Universidad de las Ciencias Informáticas

“Módulo de acceso y visualización de contenido multimedia proveniente de cámaras IP, para el Sistema Domótico del CEDIN.”

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

Autor: Lidia Mercedes Moya García

Tutor: Ing. Jordanis Viltres Chávez

Co-Tutor: Ing. Rosabel Laches Hernández

Co-Tutor: MsC. Luis Manuel Vidal Piña

La Habana, 2018

“Año 60 del triunfo de la revolución”

Declaración de autoría

Declaro ser el autor del presente trabajo de diploma y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales del mismo, con carácter exclusivo.

Para que así conste firmo el presente a los____ días del mes de _____ del año _____.

_____

Firma del Autor

Lidia Mercedes Moya García

_____	_____	_____
Firma del Tutor	Firma del Co-tutor	Firma del Co-tutor
Ing. Jordanis Viltres Chávez	Ing. Rosabel Laches Hernández	MsC. Luis Manuel Vidal Piña

Datos de contacto

Tutor: Ing. Jordanis Viltres Chávez

Título de la especialidad de graduado: Ingeniero en Ciencias Informáticas

Instituto donde se graduó: Universidad Tecnológica de La Habana (CUJAE)

Correo electrónico: jviltres@uci.cu

Co-Tutor: Ing. Rosabel Laches Hernández

Título de la especialidad de graduado: Ingeniero en Ciencias Informáticas

Instituto donde se graduó: Universidad de las Ciencias Informáticas (UCI)

Correo electrónico: rlaches@uci.cu

Co-Tutor: MsC. Luis Manuel Vidal Piña

Título de la especialidad de graduado: Máster en Ciencias Informáticas

Instituto donde se graduó: Universidad de las Ciencias Informáticas (UCI)

Correo electrónico: lmvidal@uci.cu

Dedicatoria

Dedico el presente trabajo a mi madre Mercedes García Pita.

Agradecimientos

Quisiera comenzar agradeciendo a los mayores responsables de que haya culminado satisfactoriamente este trabajo de diploma, ellos son mis tutores Viltres, Vidal y Rosabel. Gracias por su apoyo y sacrificio.

A mis padres Mercedes y Roberto por apoyarme siempre, y principalmente a mi madre por ser esa persona que cuando quería dejarlo todo y darme por vencida, nunca me lo permitió, los quiero.

A mi hermana Rachel por ser siempre sincera conmigo, te quiero.

A mi novio Zahiro, por estar junto a mi cuando más lo necesité, gracias amor de todo corazón.

A toda mi familia por apoyarme incondicionalmente.

A la Universidad y a la Revolución por brindarme la oportunidad de estudiar y ser alguien para la vida.

A mis profesores quienes me formaron a lo largo de esta carrera.

A los miembros del tribunal por sus recomendaciones, sugerencias y críticas oportunas que permitieron aumentar mi formación como profesional.

A todos mis amigos en general, gracias por su ayuda.

Resumen

El Centro de Informática Industrial (CEDIN) perteneciente a la Universidad de las Ciencias Informáticas (UCI) tiene como objetivo desarrollar productos y servicios informáticos de automatización que cumplan con las necesidades y expectativas de los clientes. Este centro está desplegado en líneas de desarrollo, una de ellas es la línea de Sistemas Embebidos. En esta línea se han desarrollado varias aplicaciones que permiten dar respuesta a las necesidades derivadas de los cambios sociales y las nuevas tendencias de nuestra forma de vida, entre los que se encuentra el Sistema de Medición y Control de Procesos de Baja y Mediana Complejidad AREX. El objetivo de la presente investigación consiste en el desarrollo de un módulo para el Sistema Domótico del CEDIN. Este módulo se implementa para visualizar video mediante el acceso a cámaras IP. La investigación comienza con un estudio de los principales conceptos enmarcados dentro del proceso de la fundamentación teórica. Se lleva a cabo un estudio de las tecnologías y herramientas para determinar cuáles utilizar en el diseño e implementación de la aplicación, por ejemplo: la plataforma *QT* como tecnología de desarrollo, el entorno de desarrollo *QT Creator*, como lenguaje de programación *C++,* se empleó como lenguaje de modelado *UML* y la herramienta *Visual Paradigm* para el modelado de los diagramas del sistema. Es utilizada la biblioteca *VLC* para transmitir y recibir flujo de datos a través de la red, y además se seleccionó como protocolos de comunicación *HTTP* y *RTSP*. Se define la arquitectura Cliente-Servidor, así como los requisitos funcionales y no funcionales que intervienen en la aplicación. Se concluye el proceso de desarrollo con la ejecución de las pruebas que garantizan la calidad y el correcto funcionamiento del *software*.

Palabras clave: Sistema Domótico, QT, cámaras IP.

Abstract

The Center for Industrial Informatics (CEDIN) belonging to the University of Computer Science (UCI) aims to develop computer automation products and services that meet the needs and expectations of customers. This center is deployed in lines of development, one of them is the line of embedded systems. In this line, several applications have been developed that allow responding to the needs derived from social changes and new trends in our way of life, among which is the System of Measurement and Control of processes of low and Medium Complexity AREX. The objective of this research is the development of a module for the Domotic System of CEDIN. This module is implemented to view video by accessing IP cameras. The investigation begins with a study of the main concepts framed within the process of the theoretical foundation. A study of the technologies and tools is carried out to determine which ones to use in the design and implementation of the application, for example: the QT platform as a development technology, the QT Creator development environment, as a C++ programming language, was used as a UML modeling language and the Visual Paradigm tool for modeling system diagrams. The vlc library is used to transmit and receive network flows, and it was also selected as communication protocols: the HTTP and RTSP protocols. The Client-Server architecture is defined, as well as the functional and non-functional requirements that intervene in the application. The development process is concluded with the execution of the tests that guarantee the quality and correct operation of the software.

Keywords: Domotic system, QT, IP cameras.

Índice

Índice de tablas

Índice de figuras

Introducción

El creciente desarrollo de la tecnología ha provocado que el control de procesos de manera automática constituya un elemento primordial para instituciones u organizaciones, razón por la cual se hace necesaria la utilización de diferentes dispositivos que contribuyan a garantizar una mayor eficiencia y calidad del proceso productivo (Almeida, 2015). Las tecnologías de información y comunicación (TIC) juegan un papel relevante en la sociedad actual. A medida que han transcurrido los años, las tecnologías han llevado un desarrollo acelerado, siendo, cada vez más necesario el uso de medios tecnológicos en las organizaciones. Estos medios a su vez necesitan ser controlados para preservar su integridad como activos de una organización, siendo así necesario el uso de equipos de vigilancia. Como parte de estos equipos de vigilancia están las cámaras, las cuales gracias a su continuo avance han sido integradas dentro de las instituciones públicas y privadas, así como en muchos hogares del mundo (Castillo, 2015). Durante mucho tiempo la televisión ha sido uno de los principales medios de comunicación en todo el mundo, ya que permite llevar al espectador una variada gama de contenidos audiovisuales de manera ágil. Desde su inicio se utilizó tecnología analógica para su captura, transmisión, almacenamiento y reproducción; esta tecnología está presente desde hace muchos años, y ha crecido con la televisión convencional. Un sucesor natural de esta ha sido la tecnología digital que junto a la masificación de los computadores han permitido la trasmisión digital de contenido multimedia. Con el transcurso de los años se ha experimentado un considerado avance en la velocidad de transmisión de las redes de computadoras, lo cual, apoyado en las actuales capacidades de procesamiento de los ordenadores, permite que las aplicaciones de video y sonido sobre redes IP sean más populares y más fáciles de implementar que hace algunos años atrás (Paez, 2003). Un elemento que ha influido en el incremento de este tipo de aplicaciones es la aparición de la tecnología *streaming*, permitiendo la transferencia de contenido audiovisual por la red desde un servidor hacia sus clientes, con la característica de que es posible visualizar el contenido en la medida que el flujo es recibido. Antes de que la tecnología "streaming" apareciera, la difusión de contenido multimedia a través de *internet* necesariamente implicaba tener que descargar completamente el archivo contenedor al disco duro local. Como los archivos de audio y especialmente los de video tienden a ser de tamaño considerable, su descarga y acceso como paquetes completos se vuelve una operación muy lenta, sin embargo, con la tecnología del streaming, el tiempo de espera es mínimo (Torres, 2009). Para poner a disposición de los clientes aplicaciones que usen el flujo de datos (*Streaming*), es necesario realizar una serie de operaciones sobre los datos que se van a transmitir. Para que sean capaces de enviar información audiovisual a través de una

red de datos con mucha rapidez. En la actualidad existen muchas aplicaciones que hacen uso de este recurso, por ejemplo: el sitio web *http://www.youtube.com* que permite a los usuarios compartir vídeos digitales a través de *Internet* (Torres, 2009). Actualmente en el mercado internacional existen un conjunto de soluciones que responden a todo tipo de necesidades de automatización, pero presentan como inconveniente que son soluciones, en su mayoría, altamente costosas, como también que son *softwares* privativos por lo que se incurre en altos gastos por concepto de compra de licencias. Es por ello que se dificulta el empleo de este tipo de aplicaciones, ya que muchas veces no se cuenta con el presupuesto necesario para ejecutar un contrato de esa índole. El gobierno cubano, desde hace algunos años, ha dictado como una política estratégica el desarrollo de soluciones nacionales, que por una parte contribuyan a la sustitución de importaciones, debido a que la automatización de los locales, dígase centros de trabajo u organizaciones en nuestro país carecen de sistemas que tributen a garantizar el ahorro energético, la seguridad y el confort que son primordiales para una empresa (Chávez, 2015). La domótica permite dar respuesta a los requerimientos que plantean estos cambios sociales facilitando el diseño de hogares más personales. Se llama domótica a los sistemas capaces de automatizar una vivienda o edificación, aportando servicios de seguridad, gestión energética, comunicación y bienestar, y pueden estar integrados por medio de redes de comunicación, controladas desde cualquier ubicación dentro o fuera del hogar. Se podría definir como la integración de la tecnología en el diseño inteligente de un recinto cerrado (Varona, 2012). Se considera un sistema domótico o inteligente a aquellos sistemas que bajo una misma central gestionan todos los servicios de una vivienda para el máximo aprovechamiento de las instalaciones y para mantener la seguridad de los centros se pueden instalar cámaras de vigilancia que garantizan la integridad de los medios contabilizados y esto asegura que no haya pérdidas en la empresa.

El Centro de Informática Industrial (CEDIN) perteneciente a la Universidad de las Ciencias Informáticas (UCI) tiene como objetivo desarrollar productos y servicios informáticos de automatización que cumplan con las necesidades y expectativas de los clientes. Este centro está desplegado en líneas de desarrollo, una de ellas es la línea de Sistemas Embebidos. En esta línea se han desarrollado varias aplicaciones que permiten dar respuesta a las necesidades derivadas de los cambios sociales y las nuevas tendencias de nuestra forma de vida, entre los que se encuentra el Sistema de Medición y Control de Procesos de Baja y Mediana Complejidad AREX que puede ser usado para la domótica y facilita una gestión eficiente de comunicación entre el usuario y el sistema. El sistema es capaz de generar seis tipos de alarmas para monitorizar variables en los procesos. Su arquitectura distribuida permite su aplicación en espacios pequeños o grandes. Su extensibilidad permite la adición de nuevos tipos de componentes gráficos a mostrar en las interfaces de

usuario, además de diversos protocolos para la comunicación con dispositivos físicos. Incorpora además el almacenamiento de datos sobre variables y alarmas, para su visualización en gráficas de tendencia o tiempo real. El sistema consume pocos recursos de *hardware*, es multiplataforma, hace uso de protocolos estándar ampliamente utilizados, y permite el control de alta velocidad sobre los procesos. Este sistema es aplicable en la domótica para diferentes escenarios, implementación de sistemas de alarma, control remoto de dispositivos, control de procesos industriales, etc. Sin embargo, actualmente existen disímiles aplicaciones en el ámbito de la domótica que integran cámaras de vigilancia para reforzar la seguridad de la empresa u organización, permitiendo observar lo que acontece en tiempo real. En el Sistema Arex no es posible visualizar este contenido multimedia obtenido de varias cámaras porque es un sistema carente de video-vigilancia, característica que integran la mayoría de los Sistemas Domóticos.

La **situación problemática** anteriormente planteada permite formular el siguiente **problema científico**:

¿Cómo acceder y visualizar el contenido multimedia proveniente de cámaras IP para el Sistema Domótico del CEDIN?

