Universidad de las Ciencias Informáticas

Facultad 6

Cliente de Grabación del sistema de Video Vigilancia, Suria Vision.

Trabajo de diploma para optar por el Título de Ingeniero en Ciencias Informáticas

Autora: Lisandra Ribeaux Cedeño

Tutor: Ing. Heliodoro Rodríguez Milián

Co-tutor: Ing. Yunier A. Pimienta Fernández

Ciudad de La Habana

Junio de 2012

"Lo fundamental es que seamos capaces de hacer cada día algo que perfeccione lo que hicimos el día anterior. "

Ernesto Guevara de la Serna

Declaración de Autoría

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

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

_____
Firma de la Autora
Lisandra Ribeaux Cedeño

_____
Firma del Tutor
Ing. Heliodoro Rodríguez Milián

Datos de Contacto

Tutor: Ing. Heliodoro Rodríguez Milián

Título: Ingeniero en Ciencias Informáticas.

Categoría Docente: Profesor.

Correo electrónico: hrodriguezm@uci.cu

Graduado de Ingeniero en Ciencias Informáticas en la UCI. Trabajador del Centro GEYSED en el Proyecto Sistema Integral de Vigilancia Suria Vision en el cual se desempeña como Planificador.

Co-tutor: Ing. Yunier Alexander Pimienta Fernández

Título: Ingeniero en Ciencias Informáticas.

Categoría docente: Instructor Recién Graduado

Correo electrónico: yapimienta@uci.cu

Graduado de Ingeniero en Ciencias Informáticas en la UCI en el año 2009. Trabajador del Centro CEDIN en el Dpto Integración y Despliegue, en el cual se desempeña como Especialista General.

Dedicatoria

A mis padres Edelia y Luis, mis más preciados tesoros. Mis ejemplos a seguir, por ser mis guías durante toda mi vida, por todo su amor y cariño. Gracias a ustedes nada me ha faltado, y he podido lograr lo que soy, que no es más que el reflejo del fruto de sus propias vidas.

A novio Alexander (Mi Chiqui), por estar ahí siempre, por tanto amor y dedicación en todos los momentos, un besote.

Agradecimientos

Llegue el más sincero y profundo agradecimiento a mis queridos padres y al resto de la familia que me han apoyado en todo momento y han depositado toda su confianza en mí.

A mi novio, por ser lo mejor que me ha sucedido en la vida. Por su apoyo incondicional, sus desvelos, ayuda y amor. Gracias por quererme.

A mis amigas Damai y Yussel por estar siempre presentes en los momentos de alegría y tristeza. Las quiero mucho.

A todos los que han sido mis compañeros de estudio, por la ayuda ofrecida y por haber tenido la suerte de compartir con ellos.

A todos los profesores que de una forma u otra aportaron su granito de arena a la formación y preparación en el transcurso de mi vida.
A mi tutor Magdiel el cual me ha apoyado y me ha ayudado en la realización de este trabajo.

A todos lo que me ayudaron en la realización de mi tesis. A Henry, Ridel, Riolvi y Zorilin porque siempre que necesitaba ayuda me la brindaron sin condición alguna.

A todos… ¡Muchas Gracias!

Resumen

Debido al auge que ha tomado la necesidad de controlar y proteger los recursos, se incorpora el uso de sistemas informáticos que sirvan de apoyo en la detección de actos delictivos. Un ejemplo que forma parte de este grupo de sistemas es la vigilancia por cámaras. Actualmente en la Universidad de las Ciencias Informáticas se está desarrollando un sistema de video vigilancia (Suria Vision) el cual brinda funcionalidades que también satisfacen las necesidades de las empresas e instituciones cubanas. Pero existe una limitante actualmente, es que este sistema no posee la portabilidad que se requiere, lo cual limita el acceso a este tipo de tecnología.

Para dar solución a dicho inconveniente se presenta el siguiente trabajo de diploma, que tiene como objetivo principal desarrollar una aplicación encargada de gestionar los procesos de grabación de flujos de video, que se integre al sistema Suria Vision. Con éste fin, se propone la utilización del lenguaje C++ y el marco de trabajo Framework Qt.

Como resultado, se pretende obtener una aplicación multiplataforma que permita gestionar los procesos de grabación (manual, por calendario o ante una ocurrencia de un evento) de múltiples flujos de video, proveniente de las cámaras IP en el sistema Suria Vision.

Palabras clave: grabación, multiplataforma, video vigilancia

Abstract

ABSTRACT

Due to the increasing importance of the need to control and protect the resources, it is incorporated the use of informatics systems to support detection of criminal acts. An example that is part of this group of systems is the surveillance via cameras. Nowadays, in the University of Informatics Sciences it is being developed a video surveillance system (Suria Vision) which provides functionalities that also satisfy the needs of Cuban enterprises and institutions. But there is a current limitation, and it is the fact that this system does not have the portability required, which limits the access to this kind of technology.

To give solution to this dilemma it is presented the following work, with the principal objective of developing an application responsible for managing the processes of recording video streams, integrated to the Suria SIV system. With this purpose, it is proposed the use of C++ language and framework Qt.

As a result, it is intended to get a multiplatform application that allows managing recording processes (manual, to schedule or for an occurrence of an event) of multiple video streams in real time, coming from the IP cameras in the Suria SIV system.

Key words: recording, multiplatform, video surveillance

Índice

Índice de Tablas

Índice de Figuras

Índice

Introducción

Hoy día, el crecimiento global de entidades y organizaciones, y su migración hacia instalaciones cada vez más complejas, eleva la necesidad de garantizar la seguridad y protección de los datos y sus bienes materiales. En el marco de esta tendencia global, se exige cada vez más la supervisión general y el despliegue de redes de vigilancia que permita un control centralizado de dichas instalaciones.

Actualmente en el mercado existe una variedad de sistemas de vigilancia, algunos de estos sistemas incluyen un dispositivo simple con cámara de video y grabación, mientras que otros son más complejos, ofreciendo además el uso de sensores de movimiento, las notificaciones por correo electrónico y las alarmas automáticas. Las posibilidades que ofrecen estos sistemas de vigilancia dependerán en gran medida del nivel de la seguridad que demande el consumidor. Por lo que, como en toda tecnología, hay sectores que parecen más interesados y son más partidarios a la adopción e incorporación de dichas herramientas.

Ejemplo de ello es el sector de administraciones públicas la cual hace su utilización en la protección de edificios públicos y el patrimonio y el control del tráfico; la industria para el control de procesos; las relaciones públicas en la prevención de acciones delictivas y de actos terroristas; el sector del comercio minorista: en el conteo de personas, análisis de conductas de compradores y el control del manejo de las cajas; en el medio ambiente para controlar y fotografiar las migraciones de la fauna; entre otros sectores importantes.

Actualmente en Cuba se cuenta con una empresa, DATYS, que comercializa una solución de video vigilancia, la cual, al ser socios de la compañía Axis[1], su precio aumenta demasiado, llegando a ser casi tan cara como las que brindan otras compañías a nivel mundial. Por lo que muchas empresas que necesitan un sistema de este tipo no puedan acceder a éste por su alto precio.

Para darle solución a este inconveniente, el centro de desarrollo GEYSED (Geoinformática y Señales Digitales) de la Universidad de las Ciencias Informáticas, comenzó el desarrollo de un sistema de Video Vigilancia hace pocos años, el cual en la actualidad cuenta con las primeras versiones de la estación de monitoreo, el módulo central gestor de todo el sistema y el de grabación de los flujos de video provenientes de las cámaras IP. Siendo este último, uno de los principales componentes de los sistemas de video vigilancia, el cual permite tener pruebas de los hechos ocurridos.

[1] Axis, compañía sueca de Tecnologías de la Información que ofrece soluciones de video IP dirigidas al mercado profesional, fundada en 1984

Sin embargo se han encontrado problemas en la usabilidad, a la hora del manejo de la grabación de flujos de video provenientes de las cámaras IP, ya que posee poco nivel de comprensión para los usuarios, dificultando así, el manejo de la aplicación al llevar a cabo una determinada acción. Además este sistema, así como sus componentes, están desarrollados sobre la plataforma propietaria .NET[2] y por ende solo se puede utilizar en sistemas operativos Windows, los cuales son propietarios, siendo estos unos inconveniente para el sistema, ya que sería necesaria la adquisición de licencias para su posterior utilización; por lo cual va en contra de las políticas de la universidad y el país de migrar bajo los principios del software libre.

Luego del análisis de la situación actual, surge el siguiente **problema a resolver**: ¿Cómo lograr que el sistema Suria Vision pueda gestionar los procesos de grabación en múltiples plataformas?

Según el problema identificado anteriormente se plantea como **objeto de estudio**: El control del proceso de grabación de flujos de video desde cámaras IP, delimitando como **campo de acción:** el control del proceso de grabación de flujos de video desde cámaras IP existentes en el sistema video vigilancia (Suria Vision) en entornos multiplataforma.

Para resolver el problema se propone el siguiente **objetivo general:** Desarrollar una aplicación multiplataforma que permita gestionar los procesos de grabación en el sistema de video vigilancia Suria Vision.

Para cumplir con el objetivo de esta investigación y resolver la situación problémica planteada, se proponen las siguientes **tareas a resolver**:

- Análisis del estado del arte de los sistemas de video vigilancia existente en la actualidad que permitan gestionar los procesos de grabación.
- Selección de las herramientas y metodologías para el desarrollo de la aplicación informática para la gestión de los procesos de grabación en el sistema de video vigilancia Suria Vision.
- Diseño del prototipo de interfaz para la aplicación.
- Realización del Modelo de Diseño de la aplicación a desarrollar.
- Implementación de la aplicación a partir del diseño realizado.
- Realización de pruebas a la aplicación para detectar errores en su funcionamiento.

[2] Entorno lanzado por Microsoft, diseñado para el desarrollo y ejecución de software en forma de servicios.

Todo lo antes expuesto conduce a plantear la siguiente **Idea a defender**: Si se desarrolla una aplicación multiplataforma que permita gestionar los procesos de grabación en el sistema de video vigilancia Suria Vision, se logrará la gestión de los procesos de grabación del mencionado sistema sobre diferentes sistemas operativos.

Para la realización de las tareas trazadas, se proponen utilizar los siguientes **métodos de investigación**:

Métodos Teóricos:

- *Analítico - sintético*: Para conocer, reflexionar y aumentar los conocimientos acerca del proceso de grabación de video digital desde cámaras IP en entornos multiplataforma a partir del estudio y análisis de documentos y bibliografías de diferentes autores, esto permitirá confeccionar un resumen coherente en el cual se muestren los resultados obtenidos del análisis.
- *Análisis histórico - lógico*: Se utilizará para hacer una valoración sobre la evolución y desarrollo que han tenido los sistemas de grabación de video, lo que permite ver en qué etapa se encuentran dichos sistemas y cuál es la tecnología más factible para el desarrollo de los mismos.

Métodos Empíricos.

- *Observación*: Permitirá realizar valoraciones y obtener informaciones a partir del funcionamiento de sistemas similares al desarrollado anteriormente, lo que da una visión de cómo tiene que ser el sistema a realizar en su forma externa.

Se espera como **posibles resultados:**

1. Informe de investigación que recoge los detalles de la solución implementada.
2. Aplicación informática capaz de gestionar los procesos de grabación en el sistema, Suria Vision.

El presente trabajo de diploma consta de 4 capítulos:

CAPÍTULO 1: Fundamentación teórica

Se mencionan algunos conceptos asociados con el tema a tratar, se describen algunas de las características de los sistemas de grabación de video digital, así como la actualidad de los mismos a

nivel nacional e internacional y la situación en la Universidad de las Ciencias Informáticas en cuanto a este tema.

CAPÍTULO 2. Herramientas y tecnologías seleccionadas

Se explican las metodologías y herramientas seleccionadas para el desarrollo de la aplicación.

CAPÍTULO 3. Características del sistema

Se describe la solución propuesta. Se explica cómo se lleva a cabo el proceso de diseño, la implementación del software y sus características generales mostrando algunos diagramas para un mejor entendimiento.

CAPÍTULO 4. Implementación y pruebas de software a la solución propuesta

Se describen los casos de prueba desarrollados por cada caso de uso y los resultados obtenidos luego de su aplicación.

CAPÍTULO 1. Fundamentación teórica

1.1. Introducción

En este capítulo se abordan aspectos relacionados con los sistemas de control del proceso de grabación de video. Además, se da una visión acerca del estado del arte de los sistemas más avanzados de gestión de grabaciones en los sistemas de video vigilancia y algunas de las aplicaciones existentes más conocidas de software libre, así como las diferentes organizaciones que son vanguardia en este mercado.

1.2. Conceptos asociados.

El control del proceso de grabación de flujos de video desde cámaras IP[3] tiene como objetivo brindar información al usuario sobre las grabaciones que se realizan, ante una petición determinada o por configuración, que puede ser por horarios determinados o por la ocurrencia de algún evento. Además de realizar reglas y tareas de grabación, así como permitir iniciar o detener las grabaciones. Anteriormente, el proceso de grabación de flujos de video era llevado a cabo por tecnologías analógicas, pero los dispositivos y transmisiones analógicos tienen, sin embargo, algunas desventajas. La más popular de ellas es la interferencia.

Las *señales analógicas* es un tipo de señal donde la información está codificada en las diferencias de amplitud y frecuencia que ésta maneja pudiendo ser periódica o no periódica. Una *tecnología analógica* se refiere a la transmisión electrónica que se consigue añadiendo señales de frecuencia o amplitud variable a ondas transportadoras de corriente electromagnética alterna con una frecuencia dada[1].

La información utilizada en estas tecnologías es llamada *información analógica*, donde la señal de video analógico es la conversión de los cambios de la intensidad de la luz en señales variables de intensidad eléctrica. Es decir, plasmar la realidad mediante señales electromagnéticas en una cinta.[2]

Con el auge del desarrollo de la tecnología de cómputo y telecomunicaciones, así como el propio avance de la sociedad de la información, trajo consigo el avance de la tecnología IP. Los últimos

[3] IP (Protocolo de Internet): protocolo de capa de red con más amplio apoyo para la Internet. Éste define y enruta datagramas a través de Internet y ofrece un servicio de transporte no orientado a conexión.

avances han hecho posible conectar cámaras directamente a una red de ordenadores basada en el protocolo IP para la comunicación entre las unidades que componen a un sistema. La tecnología IP permite al usuario tener una cámara en una localización y ver el video digital en tiempo real desde otro lugar a través de la red o de Internet.[3]

El término *video digital* se puede definir como la representación digital de la señal de video analógico, es una secuencia de imágenes o marcos[4] que son almacenadas y reproducidas a una velocidad y en un orden determinado. Dentro del proceso de grabación este término es definido como un flujo de video. Obtenido desde cámaras IP o cámaras de red, los cuales se conectan directamente a una conexión LAN[5] de la instalación de Internet[6]. A estas cámaras se les asigna una dirección IP, facilitando su acceso desde cualquier navegador al disponer de varios menús que permiten diversas funcionalidades.

