resumen	El Operador de Transmisión decide cerrar su sesión en el sistema, finalizando de esta manera el caso de uso.
Complejidad	Baja.
Prioridad	Auxiliar.
Precondiciones	El usuario con rol Operador de Monitorización existe en la Base de Datos.
Postcondiciones	La sesión del usuario quedó cerrada.
Flujo de eventos	
Flujobásico<Cerrarsesión>	

	Actor	Sistema
1.	Selecciona la opción cerrar sesión.	
2.		Elimina el token de seguridad creado cuando se registró en el sistema.
3.		Muestra la interfaz de autenticación. Termina el caso de uso.

Relaciones	**CU Incluidos**	NA
	CU Extendidos	NA
Requisitos no funcionales	NA	

Anexo 3. Diagrama de clases del diseño.



Anexo 4. Diseños de casos de prueba:

DCP CUS Autenticar usuario

Descripción general:

El usuario introduce los datos que se solicitan para acceder a la aplicación, estos se verifican y el mismo finaliza dándole los permisos, en caso de poseer los privilegios necesarios.

Condiciones de ejecución:

El usuario con rol Operador de Monitorización existe en la BD.

Escenarios a probar en el Caso de Uso:

Escenario	Descripción	Usuario	Contraseña	Respuesta del sistema	Flujo central
EC 1.1 Autenticar usuario	Se introduce usuario y contraseña.	V V1	V udp://@224,0,0,7:1234	Actualiza la lista de canales televisivos a monitorizar.	Ejecutar la aplicación
EC 1.2 Autenticar usuario. Incorrectamente.	El campo usuario o el campo contraseña tienen valores erróneos.	V I	I V	Muestra un mensaje de error Ejemplo: "Debe especificar la URL del canal".	Ejecutar la aplicación

DCP CUS Gestionar conexión a la BD del módulo Monitorización

Descripción general:

El usuario ejecuta la aplicación y la configuración de conexión a la base de datos es incorrecta. Termina cuando se hayan realizado correctamente todas las configuraciones de la base de datos.

Condiciones de ejecución:

La conexión a la Base de Datos está incorrecta.

Escenarios a probar en el Caso de Uso:

Escenario	Descripción	Usuario	Contraseña	Puerto	Servidor	Database	Respuesta del sistema	Flujo central
EC 1.1	Se introducen	V	V	V	V	V	Actualiza la lista	Ejecutar

Introduce los datos correctamente	datos para gestionar la conexión con la base de datos	V1	cadena	2535	10,0,0,2	base de datos	de canales televisivos a monitorizar.	la aplicación/ Botón "OK"/Botón "Guardar"
EC 1.2 Introduce los datos incorrectamente	Al menos un campo tiene valores erróneos.	I V V V V	V I V V V	V V I V V	V V V I V	V V V V I	Muestra un mensaje de error Ejemplo: "hay campos vacíos". Ejemplo: "dirección IP incorrecta". Ejemplo: "formato del Puerto incorrecto". Ejemplo: "usuario o contraseña incorrectos".	Ejecutar la aplicación/ Botón "OK"/Botón "Guardar"

DCP CUS Mostrar datos de canales

Descripción general:

El caso de uso inicia cuando el operador de monitorización selecciona uno de los canales que se están monitorizando, se muestran todos los datos del mismo.

Condiciones de ejecución:

Los canales planificados tienen medias asociadas. Existe conexión con la base de datos.

Escenarios a probar en el Caso de Uso:

Escenario	Descripción	Lista canales	Respuesta del sistema	Flujo central
EC 1.1 Listar los canales en transmisión.	Se listan los canales que se están monitorizando.	NA	Muestra la lista de canales televisivos que se están monitorizando monitorizar.	Botón "Canales en transmisión"

EC 1,2 Listar datos del canal en transmisión. Correctamente	Se listan los datos de un canal seleccionado.	V	Muestra los datos del canal.	Botón "Datos del canal"
EC 1.3 - Listar datos del canal en transmisión. Incorrectamente.	Se listan los datos de un canal seleccionado.	I	No muestra los datos del canal.	Botón "Datos del canal"
EC 1.4 - - Listar la programación del canal. Correctamente.	Se lista la programación de un canal seleccionado.	V	Muestra la programación del canal.	Botón "Programación del canal"
EC 1.3 -- Listar la programación del canal . Incorrectamente.	Se lista la programación de un canal seleccionado.	I	No muestra la programación del canal.	Botón "Programación del canal"

DCP CUS Registrar reportes de incidencias

Descripción general:

El Operador de Transmisión se autentica en el sistema y se comienza a recibir el flujo de los canales.