A partir del problema que se presenta se define como **objeto de estudio**: Sistemas y herramientas para el acceso, conversión y retransmisión del flujo de video y audio de cámaras de vigilancia, especificándose en el **campo de acción**: Sistemas y herramientas para el acceso, conversión y retransmisión del flujo de video y audio de cámaras IP mediante los protocolos *HTTP* y *RTSP*.

Para dar solución al problema científico se toma como **objetivo general:** Desarrollar un módulo que le permita obtener el flujo de video en tiempo real de cámaras IP para el Sistema Domótico.

Para darle cumplimiento al objetivo general definido, se elaboran las siguientes **tareas de investigación**:

- Elaborar el marco teórico de la investigación a partir del estudio del estado del arte sobre el tema propuesto.
- Identificar las herramientas, metodologías y artefactos a utilizar durante el desarrollo del trabajo.
- Identificar requisitos funcionales y no funcionales.
- Describir la propuesta de solución.
- Definir la arquitectura para la elaboración del diseño de la aplicación.
- Diseñar y aplicar las pruebas a la aplicación implementada para verificar su correcto funcionamiento.

Para darle cumplimiento a las tareas de investigación, fueron utilizados varios métodos científicos

(Morales, 2017), entre los que se destacan como **métodos teóricos:**

- **Analítico-Sintético:** se utilizó para identificar los principales fundamentos teóricos relacionados con la domótica, a través del estudio de la documentación recopilada.
- **Modelación:** mediante este método se realizarán los modelos que especifican las características que va a desarrollar la aplicación y que permitirán una mejor comprensión a los desarrolladores para una posterior implementación.

Y como **métodos empíricos**:

- **Experimental:** se empleó en la realización de pruebas, donde se crean las condiciones propicias para verificar el correcto funcionamiento del módulo desarrollado en la presente investigación.
- **Entrevista:** aplicada a especialistas con experiencia en el desarrollo de la aplicación con el fin de definir las funcionalidades que ésta tendría.

El presente documento está estructurado en tres capítulos:

En el **Capítulo 1** se muestran características, conceptos y definiciones de los elementos relacionados con flujo de video. Además, se analizaron algunos sistemas ya existentes. Se describen las herramientas y las tecnologías utilizadas, y se realiza un estudio acerca de la metodología empleada en el desarrollo del proyecto. Luego de cada epígrafe se muestra una selección precisamente de lo va a ser utilizado para el desarrollo de esta aplicación.

En el **Capítulo 2** se realiza un estudio de la arquitectura y se detallan las principales funcionalidades que posee el sistema, para una mayor claridad de lo que se desea desarrollar. Se mencionan los requisitos no funcionales para el usuario final, ya que se deben tener en cuenta para la correcta utilización del sistema desarrollado. Se describe la propuesta de solución y las principales clases que la componen. Además, se analizan patrones de diseño empleados en el proyecto.

En el **Capítulo 3** se describe la implementación del sistema dándole paso a la construcción de la solución propuesta, determinando las tareas de ingeniería que se desarrollan, el diagrama de despliegue que se utiliza y al final, se realizan las pruebas pertinentes de *software* al sistema para garantizar su correcto funcionamiento y luego la integración a su proyecto final.

Capítulo 1 Fundamentación teórica

1.1 Introducción

En este capítulo se tiene como objetivo realizar una introducción al tema de la investigación, haciendo referencia a un conjunto de conceptos asociados al dominio del problema facilitando así el rápido entendimiento del tema y logrando una mayor familiarización con los términos más comunes. Se realiza un estudio acerca de los conceptos relacionados con *streaming* de video y cámaras de vigilancia IP, y se caracterizan además las principales herramientas y tecnologías que serán utilizadas en el desarrollo de la aplicación.

1.2 Conceptos asociados al dominio del problema

A continuación se exponen una serie de conceptos que serán utilizados a lo largo de la investigación y que van a servir de ayuda para lograr una mejor comprensión de la temática abordada.

1.2.1 Domótica

La domótica es el conjunto de sistemas que automatizan las diferentes instalaciones de una vivienda o edificación de cualquier tipo, aportando servicios de gestión energética, seguridad, bienestar y comunicación, y pueden estar integrados por medio de redes interiores y exteriores de comunicación, cableadas o inalámbricas, y cuyo control goza de cierta ubicuidad, desde dentro y fuera del hogar. El principal objetivo de estas tecnologías es la mejora de la calidad de vida incrementando la comodidad de las personas, sin embargo, últimamente se está imponiendo como una tendencia en el mundo de la ecología, y es que la domótica se puede implementar en distintos ámbitos del hogar. Desde los típicos controladores de la calefacción hasta sistemas de gestión del agua, iluminación, gas o incluso sistemas automáticos de persianas y toldos basados en sensores de incidencia solar y temperatura. Todos ellos para conseguir ahorrar agua, gas y, sobre todo, electricidad. De hecho, según los datos manejados por el IDAE, se puede llegar a ahorrar más del 50 por ciento del consumo eléctrico de un hogar en función de múltiples variables. De ahí que, en muchas ocasiones la domótica no se observe como un gasto a la hora de realizar una reforma en la vivienda, sino como una verdadera inversión (Fortis, 2010).

1.2.2 Cámaras de Vigilancia

Una cámara es un dispositivo que sirve para registrar imágenes estáticas o en movimiento. Existe una gran variedad de cámaras de seguridad en cuanto a modelo y forma, pero básicamente existen dos tipos, las cámaras de seguridad analógicas y las de protocolo de *internet* (Marulanda, 2008). Las cámaras de seguridad analógicas son las que siempre se han usado en Circuito Cerrado de Televisión (CCTV). La imagen proviene de manera analógica y su principal característica es la necesidad de conectar su cableado, se requiere el uso de cable Coaxial o Par Trenzado (Diaz, 2016). Las cámaras de protocolo de *internet* (IP) también conocidas como cámaras *Web* o de *Red*, son videocámaras diseñadas para enviar las señales de audio y video, a través de *Internet* desde un navegador o a través de concentrador en una Red Local (Valeriano, 2016).

1.2.3 Streaming de video

La palabra *streaming* (significa transmisión) está formada a partir de *"stream"* cuyo significado es algo que se asemeja a una corriente de flujo. Más el sufijo *"-ing"* que indica la acción en progreso o que continúa (Zenith, 2015). La transmisión es la distribución digital del contenido multimedia a través de una red de computadoras (Huri, 2017). Se logra mediante fragmentos enviados secuencialmente a través de la red. Su principal ventaja es que elimina la barrera de tener que descargar el contenido completo para poder visualizarlo. El *streaming* de audio o vídeo se descodifica y reproduce inmediatamente o cuando se han transferido los datos suficientes para iniciar su reproducción. Los equipos de *streaming* de videos juegan un papel relevante en este proceso de transmisión (Ortiz, 2006).

1.3 Métodos para realizar *streaming* de video

Existen varias formas de realizar una transmisión, la solución adecuada para un evento puede no ser la misma para otros. El módulo implementado debe tener la capacidad de codificar o convertir el video en tiempo real, de otra forma dejaría de ser transmisión en tiempo real (Diez, 2005). Hay gran variedad de aplicaciones que realizan *streaming* de video, algunas de ellas son las siguientes:

GStreamer es prácticamente el *framework* multimedia más aplicado en *GNU/Linux*. Con *GStreamer podemos* mezclar diferentes tipos de medios como pueden ser señales de audio, video, imágenes y subtítulos. Permite aplicar diferentes tipos de filtros tanto de audio como de video, también brinda la posibilidad de transmisión y recepción de datos en tiempo real, la codificación y decodificación en diferentes

formatos y la reproducción de archivos multimedias (Gómez, 2013).

WebRTC, también conocido como *Web Real-Time Communications* (Comunicación Web en Tiempo Real por su significado) es un proyecto de código abierto promovido por *Google* y *Mozilla,* y permite comunicaciones en tiempo real sin plugins a través de una API Javascript. Facilita las aplicaciones de llamadas de voz, chat de video y compartimiento de archivos entre navegadores (Suyama, 2015).

La biblioteca ***VLC es*** una interfaz de programación de aplicaciones libre multiplataforma desarrollada por el proyecto *VideoLan*. Esta biblioteca permite crear una amplia gama de aplicaciones. Es posible añadirla a cualquier aplicación que desee contar con la capacidad de transmitir datos *streaming* a través de redes y convertir archivos multimedia en formatos distintos al original, es capaz de difundir y recibir flujos de video e información usando los protocolos, *RTP, RTSP, HTTP*, entre otros, además de otras funcionalidades relacionadas con el trabajo con multimedia y entornos de red. *VLC* puede ser utilizado como servidor y como cliente para transmitir y recibir flujos de red. Es capaz de transmitir todo lo que puede leer (Novak, 2004). *VLC* es gratuito, fácil de usar y funciona en todos los sistemas operativos (Gonzalez, 2008). Durante mucho tiempo la biblioteca *VLC* ha sido la primera opción para muchos usuarios, principalmente por ser una herramienta universal (Dommermuth, 2017).

Atendiendo al anterior análisis realizado, se selecciona la biblioteca *VLC* para el desarrollo del módulo, pues es una biblioteca capaz de transmitir datos en *streaming* a través de redes. Después de seleccionar la biblioteca *VLC* como la herramienta adecuada para realizar el envío en vivo del flujo de video, se abarcan una serie de soluciones existentes.

1.4 Análisis de otras soluciones existentes

Actualmente se cuenta con una gran variedad de servidores *streaming* tanto de *software* libre como propietario, además de una mayor cantidad de funcionalidades que garantizan la calidad en el servicio que brindan, constituyendo esto una necesidad para poder alcanzar la soberanía tecnológica que defiende el país a la hora del desarrollo de aplicaciones que hagan uso de este recurso. Tanto en Cuba como en el mundo existen algunas soluciones que se apoyan en los servidores *streaming* para poder difundir el contenido audiovisual, a continuación, se describen algunas de ellas:

YouTube es un sitio *web* que permite a los usuarios subir, bajar, ver y compartir vídeos digitales a través de *internet*. Para ver los vídeos o enviarlos a otras personas no es necesario registrarse, aunque sí para colocarlos en la página. Es uno de los servicios popularmente más conocido y usado por la comunidad

social de *Internet*. Fue fundado en febrero de 2005, aunque actualmente es propiedad de *Google*. Es muy popular gracias a la posibilidad de alojar vídeos personales de manera sencilla. Los servidores durante el proceso de “subida” del video se encargan de convertir este en dicho formato. Los formatos en los que se envía el vídeo son: *MPEG, AVI, MOV*, los utilizados por videocámaras y cámaras integrada en los teléfonos móviles y algunos otros. Para poder difundir toda la información que se encuentra alojada en el sitio, utiliza muchos servidores, dentro de ellos se puede destacar el uso del *Darwin Streaming Server* para transmitir videos a móviles (Aquiles, 2017).

PRIMICIA, es un producto informático creado en el año 2005 por la Universidad de las Ciencias Informáticas (UCI) en el Polo Productivo “Video y Sonido Digital” de la antigua Facultad 9. El canal informativo muestra de forma automática ciclos de noticias constantes y repetitivos condicionados por las informaciones publicadas para determinados períodos de tiempo, logrando integrar en sus transmisiones informaciones textuales, imágenes, sonido y video. Durante las transmisiones permite la reproducción de un fondo musical que puede ser personalizado según la noticia que se muestra, además es posible la utilización de cintillos informativos o infocintas que permitan el adelanto o emisión de breves informaciones de carácter relevante o promocional. Como información adicional a la noticia es posible mostrar la fecha y hora, tiempo restante de la noticia o pantalla y titular de la próxima noticia. El sistema permite además la transmisión de señales de video externas, tales como filmaciones en vivo o transmisiones de otras televisoras (Torres, 2009).

El Centro perteneciente a la facultad 6 de la UCI, **GEYSED**, por sus siglas Geo-informática y Señales Digitales, adscrito a la Facultad de Ciencias y Tecnologías Computacionales, desarrolla productos, servicios y soluciones informáticas en el campo del procesamiento de Señales Digitales y la Geo-informática, contribuyendo a la formación integral de profesionales que respondan a las necesidades del progreso científico técnico y socioeconómico. Entre sus funciones se encuentra: diseñar soluciones integrales de acuerdo al área temática y necesidades del cliente, desarrollar soluciones informáticas a la medida para la informatización de la UCI, la sociedad cubana, y para la exportación; prestar servicios informáticos asociados a las soluciones y productos que entrega a los clientes; ofrecer soporte técnico y seguimiento a los clientes, en cuanto a productos y soluciones desarrolladas se refiere. Uno de los proyectos de este centro es: SURIA, el cual desarrolla productos para abastecer la video-vigilancia en la universidad, por ende, la información y el código que implementan son confidenciales, ya que un uso indebido puede ocasionar problemas a su seguridad informática (Toledo, 2017).