Una de estas funciones es la grabación. Se entiende por *grabación* como el proceso de capturar datos o convertir la información a un formato almacenado en un medio de almacenamiento. La calidad de la grabación dependerá del nivel de calidad del dispositivo de grabación y las características de los flujos de video obtenidos de las cámaras IP dentro del sistema.

Un *sistema informático* es un conjunto de funciones que operan en armonía o con un mismo propósito, y que puede ser ideal o real. Por su propia naturaleza, un sistema *informático* posee reglas o normas que regulan su funcionamiento, por lo que puede ser entendido, aprendido y enseñado. [4]

Cualquier sistema es más o menos complejo pero debe poseer una coherencia discreta acerca de sus propiedades y operaciones. En general, los elementos o módulos de un sistema interactúan y se interrelacionan entre sí. Una de las características esperadas de los sistemas es la capacidad de que puedan funcionar en diversas plataformas[7].

Los términos descritos anteriormente son aspectos importantes dentro de un *sistema de Video Vigilancia IP o Vigilancia IP*, siendo este una efectiva solución de seguridad que ofrece monitorización y control avanzado desde uno o varios puntos de control. Esta solución se basa en la tecnología IP. La

[4] Se refiere a un fotograma o cuadro, una imagen particular dentro de una sucesión de imágenes que componen una animación.

[5] Red de Área Local (LAN): son redes localizadas dentro de un solo edificio o campus de hasta unos pocos kilómetros de tamaño.

[6] Se trata de una red de redes de ordenadores a escala mundial que permite transmitir datos de uno a otro ordenador.

[7] Es un sistema que sirve como base para hacer funcionar determinados módulos de hardware o software con los que es compatible. En su forma más simple consiste únicamente de un sistema operativo.

cual consta de cámaras que utilizan el protocolo de internet para transmitir datos de imagen y señales de control por una red inalámbrica o Ethernet.

1.3. Evolución de los sistemas de grabación de video.

Los cambios y transformaciones acontecidas en el orbe en los últimos años en las tecnologías, la necesidad del hombre de encontrar vías más seguras para garantizar la seguridad y protección de los datos y sus bienes materiales, ha conducido a un desarrollo progresivo en los sistemas de video vigilancia y el control centralizado de los mismos. Al mismo tiempo que los sistemas de video vigilancia han ido evolucionando, las maneras de grabar los videos obtenidos de las cámaras también han tenido un constante desarrollo. Estos avances han contribuido en gran medida a popularizar a los sistemas de Video vigilancia, permitiendo salvaguardar las pruebas de un hecho delictivo; las cuales son usadas asiduamente como evidencia ante los tribunales.

A partir de la década de los setenta se comenzaron a utilizar los Sistemas de Video Vigilancia en el sector privado y en la década de los ochenta, era común el uso de los grabadores de video analógico en cinta. Estas tecnologías tenían varias desventajas, una de ellas era el poco tiempo de almacenamiento que tenía cada cinta, por lo que debían ser cambiadas periódicamente; por otra parte, solo permitía almacenar videos desde una sola cámara, por lo que era necesario utilizar una grabadora por cada una de ellas. [5]

Para solucionar estos problemas se crearon los multiplexadores[8] digitales los cuales podían grabar a uno o menos cuadros (*frames*[9]) por segundo, reduciendo así el espacio en cinta, necesario para almacenar el video. Esto posibilitaba no tener que cambiar la cinta hasta dentro de cuatro o cinco días, sin embargo esta tarea aún debía ser realizada de forma manual, además se perdían detalles de incidentes grabados. Los multiplexadores además podían tomar los videos de varias cámaras y juntarlos en uno solo, esto hacía que el tamaño de las imágenes fuera muy reducido; sin posibilidades de aumentarla, lo que dificultaba distinguir los detalles de la escena. [5]

Los sistemas de Grabación de Video Digital (DVR, *Digital Video Recorder*) surgen a principios de los noventa. Su aspecto externo era similar al de los grabadores de cinta que eran operados por botones o mediante control remoto. Desafortunadamente, debido a limitaciones tecnológicas los primeros DVRs

[8] Dispositivo que puede recibir varias entradas y transmitirlas por un medio de transmisión compartido.

[9] Frame en inglés, es un fotograma o cuadro, una imagen particular dentro de una sucesión de imágenes que componen una animación.

no eran fiables y brindaban poca calidad de video, por ello no eran lo suficientemente potentes para brindar soluciones que pudieran manejar una gran cantidad de cámaras. [5]

En la actualidad los DVRs se han dividido en tres categorías muy especializadas:

- **Dedicadas, Autónomas. (*Non- PC based*)**

Unidad compacta con entradas análogas de video (conector BNC[10]), que físicamente no tiene parecido a una ordenador, pero que en su arquitectura electrónica está basada en un esquema de cómputo (fuente, tarjeta madre, disco duro, puertos de comunicaciones, tarjeta de captura de video, dispositivos de entrada y salida). Estas soluciones utilizan lo que se conoce como "Chip ASIC" (Circuito Integrado de Aplicaciones Específicas, por sus siglas en inglés) estos chips están diseñados para un propósito específico por lo que no pueden ser reprogramados para agregarle nuevas funcionalidades.[6]

Fig. 1 Esquema de un sistema que incluye un DVR Non-PC based.

- **Basadas en PC. (*PC based)***

Los DVR basados en PC requieren de una tarjeta de video que se instala físicamente dentro de una computadora personal, esta instalación es más sencilla ya que actualmente los periféricos de una PC son más conocidos y su configuración es basada en software. Esta tecnología suele ser más flexible y adaptable, pues permite actualizar las técnicas de compresión y poseen una interfaz que hace más cómoda la interacción del usuario con la misma, facilitando así, la gestión del video.[6]

[10] BNC (del inglés Bayonet Neill-Concelman): es un tipo de conector de cable coaxial para cables de video y sincronismos.

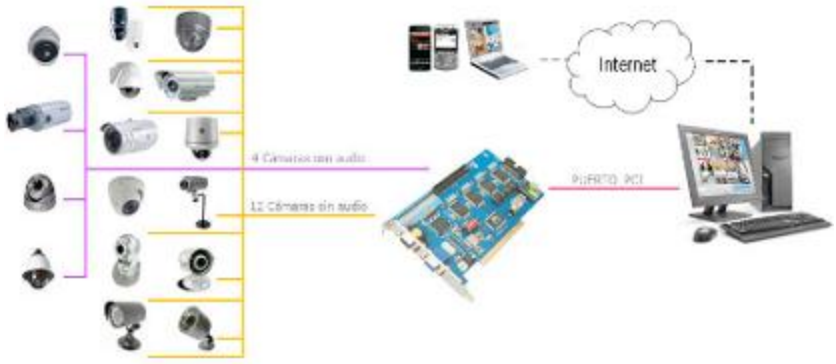


Fig. 2 Esquema de un sistema que incluye un DVR basado en PC.

- **Grabadores de Video basados en Red, NVR. (*Network Video Recorder*)**

Los grabadores NVR son una alternativa profesional para la gestión y grabación de cámaras IP, permitiendo prescindir de un ordenador como medio de grabación y gestión de las cámaras IP.

Un NVR está diseñado para ofrecer un rendimiento óptimo para un conjunto de cámaras y normalmente es menos escalable que un sistema basado en servidor de una computadora personal. Esto permite que la unidad resulte más adecuada para sistemas más pequeños donde el número de cámaras se encuentra dentro de los límites de la capacidad de diseño de un NVR. Un sistema de video IP que utiliza cámaras IP añade varias ventajas, por ejemplo: calidad de imagen constante, mayor flexibilidad y escalabilidad en el sistema.[7]

Fig. 3 Esquema de un sistema que incluye un NVR.

1.4. Descripción actual del dominio del problema

Actualmente en el sistema de Video Vigilancia en la fase de Requisitos y Análisis se definió el control de proceso de grabación de video. Según el levantamiento de información realizado, los clientes necesitan planificar las capturas de señales emitidas por cámaras IP, especificando parámetros para estas actividades a partir de las cuales se efectuará la grabación de video:
Regla de grabación: define en qué momento se graba desde una cámara y durante qué espacio de tiempo. Es repetitiva pudiendo tener una frecuencia mensual, semanal o diaria.
Tarea de grabación: define en qué momento se graba desde una cámara especificando la fecha y hora de inicio y la fecha y hora de fin.

Este sistema cuenta con distintos módulos como son el Gestor, Visor, Recuperador y el Grabador:

- El ***Gestor*** es el módulo fundamental del sistema Suria Vision, su tarea principal es gestionar todo el tráfico de información.
- El ***Visor*** es la estación de visualización del sistema, que permite el manejo de las cámaras internamente.
- El módulo ***Recuperador*** es el encargado de recuperar los datos almacenados por el Grabador y hacer búsquedas en estos. El Recuperador tiene su propio visualizador, los videos recuperados se visualizan por éste.
- El ***Grabador*** es el encargado de la grabación de video obtenido de las cámaras IP, bajo petición de un cliente determinado o por configuración, que puede ser por horarios determinados o por la ocurrencia de algún evento. Todo el funcionamiento del Grabador, así como su colaboración con otros módulos, es coordinado por el Gestor.

El módulo de grabación está compuesto por varias aplicaciones: Servidor de Grabación, Plugins de Grabación y el Cliente.

- ***El servidor de grabación***: consistente en una aplicación tipo servicio de Windows, que atiende los pedidos de grabación apoyándose en un grupo de librerías denominadas plugins de grabación.
- **Editor de Configuración,** es una aplicación sencilla que permite configurar los parámetros de grabación del Servidor.
- ***Un plugin de grabación***: implementa la funcionalidad de grabación para uno o varios tipos de cámaras, usando una tecnología de grabación específica.

- ***El cliente***: es una aplicación gráfica que funciona como la interfaz de usuario principal del módulo, permitiendo la planificación de las grabaciones por calendario.

El proceso de grabación en el sistema se efectúa a partir de la inserción de reglas de grabación, las cuales pueden generar una o varias tareas de grabación, luego estas a su vez generan una grabación respectivamente. Los pedidos de grabación son atendidos en el Servidor de Grabación que ejecuta las tareas de grabación como tal, almacenando las señales capturadas desde las cámaras IP. Este proceso es coordinado por el Gestor, el cual dispone de todo el funcionamiento del sistema.

Específicamente la planificación del proceso de grabación llevado a cabo por el Cliente de Grabación ocurre de la manera antes mencionada; los usuarios pueden crear reglas de grabación para cada cámara seleccionada, definiendo los parámetros de grabación correspondientes; además de comenzar la grabación manualmente. Se tiene una versión sobre el marco de trabajo .NET.

1.5. Situación problemática

La versión sobre .Net del Cliente de Grabación, actualmente tiene un gran número de las funcionalidades iniciales identificadas y aún está en desarrollo. Sin embargo, este presenta varios inconvenientes como, limitante en el entorno de aplicación pues debe ser desplegada en sistemas operativos Windows. Entre los objetivos de la Universidad de las Ciencias Informática es lograr la informatización del país e insertarse en el mercado internacional de software y que lo que se posee hasta ahora en el Sistema Video Vigilancia no cumple con las expectativas debido a que este es un elemento que todos los clientes potenciales del sistema admitirán, sobre todo si se tienen en cuenta las políticas de soberanía tecnológica que ha implementado Cuba.
Otro de los inconvenientes que presenta este Cliente de Grabación, es la interfaz de usuario de la aplicación. Esta no presenta una alta usabilidad ya que los componentes visuales utilizados no permiten la fácil e intuitiva interacción del usuario con el mismo, además los usuarios que presentan poco conocimiento de su funcionamiento, les dificulta su uso por su bajo nivel de comprensión al llevar a cabo una determinada acción.
Las problemáticas identificadas son: la versión actual es privativa limitando su posibilidad de mercado y la interfaz de usuario no es amigable influyendo en la usabilidad de la aplicación.

1.6. Análisis de soluciones existentes.

1.6.1. Axis Communications

Compañía sueca de Tecnologías de la Información que ofrece soluciones de video IP dirigidas al mercado profesional, fundada en 1984. La compañía es líder del mercado del video IP, conduciendo el cambio del video vigilancia analógica hacia las soluciones digitales. El nombre de su principal producto es Axis Camera Station.

Esta solución es un software de vigilancia IP que funciona con cámaras de red y codificadores de video de Axis y proporciona funciones de supervisión de video, grabación y gestión de eventos. Los usuarios pueden realizar una grabación de video continua, programada, activada por alarma y/o por detección de movimiento.[8] Además, este sistema cuenta con un calendario gráfico, el cual permite planificar grabaciones de manera visual.

Fig. 4 Sistema Axis Camera Station.

1.6.2. SCATI LABS, S.A.

Empresa fundada en 1998, dedicada a la fabricación de equipos de grabación digital de videos y a la realización de consultorías de alto valor añadido, para el desarrollo de proyectos en los que se requiera una gestión de audio y video específica. Esta presenta al mercado su completa Suite de Gestión Remota VisionSurfer para la explotación y mantenimiento eficaz de grandes parques de videograbación digital o de instalaciones singulares. Esta suite, está compuesta por un potente conjunto de aplicaciones integrables entre sí, tales como grabación, explotación, gestión local y/o remota, control y monitorización. Aunque sus soluciones son muy caras brindan la perfecta

combinación de robustez, funcionalidad y servicio. El módulo que se encarga de la vigilancia IP es el SurferWatcher, pero ningún módulo se comercializa de manera independiente.[9]

Fig. 5 Sistema SCATI LABS S.A SurferWatcher.

1.6.3. ALNET SYSTEM Inc.

Empresa fundada en 1989, la cual trata el desarrollo de los sistemas de grabación digital y la identificación y reconocimiento de imagen. Su principal producto es el VDR (grabadora de video digital), además de los sistemas digitales de grabación de video que ofrece el sistema ERP (sistema de identificación y reconocimiento de matrículas).

Además la empresa cuenta con NetStation, avanzado sistema de red de grabación de audio y video que interactúa tanto con cámaras IP como análogas. Es un sistema híbrido, es decir, que a partir de 64 cámaras, 16 pueden ser cámaras analógicas y 48 cámaras IP. Cuenta con un número ilimitado de conexiones de cliente remoto y apoyo a todos los fabricantes principales de cámaras IP. Los usuarios pueden planificar una grabación de video forma continua, programada o activada por alarma.[10]

Fig. 6 Sistema ALNET SYSTEM NetSation.

1.6.4. VIVOTEK INC.

Empresa establecida en 2000, se ganó su lugar rápidamente como fabricante líder en la industria de la vigilancia IP. VIVOTEK se especializa en la integración de componentes audiovisuales en la operación de redes. Ofrece un software de gestión y grabación central, llamado VAST, siendo un sistema fiable y fácil de administrar, además proporciona una gran escalabilidad para las diversas aplicaciones de vigilancia IP.

La empresa proporciona 3 series de software de grabación: ST2403, ST3402 y ST7501, ambos gratuitos. El ST2403 destinado a las cámaras de red y servidores de video VIVOTEK de la serie 2000 y el ST7501 y ST3402 destinado a las cámaras de red de serie 3000, 6000, 7000 y 8000 y servidores video de serie 3000.[11]

Fig. 7 Sistema VIVOTEK ST7501.