Condiciones de ejecución:

El usuario con rol Operador de Monitorización existe en la Base de Datos. Existe al menos un canal en transmisión.

Escenarios a probar en el Caso de Uso:

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Registrar reportes de incidencias.	Se registran en logs, la descripción del flujo cada cierto intervalo de tiempo.	Registra reportes de incidencias en logs.	Ejecutar la aplicación

DCP CUS Actualizar transmisión

Descripción general:

El Operador de Monitorización comienza a monitorizar al actualizarse la base de datos, el sistema actualiza el flujo de los canales y la programación del día, así como los datos de dichos canales. Termina este caso de uso cuando se reproducen los canales actualizados.

Condiciones de ejecución:

Existe conexión con la BD.

Escenarios a probar en el Caso de Uso:

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Actualizar transmisión	Se actualiza el sistema cuando se hagan cambios en la base de datos.	Actualiza el flujo de streaming de los canales en transmisión así como los datos asociados a estos.	Ejecutar la aplicación

DCP CUS Visualizar canal seleccionado

Descripción general:

El Operador de Monitorización desea visualizar a mayor escala el canal que se está transmitiendo. Termina este caso de uso cuando se visualiza dicho canal en el visor, a mayor resolución conjuntamente con la medición de los decibeles de audio de dicho canal.

Condiciones de ejecución:

Existe al menos un canal en transmisión.

Escenarios a probar en el Caso de Uso:

Escenario	Descripción	Canales en reproducción	Respuesta del sistema	Flujo central
EC 1.1	Se introduce	V	Muestra en el	Pulsar en el flujo

Visualizar canal seleccionado. Correctamente.	usuario y contraseña.		visor el flujo del canal seleccionado a mayor escala y se muestra la medición de decibeles de audio.	que se monitoriza.
EC 1.2 Visualizar canal seleccionado. Incorrectamente.	El campo usuario o el campo contraseña tienen valores erróneos.	I	No muestra en el visor el flujo del canal seleccionado a mayor escala, ni se muestra la medición de decibeles de audio.	Pulsar en el flujo que se monitoriza.

DCP CUS Notificar error de transmisión al transmisor

Descripción general:

El CU se inicia cuando el Operador de Monitorización presiona en la opción enviar mensaje. Termina este caso de uso cuando aparece un mensaje de confirmación.

Condiciones de ejecución:

El usuario con rol Operador de Monitorización existe en la BD.

Escenarios a probar en el Caso de Uso:

Escenario	Descripción	IP	Respuesta del sistema	Flujo central
EC 1.1 Notificar error de transmisión al transmisor	Se introduce usuario y contraseña.	V 10,0,0,25	Envía una notificación al transmisor. Muestra un mensaje de confirmación.	Botón "enviar mensaje de error"/ Botón "OK"
EC 1.2 Notificar error de transmisión al transmisor.	El campo usuario o el campo contraseña	I	No envía una notificación al transmisor.	Botón "enviar mensaje de error"/ Botón "OK"

Incorrectamente.	tienen valores erróneos.	abs		

DCP CUS Cerrar sesión

Descripción general:

El Operador de Transmisión decide cerrar su sesión en el sistema, finalizando de esta manera el caso de uso.

Condiciones de ejecución:

El usuario con rol Operador de Monitorización existe en la Base de Datos.

Escenarios a probar en el Caso de Uso:

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Cerrar sesión	Se cierra la sesión por la que se autenticó el usuario.	Muestra la interfaz de autenticación.	Botón cerrar.

1