La revisión y estudio de las soluciones existentes evidenció la no existencia de una solución que pueda integrarse con el Sistema Arex. Las soluciones analizadas son aplicaciones propietarias, por lo tanto, son altamente dependientes de estas y además no son compatibles con este Sistema Domótico.

1.5 Protocolos de comunicación

Un protocolo es un método estándar que permite la comunicación entre procesos, es un conjunto de reglas y procedimientos que están obligadas a cumplir todas las máquinas y programas que intervienen en una comunicación de datos, las cuales deben respetarse para el envío y la recepción de datos a través de una red, ya que sin estas la comunicación resultaría caótica y por lo tanto imposible. A pesar de que los ficheros multimedia son enormes, normalmente son enviados en flujos de datos a través de la red. Los flujos de datos se fragmentan en paquetes con un determinado tamaño para la transmisión entre los servidores y los clientes. Un cliente puede escuchar el primer paquete, descomprimir el segundo mientras recibe el tercero. De esta forma los usuarios pueden empezar a disfrutar de la multimedia sin esperar al final de la transmisión (Tanenbaum, 2003). Existen varios protocolos que han sido normalizados para permitir la comunicación entre los servidores *streaming* y los ordenadores cliente. Los protocolos de comunicaciones definen las normas que posibilitan que se establezca una comunicación entre varios equipos o dispositivos, ya que estos equipos pueden ser diferentes entre sí. Entre los principales protocolos de comunicación se encuentran los siguientes:

- Protocolo de red para transmitir los videos en tiempo real (*RTSP* significa *Real Time Streaming Protocol*)
- Protocolo de Transferencia de HiperTexto (*HTTP* significa *Hypertext Transfer Protocol*)

1.5.1 Protocolo de red para Transmitir videos en Tiempo Real (RTSP)

El protocolo *RTSP* proviene de las siglas en inglés *Real Time Streaming Protocol* y como su nombre indica es un protocolo de flujos de datos en tiempo real. Su función consiste en establecer y controlar uno o muchos flujos sincronizados de datos (pudiendo ser audio o vídeo). Fue desarrollado por *Progressive Networks, Netscape Communications* y la Universidad de Colombia ante la necesidad de crear un sistema de transmisión de los medios de comunicación a través de redes de internet. Es un sistema muy práctico para los usuarios ya que permite controlar la reproducción multimedia de manera que pueden reproducir, pausar, avanzar rápido, rebobinar un vídeo en directo o grabar guardando el flujo en un servidor Web (Bustio,

2018). Una cámara IP es capaz de transmitir videos en vivo en reproductores multimedia compatibles con *RTSP* como *VLC Media Player y QuickTime*. Para ver transmisiones de video en vivo en un reproductor multimedia habilitado para *RTSP*, como *VLC*, los usuarios necesitan establecer el enlace al obtener primero la URL de la secuencia *RTSP* de sus cámaras IP y configurar el reproductor multimedia. Las diferentes cámaras IP vienen con diferentes URL y muchas direcciones URL de cámara IP son proporcionadas por los fabricantes o pueden encontrarse en línea (Zamudio, 2017).

1.5.2 Protocolo de Transferencia de HiperTexto (HTTP)

El protocolo *HTTP* por sus siglas en inglés *Hypertext Transfer Protocol*, fue propuesto por Tim BernersLee en 1945, atendiendo a las necesidades de un sistema global de distribución de información como el *World Wide Web*. Desde el punto de vista de las comunicaciones, está soportado sobre los servicios de conexión y funciona de la misma forma que el resto de los servicios comunes de los entornos *UNIX*: un proceso servidor escucha en un puerto de comunicaciones *TCP* (por defecto, el 80), y espera las solicitudes de conexión de los clientes *Web*. Una vez que se establece la conexión, el protocolo se encarga de mantener la comunicación y garantizar un intercambio de datos libre de errores. La comunicación se puede simplificar en dos pasos: el navegador manda una petición *HTTP* y solicita un archivo, y el servidor responde con la información, que es descifrado por el navegador (Zamudio, 2017). Todas las operaciones pueden adjuntar un objeto o recurso sobre el que actúan; cada objeto *Web* es conocido por su URL. Su principal objetivo es la transferencia de archivos entre un cliente, por ejemplo, un navegador *web*, y un servidor, como puede ser un ordenador, siguiendo el esquema de cliente-servidor. Es el protocolo usado en cada transacción de la *World Wide Web* (WWW) haciendo que la información de las páginas web pueda verse en los navegadores (Bustio, 2018).

1.6 Tecnologías y herramientas usadas para el desarrollo del software

Para el desarrollo del trabajo de diploma se emplearon un conjunto de tecnologías y herramientas de *software* libre de las cuales se expone a continuación una breve descripción:

1.6.1 Marco de trabajo QT

Qt es una plataforma de desarrollo que incluye clases, bibliotecas y herramientas para la producción de aplicaciones de interfaz gráfica en C++ entre otras funcionalidades. Es un *software* libre, multiplataforma y

de código abierto. QT dispone de una amplia gama de herramientas que facilitan entre otras cosas la creación de formularios, botones y ventanas de diálogo con el uso del ratón. Además, provee soporte para la internacionalización de su contenido (Hernández, 2010).

1.6.2 IDE de desarrollo QT Creator

Qt Creator es un entorno de desarrollo (IDE) multiplataforma muy completo. Fue de gran ayuda para agilizar el desarrollo gracias a su completamiento, generación de código, búsquedas avanzadas y reemplazamiento de frases (Molkentin, 2010).

1.6.3 Lenguaje de programación C++

Es un lenguaje imperativo orientado a objetos derivado de C. En realidad, un superconjunto de C, que nació para añadirle cualidades y características de las que carecía. Este lenguaje tiene una alta potencia para la programación a bajo nivel. Se le han añadido elementos que le permite un estilo de programación con alto nivel de abstracción. Entre las grandes ventajas de C++ se pueden mencionar la variedad tipos de datos, clases, plantillas, mecanismo de excepciones, sistema de espacios de nombres, sobrecarga de operadores, referencias y operadores para manejo de memoria persistente. Es orientado a objetos, lo que permite estructurar mejor las ideas y encapsular los conceptos de la realidad en clases (Stroustrup, 2007).

1.6.4 Lenguaje de modelado UML

El Lenguaje de Modelado Unificado o *Unified Modeling Language* (UML por sus siglas en inglés), es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. Ofrece un estándar para describir un plano del sistema, incluyendo aspectos conceptuales tales como procesos de negocio y funcionalidades, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables. Permite modelar sistemas utilizando conceptos orientados a objetos y crear un lenguaje de modelado que puede ser utilizado tanto por humanos como por máquinas. Es un lenguaje de modelado para especificar o para describir métodos o procesos (Orallo, 2017).

1.6.5 Lenguaje de modelado UML 2.1

Es un lenguaje para especificar, construir, visualizar y documentar los artefactos de un sistema de *software* orientado a objetos (OO). Para realizar los modelos del sistema propuesto se hará uso del Lenguaje

Unificado de Modelado 2.1 (UML, por sus siglas en inglés, *Unified Modeling Language*), porque tiene una notación gráfica muy expresiva que permite representar en mayor o menor medida todas las fases de un proyecto informático: desde el análisis con los casos de uso, el diseño con los diagramas de clases, objetos, hasta la implementación y configuración con los diagramas de despliegue (Orallo, 2017).

1.6.6 Herramienta de modelado Visual Paradigm

La herramienta CASE utilizada fue el *Visual Paradigm* para el modelado de la solución. La decisión está basada en las facilidades que brinda esta aplicación de representar los diagramas de clases, generar código desde diagramas y diagramas desde el código, facilita el modelado de UML, ya que proporciona herramientas específicas para ello. También permite la estandarización de la documentación, ya que la misma se ajusta al estándar soportado por la herramienta. Controla que el modelado con UML sea correcto. Además, es un *software* multiplataforma. Facilita la reutilización, ya que es una herramienta centralizada donde se encuentran los modelos utilizados para otros proyectos. Permite generar código de forma automática, reduce los tiempos de desarrollo y evita errores en la codificación del *software* (James Rumbaugh, 2013). Gracias a esta herramienta se logró modelar los diagramas del sistema de manera ágil.

1.7 Metodologías de desarrollo de software

Las metodologías de desarrollo consisten en una colección de documentación formal referente a los procesos, las políticas y los procedimientos que intervienen en el desarrollo del *software*. Constituye una guía que nos indica cómo actuar y qué hacer para lograr el resultado esperado en una investigación. La metodología que se utilizará debe estar en correspondencia con las características que tenga el proyecto que se pretende desarrollar. La finalidad de una metodología de desarrollo es garantizar la eficacia y la eficiencia en el proceso de generación de *software*. En los últimos años se han desarrollado dos corrientes fundamentales en lo referente a los procesos de desarrollo, las metodologías ágiles y las tradicionales. La diferencia fundamental entre ambos tipos de metodologías consiste en que las tradicionales pretenden lograr sus objetivos mediante el orden y la documentación, mientras que las ágiles intentan mejorar la calidad del *software* mediante la comunicación directa e inmediata entre las personas que intervienen en el proceso (Pressman, 1997).

1.7.1 Proceso Unificado de Desarrollo (RUP)

Es una metodología de desarrollo de *software* que está basada en componentes e interfaces bien definidas, y junto con el UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. Es un proceso que puede especializarse para una gran variedad de sistemas de *software*, en diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyecto (Hidalgo, 1997).

Por sus características se clasifica como un método pesado, pues requiere de equipos de desarrollo con un gran número de personas para su uso, además, un cambio en las etapas de vida del sistema incrementaría notablemente el costo (Hidalgo, 1997).

1.7.2 Metodología eXtreme Programming (XP)

Desarrollada por *Kent Beck*, la metodología XP se clasifica como ágil. Consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar el éxito del proyecto. Está centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de *software*, promoviendo el trabajo en equipo y propiciando un buen clima de trabajo. XP se basa en realimentación continua y comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. Es especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, donde existe un alto riesgo técnico. Intenta reducir la complejidad del *software* por medio de un trabajo orientado directamente al objetivo, basado en las relaciones interpersonales y la velocidad de reacción (Letelier, 2006).

1.7.3 Proceso Unificado Ágil (AUP) con el modelo CMMI-DEV v 1.3

El Proceso Unificado Ágil de *Scott Ambler o Agile Unified Process (AUP)* en inglés es una versión simplificada del *Proceso Unificado de Rational (RUP).* Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de *software* de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP (Diaz, 2015).

En AUP se establecen cuatro fases que transcurren de manera consecutiva (Torrecilla, 2012):

- **Inicio:** el objetivo de esta fase es obtener una comprensión común cliente - equipo de desarrollo del alcance del nuevo sistema y definir una o varias arquitecturas candidatas para el mismo.

- **Elaboración:** el objetivo es que el equipo de desarrollo profundice en la comprensión de los requisitos del sistema y en validar la arquitectura.
- **Construcción:** durante esta fase el sistema es desarrollado y probado completo en el ambiente de desarrollo.
- **Transición:** el sistema se lleva a los entornos de preproducción donde se somete a pruebas de validación y aceptación y finalmente se despliega en los sistemas de producción.

1.8 Selección de la metodología de desarrollo de *software*

Al no existir una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto, la universidad decide hacer una variación de la metodología AUP, de forma tal que se adapte al ciclo de vida definido para la actividad productiva (Diaz, 2015).

Una metodología de desarrollo de *software* tiene entre sus objetivos aumentar la calidad del *software* que se produce, de ahí la importancia de aplicar buenas prácticas, para ello se apoya en el Modelo CMMI-DEV v1.3, el cual constituye una guía para aplicar las mejores prácticas en una entidad desarrolladora. Estas prácticas se centran en el desarrollo de productos y servicios de calidad (Diaz, 2015).