1.6.5. DATYS, Tecnología y Sistemas

Empresa creada desde el año 2005, esta adquiere personalidad jurídica propia el 16 de abril de 2007, para asumir el desarrollo de sistemas complejos y especializados, de alta demanda en el mercado. Uno de sus productos es XYMA SAFE VISION, el cual permite el monitoreo y control en tiempo real y de forma histórica de cada uno de los movimientos que ocurren en áreas e inmuebles. Este producto es un software de video vigilancia profesional basado en tecnología IP, con un alto grado de modularidad, adaptable a una gran cantidad de entornos, flexible, escalable y que admite también el uso de tecnologías analógicas. Además permite ver y grabar a través de múltiples cámaras continuamente.[12] Esta aplicación cuenta con el inconveniente que requiere de sistemas operativo Windows para su implantación.

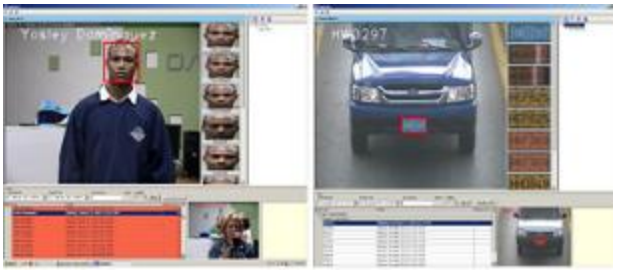


Fig. 8 Sistema XYMA SAFE VISION

1.7. Sistemas en el contexto del Software Libre

Luego de analizar la existencia de empresas dedicadas a la fabricación de software de vigilancia por cámaras, tanto a nivel internacional como nacional, se observa que han sido muchos los productos desarrollados sobre esta temática para aplicarse en diferentes sectores. En el contexto de software libre se pueden mencionar algunas de las aplicaciones existentes más utilizadas:

- **Wxcam**: aplicación de cámara web para Linux cuando detecta cualquier movimiento. Es compatible con grabación de video sin audio, en un formato avi[11] sin comprimir y sin pérdidas, y con audio en formato Xvid[12] con pérdida, pero con un gran ahorro de espacio en disco duro. Además, es compatible tanto con video4linux[13] 1 y 2 API[14], por lo que trabaja en un gran número de dispositivos.[13]
- **Devolution Security:** soporta hasta 16 cámaras de vigilancia con detección de movimientos, streaming unicast y multicast[15]. Inicia el proceso de grabación al detectar movimientos, con una calidad de 25 fotogramas por segundo y formato MPEG-4[16]. Además, ofrece diferentes temas de escritorio y layouts para las cámaras. Soporta multicast a través de una red local o unicast a una dirección IP.[14]
- **ZoneMinder:** es un conjunto integrado de aplicaciones que proporcionan una solución completa de vigilancia que permite la captura, análisis, registro y monitoreo de cualquier circuito cerrado de televisión o cámaras de seguridad conectado a una máquina basada en Linux. Está

11 AVI (del inglés Audio Video Interleave) es un formato contenedor de audio y video.

12 Es un códec de código abierto, gratuito, que sigue el estándar de codificación MPEG-4.

13 Video4linux o V4L: es una API de captura de video para Linux.

14 API (del inglés Application Programming Interface): especificación de una librería o utilidad que documenta su interfaz y permite su uso sin conocimiento de su interior.

15 Distribución de video ya sea desde un único emisor a un único receptor (unicast) o a varios de ellos (multicast)

16 Es un formato contenedor especificado como parte del estándar internacional MPEG (en inglés: Moving Picture Experts Group) para almacenar formatos audiovisuales.

diseñado para funcionar en distribuciones que apoyan el video para Linux (V4L) y puede soportar múltiples cámaras sin pérdida aparente de rendimiento.[15]

- **Motion:** es un programa que supervisa la señal de video de una o más cámaras, utilizado el V4L para recibir imágenes, y es capaz de detectar si una parte significativa de la situación ha cambiado, es decir, puede detectar el movimiento. Una vez detectado, desde ese momento, comienza a grabar o a generar eventos previamente configurados.[16]

1.8. Conclusiones parciales

El estudio de los diferentes sistemas de video-vigilancia en el control del proceso de grabación con funcionalidades similares a las requeridas por el producto a desarrollar, así como el estudio de los distintos temas relacionados con los sistemas de grabación de video, ha permitido obtener un conjunto de conocimientos necesarios para el desarrollo de la aplicación. El análisis permitió concluir que ninguna satisface todos los requisitos de la aplicación que se desea implementar; pudiéndose coger como referencia algunas de ellas para intégrasela a la aplicación a desarrollar, como es la incorporación de un calendario grafico para la planificación de grabaciones.

Además el análisis del proceso de grabación de video en la Universidad de las Ciencias Informáticas llevado a cabo por el sistema Suria Vision, el cual ha permitido conocer los elementos importantes de éste; así como el inconveniente que este presenta, el cual se pretende dar solución.

CAPÍTULO 2. Herramientas y tecnologías seleccionadas

2.1. Introducción

En este capítulo se hace referencia a la selección de las herramientas y tecnologías a emplear, las cuales se ajustan a la solución adecuada. Definiendo la metodología de desarrollo, el lenguaje de modelado, el lenguaje de programación y el gestor de Base de Datos, así como se tendrán en cuenta las herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Computadora) como comúnmente se llaman e IDEs para el desarrollo, entre otras que son importantes dentro del desarrollo de la aplicación.

2.1.1. Metodologías de Desarrollo: RUP

Se entiende por metodología de desarrollo una colección de documentación formal referente a los procesos, las políticas y los procedimientos que intervienen en el desarrollo del software. La finalidad de una metodología de desarrollo es garantizar la eficacia (p.ej. cumplir los requisitos iniciales) y la eficiencia (por ej. minimizar las pérdidas de tiempo) en el proceso de generación de software.[17]

Para la selección de la metodología de desarrollo se estableció anteriormente una comparación entre Proceso Unificado Racional (*Rational Unified Process, RUP*) y Programación Extrema (*Extreme Programming, XP*), siendo estas las tecnologías más famosas de su tipo. Entre estas metodologías se decidió escoger RUP debido a que esta es una metodología usada y probada ya por la universidad y además por las ventajas que brinda, esta selección es debido a que en caso de utilizar la metodología XP, se debe tener a un integrante por parte del cliente trabajando con el equipo de desarrollo, lo cual se ha querido pero ha sido imposible.

Además, esta selección es debido a su eficaz adaptación a las necesidades exclusivas del proyecto, la cual permite enfocarse en trabajar de forma organizada, donde se controla y documenta todo lo relacionado con el proyecto que se está realizando. Es un proceso que proporciona un acercamiento disciplinado a la asignación de tareas y responsabilidades; intenta integrar todos los aspectos a tener en cuenta durante todo el ciclo de vida del software, con el objetivo de hacer abarcables tanto pequeños como grandes proyectos software.

RUP es un proceso en el que se define claramente quien, cómo, cuándo y qué debe hacerse en el proyecto. Se caracteriza por ser iterativo e incremental, estar centrado en la arquitectura y dirigido por los casos de uso.

- *Dirigido por casos de uso*[17]: Significa que el proceso de desarrollo sigue un hilo, avanza a través de una serie de flujos de trabajo que parten de los casos de uso.[18]
- *Iterativo e incremental*: Posibilita que se pueda dividir el trabajo en partes más pequeñas o en miniproyectos. Cada miniproyeto es una iteración que resulta en un incremento. Las iteraciones hacen referencia a pasos en flujo de trabajo, y los incrementos, al incremento del producto.[18]
- *Centrado en la arquitectura:* Facilita la visión completa del sistema. La arquitectura incluye los aspectos estáticos y dinámicos más significativos del sistema, los cuales son importantes para su construcción, o sea, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente.[18]

2.1.2. Lenguaje de modelado: UML

El Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, Unified Modeling Language) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software.

- UML permite ***visualizar*** de una forma gráfica un sistema de forma que otro lo puede entender.
- UML permite ***especificar*** cuáles son las características de un sistema antes de su construcción.
- A partir de los modelos especificados se pueden ***construir*** los sistemas diseñados.
- Los propios elementos gráficos sirven como ***documentación*** del sistema desarrollado que pueden servir para su futura revisión.

Se puede aplicar en una gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como el Proceso Unificado Racional), pero no especifica en sí mismo qué metodología o proceso usar.[19]

UML también contiene construcciones organizativas para agrupar los modelos en paquetes, lo que permite a los equipos de software dividir grandes sistemas en piezas de trabajo, para entender y controlar las dependencias entre paquetes, y para gestionar las versiones de las unidades del modelo,

[17] Casos de uso: es un fragmento de funcionalidad del sistema que proporciona al usuario un resultado importante. Estos representan los requisitos funcionales.

en un entorno de desarrollo complejo. Otra característica de este modelado visual es que sea independiente del lenguaje de implementación, de tal forma que los diseños realizados usando UML se puedan implementar en cualquier lenguaje que soporte las posibilidades de UML (principalmente lenguajes orientados a objetos, como C++). Permitiendo así una mayor escalabilidad del software y reutilización del diseño.

Debido a que la metodología de desarrollo seleccionada (RUP) utiliza UML como lenguaje para la modelación, se decide utilizar el mismo como lenguaje de modelado del sistema Cliente de Grabación, teniendo en cuenta además, las ventajas que propicia este lenguaje de modelado.

2.1.3. Herramienta CASE: Enterprise Architect

Una herramienta CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Computadora), es un conjunto de métodos, utilidades y técnicas que facilitan la automatización del ciclo de vida del desarrollo de sistemas de información, completamente o en alguna de sus fases.[20] Se puede concluir que una herramienta CASE es un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un software; aumentando la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero.

Enterprise Architect 7.0 (EA) es una herramienta CASE que combina el poder de la última especificación UML 2.1 con alto rendimiento, interfaz intuitiva, para traer modelado avanzado al escritorio, y para el equipo completo de desarrollo e implementación. Entre las principales características que éste brinda se pueden mencionar:[21]

- Soporta los 13 diagramas de UML 2.1.
- Interfaz de usuario intuitiva.
- Documentación flexible y comprensible.
- Puede generar código fuente C++, entre otros lenguajes de programación.
- Generación directa de los scripts DLL para crear las estructuras de Base de Datos.
- Soporta Control de Versiones.
- Soporte para Prueba.
- Soporte para Administración de Proyecto.

Se decide seleccionar Enterprise Architect como herramienta CASE debido a sus características y además por ser una herramienta multi-usuario, diseñada para ayudar a construir software robusto y fácil de mantener. Ofrece salida de documentación flexible y de alta calidad.

2.1.4. Lenguaje de programación: C++

Un lenguaje de programación es un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Es utilizado para controlar el comportamiento físico y lógico de una máquina.[22]

Por políticas de la universidad y del país de contar con aplicaciones desarrolladas bajo software libre y por decisión del proyecto de contar con una aplicación rápida, robusta, multiplataforma y que requiriera el mínimo de esfuerzo en cuanto a tiempo de desarrollo, se decide utilizar el framework de desarrollo Qt el cual se basa en C++.

C++ es un lenguaje de programación que está considerado por muchos como el lenguaje más potente, debido a que permite trabajar tanto a alto como a bajo nivel. Éste se plantea como una mejora del lenguaje de programación C, pero en la actualidad es un lenguaje independiente, versátil y general, además es compatible con diversos sistemas operativos.

Una particularidad del C++ es la posibilidad de redefinir los operadores (sobrecarga de operadores), y de poder crear nuevos tipos que se comporten como tipos fundamentales. Dentro de las principales características de este lenguaje de programación está el soporte para programación orientada a objetos.

Se decide utilizar C++ como lenguaje de programación para el desarrollo del sistema debido a:

- Es un lenguaje compilado, lo que le hace muy rápido.
- Es un lenguaje que permite el manejo de todo tipo de estructuras de datos por lo que es capaz de resolver todo tipo de problemas.
- Gracias a su capacidad de uso de objetos y clases, facilita la reutilización del código.
- Permite incorporar librerías, según la necesidad del desarrollo de aplicaciones.
- Es muy utilizado en el mundo por lo que existe abundante información sobre su uso.

2.1.5. Framework para el desarrollo: Qt

Un marco de trabajo (*framework*) en el desarrollo de software, es una estructura de soporte definida mediante la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.[23]

Representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio.

Qt es un marco de trabajo multiplataforma, que se utiliza para el desarrollo de aplicaciones, está escrito en C++, sin embargo, es posible utilizar Qt con otros lenguajes a través de ligaduras (bindings[18]). Funciona en todas las principales plataformas, y tiene un amplio apoyo. El API de la biblioteca cuenta con métodos para acceder a bases de datos mediante SQL, así como uso de XML[19], gestión de hilos, soporte de red, una API multiplataforma unificada para la manipulación de archivos y una multitud de otros para el manejo de ficheros, además de estructuras de datos tradicionales.[24]

Se selecciona Qt como marco de trabajo para el desarrollo de la interfaz gráfica de la aplicación debido a que está distribuida bajo los términos de GNU Lesser General Public License (Licencia Pública General Reducida de GNU), siendo un software libre y de código abierto.[24] Además, posee características que la convierten en una herramienta de desarrollo muy potente, tales como:

- Es rápido de aprender y fácil de usar.
- Se cuenta con una amplia internacionalización de apoyo.
- Permite escribir aplicaciones avanzadas e interfaces de usuario, y desplegarlas a través de escritorio y sistemas operativos embebidos sin tener que reescribir el código fuente.
- Proporciona un grupo de elementos gráficos predefinidos, llamados “widgets” personalizables, lienzo gráfico, el motor de estilo y lo necesario para construir interfaces de usuario modernas.

[18] Ligadura o binding, es una adaptación de una biblioteca para ser usada en un lenguaje de programación distinto de aquél en el que ha sido escrita.

[19] Siglas en inglés: eXtensible Markup Language (Lenguaje de marcado extensible). Diseñado especialmente para la web.

2.1.6. Entorno de desarrollo: Qt Creator

Un entorno de desarrollo integrado (en inglés, *Integrated Development Environment* o *IDE*), es un programa compuesto por una serie de herramientas que utilizan los programadores para desarrollar código. Esta herramienta puede estar pensada para su utilización con un único lenguaje de programación o bien puede dar cabida a varios de estos.[25]

Se decide seleccionar Qt Creator por ser un potente IDE multiplataforma, diseñado a la medida de los desarrolladores de Qt, para hacer que el desarrollo en C++ de las aplicaciones Qt sea más rápido y fácil. Permite a los desarrolladores crear aplicaciones para escritorio y plataformas de dispositivos móviles. Otras de las principales características de Qt Creator son:[26]

- Posee un avanzado editor de código C++ y JavaScript.
- Posee interfaz de usuario integrada de diseño.
- Herramienta para proyectos y administración.
- Ayuda integrada sensible al contexto.
- Depurador visual.
- Resaltado y auto-completado de código.
- Soporte para refactorización de código.
- Simulador de interfaces de usuario móviles.
- Soporte para el control de versiones, como SubVersion.