Entre las técnicas ágiles que utiliza AUP se encuentra el modelado ágil. Se hace uso de esta técnica para el proyecto, ya que por sus características encapsula sus requisitos funcionales en historias de usuarios. De forma general se siguen utilizando los productos de trabajos definidos en el expediente de proyecto, algunos son obligatorios independientemente del tipo de proyecto y otros son opcionales basándose en las particularidades del mismo.

Considerando las características de las funcionalidades a implementar y teniendo en cuenta que se necesita obtener un *software* en poco tiempo y que a la vez sea confiable, es seleccionada la metodología AUP en su variante UCI. Además, es flexible, no requiere de una gran cantidad de desarrolladores para favorecer a la estandarización de una metodología en la universidad.

Conclusiones parciales

En este capítulo se realizó un análisis de los principales conceptos para una mejor comprensión de la investigación, abordando definiciones relacionadas con los Sistemas Domóticos, las cámaras de vigilancia y el flujo de video. A partir de la revisión de algunas aplicaciones existentes se pudo verificar que los sistemas más completos utilizan *streaming* de video para realizar la transmisión en tiempo real. Por último, se hizo

una selección de herramientas y tecnologías para el desarrollo de la solución propuesta, la biblioteca *VLC* para realizar el envío en vivo del flujo de video, *Qt Creator* como entorno de desarrollo integrado, *C++* como lenguaje de programación, *UML* como lenguaje de modelado, *Visual Paradigm* como herramienta de modelado, y AUP en su variante UCI como metodología de desarrollo del *software*.

Capítulo 2 Diseño de la solución propuesta

2.1 Introducción

El objetivo de este capítulo es describir el sistema propuesto como solución. Comienza con una propuesta de arquitectura, luego se explica mediante una descripción detallada cada una de las funcionalidades que debe poseer el sistema a partir de las historias de usuario, para una mayor claridad de lo que se desea desarrollar. Se describen también los requisitos no funcionales con gran utilidad para el usuario final y estos se deben tener en cuenta para la perfecta utilización del sistema desarrollado. Además, se describen las principales clases que la componen para ayudar al desarrollo de la aplicación y una correcta estructuración del código.

2.2 Propuesta de solución

Para darle respuesta a la situación problemática antes descrita se propone desarrollar un módulo de acceso y visualización de contenido multimedia proveniente de cámaras IP para el Sistema Domótico del CEDIN. Este módulo estará diseñado para visualizar en tiempo real el flujo de video capturado por las cámaras. El Sistema Domótico contará con un componente grafico donde se podrá visualizar el contenido transmitido por cada cámara después de ser procesado por la solución propuesta, desde el cual se podrán invocar la ejecución de comandos mediante peticiones *HTTP*. Los comandos permitirán rotar la cámara, modificar el brillo, contraste, saturación y tono de color del video, además aumentar o disminuir el acercamiento; estos comandos se definen como: *rotate*, *brightness*, *contrast*, *saturation*, *hue* y *zoom*. La solución propuesta está compuesta por los elementos que se describen e ilustran a continuación:

Runtime: permite la representación en tiempo de ejecución de los procesos mediante la actualización de los componentes gráficos ubicados en los despliegues.

Ejecutador de Comandos (*CommandsExecuter*): es el encargado de la ejecución de comandos.

Manejador de Cámaras (*CamerasManager*): es el encargado de crear y utilizar las interfaces de cada una de las cámaras.

Biblioteca de cámara: permite ejecutar acciones sobre las cámaras, entre ellas solicitar *streaming* y ejecutar comandos. Además, representa una interfaz de biblioteca que implementa la comunicación con un

modelo de cámara.

Biblioteca VLC: es la biblioteca encargada de transmitir datos en *streaming*.

Figura 1 Propuesta de solución.

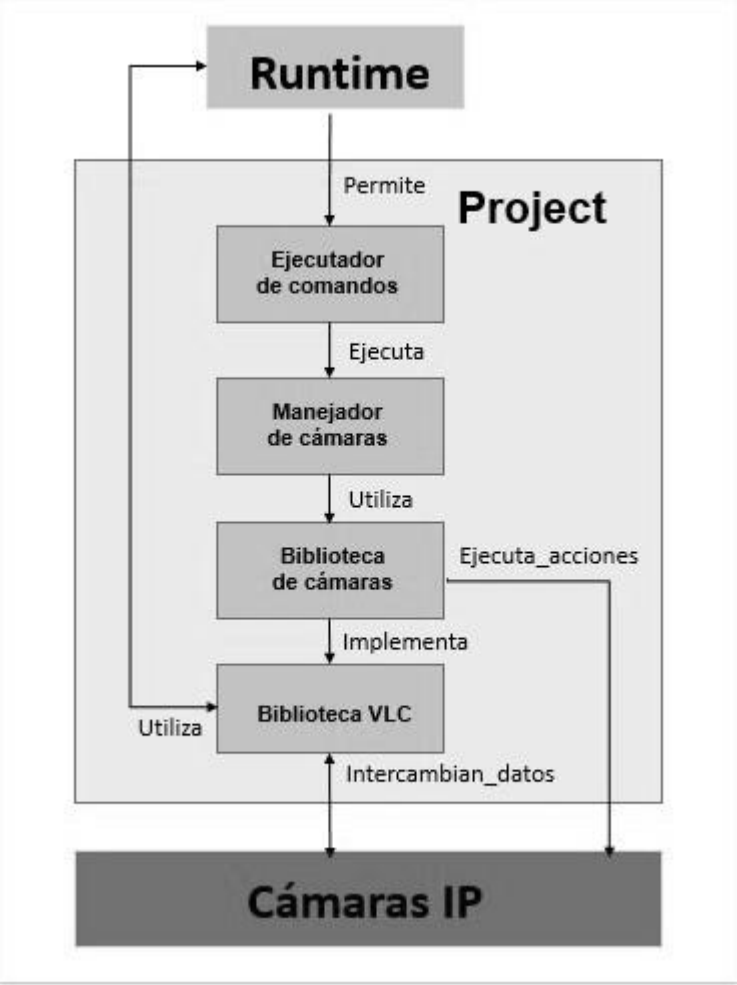


Tabla 1 Comandos para cámaras IP.

Parámetros	Valores
rotate	left, right, up, down
brightness	more, less, [-10,10]

contrast	more, less, [0, 20]
saturation	more, less, [0, 20]
hue	more, less, [-10, 10]
zoom	in, out, more, less, [0, 100]

2.3 Arquitectura Cliente-Servidor

La Arquitectura Cliente Servidor es un modelo para el desarrollo de sistemas de información en el que las transacciones se dividen en procesos independientes que cooperan entre sí para intercambiar información, servicios o recursos (Buschmann, 1996). Permite la combinación de un cliente que interactúa con el usuario, y un servidor que interactúa con los recursos compartidos. Cliente/Servidor establece una relación entre procesos distintos, los cuales pueden ser ejecutados en la misma máquina o en máquinas diferentes distribuidas a lo largo de la red. En esta arquitectura al estar distribuidas las funciones y responsabilidades independientes, es posible reemplazar, reparar, actualizar o incluso trasladar un servidor, mientras que sus clientes no se verán afectados por ese cambio. La figura muestra la arquitectura del sistema propuesto compuesta por el *Rutime* de Arex, el ejecutador de comandos (*CommandsExecuter*), el controlador de cámaras (*CamerasManager*), la biblioteca de cámara, la biblioteca VLC y por último las cámaras IP conectadas. La arquitectura Cliente/Servidor que se muestra en las figuras en un primer momento que el *Runtime* de Arex le realiza peticiones de video al ejecutador de comandos, refleja al *Rutime* como cliente y al ejecutador como servidor, y en un segundo instante el módulo implementado es denominado cliente y las cámaras conectadas serían servidores, se observa explícitamente en la Figura 2 y 3. Por esta razón se frecuenta decir que en este caso existe una comunicación en un entorno Cliente/Servidor, la cual se muestra a continuación:

Figura 2 Arquitectura del sistema.

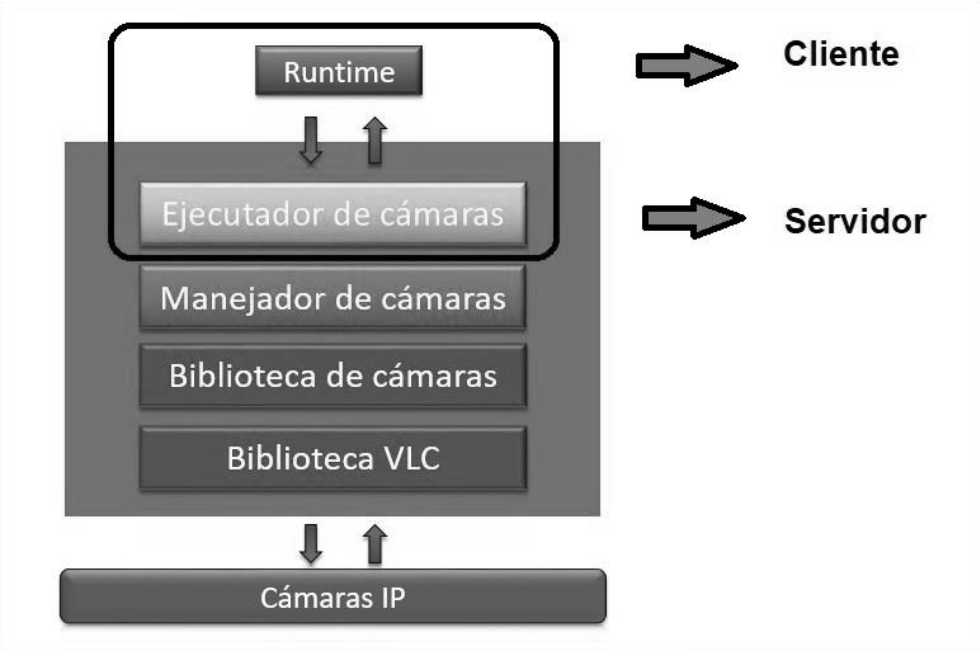
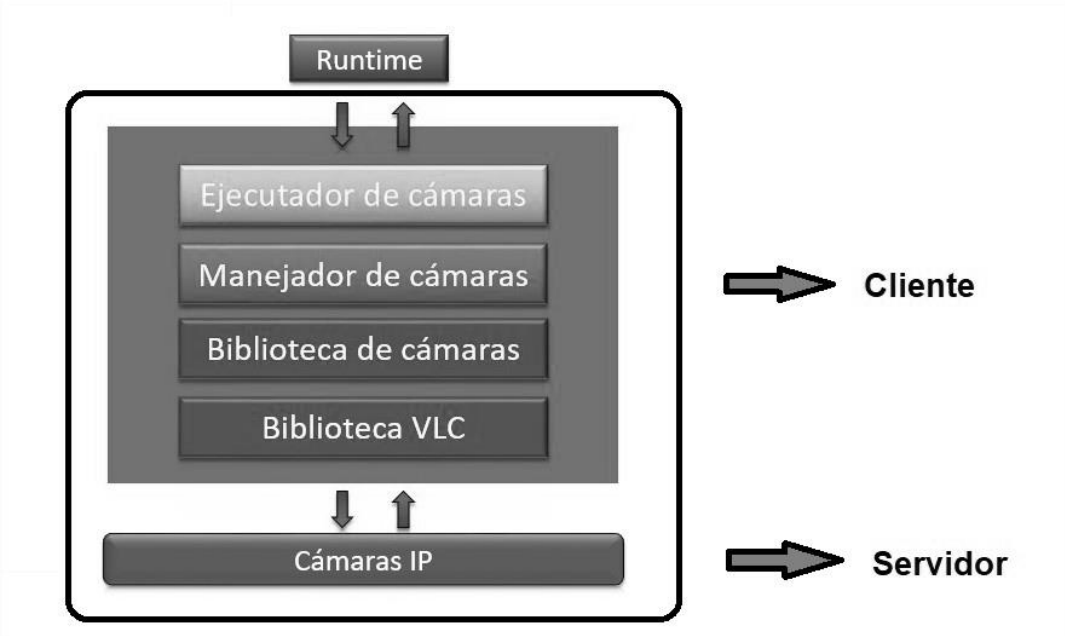


Figura 3 Arquitectura del sistema.



2.4 Requisitos funcionales del sistema

A través de los requisitos funcionales, se puede expresar una especificación más detallada de las responsabilidades del sistema. Con ellos, se pretende determinar de manera clara y concisa lo que debe hacer el sistema siguiendo un enfoque funcional. Como nos permite la metodología AUP-UCI, las funcionalidades del sistema serán descritas a través de historias de usuario generadas.