2.1.7. Tecnología para la comunicación: XML RPC

XML-RPC es a protocolo de llamada a procedimiento remoto (*Remote Procedure Call),* que utiliza XML para codificar sus llamadas y HTTP como mecanismo del transporte. Es un protocolo muy simple ya que solo define unos cuantos tipos de datos y comandos útiles, además de una descripción completa de corta extensión. La simplicidad del XML-RPC contrasta con la mayoría de protocolos RPC que tiene una documentación extensa y requiere considerable soporte de software para su uso[27].
Existen implementaciones para varios sistemas operativos, lenguajes de programación, licencias comerciales y de software libre: C/C++, Microsoft .NET, Perl, PHP, etc.

2.1.8. Control de versiones: SubVersion

Un sistema de control de versiones es un software que administra el acceso a un conjunto de ficheros, y mantiene un historial de cambios realizados. El control de versiones es útil para guardar cualquier documento que cambie con frecuencia, como código fuente, documentación o ficheros de configuración.[28] Estos ofrecen una alternativa más poderosa que guardar archivos de seguridad.

SubVersion es un software para el control de versiones distribuido bajo licencia libre de tipo Apache/BSD[20], es multiplataforma, está escrito en lenguaje C, preparado para funcionar en red y diseñado específicamente para reemplazar al popular CVS[21], el cual posee varias deficiencia.[29] SubVersion trabaja replicando el modelo cliente/servidor, es decir uno o más clientes se conectan a un servidor central que tiene la última copia del proyecto, como también copias de sus versiones anteriores. Además, proporciona otras características en su utilización:[30]

- Mantiene versiones no sólo de archivos, sino también de directorios.
- Mantiene versiones de los metadatos asociados a los directorios.
- Además de los cambios en el contenido de los documentos, se mantiene la historia de todas las operaciones de cada elemento, incluyendo la copia, cambio de directorio o de nombre.
- La creación de ramas y etiquetas es una operación más eficiente; el coste de ramificación y etiquetado no necesita ser proporcional al tamaño del proyecto.
- Mejor uso del ancho de banda, ya que en las transacciones se transmiten sólo las diferencias y no los archivos completos.

Se selecciona SubVersion por sus amplias características, además es respaldada por toda una comunidad de desarrollo especializada en el tema. Tiene un funcionamiento muy independiente de los IDEs de desarrollo. Igualmente es la tecnología recomendada por la universidad teniendo varios años de experiencia en su uso.

[20] Se refiere a dos tipos de licencia, Apache la cual permite al desarrollador hacer lo que desee con el código fuente, incluso productos propietarios, sin entregar el código fuente y la licencia BSD (del inglés Berkeley Software Distribution o Distribución de Software de Berkeley), que no impone ninguna restricción a los desarrolladores de software en lo referente a la utilización posterior del código en derivados y licencias de estos programas.

[21] CVS(del inglés Concurrent Versions System) es una aplicación informática que implementa un sistema de control de versiones.

2.2. Conclusiones parciales

La caracterización de las tecnologías y herramientas seleccionadas para la construcción y desarrollo de la aplicación propuesta, permitió conocer las ventajas que ofrecen las mismas para el desarrollo, viabilizando un desarrollo eficaz y efectivo del sistema. De esta manera se está en condiciones de realizar el desarrollo de la aplicación, utilizando las herramientas y tecnologías para dar cumplimiento a las funcionalidades del sistema.

CAPÍTULO 3. Diseño de la solución propuesta

3.1. Introducción

En este capítulo se presentan los temas referentes al diseño e implementación del sistema. Se hace referencia a las principales tareas que se llevan a cabo en este flujo de trabajo, así como la descripción de los artefactos que se generan.

Se explicará cómo se distribuye físicamente el hardware utilizado y cómo se llevaron a cabo las diferentes pruebas de sistema realizadas al software.

3.2. Generalidades del sistema

El sistema Suria Vision está diseñado siguiendo una Arquitectura de Pizarra, en su variante de tablero de control. Esta arquitectura se ingresa en la familia de los Estilos Centrados en Datos, la cual enfatiza la integrabilidad de los datos. En esta arquitectura hay dos componentes principales: una estructura de datos que representa el estado actual y una colección de componentes independientes que operan sobre él[31].

La Arquitectura de Pizarra desacopla el sistema en componentes denominados agentes autónomos, en éste caso los módulos del sistema de video vigilancia, los cuales son independientes en la realización atómica de su funcionalidad. Éste depende de una entrada de información externa, que es provista por otros módulos, y a su vez, producen un resultado que puede ser entrada de otros. Cada módulo se rige por interfaces estándares que permiten cambios en el ambiente externo sin que éste sufra cambios en su funcionamiento interno. Todo el funcionamiento está coordinado por un elemento central, denominado Repositorio Activo, el cual entrega y recibe información de los módulos y coordina su funcionamiento. En el caso del sistema de video vigilancia Suria Vision, el Gestor cumple con la función de Repositorio activo al que se pueden conectar diferentes agentes autónomos. Esta arquitectura brinda una sólida estructura, que es flexible al cambio, por lo cual es una respuesta adecuada a las necesidades del sistema.

Aunque el sistema mantiene una Arquitectura de Pizarra a manera global, en cada uno de los módulos se usa el estilo Modelo-Vista-Controlador (MVC) y el de 3 Capas. El primero afronta una de las consecuencias de los cambios frecuentes en los soportes de hardware y de añadir nuevas

funcionalidades, que es la de constantes modificaciones en la interfaz de usuario. Los módulos, como el Grabador se separan en 3 componentes fundamentales, el Modelo, la Vista y el Controlador. El Modelo es inmutable a los cambios en la interfaz de usuario, se encarga de toda la gestión interna del negocio, la Vista se encarga de presentar los datos y de interactuar con el usuario, mientras que el controlador maneja todos los cambios de estado de la Vista. Mediante el uso de este patrón el Grabador es capaz de utilizar diferentes interfaces de usuario manteniendo la misma lógica de negocio. El segundo distribuye todo el diseño del módulo en 3 capas bien definidas, cada una de ellas agrupando clases con tareas muy relacionadas en pos de una meta general. En el caso del Sistema se usa para delimitar las capas de Presentación, Negocio y Acceso a Datos.

El Grabador está compuesto por varias partes que colaboran entre sí, algunas de las cuales constituyen agentes autónomos dentro del sistema Suria Vision. Esto se puede observar en la siguiente imagen que muestra al módulo Grabador integrado en el sistema Suria Vision.

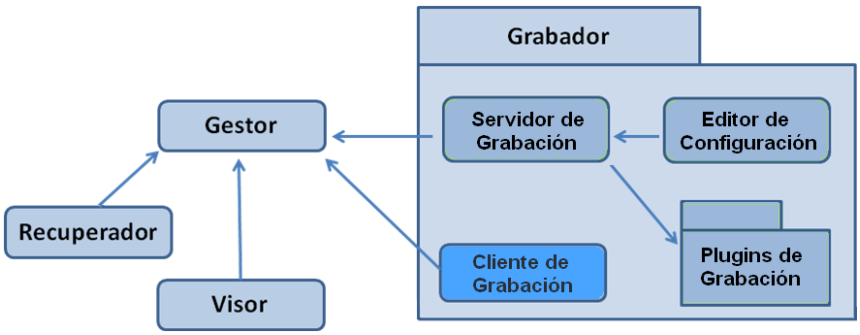


Fig. 9 Módulo Grabador como un conjunto de aplicaciones

3.3. Patrones de diseño

Un patrón es un modelo que se puede seguir para realizar algo. Éste se plantea además, como una solución a un problema en un contexto, codifica conocimiento específico acumulado por la experiencia en un dominio determinado.

Los Patrones de Diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software. Un patrón de diseño es una solución a un problema de diseño, además, estos pretenden proporcionar catálogos de elementos reusables en el diseño de sistemas software, formalizar un vocabulario común entre diseñadores y estandarizar el modo en que se realiza el diseño[32].

Con la selección de los patrones de diseño, el sistema tienden a ser más fácil de entender, mantener y ampliar, y existen más oportunidades para reutilizar componentes en futuras aplicaciones.

3.3.1. Patrones GRASP

GRASP, Acrónimo de General Responsability Assignment Software Patterns (Patrones de Software para la Asignación General de Responsabilidad). Estos patrones describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades[33].

3.3.1.1. Patrón Experto

El propósito del patrón *Experto* es asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad.

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. De este modo se obtendrá un diseño con mayor cohesión y así la información se mantiene encapsulada (disminución del acoplamiento)[34].

Dicho patrón es evidenciado en la definición de las clases de acuerdo a las funcionalidades que deben realizar a partir de la información que manejan. Específicamente: la clase SIS_RecorderClientMaster, será la responsable de realizar todo el negocio del cliente de grabación manteniendo la interfaz actualizada. Sobre este mismo principio se realiza el diseño de las restantes funcionalidades.

3.3.1.2. Patrón Creador

El patrón *Creador* guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos El propósito fundamental de este patrón es encontrar un creador que debe conectar con el objeto producido en cualquier evento. Al escoger este patrón, se da soporte al bajo acoplamiento.

El patrón creador se percibe en las clases ISIS_Tabo_Cameras_ServerModel y ISIS_StorageServerModel ya que son las que poseen los datos necesarios para crear las estructuras dentro del negocio.

3.3.1.3. Patrón Bajo acoplamiento

El patrón Bajo acoplamiento asigna una responsabilidad para mantener bajo acoplamiento. El acoplamiento es una medida de fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre ellas. El propósito de esta patrón es tener las clases lo menos ligadas entre sí que se pueda. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases.

Este patrón se evidencia con la definición de interfaces e implementaciones, como puede ser las interfaces ISIS_ ISIS_Tabo_Cameras_ServerModel e ISIS_StorageServerModel y su implementación, que permiten que la clase SIS_RecorderClientMaster se relacionen únicamente con ellas para realizar sus operaciones, reduciendo el impacto de cambios posteriores en el negocio del sistema.

3.3.1.4. Patrón Alta cohesión

El patrón *Alta cohesión* plantea asignar una responsabilidad de modo que la cohesión siga siendo alta. La cohesión es una medida de la fuerza con la que se relacionan las clases y el grado de focalización de las responsabilidades de un elemento. Cada elemento del diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable.

Una clase con baja cohesión hace muchas cosas no afines o realiza un trabajo excesivo. Estas a menudo representan un alto grado de abstracción o han asumido responsabilidades que deberían haber delegado a otros objetos.

Este patrón fue utilizado en el diseño de la aplicación de manera general; donde se agruparon las clases, según la premisa de que cada clase debe implementar las operaciones que estén sobre la misma área funcional.

3.3.1.5. Patrón Controlador

El patrón C*ontrolador* es un patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado.

Este patrón sugiere que la lógica de negocios debe estar separada de la capa de presentación, esto para aumentar la reutilización de código y a la vez tener un mayor control. Se recomienda dividir los eventos del sistema en el mayor número de controladores para poder aumentar la cohesión y disminuir el acoplamiento.

La clase controladora SIS_RecorderClientMaster, constituye un ejemplo de la aplicación de este patrón, la misma realiza todo el negocio del cliente de grabación manteniendo la interfaz actualizada.

3.3.2. Patrones GOF

Los patrones Gang-of-Four (“pandilla de los cuatro”) o comúnmente llamados Patrones GOF, descritos en el libro Design Patterns (Gama 1995) definen un catálogo con 23 patrones básicos. Según el libro GOF existen 3 tipos de patrones:

- *De Creación*: abstraen el proceso de creación de instancias.
- *Estructurales*: se ocupan de cómo clases y objetos son utilizados para componer estructuras de mayor tamaño.
- *De Comportamiento*: atañen a los algoritmos y a la asignación de responsabilidades entre objetos.

3.3.2.1. Patrón Singleton

El patrón de diseño *Singleton* (instancia única) como patrón de creación, está diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto.

Su intención consiste en garantizar que una clase sólo tenga una instancia y proporcionar un punto de acceso global a ella. Este patrón se implementa creando en una clase un método que crea una instancia del objeto sólo si todavía no existe alguna. Para asegurar que la clase no puede ser instanciada nuevamente se regula el alcance del constructor (con atributos como protegido o privado).

Este patrón ha sido escogido para ponerlo en uso en la conexión con las clases del gestor, así como en la configuración del cliente de grabación, logrando tener una instancia global de estas

3.4. Modelo de Diseño

El objetivo fundamental de este flujo de trabajo es de traducir los requisitos funcionales a una descripción de cómo quedará implementado el sistema. Debe ser lo suficiente robusto para no permitir ambigüedades. El objetivo de este flujo de trabajo es producir un modelo lógico del sistema a implementar. Algunos de los propósitos del diseño son[33]:

- Adquirir una compresión en profundidad de los aspectos relacionados con los requisitos no funcionales, restricciones impuestas por el lenguaje de programación, el Sistema Operativo donde se va a ejecutar, el tipo de interfaz, entre otras.
- Punto de partida para la implementación capturando requisitos de las clases del análisis.
- Ser capaz de visualizar y razonar acerca del diseño usando una notación común.

3.4.1. Diagrama de clases del Diseño

Los diagramas o modelos de clases del diseño sirven para la toma de decisiones, en el diseño y la implementación. Los modelos brindan una vista a menor escala, de la solución final y resulta mucho más fácil efectuar cambios a los mismos, una vez detectados los problemas, que tener que efectuar cambios a la solución ya implementada.

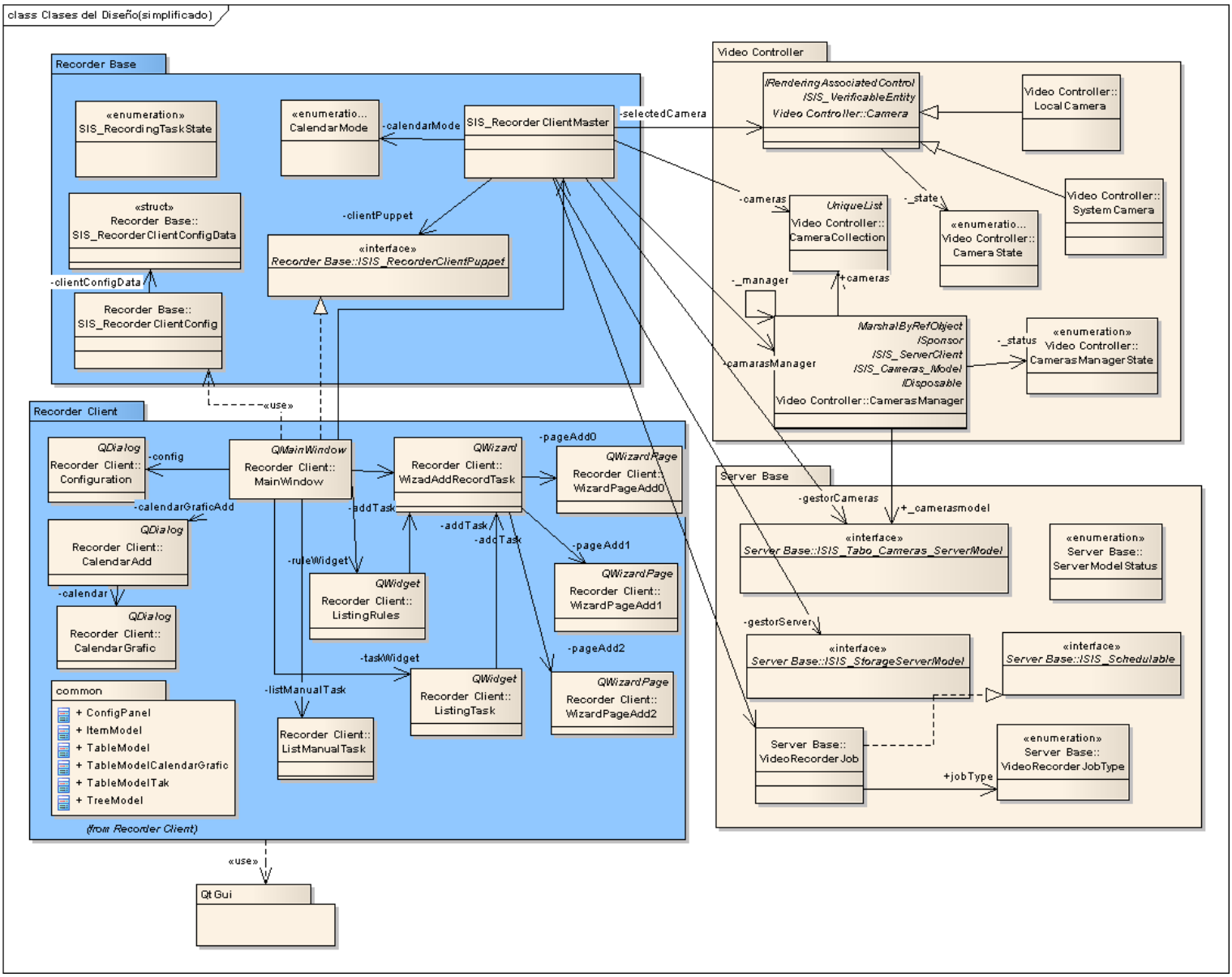


Fig. 10 Clases del diseño del Cliente de Grabación, Simplificado (Detallado Véase Anexo #1)

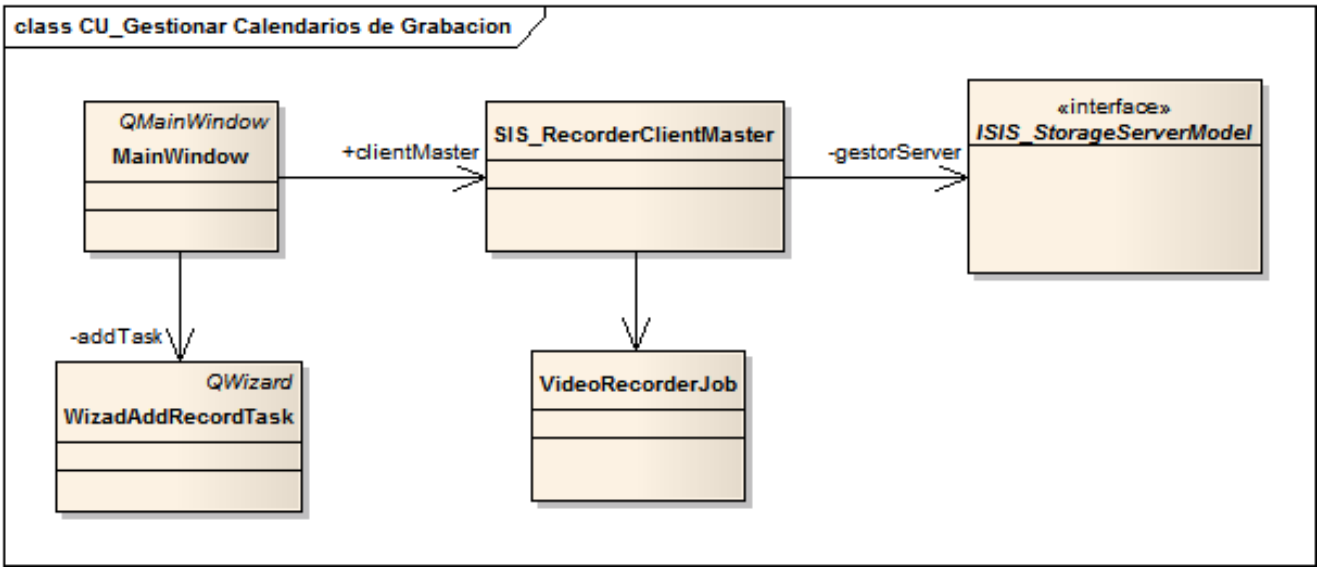


Fig. 11 CU_Gestionar Calendarios de Grabación

El caso de uso se inicia cuando un usuario con privilegios selecciona la opción de adicionar o editar una regla de grabación. El sistema muestra un wizard ***WizadAddRecordTask*** que contiene los parámetros de una regla de grabación, luego el usuario introduce los datos. La clase interfaz ***MainWindow*** envía un evento a la controladora ***SIS_RecorderClientMaster*** indicando la operación a realizar. Luego, la clase controladora manda una petición en dependencia de la acción desarrollada; crea o actualiza una instancia de la clase ***VideoRecorderJob*** que define una regla o tarea de grabación y solicita al Gestor a través de su interfaz ***ISIS_StorageServerModel*** que adicione, modifique o elimine la regla de grabación en la Base de Datos.

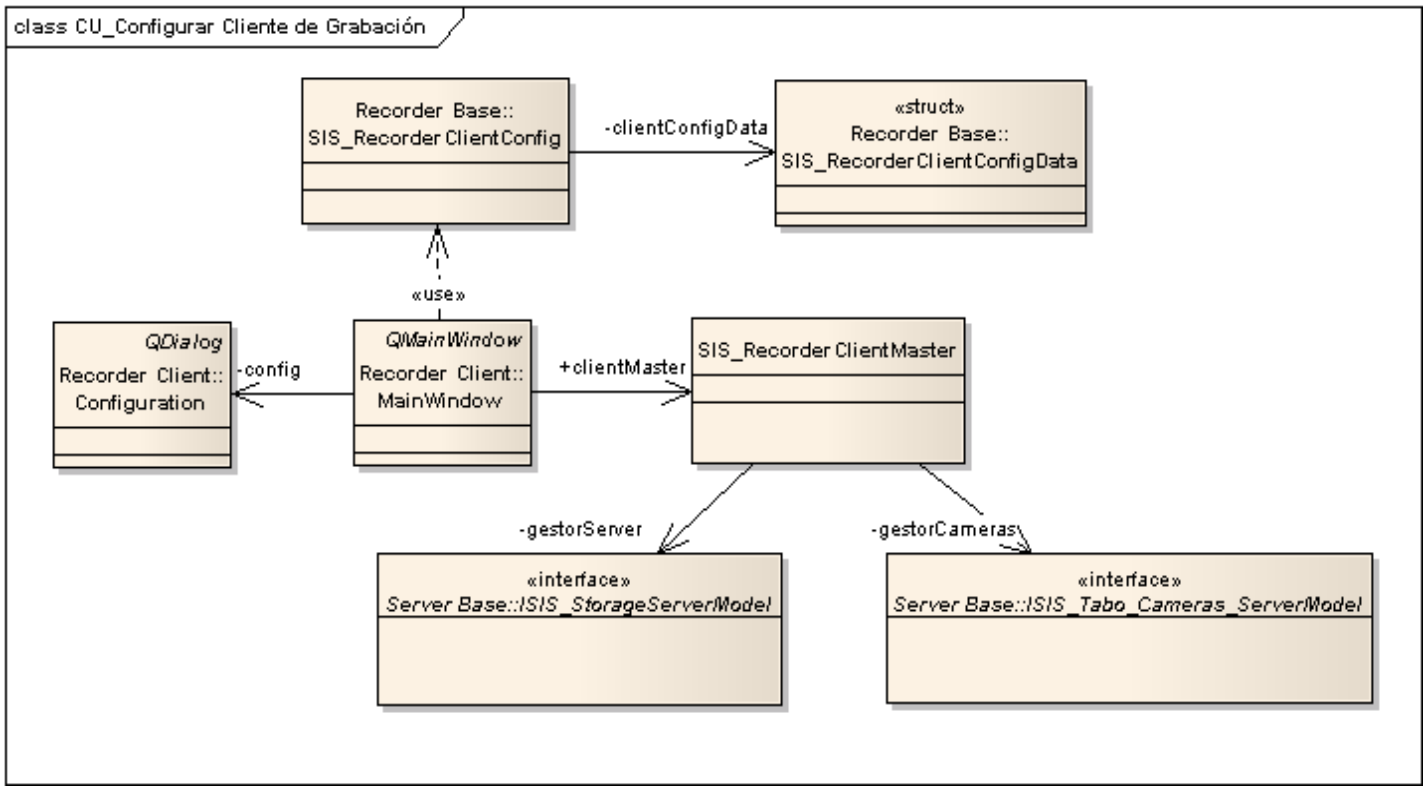


Fig. 12 CU_ Configurar Cliente de Grabación

El caso de uso Configurar Cliente de Grabación se inicia cuando el usuario con privilegios selecciona la pestaña “Configuración” en la interfaz ***MainWindow***. El sistema muestra una ventana con los parámetros de configuración obtenidos de las clases ***SIS_RecorderClientConfig*** y ***SIS_RecorderClientConfigData***. El usuario modifica los parámetros necesarios y pulsa la opción “Aceptar”, luego ***MainWindow*** envía un evento a la controladora ***SIS_RecorderClientMaster*** indicando un cambio en la configuración. La clase controladora ***SIS_RecorderClientMaster*** crea instancias de ***ISIS_Tabo_Cameras_ServerModel*** e ***ISIS_StorageServerModel*** y descarga la información del gestor, actualizando posteriormente la interfaz ***MainWindow***.

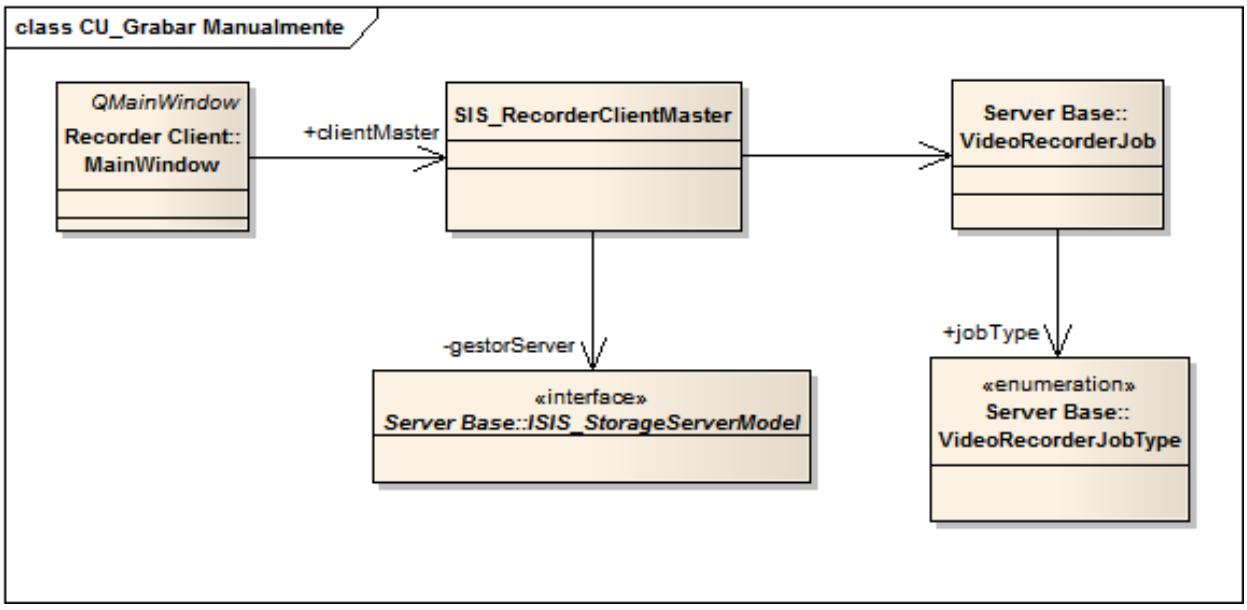


Fig. 13 CU_Grabar Manualmente

El caso de uso Grabar Manualmente se inicia cuando un usuario con privilegios selecciona una cámara existente y elige la opción “Comenzar a grabar manualmente” o “Detener la grabación”. Luego la clase interfaz ***MainWindow*** envía un evento a la controladora ***SIS_RecorderClientMaster*** indicando que el usuario solicita iniciar o detener una grabación. La clase controladora hace una petición de inicio o fin de grabación el gestor a través de su interfaz ***ISIS_StorageServerModel***; en el caso de inicio de grabación, se crea una instancia de la clase ***VideoRecorderJob*** y se define que esta es manual a través de la clase ***VideoRecorderJobType.***

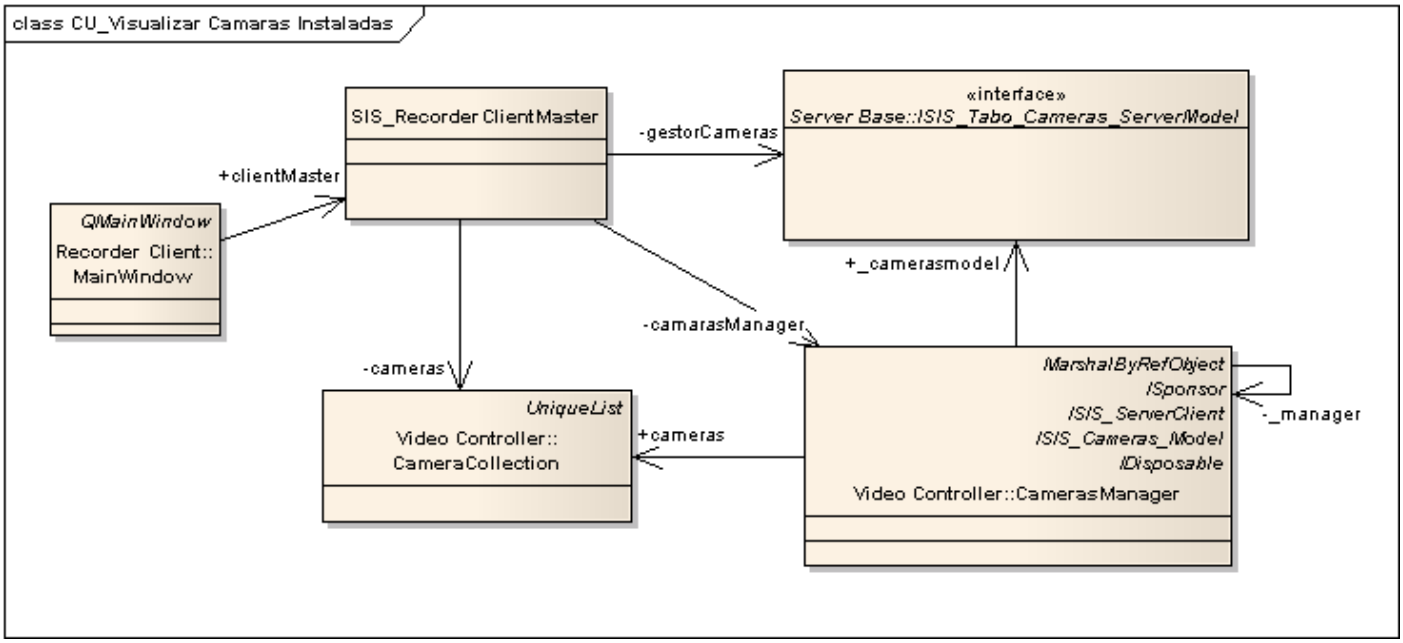


Fig. 14 CU_Visualizar Cámaras Instaladas

Una vez que el usuario se ha logueado correctamente, el caso de uso Visualizar Cámaras Instaladas se inicia cuando la clase controladora ***SIS_RecorderClientMaster*** manda una petición a la clase ***CamerasManager***, obteniendo acceso a la información relacionada con las cámaras en el sistema, como el listado de cámaras existentes ***CameraCollection,*** actualiza la clase interfaz del gestor ***ISIS_Tabo_Cameras_ServerModel*** permitiendo acceder a sus funciones como servidor de la información relacionada con las cámaras y luego solicita a la clase ***MainWindow*** que muestre la lista de cámaras.