A continuación, se muestran las principales funcionalidades del sistema:

- RF1: Cargar archivo de configuración del Sistema Domótico e iniciar la comunicación con las cámaras.
- RF2: Retransmitir el contenido recibido de cada cámara a los clientes que lo soliciten.
- RF3: Ejecutar comandos sobre las cámaras conectadas a solicitud de los usuarios.

2.4.1 Historias de usuarios

En el desarrollo de *software* una de las actividades más importantes está dirigida a la descripción de requisito, la misma permitirá al equipo de desarrollo ejecutar con precisión lo que debe cumplir el sistema para satisfacer a su proveedor. El ciclo de desarrollo del *software*, según la metodología AUP, comienza con la fase de inicio, en la cual, los clientes plantean a grandes rasgos las Historias de Usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán durante el desarrollo (Diaz, 2015).

Las Historias de usuarios (HU), son descripciones cortas y escritas en el lenguaje del usuario, donde el nivel de detalle debe ser el mínimo posible, para que de manera sencilla se determine cuánto costará la implementación del sistema. El cliente redacta, según su consideración, los requisitos que debe tener la aplicación mediante las HU, expresando de esta manera su punto de vista en cuanto a las necesidades del sistema.

Descripción de Requisitos de *Software*:

Tabla 2 HU Cargar archivo de configuración del Sistema Domótico e iniciar la comunicación con las cámaras.

Número: RF1	**Nombre del requisito:** Cargar archivo de configuración del Sistema Domótico e iniciar la comunicación con las cámaras.

Programador: Lidia Mercedes Moya García	**Iteración Asignada:** V 1.0
Prioridad: Alta	**Tiempo Estimado:** 48 horas
Riesgo en Desarrollo: El programador puede presentar estado de enfermedad que mermen su rendimiento laboral o le impidan cumplir con sus tareas.	**Tiempo Real:** 2 semanas
Descripción: El sistema explora el archivo de configuración del Sistema Domótico, reconoce la sección correspondiente a las cámaras de video, carga la configuración de cada cámara, configura e inicializa la comunicación con cada una, y se comienza la retransmisión del contenido recibido con velocidades de video, audio y fotogramas configurados.	

Tabla 3 HU Retransmitir el contenido recibido de cada cámara a los clientes que lo soliciten.

Número: RF2	**Nombre del requisito:** Retransmitir el contenido recibido de cada cámara a los clientes que lo soliciten.
Programador: Lidia Mercedes Moya García	**Iteración Asignada:** V 1.0
Prioridad: Alta	**Tiempo Estimado:** 56 horas
Riesgo en Desarrollo: El programador puede presentar estado de enfermedad que mermen su rendimiento laboral o le impidan cumplir con sus tareas.	**Tiempo Real:** 2 semanas
Descripción: El sistema acepta la retransmisión del contenido recibido de cada cámara a cada cliente que lo solicite mediante una petición *HTTP*.	

Tabla 4 HU Ejecutar comandos sobre las cámaras conectadas a solicitud de los usuarios.

Número: RF3	**Nombre del requisito:** Ejecutar comandos sobre las cámaras conectadas a

	solicitud de los usuarios.
Programador: Lidia Mercedes Moya García	**Iteración Asignada:** V 1.0
Prioridad: Alta	**Tiempo Estimado:** 80 horas
Riesgo en Desarrollo: El programador puede presentar estado de enfermedad que mermen su rendimiento laboral o le impidan cumplir con sus tareas.	**Tiempo Real:** 3 semanas
Descripción: Se ejecutan comandos a las cámaras conectadas según la solicitud de los usuarios con peticiones *HTTP*.	

2.5 Requisitos no funcionales del sistema

Los requisitos no funcionales del sistema son restricciones de los servicios o funciones ofrecidas por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. Se puede decir que los requisitos no funcionales constituyen la forma en la que debe actuar el sistema para que funcione de manera eficaz, atendiendo aspectos tales como la disponibilidad, flexibilidad, seguridad y facilidad de uso. En muchos casos los requisitos no funcionales son fundamentales en el éxito del producto debido a que forman una parte significativa de la especificación.

Requisitos no funcionales de software:

- *Framework QT*.
- PC con Sistema Operativo *Linux*.

Requisitos no funcionales de hardware:

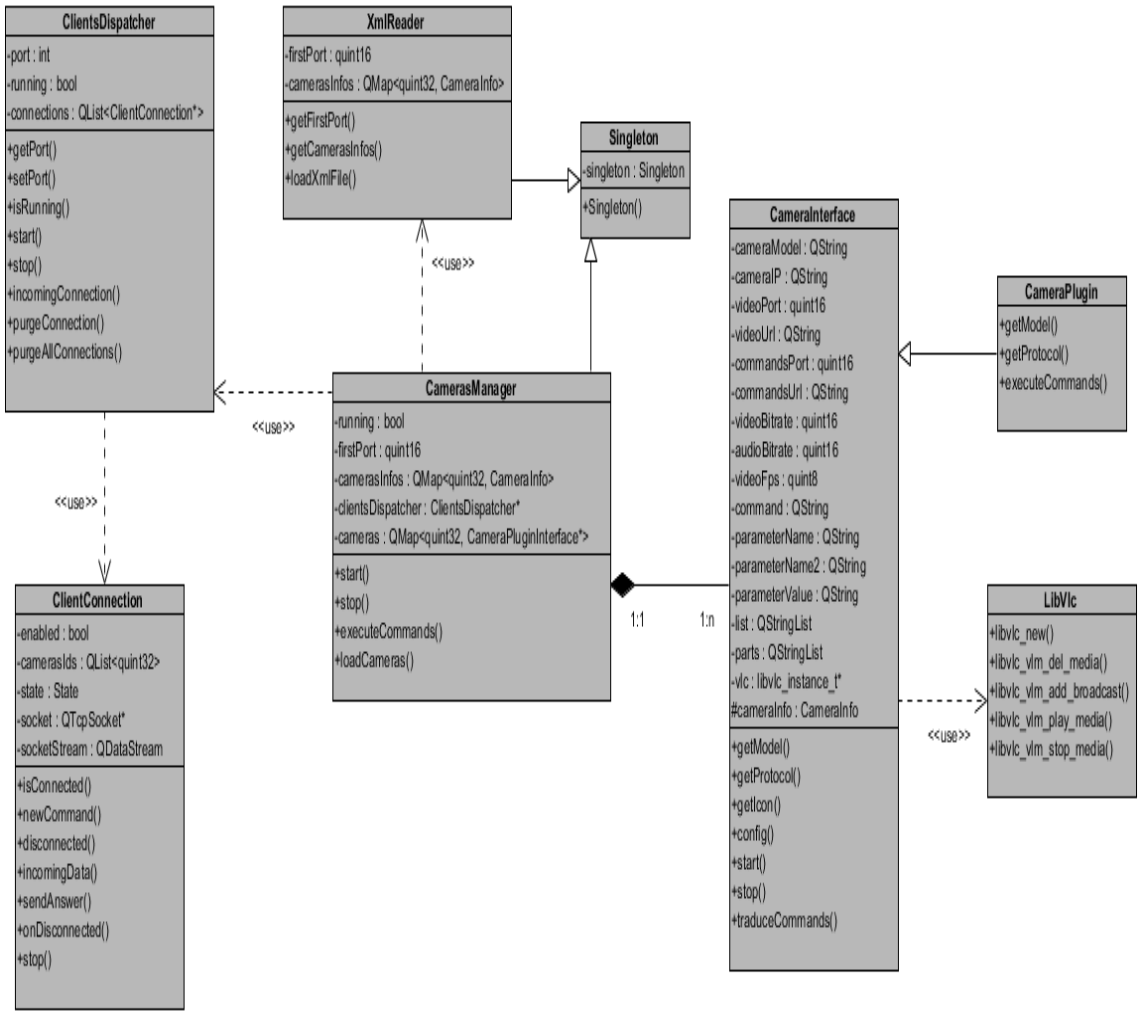
- PC con microprocesador *Pentium* 4 o superior.
- PC con memoria *RAM* mínima de 256 MB.

2.6 Diagrama de clases del sistema

Los diagramas de clases muestran cómo se compone un sistema. Se dice que son diagramas «estáticos» porque muestran las clases, junto con sus métodos y atributos, pero no muestran los métodos mediante los que se invocan entre ellas. Para lograr un mejor entendimiento de la solución a continuación se describe

brevemente el diseño de clases del módulo implementado como parte del Sistema Domótico del CEDIN. En el diagrama que se observa en la Figura 4, la clase *CamerasManager* representa la clase principal y controladora de dicho módulo, esta clase es la encargada de crear cada instancia de las interfaces de las cámaras y utiliza la clase *CameraInterface*, *XmlReader* y *CommandsExecuter*. La clase *CameraInterface* usa la biblioteca *VLC* para recibir el flujo de video de cada cámara configurada. Las peticiones de los clientes son atendidas por instancias de la clase *CommandsExecuter* que tiene la responsabilidad de establecer comunicación con la clase *ClientConnection*.

Figura 4 Diagrama de clases del sistema.



2.7 Patrones de diseño

Los patrones de diseño pueden usarse durante el diseño del *software*. Estos patrones se aplican a un elemento específico del diseño como un agregado de componentes para resolver algún problema de diseño, relaciones entre los componentes o los mecanismos para efectuar esta comunicación. (Pressman, 2010)

2.7.1 Patrones GRASP

Los patrones *GRASP*, más que patrones propiamente dichos, son una serie de “buenas prácticas” de aplicación recomendable en el diseño de *software*. Entre ellos, los patrones Experto y Creador. Son patrones generales de software para asignación de responsabilidades, es el acrónimo de "*GRASP* (*General Responsibility Assignment Software Patterns*)" (Tabares, 2010).

Experto en Información

El patrón Experto es el principio básico de asignación de responsabilidades. Nos indica, por ejemplo, que la responsabilidad de la creación de un objeto o la implementación de un método debe recaer sobre la clase que conoce toda la información necesaria para crearlo, en este caso la clase experta es *CamerasManager*. De este modo la información se mantiene encapsulada. Además, plantea el problema “¿Cuál es el principio fundamental mediante el cual se asignan responsabilidades a los objetos?” y aporta la solución “Asignar la responsabilidad a la clase que tiene la información necesaria para cumplirla”. El patrón Experto posibilita una adecuada asignación de responsabilidades facilitando la comprensión del sistema, su mantenimiento y adaptación a los cambios con reutilización de componentes. Los objetos utilizan su propia información para llevar a cabo sus tareas. Se distribuye el comportamiento entre las clases que contienen la información requerida. Son más fáciles de entender y mantener (Tabares, 2010).

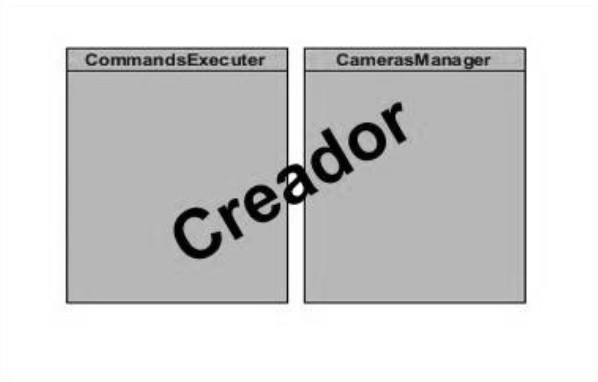
Figura 5 Patrón Experto en información.

Creador

El patrón Creador nos ayuda a identificar quién debe ser el responsable de la creación de nuevos objetos o clases, este patrón se refleja en las clases *CommandsExecuter y CamerasManager*. Una de las consecuencias de usar este patrón es la visibilidad entre la clase creada y la clase creador. Una ventaja es la facilidad de mantenimiento y reutilización. La creación de instancias es una de las actividades más comunes en un sistema orientado a objetos. En consecuencia, es útil contar con un principio general para la asignación de las responsabilidades de creación. Si se asignan bien el diseño puede soportar una mayor claridad, encapsulación y reutilización (Botella, 1996). Además, aporta un principio general para la creación de objetos, una de las actividades más frecuentes en programación. La nueva instancia deberá ser creada por la clase que cumpla una de las siguientes condiciones:

- Tiene la información necesaria para realizar la creación del objeto.
- Usa directamente las instancias creadas del objeto.
- Almacena o maneja varias instancias de la clase.
- Contiene o agrega la clase.

Figura 6 Patrón Creador.



2.7.2 Patrones GoF

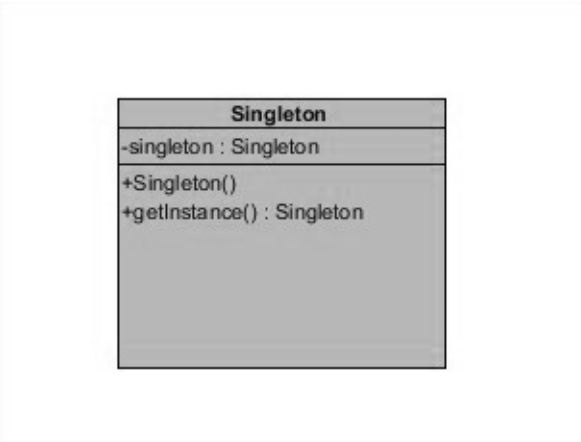
El término *GoF* proviene del inglés *Gang of Four*, en español pandilla de cuatro. Dicha pandilla estaba compuesta por *Erich Gamma, Richard Helm, Ralph Johnson y John Vlisides*, este grupo escribió el libro *"Design Patterns".* En este famosísimo libro, a menudo referenciado como *GoF*, los autores recogían 23 patrones comunes en el diseño de *software*, explicando al detalle en qué consistían y describiendo las

soluciones adoptadas típicamente para cada problema. Toda una enciclopedia sobre diseño de *software* para el programador (Pressman, 2010).

Singleton

El patrón de diseño *Singleton* está diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto. En la solución que se propone se utiliza el patrón *Singleton* en las clases *CamerasManager* y *XmlReader*. Su intención consiste en garantizar que una clase sólo tenga una instancia y proporcionar un punto de acceso global a ella. No se encarga de la creación de objetos en sí, sino que se enfoca en la restricción en la creación de un objeto. En el módulo desarrollado fue necesaria la utilización de este patrón para que se posea una sola instancia (Larman, 2010).

Figura 7 Patrón Singleton.



Facade

El patrón *Facade* (Fachada) viene motivado por la necesidad de estructurar un entorno de programación y reducir su complejidad con la división en subsistemas, minimizando las comunicaciones y dependencias entre estos. Los patrones de diseño estructural dan una solución probada y documentada a problemas de desarrollo de *software* que aparecen en un contexto similar. Se aplicará el patrón *Facade* en la clase CameraInterface. Surge de la necesidad de desacoplar un sistema de otros subsistemas, haciéndolo más independiente, portable y reutilizable (Peña, 2017).

Figura 8 Patrón Fachada.

Conclusiones parciales

En este capítulo se cumplen los objetivos del trabajo de diploma propuesto como solución. Se describe el diseño del módulo especializado para visualizar el flujo de video de cámaras IP en el área de la domótica.

Capítulo 3 Implementación y pruebas del sistema

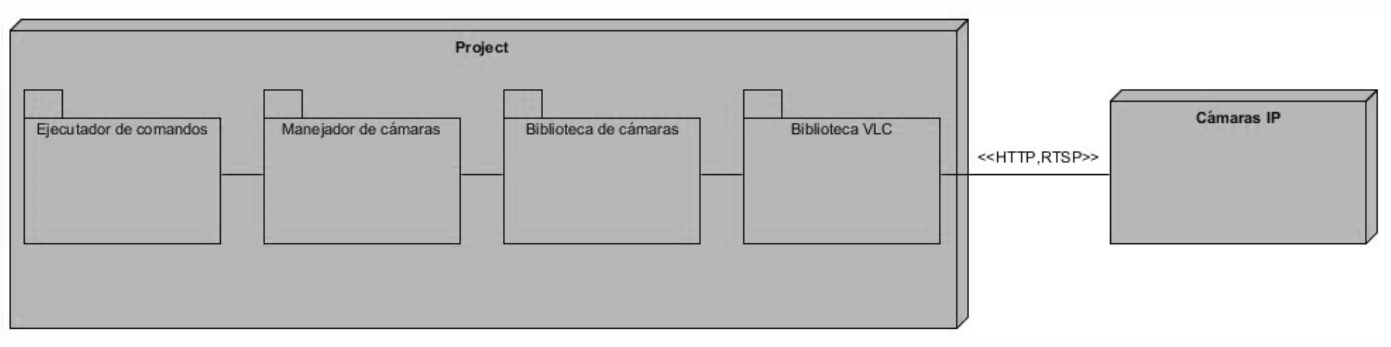
3.1 Introducción

En este capítulo se comienza con la descripción del diagrama donde va a estar desplegado el sistema. Una vez que la implementación culmine se realizan un conjunto de pruebas funcionales para validar el correcto funcionamiento del sistema, con el objetivo de verificar la calidad de un producto, detectando los posibles errores que puede presentar.

3.2 Diagrama de despliegue del sistema

Los diagramas de despliegue describen la arquitectura física del sistema durante la ejecución y su topología: la estructura de los elementos de hardware y *software* que ejecuta cada uno de ellos. El diagrama de despliegue está constituido por nodos físicos en los cuales se ejecutan los componentes. Se describe el mapeo del *software* en el hardware y refleja los aspectos de distribución. También se estarán abordando temas como el diseño del despliegue el cual establece una correspondencia entre la arquitectura de *software* y la arquitectura de hardware del sistema; lo que permite modelar la disposición física o la topología de un sistema, mostrar el hardware usado y los componentes instalados en el hardware, y además mostrar las conexiones físicas entre el hardware y las relaciones entre componentes.

Figura 9 Diagrama de despliegue del sistema.



Project: en el nodo de la izquierda se ejecutará el módulo desarrollado.

Cámaras IP: en el nodo de la derecha se encuentran las cámaras IP.

3.3 Pruebas del sistema

Las pruebas son procesos que se llevan a cabo con el objetivo de verificar la calidad de un producto, detectando los posibles errores que puede presentar. Existen disímiles tipos de pruebas, y cada una de ellas fue diseñada con el objetivo de evaluar diferentes aspectos de un producto. Se estará abordando acerca de lo que es la validación y verificación a lo largo del desarrollo del sistema. El papel de la verificación comprende comprobar que el *software* está de acuerdo con su especificación. Se comprueba que el sistema cumple los requerimientos funcionales y no funcionales que se le han especificado. La validación es un proceso más general, puesto que se debe asegurar que el *software* cumple las expectativas del cliente. Existen técnicas complementarias para el análisis y comprobación de los sistemas tales como las inspecciones del *software* donde se analizan y comprueban las representaciones del sistema como el documento de requerimientos, los diagramas de diseño y el código fuente del programa, y las pruebas del software consisten en contrastar las respuestas de una implementación del *software* a series de datos de prueba y examinar las respuestas del software y su comportamiento operacional, para comprobar que se desempeñe conforme a lo requerido. Para detectar los errores que puede llegar a presentar nuestro sistema se puede hacer uso de los métodos de prueba Caja Blanca y Caja Negra. Las pruebas de caja blanca requieren del conocimiento de la estructura interna del programa y las pruebas de caja negra se centran principalmente en los requisitos funcionales del software permitiendo así encontrar funciones incorrectas o ausentes (Roger, 1997). Al sistema se le realizaron tres tipos de pruebas: las pruebas unitarias empleando el método de caja negra, las pruebas de aceptación revisadas por el cliente, y por último las pruebas de liberación junto a un personal con vasta experiencia en el tema tratado y apto para realizar estas pruebas.

3.3.1 Pruebas unitarias

Las **pruebas unitarias** se realizaron con el objetivo de validar que el código sea funcional. Para ello se utilizó la técnica o **método de caja negra.** Estas pruebas comenzaron a mucho antes de terminar la implementación, ya que era necesario probar las funcionalidades a medida que se iban desarrollando. Esto garantizó la culminación de una aplicación que posee el comportamiento esperado por el usuario. La prueba de caja negra intenta encontrar funciones incorrectas o ausentes, errores en estructuras de datos, de inicialización o de terminación. A continuación, se muestran las pruebas de caja negra pertenecientes a la solución propuesta:

Tabla 5 Prueba de caja negra " Cargar archivo de configuración del Sistema Domótico e iniciar la comunicación con las cámaras".

Escenario 1.1 Existe el archivo de configuración.	
Descripción	Cargar el archivo de configuración del Sistema Domótico e iniciar la comunicación con las cámaras.
Variables	**NA**
Respuesta del sistema	Se carga el archivo de configuración del sistema y luego se inicia la comunicación con las cámaras.
Flujo central	La aplicación se inicia automáticamente con el sistema operativo como proceso. / Se inicia o reinicia el proceso mediante comandos de consola siguientes: Detener: /etc/init.d/cedin-domotic-camerastreamer stop Iniciar: /etc/init.d/cedin-domotic-camerastreamer start Reiniciar: /etc/init.d/cedin-domotic-camerastreamer restart
Escenario 1.2 No existe el archivo de configuración.	
Descripción	Cargar archivo de configuración del Sistema Domótico e iniciar la comunicación con las cámaras.
Variables	**NA**
Respuesta del sistema	No se puede establecer la comunicación con las cámaras, ya que no existe el archivo de configuración del sistema.
Flujo central	La aplicación no se inicia automáticamente con el sistema operativo como proceso.

Tabla 6 Prueba de caja negra "Retransmitir el contenido recibido de cada cámara a los usuarios que lo soliciten".

Escenario 1.1 Retransmitir el contenido recibido de cada cámara.			
Descripción	Se retransmite el contenido recibido de una cámara al usuario que lo solicite.		
Variables	**Recurso**	**V**	**Image.cgi**
Respuesta del sistema	En la interfaz donde se realizó la solicitud se debe obtener el video y el audio recibido de cada cámara.		
Flujo central	El usuario realiza la petición HTTP para solicitar *streaming* de video y audio de una cámara, mediante la URL siguiente: http://ipApp:Puerto/Recurso donde ipApp es el IP de la aplicación y Puerto sería el puerto de comunicación definido para la transmisión de		

	video y audio de una cámara.		
Escenario 1.2 Retransmitir el contenido recibido de cada cámara/ URL con error.			
Descripción	Se retransmite el contenido recibido de una cámara al usuario que lo solicite.		
Variables	**Recurso**	**V**	**Image.cgi**
Respuesta del sistema	El sistema empieza la transmisión del video y el audio con los parámetros configurados de velocidad de video, velocidad de audio y fotogramas por segundo.		
Flujo central	El usuario realiza la petición HTTP para solicitar *streaming* de video y audio de una cámara, mediante la URL siguiente: http://ipApp:Puerto/Recurso donde ipApp es el IP de la aplicación y Puerto sería el puerto de comunicación definido para la transmisión de video y audio de una cámara.		

Tabla 7 Prueba de caja negra “Ejecutar comandos sobre las cámaras conectadas a solicitud de los usuarios”.

Escenario 1.1 Ejecutar comandos sobre las cámaras conectadas.			
Descripción	Ejecuta un comando sobre una cámara a solicitud de un usuario.		
Variables	**Cámara**	**V**	**1**
	Parámetro	**V**	**rotate**
	Valor	**V**	**right**
Respuesta del sistema	Se ejecuta el comando sobre la cámara indicada. Se notifica al usuario de la ejecución del comando.		
Flujo central	El usuario realiza la petición HTTP para solicitar la ejecución de un comando a una cámara mediante la URL siguiente: http://ipApp:Puerto/Cámara?Parámetro=Valor donde **ipApp** es el IP de la aplicación, **Puerto** sería el puerto del servidor ejecutor de comandos, **Cámara** es el número de la cámara, **Parámetro** es el nombre de uno de los parámetros soportados, y **Valor** es el valor que se le quiere asignar al parámetro. Los posibles valores de **Cámara** son: 1, …, N; donde N es la cantidad de cámaras configuradas. Los posibles valores de **Parámetro** y **Valor** son los siguientes: Parámetros Valores rotate left, right, up, down brightness more, less, [-10,10]		

	contrast	more, less, [0, 20]
	saturation	more, less, [0, 20]
	hue	more, less, [-10, 10]
	zoom	in, out, more, less, [0, 100]

Escenario 1.2 Ejecutar comandos sobre las cámaras conectadas/ Error en "Cámara".			
Descripción	Ejecuta un comando sobre una cámara a solicitud de un cliente.		
Variables	**Cámara**	**F**	**a**
	Parámetro	**V**	**contrast**
	Valor	**V**	**less**
Respuesta del sistema	No se ejecuta el comando sobre la cámara indicada. Se notifica al usuario de un error en el número de cámaras conectadas.		
Flujo central	El usuario realiza la petición HTTP para solicitar la ejecución de un comando a una cámara mediante la URL siguiente: http://ipApp:Puerto/Cámara?Parámetro=Valor.		