3.4.1.1. Descripción de las clases del diseño y las relaciones

Paquete Recorder Base: paquete que contiene las clases bases del módulo Grabador, es decir las clases comunes para todas las aplicaciones del grabador.

- **SIS_RecordingTaskState**: esta define el estado en que puede encontrarse una tarea de grabación.
- **SIS_RecorderClientConfigData**: clase que contiene los datos de configuración del Cliente de grabación.
- **SIS_RecorderClientConfig**: clase de configuración del cliente de grabación.

- **SIS_RecorderClientMaster:** clase controladora que realiza todo el negocio del cliente de grabación manteniendo la interfaz actualizada.
- **ISIS_RecorderClientPuppet**: esta clase define las funcionalidades que debe tener la interfaz gráfica del cliente de grabación.
- **CalendarMode**: clase que indica si MainWindow se ejecuta como la ventana principal de una aplicación o es instanciada por otra aplicación.

Paquete Video Controller: éste contiene las clases de control de las fuentes de video, es decir las cámaras.

- **Camera:** clase que representa una cámara en el sistema.
- **LocalCamera:** una cámara local, conectada directamente a la PC o integrada en la PC (Webcam).
- **CameraState:** define los estados en que se puede encontrar una cámara.
- **CamerasManagerState:** estados de CameraManager.
- **CamerasManager:** clase que permite el acceso a la información relacionada con las cámaras en el sistema.
- **CameraCollection:** es la lista de cámaras.

Paquete Server Base: paquete que contiene las clases bases y las interfaces comunes para todo el sistema de video vigilancia.

- **ISIS_Tabo_Cameras_ServerModel:** interfaz del Gestor que permite acceder a sus funciones como servidor de la información relacionada con las cámaras.
- **ISIS_StorageServerModel:** interfaz del Gestor que permite acceder a sus funciones como servidor de la información relacionada con la grabación.
- **VideoRecorderJob:** define una regla o tarea de grabación.
- **ServerModelStatus:** esta clase indica el estado en que puede encontrarse el gestor.
- **ISIS_Schedulable:** clase base para las operaciones programables por calendario.
- **VideoRecorderJobType:** define el tipo de regla de grabación: si se realiza de forma manual, si es planificada por calendario, si es por detección de movimiento o si es una regla.

Paquete Recorder Client: contiene las clases de la capa de presentación del cliente de grabación.

- **MainWindow:** ventana principal de la aplicación.

- **Configuration:** ventana que permite editar los parámetros de configuración del cliente de grabación.
- **WizadAddRecordTask:** ventana asistente que permite crear o editar una regla de grabación.
- **WizardPageAdd0, WizardPageAdd1 y WizardPageAdd2:** ventanas que contiene los parámetros necesarios para crear o editar una regla de grabación.
- **CalendarAdd:** pequeña ventana que permite crear, editar y eliminar los calendarios gráficos, mostrando además los calendarios gráficos disponibles en la aplicación.
- **CalendarGrafic:** interfaz que permite crear reglas de grabación de manera visual, mediante el sombreado de intervalos de tiempos.
- **ListingRules:** representa el listado de reglas de grabación de una cámara seleccionada.
- **ListingTasks:** representa el listado de tareas de grabación de una cámara seleccionada.

Paquete Common: contiene las clases y componentes visuales comunes para el cliente de grabación.

- **ConfigPanel:** representa la paleta de herramientas que contiene todas las funcionalidades de la aplicación.
- **TreeModel:** representa el árbol de cámaras existentes en el sistema.
- **TableModel, TableModelTask y TableModelCalendarGrafic:** representan los modelos para llenar las tablas para mostrar la información de las reglas, las tareas y los calendarios gráficos. **ItemModel:** contiene todos los elementos a mostrar en TableModel, TableModelTask y TableModelCalendarGrafic.

Paquete QtQui: módulo extendido de QtCore con la funcionalidad de interfaz gráfica de usuario.

Relaciones de asociación: permite asociar objetos de las clases asociadas que colaboran entre si. Cabe destacar que no es una relación fuerte, es decir, el tiempo de vida de un objeto no depende del otro.

3.4.2. Modelo de Datos

El modelo de datos es el modelo utilizado para el diseño de la base de datos. Por lo general, un modelo de datos permite describir las estructuras de datos de la base (el tipo de los datos que incluye la base y la forma en que se relacionan), las restricciones de integridad (las condiciones que los datos deben cumplir para reflejar correctamente la realidad deseada) y las operaciones de manipulación de los datos (agregado, borrado, modificación y recuperación de los datos de la base)[35].

3.4.2.1. Descripción de las tablas

Tabla 1 Descripción de la tabla tbl_camara_to_record

Nombre: tbl_camara_to_record		
Representa una regla o planificación de grabación definida para una cámara del sistema.		
Atributo	**Tipo**	**Descripción**
cameraid	integer	Identificador de la cámara.
rstart	timestamp(0)	Hora de inicio de la grabación.
rend	timestamp(0)	Hora de fin de la grabación
Id_schedule	integer	Identificador de la regla de grabación.
formato	varchar(20)	Formato de video en que se grabará.
kind	varchar(20)	Tipo de regla de grabación (Exacta /Diaria /Mensual/Semanal).
days	varchar(60)	Días en que se graba (de 1-7 si es Semanal, de 1-31 si es Mensual, si es Exacta o Diaria queda vacío).
descripcion	varchar(200)	Descripción de la regla.
nombre	varchar(60)	Nombre de la regla.

3.5. Conclusiones parciales

El uso de patrones de diseño permitió elaborar diagramas de clases de diseño fiables para la implementación de los diferentes teniendo en cuenta la arquitectura definida por el proyecto. La elaboración los diagramas de clases del diseño, ha formado la base para el inicio del desarrollado de las funcionalidades de la aplicación.

CAPÍTULO 4. Implementación y pruebas de software a la solución propuesta

4.1. Introducción

En el presente capítulo se describen los principales artefactos desarrollados durante la fase de elaboración. Durante esta fase los flujos de trabajo de mayor importancia son los flujos de trabajo de Implementación y Pruebas por lo que se detallan los diagramas de componentes y diagramas de despliegue, herramientas fundamentales elaboradas durante el flujo de implementación, y se describen algunas de las pruebas realizadas a la aplicación así como los resultados.

4.2. Estándares de Codificación

Los estándares de codificación se utilizan en los flujos de implementación, mantenimiento y despliegue. Las convenciones o estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código, brindan legibilidad y claridad[36]. En el desarrollo de la aplicación se hizo uso del estándar de codificación definido en el proyecto Video Vigilancia para el lenguaje C++. A continuación se presentan algunas de estas convenciones.

Comentarios

Cada programa deberá comenzar con un comentario que incluya:

- Autor.
- Fecha.
- Objetivo, o problema que resuelve el programa.
- Fecha de creación y bitácora de versiones con las dos últimas fechas de modificación.
- Algoritmo.

Cada función debe tener un encabezado que contenga:

- Objetivo de la función y no descripción del procedimiento.
- Comentarios de apoyo a variables, llamadas a función o inclusión de archivos que no sean obvios al proceso.

- Explicación de uso de argumentos (parámetros) no obvios.
- Explicación de uso de valores devueltos (de retorno).

Nombres de identificadores.

Se considera como identificador a los nombres de variables (arreglos, matrices, apuntadores), funciones, así como cualquier tipo de dato definido por el usuario (estructura, clase). Dichos identificadores deberán seguir las siguientes normas, además de las definidas por el propio lenguaje.

- Deberán tener un nombre significativo para que por su simple lectura, pueda conocerse su función, sin tener que consultar manuales o hacer demasiados comentarios.
- Para nombres que se usen con frecuencia o para términos largos, se recomienda usar abreviaturas estándar para que éstos tengan una longitud razonable. Si usa abreviaturas deben manejar la misma lógica en todo el programa.
- Evitar identificadores que comiencen con uno o dos caracteres de subrayado para evitar que se confundan con los que el compilador selecciona.
- Cada identificador de función, variable o procedimiento deberá ser precedido por la abreviación del tipo de dato de que es la variable, o si se trata de una función o procedimiento del tipo de dato que regresa.

Tabla 2 Abreviación de los identificadores según el tipo de dato

Tipo de dato	Abreviatura
integer	I
float	f
double	d
Arreglo	ar
char	c
Enumeración	e
Estructura	st
constantes	TODAS LAS LETRAS DEL IDENTIFICADOR CON MAYÚSCULA.
punteros	p

Identificadores de variables

Comenzarán siempre con la primera letra minúscula correspondiente a su tipo de dato. Para distinguir palabras dentro del nombre deberá emplearse una letra mayúscula o un guión bajo (_), sin mezclar ambas formas en un mismo programa.

Ejemplos:

- temperaturaDeVapor
- temperatura_de_vapor

Identificadores de punteros (apuntadores)

Su nombre deberá comenzar con la letra p.
Ejemplo: pAlumno
Dónde pAlumno es un puntero que podrá tener la dirección del lugar donde se almacena información de un alumno.

Identificadores de variables dimensionadas (arreglos, matrices)

Su nombre deberá comenzar con las letras ar.
Ejemplo: arAlumnos
Donde arAlumnos es un arreglo de datos para guardar información de alumnos.

Identificadores de datos constantes

Estas serán declaradas en letras mayúsculas.
Ejemplo: const IVA = 0.15;

Identificadores de funciones

La primera letra deberá ser minúscula.
Ejemplo: void vFuncion( );

Identificadores de tipos definidos por el usuario

La primera letra será mayúscula, las interfaces deben de comenzar con I
Ejemplo: class Clase o struct Estructura o class IEnumerable

En el caso de los identificadores pertenecientes a un componente determinado el identificador debe comenzar con el nombre del componente o una palabra sugerente a éste, un _ y se continúa con la regla anterior.

Ejemplo:

```
class Planificador_Tarea
class Reproductor_IFuente
```

Organización Visual del Programa

Generales

- No manejar en los programas más de una instrucción por línea.
- Declarar las variables en líneas separadas
- Añadir comentarios descriptivos junto a cada declaración de variables, si es necesario.

Sangrías

- Las sangrías tendrán una longitud de tres espacios.
- Para las llaves que definen el cuerpo de una función, sangría un nivel.

Ejemplo:

```
void Funcion ( )
{
        //Instrucciones de la función
}
```

- Adicionar sangría a las instrucciones del cuerpo de cada estructura de control.

Ejemplo:

```
for (int x = 0; x < 5; x++)
{
        //Instrucciones a ejecutar
}
```

- Tratar de evitar codificar más de tres niveles de sangrado.

Líneas y espacios en blanco

- Insertar una línea en blanco antes y después de una declaración de datos que aparezca entre instrucciones ejecutables.

```
a = b + c;
// línea en blanco
int f; //declaración entre instrucciones
// línea en blanco
f = a;
```

- Las declaraciones de datos dentro de una función, deberán ir al inicio y separadas de las instrucciones ejecutables de la función por medio de una línea en blanco.
- Deben incluirse espacios en ambos lados de los operadores binarios.

Ejemplo:

```
y = 50 + 15 - x;
```

- Es posible distribuir una instrucción grande sobre varias líneas. En caso de realizarse, seleccionar puntos de ruptura que tengan sentido, como después de una coma en el caso de una lista, o después de un operador en el caso de una expresión larga. Adicionar sangría a todas las líneas subsecuentes.

Ejemplo:

```
cout << “Ejemplo de ruptura de una instrucción en más de una”
<<” línea de comandos”;
```

- Los operadores unarios (++, --, etc.) deben ponerse junto a sus operandos, sin espacios intermedios.
- Antes y después de cada estructura de control deberá poner una línea en blanco.

Paréntesis

- Para hacer más clara una expresión, es aceptable agregarle paréntesis innecesarios. Dichos paréntesis se llaman paréntesis redundantes.

4.3. Modelo de Implementación

El flujo de trabajo de implementación describe cómo los elementos del modelo del diseño se implementan en términos de componentes y cómo estos se organizan de acuerdo a los nodos específicos en el modelo de despliegue. El Modelo de implementación representa la composición física

de la implementación en términos de subsistemas de implementación y elementos (carpetas y ficheros, incluyendo código fuente, datos y ejecutables).

4.3.1. Diagrama de Despliegue

En el diagrama de despliegue se indica la situación física de los componentes lógicos desarrollados. Es decir se sitúa el software en el hardware que lo contiene.

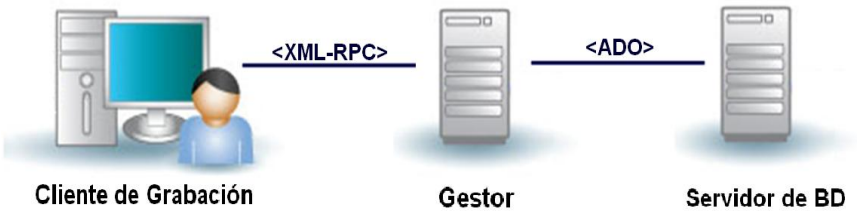


Fig. 15 Diagrama de despliegue del Cliente de Grabación

4.3.2. Diagrama de Componentes

Un componente es la parte modular de un sistema, desplegable y reemplazable que encapsula implementación, un conjunto de interfaces que proporciona la realización de los mismos. Típicamente contiene clases y puede ser implementado por uno o más artefactos (ficheros ejecutables, binarios, etc.). Son las piezas reutilizables de alto nivel a partir de las cuales se pueden construir los sistemas.

Algunos estereotipos estándar de componentes son:

- *Ejecutable*: Es un programa que se puede ejecutar en un nodo.
- *Biblioteca*: Es una biblioteca de objetos estática o dinámica.
- *Tabla*: Es una tabla de una Base de Datos.
- *Archivo*: Es un fichero que contiene código fuente o datos.
- *Documento*: Es un documento.
- *Página Web*: Es una página que se obtiene de la ejecución del sistema.

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Estos muestran las opciones de realización incluyendo código fuente, binario y ejecutable. Los componentes representan todos los tipos de elementos software que entran en la fabricación de aplicaciones informáticas. Los diagramas de componentes son usados para estructurar el modelo de implementación en términos de subsistemas de implementación y mostrar las relaciones entre sus

elementos. Los componentes representados pueden ser datos, archivos, ejecutables, código fuente y directorios[37]. A continuación se muestra el correspondiente al sistema desarrollado.

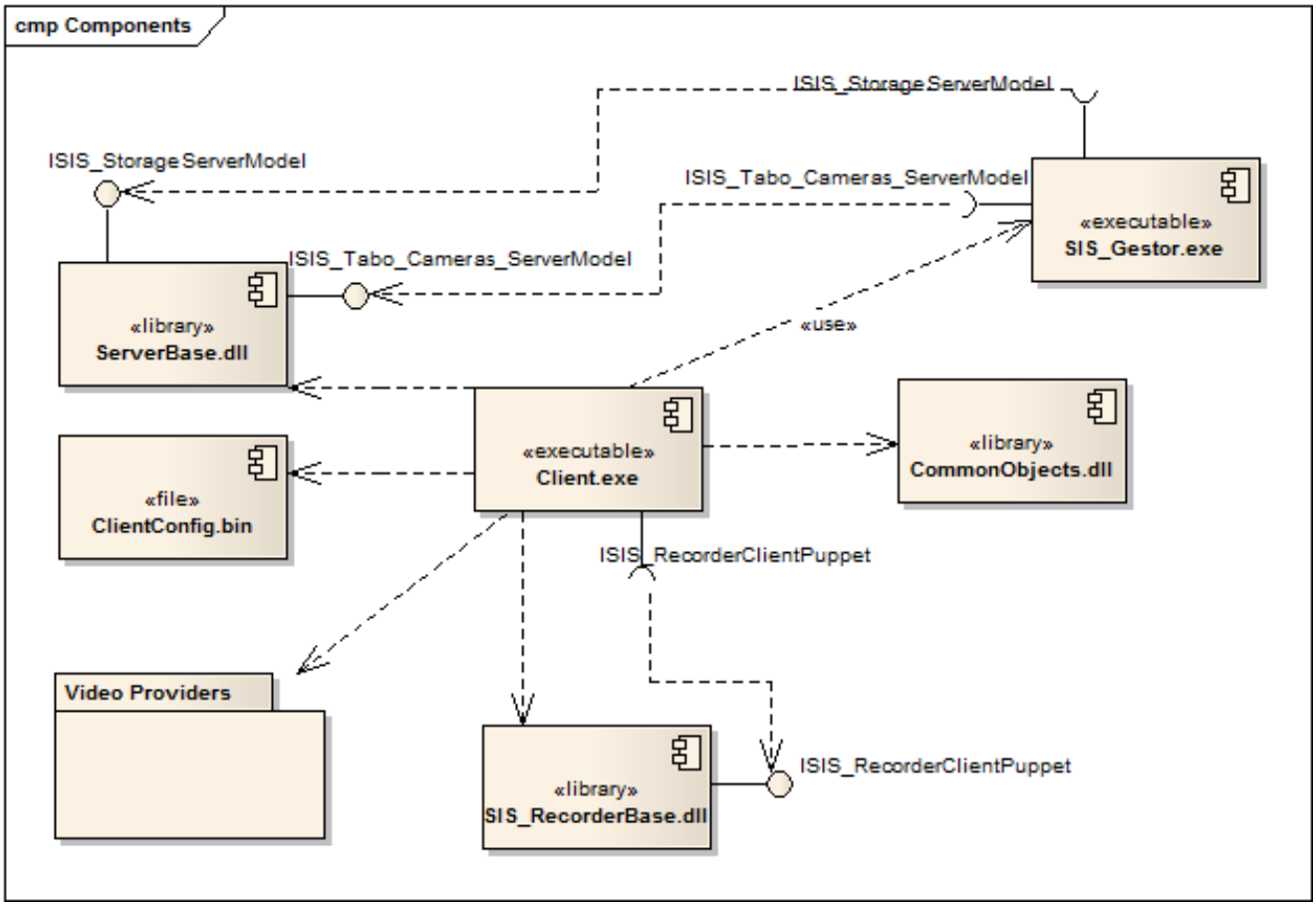


Fig. 16 Diagrama de componentes del Cliente de Grabación

4.4. Pruebas de software a la solución propuesta

Las pruebas es una actividad en la cual un sistema o componente es ejecutado bajo unas condiciones o requerimientos especificados, los resultados son observados y registrados, y una evaluación es hecha de algún aspecto del sistema o componente[38]. Las pruebas son técnicas de validación predominantes o procesos de ejecución de un programa con la intención de descubrir errores de una aplicación que antes no se habían encontrado.

4.4.1. Pruebas de Caja Negra

Las pruebas son una actividad en la cual un sistema o componente es ejecutado bajo unas condiciones o requerimientos especificados, los resultados son observados y registrados, y una

evaluación es hecha de algún aspecto del sistema o componente[39]. La prueba de software es un elemento crítico para la garantía de la calidad del software y representa una revisión final de las especificaciones del diseño y de la codificación.

La prueba de caja negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del software. A este tipo de pruebas se les además, pruebas de comportamiento, ya que los probadores o analistas de pruebas no enfocan su atención a como se generan las respuestas del sistema, básicamente el enfoque de este tipo de prueba se basa en el análisis de los datos de entrada y en los de salida, los que generalmente se define en los casos de prueba preparados antes del inicio de las pruebas

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 se mantiene[40].

4.4.1.1. Casos de Pruebas

El propósito de los casos de pruebas es especificar una forma de probar el sistema, incluyendo las entradas con las que se prueba, los resultados esperados y las condiciones bajo las que ha de probarse. Los casos de prueba ayudan a validar y verificar:

- Si el producto satisface los requerimientos del usuario, tal y como se describe en las especificación de los requerimientos.
- Si el producto se comporta como se desea, tal y como se describe en las especificaciones funcionales del diseño.

Normalmente, un Caso de Prueba se deriva de un caso de uso en el modelo de casos de uso. Con estos Casos de Prueba se validan los requerimientos funcionales del sistema.

Caso de prueba para el caso de uso del sistema Configurar Cliente de Grabación

Descripción General: Se inicia cuando el Usuario con privilegios ejecuta el Editor de Configuración o selecciona la pestaña “Configuración” en el Cliente. Modifica algunos parámetros de configuración y pulsa “Guardar Cambios”.

Condiciones de Ejecución: El usuario tiene que tener permisos para realizar esta acción.

Secciones a probar en el Caso de Uso.

Tabla 3 Caso de Prueba del caso de uso Configurar Cliente De Grabación

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo Central

SC 1 Configurar cliente	EC 1.1: Configurar cliente con éxito.	El Usuario con privilegios configura el software cliente. Para ello selecciona la pestaña Configuración, se selecciona la opción "Editar Configuración" y pulsa la opción "Aceptar". El sistema valida los datos, actualizar el fichero de configuración, cierra la ventana y queda actualizada la configuración del Cliente.	Cliente de Grabación - Suria Vision Recorder/ Ficha Configuración /En "Archivo de configuración" / botón "Editar Configuración"/ botón Aceptar
	EC 1.2: Cancelación de Configuración del Cliente.	El Usuario con privilegios cancela la operación de configuración. Pulsa el botón "Cancelar", luego el Sistema restituye la configuración original y el Sistema cierra la ventana.	Cliente de Grabación - Suria Vision Recorder / Ficha Configuración /En Archivos de configuración / botón Editar Configuración/ botón Cancelar
	EC 1.3 Modificación de algunos de los parámetros de configuración.	El Usuario con privilegios modifica algunos parámetros de configuración. El Sistema valida los datos, guarda una copia temporal y actualiza el fichero de configuración con la nueva configuración.	Cliente de Grabación - Suria Vision Recorder / Ficha Configuración /En Archivos de configuración / botón Editar Configuración/ botón Guardar Cambios/botón aceptar.
	EC 1.4: Configurar cliente con falla.	En caso que se haya introducido algún dato mal el Sistema muestra un mensaje indicando que los datos introducidos no son válidos.	Cliente de Grabación - Suria Vision Recorder / Configuración/Editar Configuración/ botón Aceptar.

Caso de prueba para el caso de uso del sistema Gestionar Calendario de Grabación

Descripción General: El caso de uso se inicia cuando el Usuario con Privilegios selecciona la opción de adicionar, eliminar o editar una regla de grabación. O cuando modifica de forma manual la representación gráfica del Calendario de Grabación, de forma que automáticamente se modifican, agregan o eliminan reglas de grabación para ajustarse a la representación gráfica realizada. Para la realización de este caso de uso es necesario registrar los log (ver caso de uso "Registrar Log") y además se puede obtener información (ver caso de uso Obtener Información).

Condiciones de Ejecución: Debe haber conexión con el Gestor. El usuario debe tener los permisos para llevar a cabo esta acción.

Secciones a probar en el Caso de Uso.

Tabla 4 Caso de Prueba del caso de uso Gestionar Calendario de Grabación

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo Central

SC 1 Adicionar Regla mediante Wizard	EC 1.1: Adicionar Regla mediante Wizard correctamente.	El usuario con privilegios adiciona reglas mediante Wizard para ello el sistema muestra una interfaz para introducir los datos generales de la regla de grabación, el usuario introduce los datos y si estos están correctamente y la regla de grabación definida no se solape con una de las ya existentes, el sistema solicita al Gestor que agregue la nueva regla de grabación.	1- Cliente de Grabación - Suria Vision Recorder / Pestaña Calendario/Seleccionar una cámara/en Reglas de Grabación/ botón Adicionar Regla de grabación. 2- Cliente de Grabación - Suria Vision Recorder/clic derecho sobre una cámara/ Adicionar regla.
	EC 1.2: Cancelar Adicionar Regla mediante Wizard.	El usuario selecciona la opción “Cancelar”. El sistema cierra el Wizard y termina el Caso de Uso.	1- Cliente de Grabación - Suria Vision Recorder / Pestaña Calendario/Seleccionar una cámara/Gestionar Calendarios de Grabaciones/Opción Adicionar Regla de grabación/ botón “Cancelar”. 2- Cliente de Grabación - Suria Vision Recorder/clic derecho sobre una cámara/ Adicionar regla/botón Cancelar.
	EC: 1.3: Fallas en Adicionar Regla mediante Wizard.	El usuario con privilegios adiciona reglas mediante Wizard, dejando algún campo vacío, el sistema muestra un mensaje indicándole al usuario que llene todos los campos. Mensaje: “Llene los datos requeridos”.	Cliente de Grabación - Suria Vision Recorder/Pestaña Calendario/Seleccionar una cámara/Gestionar Calendarios de Grabaciones/Opción Adicionar Regla de grabación/botón “Siguiente”. Cliente de Grabación - Suria Vision Recorder / Pestaña Calendario/Seleccionar una cámara/Gestionar Calendarios de Grabaciones/Opción Adicionar Regla de grabación/botón “Siguiente”/ botón “Siguiente” Cliente de Grabación - Suria Vision Recorder / Pestaña Calendario/Seleccionar una cámara/Gestionar Calendarios de Grabaciones/Opción Adicionar Regla de grabación/botón “Siguiente”/ botón

	EC 1.4: Adicionar Regla mediante Wizard cuando la regla adicionada se solapa.	Si la regla adicionada se solapa con alguna existente, el sistema muestra un mensaje indicando que no puede ser creada, porque se solapa con una regla existente y de esta forma termina el Caso de Uso.	Cliente de Grabación - Suria Vision Recorder / Pestaña Calendario/Seleccionar una cámara/ botón "Adicionar Regla de grabación"/(Se llenan todos los campos) botón "Siguiente" (se repite este paso tres veces)/botón "Aceptar"/ botón "Aceptar". Cliente de Grabación - Suria Vision Recorder / Pestaña Calendario/clic derecho sobre una cámara/ /Opción "Adicionar Regla "/(Se llenan todos los campos) botón "Siguiente" (se repite este paso tres veces)/botón "Aceptar"/ botón "Aceptar".
SC 2 Modificar Regla de Grabación mediante Wizard.	EC 2.1: Modificar Regla de Grabación mediante Wizard correctamente.	El usuario con privilegios selecciona una regla de grabación y pulsa la botón "Editar Regla de Grabación", modifica los datos, el sistema verifica que todos los datos están llenos y que la regla de grabación no se solape con una de las ya existente, el sistema solicita al Gestor que modifique la regla de grabación en la base de datos (si ninguno de los datos es la hora). En caso de haberse modificado la hora sistema solicita al Gestor que cree una nueva regla de grabación con los datos especificados.	1- Cliente de Grabación - Suria Vision Recorder / Pestaña Calendario/Seleccionar una regla en el listado de regla/ Opción Editar Regla de grabación/se modifican todos los campos y pulsa botón "Siguiente" (se repite este paso, dos veces)/ se modifican los campos, clic botón "Aceptar" 2- Cliente de grabación – Suria Vision Recorder/ Seleccionar una regla en el listado de regla /clic derecho sobre la regla/ Opción Editar Regla/ se modifican todos los campos y pulsa botón "Siguiente" (se repite este paso, dos veces)/ se modifican los campos, clic botón "Aceptar".

	EC 2.2: Cancelar Modificar Regla mediante Wizard.	El usuario selecciona la opción "Cancelar". El sistema cierra el Wizard y termina el Caso de Uso.	1- Cliente de Grabación - Suria Vision Recorder / Pestaña Calendario/Seleccionar una regla en el listado de regla/ Seleccionar una regla en el listado de regla/ Opción Editar Regla de grabación/se modifican o no todos los campos y pulsa botón "Cancelar". Cliente de Grabación - Suria Vision Recorder / Pestaña Calendario/Seleccionar una regla en el listado de regla/ Opción Editar Regla de grabación/se modifican todos los campos y pulsa botón "Siguiente"/ se modifican o no todos los campos y pulsa botón "Cancelar". Cliente de Grabación - Suria Vision Recorder / Pestaña Calendario/Seleccionar una regla en el listado de regla/ Opción Editar Regla de grabación/se modifican todos los campos y pulsa botón "Siguiente" (se repite esta acción una vez más)/ se modifican o no todos los campos y pulsa botón "Cancelar". 2- Cliente de Grabación - Suria Vision Recorder / Pestaña Calendario/Seleccionar una regla en el listado de regla /clic derecho sobre la regla/ Opción Editar Regla/ seguir los pasos anteriores.
	EC: 2.3: Modificar Regla mediante Wizard con campos vacíos.	El usuario con privilegios desea modificar alguna regla, dejando algún campo vacío, el sistema muestra un mensaje indicándole al usuario que debe llenar todos los campos. Mensaje: "Llene los datos requeridos"	Cliente de Grabación - Suria Vision Recorder / Pestaña Calendario/Seleccionar una cámara/ botón Editar Regla de grabación/ se modifican dejando campos en blanco/ botón "Siguiente"/ botón Aceptar.