Escenario 1.3 Ejecutar comandos sobre las cámaras conectadas/ Error en "Parámetro".			
Descripción	Ejecuta un comando sobre una cámara a solicitud de un cliente.		
Variables	**Cámara**	**V**	**5**
	Parámetro	**F**	**prightness**
	Valor	**V**	**more**
Respuesta del sistema	No se ejecuta el comando sobre la cámara indicada. Se notifica al usuario de un error en el nombre del parámetro.		
Flujo central	El usuario realiza la petición HTTP para solicitar la ejecución de un comando a una cámara mediante la URL siguiente: http://ipApp:Puerto/Cámara?Parámetro=Valor.		

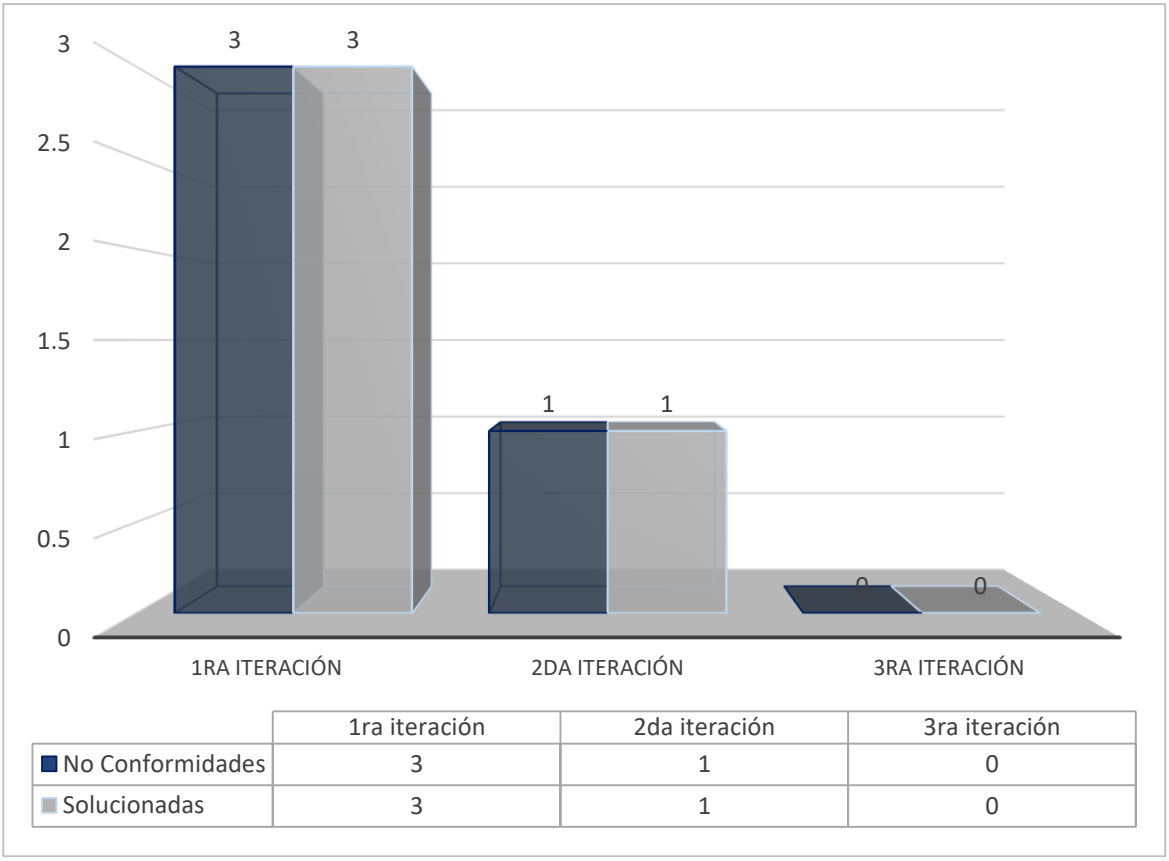
Escenario 1.4 Ejecutar comandos sobre las cámaras conectadas/ Error en "Valor".			
Descripción	Ejecuta un comando sobre una cámara a solicitud de un cliente.		
Variables	**Cámara**	**V**	**8**
	Parámetro	**V**	**saturation**
	Valor	**F**	**lest**
Respuesta del	No se ejecuta el comando sobre la cámara indicada. Se notifica al		

sistema	usuario de un error en el valor del parámetro.
Flujo central	El usuario realiza la petición HTTP para solicitar la ejecución de un comando a una cámara mediante la URL siguiente: http://ipApp:Puerto/Cámara?Parámetro=Valor.

3.3.2 Pruebas de aceptación

Las **pruebas de aceptación** se realizaron con el objetivo de validar que el sistema cumple con el funcionamiento esperado y permitió al usuario determinar su aceptación, desde el punto de vista de su funcionalidad y rendimiento. Las pruebas de aceptación fueron definidas por el usuario del sistema y preparadas por el desarrollador. Cada HU sólo se consideró terminada cuando pasó correctamente todas las pruebas de aceptación.

Figura 10 Pruebas de aceptación.



	1ra iteración	2da iteración	3ra iteración
No Conformidades	3	1	0
Solucionadas	3	1	0

La gráfica muestra el resultado de las pruebas de aceptación realizadas al sistema, la misma fue dividida en tres iteraciones. En la **primera iteración** fueron detectadas tres no conformidades: una de ortografía y dos de redacción, las cuales fueron solucionadas. En la **segunda iteración** fue detectada una no conformidad de redacción, la cual fue solucionada. En la **última iteración** no se detectan no conformidades. Estas pruebas de aceptación fueron realizadas al sistema antes de culminar.

3.3.3 Pruebas de liberación

Finalmente se realizaron las pruebas de liberación al sistema final. Un equipo de proyecto especializado del centro CEDIN en donde se efectuaron las pruebas, determinó que no se encontraron no conformidades, por ello se puede decir que el *software* funciona correctamente y cuenta con la calidad requerida para su posterior uso en el área de la domótica.

Conclusiones parciales

En este capítulo se muestra el diagrama de despliegue de la solución propuesta que brinda el conocimiento de cómo se distribuyen los elementos que intervienen en la aplicación. Las pruebas de validación del *software* realizadas, permitieron comprobar el buen funcionamiento del producto domótico, además, esta aplicación puede ser ejecutada e integrada al Sistema Domótico satisfactoriamente.

Conclusiones

En el presente trabajo se propuso desarrollar un módulo de acceso y visualización de contenido multimedia proveniente de cámaras IP, para el Sistema Domótico del CEDIN. Al finalizar la investigación sobre el desarrollo de este módulo se puede concluir que se dio solución al problema planteado, con lo cual se logró:

- El desarrollo de un módulo que pueda ser utilizado e integrado al Sistema Domótico, para resolver la carencia de captura de *streaming* de video utilizando cámaras IP.
- Descartar la existencia de alguna herramienta que brinde solución al problema identificado en esta investigación, luego de la búsqueda y el análisis de varias soluciones existentes.
- Diseñar el modelo a desarrollar a través del uso de la metodología, tecnologías y herramientas seleccionadas.
- Realizar las pruebas pertinentes al *software* para validar su correcto funcionamiento.

Recomendaciones

Al concluir la investigación del trabajo de diploma se propone como recomendación para dar continuidad:

- Implementar nuevos reconocimientos de modelos y protocolos de comunicación de cámaras.

Bibliografía

Almeida, Zahiro Lamadrid. 2015. *Documento de tesis.* 2015.

Almenara, Julio Cabero. 1996. Nuevas tecnologías, comunicación y educación. *Revista electrónica de tecnología educativa.* [En línea] 1996. http://www.edutec.es/revista/index.php/edutec-e/article/download/576/305.

ANDALUCÍA, JUNTA DE. 2015. Patrones de Diseño. [En línea] 2015. http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/13.

Aquiles, Alfredo. 2017. Cómo hacer transmisiones en vivo. [En línea] 24 de junio de 2017. [Citado el: 9 de enero de 2018.] https://www.entrepreneur.com/article/264962.

Botella, Ricardo Devis. 1996. *PATRONES DE DISEÑO.* 1996.

Botero Tabares, Ricardo. 2010. GRASP: Más patrones para asignar responsabilidades. *Patrones Grasp y Anti-Patrones: un Enfoque Orientado a Objetos desde Lógica de Programación.* [En línea] 2010. [Citado el: 10 de Junio de 2015.] http://biblioteca.ucp.edu.co/ojs/index.php/entrecei/article/download/760/721.

Buschmann, Frank. 1996. *Pattern Oriented Software Architecture.* 1996. ISBN 0-471-95869-7.

Bustio, José Andrés Hernández. 2018. Protocolos y modelos de comunicación de las cámaras IP WiFi de videovigilancia y domótica de tenealive. *Protocolos y modelos de comunicación de las cámaras IP _TeneaLive.* [En línea] 8 de junio de 2018. [Citado el: 22 de enero de 2018.] http://tenealive.com/protocolosymodelosdecomunicaci%C3%B3ndelasc%.

Castillo, Eddye Arturo Sanchez. 2015. El importante rol del sistema de cctv en la seguridad privada. [En línea] 6 de julio de 2015. [Citado el: 5 de octubre de 2018.] http://siesa.com.ar/el-importante-rol-del-sistema-de-cctv-en-la-seguridad-privada.

Chavez Frias, Hugo Rafael. 2003. El golpe fascista contra Venezuela. *discursos e intervenciones.* [En línea] Enero de 2003.

Chavez, Jordanis Viltres. 2018. *“ SISTEMA DOMÓTICO, ALTERNATIVA NACIONAL PARA LA AUTOMATIZACIÓN ”.* 2018.

Chávez, Jordanis Viltres. 2015. *Arex, una alternativa de automatización orientado a la domótica.* 2015.

Conde García, Luis. 2015. gestiopolis. [En línea] 2015. http://www.gestiopolis.com/protocolo-de-control-de-transmision-tcp/.

Diaz, Katia Roselló. 2015. *Metodología de desarrollo para la Actividad productiva de la UCI.* San Antonio : s.n., 2015.

Diaz, Yassiel Gómez. 2016. *Control de cámara Pan Tilt.* La Habana : s.n., 2016.

Diez, Alfredo Aquiles. 2005. Cómo hacer una transmisión en directo en Internet por *streaming*. *STREAMINGDIEZ.* [En línea] 8 de abril de 2005. [Citado el: 16 de febrero de 2012.] https://streamingdiez.com/como-hacer-una-transmision-en-directo-en-internet-por-streaming/.

Dommermuth, Ralph. 2017. Digital Guide. [En línea] 8 de febrero de 2017. [Citado el: 28 de noviembre de 2018.] https://www.1and1.es/digitalguide/correo-electronico/.

Eckel, Bruce . 2012. Pensar en C++. [En línea] Enero de 2012.

Empresas NorteSur. 2001. Prpductos GE Fanuc. *Empresas Norte Sur, C.A. .* [En línea] 2001.

http://www.empresasnortesur.com/lineas2.htm.

Fontela, Carlos. 2011. *UML: modelado de software para profesionales.* Primera. Buenos Aires : Alfaomega Grupo Editor Argentino, 2011. pág. 184.

Fortis, Javier Flores. 2010. ¿Qué es la domótica? [En línea] 6 de noviembre de 2010. [Citado el: 20 de mayo de 2018.] https://www.muyinteresante.es/innovacion/articulo/ique-es-la-domotica.

Gamma, Erich, y otros. 1994. Design Patterns. *Elements of Reusable Object-Oriented Software.* [En línea] 1994.

GENBETA. 2011. Los patrones de diseño de software. [En línea] 6 de Abril de 2011. http://www.genbetadev.com/metodologias-de-programacion/los-patrones-de-diseno-de-software.

Gómez, Raidel Morales. 2013. Espacio Linux. *Super tutorial de introducción a GStreamer.* [En línea] 15 de diciembre de 2013. [Citado el: 2 de noviembre de 2018.] http://www.espaciolinux.com.

Gonzalez, Barahona. 2008. Digitaliza tu vida. [En línea] WordPress.com, 19 de 01 de 2008. [Citado el: 8 de diciembre de 2018.] https://mivideo.wordpress.com/2008/01/19/vlc-media-player/.

González, Oscar Díaz. 2007. Departamento de Lenguajes y Sistemas Informáticos e Ingeniería de Software. [En línea] 2007.

Hernández Meléndrez, Edelsis. 2006. *Metodología de la invaetigación. Cómo escribir una tesis.* La Habana : Escuela Nacional de Salud Pública, 2006. pág. 51. Vol. 1.

Hernández, José. 2010. *Framework QT.* 2010.

Hidalgo, Mario Mesa. 1997. *Proceso Unificado de Desarrollo.* 1997.

Himmelblau, David M y Bischoff, Kenneth B. 1992. *Análisis y simulación de procesos.* Barcelona : Reverté, 1992.

Huri, Mauricio Leonardo. 2017. Ventajas de la retransmisión por Vídeo *Streaming*. [En línea] 16 de marzo de 2017. [Citado el: 28 de octubre de 2018.] http://huribroadcast.com/ventajas-retransmision-video-streaming/.

James Rumbaugh. 2013. *El Lenguaje Unificado de Modelado. Manual de Referencia.* 2013.

JUNTA DE ANDALUCÍA. 2015. Patrones de Diseño. *Marco de Desarrollo de la Junta de Andalucía.* [En línea] 2015. http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/13.

Kord, Maia. 2013. Patrones. Diferencia entre GRASP y GoF. [En línea] 3 de 12 de 2013. https://sites.google.com/site/tuxnots/materiasdelafacu/metodologiadesistemas/patronesgrasppatronesgofdiferenciaentregraspygof.

Larman, Craig. UML y Patrones. *Una introducción al análisis y diseño orientado a objetos y al proceso unificado.* [En línea] http://sunshine.prod.uci.cu/gridfs/sunshine/books/Craig_Larman_-_UML_y_Patrones.pdf.

Letelier, Patricio. 2006. Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP). [En línea] 15 de junio de 2006. [Citado el: 3 de febrero de 2012.] http://www.cyta.com.ar/ta0502/b_v5n2a1.htm.

LibrosWeb. 2017. http://librosweb.es. [En línea] 2017. [Citado el: 11 de marzo de 2017.] http://librosweb.es/libro/symfony_1_2_en/chapter_6/the_front_controller.html.

Lucid Software Inc. 2018. *UML Class Diagram Tutorial.* 2018.

Marín Moreno, William. 2003. Modelo OSI. [En línea] 2003. http://www.ie.itcr.ac.cr/marin/telematica/trd/01_modelo_OSI_v2.pdf.

Marquez, Lis. 2004. *Implementación de un reconocedor de voz gratuito a el sistema de ayuda a invidentes Dos-Vox en español .* Mexico : s.n., 2004.

MARTÍNEZ JUAN, FRANCISCO JAVIER. GUÍA DE CONSTRUCCIÓN DE SOFTWARE EN JAVA CON PATRONES DE DISEÑO. *PROYECTO FIN DE CARRERA.* [En línea] http://sunshine.prod.uci.cu/gridfs/sunshine/books/GUIA_DE_CONSTRUCCION_DE_SOFTWARE_EN_JAVA_CON_PATRONES_DE_DISEO.pdf.

Marulanda, Jhon Alexander Gomez. 2008. Cámaras de Vigilancia y Seguridad. *Video Vigilancia.* [En línea] INTPLUS, 15 de abril de 2008. [Citado el: 4 de febrero de 2016.] http://www.videovigilancia.com/camaras.htm.

Medina, Francisco Javier Medina. 2018. Patrón de diseño Singleton. [En línea] InformaticaPC.com, 2018. https://informaticapc.com/patronesdediseno/singleton.php.

Molkentin, Daniel. 2010. *APRENDA QT HOY MISMO.* 2010.

Morales, Hernán Venegas. 2017. Métodos de Investigación Científica. *Métodos Teóricos y Empíricos.* [En línea] 4 de diciembre de 2017. [Citado el: 24 de octubre de 2018.] http://www.ecotec.edu.ec/documentacion%5Cinvestigaciones%5Cdocentes_y_directivos%5Carticulos/4937_Fcevallos_00004.pdf.

Novak, Tadej. 2004. VLC-QT Library. [En línea] 13 de junio de 2004. [Citado el: 25 de noviembre de 2018.] http://vlc-qt.tano.si.

Orallo, Enrique Hernández. 2017. *El Lenguaje de Modelado Unificado de UML.* Buenos Aires : Alfaomega Grupo Editor Argentino, 2017.

Ortiz, Quintero. 2006. EVALUACION DE SERVIDORES DE STREAMING DE VIDEO ORIENTADO A DISPOSITIVOS MOVILES. [En línea] 9 de julio de 2006. [Citado el: 17 de enero de 2010.]

Paez, Ginebra Cabrera. 2003. Taller Internacional de Impacto de las TIC en la Sociedad. [En línea] 8 de enero de 2003. [Citado el: 25 de septiembre de 2018.] https://eventos.uci.cu/es/iii-taller-internacional-de-impacto-de-las-tic-en-la-sociedad.

Peña, J. 2017. *Patrones: Facade y Template Method. Ingeniería del Software II (Teoría).* Universidad de Sevilla : s.n., 2017.

Perez Javier, Maikel. 2015. Manual Organizacional: CEDIN. [En línea] Febrero de 2015. [Citado el: 12 de Marzo de 2015.] http://gespro.cedin.prod.uci.cu/.

Pressman, Roger S. 1997. Ingeniería del Software: Un enfoque práctico. *Mikel Angoar.* [En línea] 1997. https://www.google.com/books?hl=en&lr=&id=8UV5jxkuBZIC&oi=fnd&pg=PP3&dq=Las+metodolog%C3%ADas+de+desarrollo+de+software&ots=wJQr_TKmCO&sig=-yLiS3M6nG4L3s1fhV_oRgyDJ0Q.

Pressman, Roger S. 1997. *Ingeniería del Software: Un enfoque práctico.* s.l. : McGraw Hill, 1997. ISBN 9701054733.

Qt Documentation. 2015. Qt Creator Manual. [En línea] 2015. http://doc.qt.io/qtcreator/.

QT, Company. 2015. Framework QT. [En línea] 2015. https://www.qt.io/qt-framework/.

Quintero, Ortiz. 2006. [En línea] 2006.

Roger, Pressman. 1997. Ingeniería del software: Un Enfoque Práctico. *Mikel Angoar.* [En línea] 1997. https://www.google.com/books?hl=en&lr=&id=8UV5jxkuBZIC&oi=fnd&pg=PP3&dq=Las+metodolog%C3%ADas+de+desarrollo+de+software&ots=wJQr_TKmCO&sig=-yLiS3M6nG4L3s1fhV_oRgyDJ0Q.

Romero Peñalver, Gladys Marsi. 2011. "SXP, metodología de desarrollo de software. *Serie Científica 4.11.* [En línea] 2011. http://publicaciones.uci.cu/index.php/SC/article/view/429.

Stroustrup, Bjarne. 2007. El lenguaje de progamación C++. [En línea] 12 de diciembre de 2007. [Citado el: 3 de febrero de 2018.] http://dialnet.unirioja.es/servlet/libro?codigo=370770.

Suyama, Maria. 2015. ¿Qué es WebRTC? [En línea] 22 de enero de 2015. [Citado el: 5 de noviembre de 2018.] https://www.3cx.es/webrtc/que-es-webrtc/.

Tabares, Ricardo Botero. 2010. *GRASP: Más patrones para asignar responsabilidades.* México : s.n., 2010.

Tanenbaum, Andrew S. 2003. *Redes de Computadoras.* Cuarta. s.l. : Pearson, 2003. ISBN 0-13-066-102-3.

Tanenbaum, Andrew.S. 2003. *Redes de Computadoras.* s.l. : Pearson, 2003. ISBN 0-13-066-102-3.

Tedeschi, Nicolás . 2015. ¿Qué es un Patrón de Diseño? *Microsoft.* [En línea] 2015. https://msdn.microsoft.com/es-es/library/bb972240.aspx.

Toledo, Reidel Morales. 2017. *Geo-Informática y Señales Digitales.* UCI : s.n., 2017.

Torrecilla, Pablo. 2012. El Proceso Unificado Ágil: fases y disciplinas. [En línea] 6 de Junio de 2012. [Citado el: 20 de febrero de 2018.] http://nosolopau.com/2012/06/07/mas-sobre-el-proceso-unificado-agil-fases-y-disciplinas/.

Torres, Angel Laguna. 2009. *Propuesta de Servidor Streaming de software libre para la captura y transmisión de video y sonido digital.* UCI : s.n., 2009.

UCI. 2018. UCIENCIA. *Eventos.* [En línea] 2018. http://uciencia.uci.cu http://eventos.uci.cu.

UML, Guión Visual Paradigm for. 2013. 2013.

Valeriano, Serena Castro. 2016. *Cámaras IP.* 2016.

Varona, Florencia Llanos. 2012. ¿Qué es domótica? *CEDOM.* [En línea] ASOCIACIÓN ESPAÑOLA DE DOMÓTICA E INMÓTICA, 13 de febrero de 2012. [Citado el: 22 de octubre de 2018.] http://www.cedom.es/sobre-domotica/que-es-domotica.

Zambrano Álvarez, Miguel Angel. 2011. *¿CÓMO HACER Y DEFENDER UNA TESIS?* San Cristobal : EDITORIAL TEXTO, C.A., 2011.

Zamudio, Fabian Romo. 2017. Camera IP and best RTSP camera buying guide. [En línea] 11 de mayo de 2017. [Citado el: 27 de enero de 2018.] https://reolink.com/rtsp-ip-camera-and-best-rtsp-camera-buying-guide/.

Zenith, Cristian Andrey. 2015. ¿Qué es y cómo funciona el live streaming? [En línea] 25 de enero de 2015. [Citado el: 12 de septiembre de 2016.] http://blogginzenith.zenithmedia.es/que-es-y-como-funcina-el-live-streaming-diccionario/.

Glosario de términos

Módulo: es una porción de un programa de ordenador. De las varias tareas que debe realizar un programa para cumplir con su función u objetivos, un módulo realizará, comúnmente, una de dichas tareas.

Domótica: se llama domótica a los sistemas capaces de automatizar una vivienda o edificación de cualquier tipo, aportando servicios de gestión energética, seguridad, bienestar y comunicación, y que pueden estar integrados por medio de redes interiores y exteriores de comunicación, cableadas o inalámbricas desde dentro y fuera del hogar. Se podría definir como la integración de la tecnología en el diseño inteligente de un recinto cerrado.

***Streaming*:** La retransmisión (en inglés *streaming*, también denominado transmisión de flujo, transmisión por secuencias, lectura en continuo, difusión en continuo o descarga continua) es la distribución digital de contenido multimedia a través de una red de computadoras, de manera que el usuario utiliza el producto a la vez que se descarga.

Vigilancia: es el monitoreo del comportamiento. La vigilancia por sistema es el proceso de monitoreo de personas, objetos o procesos dentro de sistemas para la conformidad de normas esperadas o deseadas en sistemas confiables para control de seguridad o social.

***Plugin*:** En informática, un complemento o plugin es una aplicación (o programa informático) que se relaciona con otra para agregarle una función nueva y generalmente muy específica. Esta aplicación adicional es ejecutada por la aplicación principal e interactúan por medio de la interfaz de programación de aplicaciones.

LAN: *Local Área Network*. Ambiente de comunicaciones locales, típicamente construido en base a cableados dentro de establecimientos de comunicaciones operados privadamente.

***Framework*:** en los sistemas orientados a objeto, un *framework* es un conjunto de clases que encapsulan diseños abstractos de soluciones a un determinado número de problemas en relación. Los objetivos principales que persigue un *framework* son: acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo como el uso de patrones.

Protocolos de comunicación: son como reglas de comunicación que permiten el flujo de información entre computadoras o dispositivos diferentes que manejan lenguajes distintos.

URL: es una cadena que identifica los recursos de una red.

GNU/Linux: El núcleo Linux se complementa con una serie de aplicaciones desarrolladas por el grupo GNU para conformar el sistema operativo *software* libre GNU/Linux. Linux/GNU es además multiusuario, multitarea, multiprocesador, multiplataforma y multilingüe, nacido en la red de redes Internet.

RAM: Siglas de *Random Access Memory*, que en español significa Memoria de Acceso Aleatorio. Es donde el computador guarda los datos que está utilizando en el momento presente. El almacenamiento es considerado temporal por que los datos y programas permanecen en ella mientras que la computadora esté encendida o no sea reiniciada.