	EC 2.4: Modificar Regla mediante Wizard cuando la regla modificada se solapa.	Si la regla modificada se solapa con alguna existente, el sistema muestra un mensaje indicando que no puede ser modificada, porque se solapa con una regla existente.	Cliente de Grabación - Suria Vision Recorder / Pestaña Calendario/Seleccionar una regla/ botón Editar Regla de grabación´/(Se modifican los campos) botón "Siguiente" (se repite este paso dos veces)/botón "Aceptar"/ botón
SC 3 Eliminar Regla de Grabación	EC 3.1: Eliminar Regla de Grabación correctamente.	El usuario con privilegio selecciona una o varias reglas de grabación, selecciona la opción de eliminar reglas seleccionadas, el sistema pide confirmación al usuario, confirma que desea eliminar, el Gestor elimina las reglas de grabación de la base de	Cliente de Grabación - Suria Vision Recorder / Pestaña Calendario/Seleccionar una o varias reglas/botón Eliminar las reglas seleccionadas/ botón "Sí".
	EC 3.2: Eliminar Regla de Grabación cuando no se confirma.	El actor no confirma que desea eliminar las reglas de grabación, entonces el sistema cierra el diálogo y no se elimina la regla seleccionada.	Cliente de Grabación - Suria Vision Recorder / Pestaña Calendario/Seleccionar una o varias reglas/botón Eliminar las reglas seleccionadas/ botón "No".

Caso de prueba para el caso de uso del sistema Grabar Manualmente

Descripción General: El caso de uso se inicia cuando el Usuario con privilegios selecciona una cámara y selecciona la opción "Comenzar a Grabar", el sistema hace una petición de inicio de grabación al Gestor. El caso de uso termina cuando el Usuario con privilegios selecciona la opción "Terminar Grabación". Además se puede obtener información (ver caso de uso Obtener Información).

Condiciones de Ejecución: El Gestor debe estar "Disponible". El usuario tiene que tener permisos para realizar esta acción.

Secciones a probar en el Caso de Uso.

Tabla 5 Caso de Prueba del caso de uso Grabar Manualmente

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo Central
SC 1: Grabar de forma manual.	EC 1.1: Grabación con éxito	El usuario con privilegios selecciona una cámara y la opción "Comenzar Grabación". Se agrega la nueva tarea a grabar.	Cliente de Grabación - Suria Vision Recorder / Clic derecho sobre una de las cámaras/ Opción "Comenzar Grabación"

	EC 1.2: Grabación falla.	Notifica al Gestor si el proceso de grabación falló y la causa del fallo (las causas de fallo pueden ser por desconexión del servidor de grabación o por desconexión tanto de la red como de la corriente de una cámara determinada.) y se termina el caso de uso.	Cliente de Grabación - Suria Vision Recorder / Clic derecho sobre una de las cámaras/ Opción "Comenzar Grabación" Cambia estado de la tarea de grabación. • No iniciada • Grabando • Error o desconexión (ocurre cuando ahí desconexión de la red, el servidor de grabación o por fallo en la corriente).
SC 2: Terminar grabación manual	EC 2.1: Terminar grabación con éxito	El usuario con privilegios detiene la grabación para ello selecciona una cámara y escoge la opción "Detener Grabación", selecciona la grabación en curso que se desea terminar y confirma la detención de la grabación.	Módulo Visor/Ficha Grabación/Botón Cliente de Grabación - Suria Vision Recorder / Clic derecho sobre una de las cámaras/ Opción "Comenzar Grabación"/ Clic derecho sobre la cámara/ Opción "Detener Grabación" (Menú con grabaciones activas)/ Clic sobre una de ellas/ Clic en el botón Si.
	EC 2.2: Terminar grabación sin éxito.	El usuario no confirma que desea terminar la grabación.	Módulo Visor/Ficha Grabación/Botón Cliente de Grabación - Suria Vision Recorder / Clic sobre una de las cámaras/ Opción "Comenzar a Grabar"/ Clic derecho sobre la cámara/ Opción "Detener Grabación" (Menú con grabaciones activas)/ Clic sobre una de ellas/ Clic en el botón No.

Caso de prueba para el caso de uso del sistema Mostrar Cámaras Instaladas

Descripción General: El caso de uso se inicia cuando el Usuario con privilegios se ha autenticado en el Sistema y termina cuando el Sistema muestra una vista de árbol que incluye todas las cámaras manejadas por el sistema. Para esto se necesita registrar log (ver caso de uso Registrar Log) y además se puede obtener información (ver caso de uso Obtener Información).

Condiciones de Ejecución: El Gestor debe estar "Disponible".

El usuario tiene que tener permisos para realizar esta acción.

Secciones a probar en el Caso de Uso.

Tabla 6 Caso de Prueba del caso de uso Mostrar Cámaras Instaladas

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo Central
SC 1: Mostrar Cámaras Instaladas.	EC 1.1: Mostrar Cámaras Instaladas satisfactoriamente.	El Sistema carga la lista de cámaras desde el Gestor, el mismo construye una vista de árbol con la jerarquía de cámaras, agrupadas por su ubicación física.	Módulo Visor/Ficha Grabación/Botón Cliente de Grabación - Suria Vision Recorder / lista de cámaras.

Caso de prueba para el caso de uso del sistema Visualizar Calendario de Grabación

Descripción General: El caso de uso se inicia cuando el Usuario con privilegios selecciona la opción "Mostrar Calendario en forma de Listas" o cuando selecciona la opción Mostrar Calendario Gráfico. Además se puede obtener información.

Condiciones de Ejecución: El Gestor debe estar disponible y el usuario tiene que tener permisos para realizar esta acción.

Secciones a probar en el Caso de Uso.

Tabla 7 Caso de Prueba del caso de uso Visualizar Calendario de Grabación

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo Central
SC 1: Visualizar Calendario en forma de Listas.	EC 1.1: Visualizar Calendario en forma de Listas con éxito.	El usuario con privilegios selecciona la opción visualizar calendario en forma de lista de reglas y tareas de grabación. Luego el sistema muestra una lista de las reglas de grabación y una lista de las tareas de grabación pendientes y en ejecución.	Módulo Visor/Ficha Grabación/Botón Cliente de Grabación - Suria Vision Recorder / Listas horizontales o listas verticales.

4.4.1.2. Resultados de las pruebas

Las pruebas diseñadas se realizaron a lo largo de todo el ciclo de desarrollo de la aplicación. Como resultado, se obtuvo una aplicación con alta funcionalidad y 100% de cumplimiento de los objetivos trazados.

La ejecución de estas pruebas permitió localizar errores y que estos fueran corregidos mucho antes de dar los toques finales a la aplicación, posibilitando que se hiciera una mejor distribución del tiempo de trabajo. De esta forma, se obtiene una aplicación cuyas funcionalidades han sido comprobadas con anterioridad antes de ponerse en manos de los usuarios finales, lo que garantiza la confiabilidad en la aplicación y la obtención de resultados satisfactorios durante su uso.

A continuación se detallan algunos de los casos de prueba elaborados por cada caso de uso para comprobar el correcto funcionamiento de la aplicación bajo las diferentes condiciones a las que puede someterse.

Tabla 8 Resultados del Caso de Prueba del Caso de Uso Configurar Cliente De Grabación

Nombre del caso de uso: Configurar Cliente De Grabación		
Caso de Prueba del caso de uso Configurar Cliente De Grabación		
Entrada	**Resultados esperados**	**Resultados obtenidos**

Campos vacíos.	Acción fallida. El sistema muestra el mensaje de advertencia “Debe llenar todos los campos”.	Prueba exitosa. Se obtienen los resultados esperados.
Campos con datos incorrectos.	Acción fallida. El sistema muestra un mensaje indicando que los datos introducidos no son válidos.	Prueba exitosa. Se obtienen los resultados esperados.
Campos llenos correctamente.	Acción válida. La aplicación actualiza el fichero de configuración.	Prueba exitosa. Se obtienen los resultados esperados.

Tabla 9 Resultados del Caso de Prueba del caso de uso Gestionar Calendario de Grabación

Nombre del caso de uso: Gestionar Calendario de Grabación		
Caso de prueba para el caso de uso Gestionar Calendario de Grabación		
Entrada	**Resultados esperados**	**Resultados obtenidos**
Sección “Adicionar regla mediante Wizard”		
No seleccionar una cámara.	Acción fallida. El sistema muestra un mensaje indicando que debe seleccionar una cámara para crear regla de grabación.	Prueba exitosa. Se obtienen los resultados esperados.
Campos vacíos.	Acción fallida. El sistema bloquea los botones de navegación hasta que el usuario llene los campos necesarios.	Prueba exitosa. Se obtienen los resultados esperados.
Hora de inicio de grabación: **=** hora de inicio ya existente.	Acción fallida. El sistema muestra un mensaje indicando que la regla a adicionar se solapa con otra.	Prueba exitosa. Se obtienen los resultados esperados.
Hora de fin de grabación: **>** hora de inicio existente. Hora de inicio de grabación: **<** hora de fin existente.	Acción fallida. El sistema muestra un mensaje indicando que la regla a adicionar se solapa con otra.	Prueba exitosa. Se obtienen los resultados esperados.
Hora de fin de grabación: **<** hora de inicio existente. Hora de fin de grabación: **>**	Acción fallida. El sistema muestra un mensaje indicando que la regla a adicionar se solapa con otra.	Prueba exitosa. Se obtienen los resultados esperados.

hora de inicio existente.		
Campos llenos correctamente.	Acción válida. La aplicación solicita al Gestor que agregue la nueva regla de grabación a la base de datos.	Prueba exitosa. Se obtienen los resultados esperados.
Sección "Modificar regla mediante Wizard"		
Campos vacíos.	Acción fallida. El sistema bloquea los botones de navegación hasta que el usuario llene los campos necesarios.	Prueba exitosa. Se obtienen los resultados esperados.
Hora de inicio de grabación: **=** hora de inicio ya existente.	Acción fallida. El sistema muestra un mensaje indicando que la regla a modificar se solapa con otra.	Prueba exitosa. Se obtienen los resultados esperados.
Hora de fin de grabación: **>** hora de inicio existente. Hora de inicio de grabación: **<** hora de fin existente.	Acción fallida. El sistema muestra un mensaje indicando que la regla a modificar se solapa con otra.	Prueba exitosa. Se obtienen los resultados esperados.
Hora de fin de grabación: **<** hora de inicio existente. Hora de fin de grabación: **>** hora de inicio existente.	Acción fallida. El sistema muestra un mensaje indicando que la regla a modificar se solapa con otra.	Prueba exitosa. Se obtienen los resultados esperados.
Campos llenos correctamente.	Acción válida. La aplicación solicita al Gestor que actualice la regla de grabación en la base de datos.	Prueba exitosa. Se obtienen los resultados esperados.
Sección "Eliminar regla mediante Wizard"		
No seleccionar regla de grabación.	Acción fallida. El sistema muestra un mensaje indicando que debe seleccionar reglas de grabación.	Prueba exitosa. Se obtienen los resultados esperados.
Selección de una o varias reglas de grabación.	Acción válida. El sistema pide confirmación al usuario de la acción.	Prueba exitosa. Se obtienen los resultados esperados.
No existen reglas de grabación.	Acción fallida. El sistema muestra el mensaje de advertencia indicando que no existen reglas de grabación.	Prueba fallida. El sistema muestra un mensaje indicando que debe seleccionar reglas de

		grabación.

4.5. Conclusiones parciales

En el desarrollo del capítulo se especificaron los resultados obtenidos. Se presentó la distribución física del sistema y sus componentes mediante el modelo de implementación y el modelo de despliegue; estos permitieron un mejor entendimiento de la distribución física y lógica del sistema. Se definieron los estándares de codificación que se utilizan en la implementación de la aplicación posibilitando un mejor entendimiento del código fuente.

Finalmente, se valida el sistema desarrollado mediante pruebas de caja negra, realizando casos de prueba a todas las funcionalidades. Este proceso permitió detectar la mayor cantidad de no conformidades que éste presentaba para darle solución a las mismas.

Conclusiones Generales

Durante el desarrollo de la presente investigación, mediante el cumplimiento de las tareas y el objetivo propuesto se llegó a las siguientes conclusiones:

- El análisis del control del proceso de grabación de video llevado a cabo por el sistema Suria Vision, permitió conocer los elementos importantes de este; así como el inconveniente que este presenta.
- El análisis de las principales características de las tecnologías y herramientas seleccionadas para la construcción y desarrollo del sistema propuesto, garantizó que la selección de estas viabilice un desarrollo eficaz y efectivo del sistema.
- El rediseño de la interfaz gráfica de la aplicación, permitió una mejor usabilidad a la hora del manejo de la aplicación.
- Se desarrolló una aplicación multiplataforma capaz de gestionar el proceso de grabación de los flujos de video provenientes de las cámaras IP, ya sea por un grupo de reglas de grabación definidas para cada cámara, ante la ocurrencia de un evento en el sistema o de forma manual.
- Con el desarrollo de esta aplicación se ha cumplido con la política tecnológica que sigue el país de migrar al software libre.
- Se ampliará el espectro de clientes potenciales del Sistema Suria Vision debido a que la dependencia de tecnologías privativa ya no es una barrera en su uso.

Recomendaciones

Al concluir el presente trabajo, se recomienda:

- Incorporar a la aplicación la posibilidad de visualizar un registro de las incidencias del sistema y las operaciones realizadas por el usuario cuando ejecuta alguna operación en el sistema.
- Implementar un mecanismo que permita la autenticación de usuarios para mayor seguridad de la aplicación.

Referencias Bibliográficas

1. *Introduccion a las Telecomunicaciones, Señales Electrónicas*. [Página Web]; Disponible en: http://telecomunicaciones-ibeth.blogspot.com/2010_10_01_archive.html.
2. *Diferencia entre Video Digital y Video Analógico*. [Página Web]; Disponible en: http://www.desarrollomultimedia.es/articulos/diferencia-entre-video-digital-y-video-analogico.html.
3. *Tecnología IP para videovigilancia*. Disponible en: http://www.dixita.net/%5Cdocs%5CDixitaseguridadIP.pdf.
4. Victoria. *DefiniciónABC*. Disponible en: http://www.definicionabc.com/general/sistema.php.
5. Guide, V.S. *The history of video surveillance*. [Página Web]; Disponible en: http://www.video-surveillance-guide.com/history-of-video-surveillance.htm.
6. IP, V.C. *Distintas tecnologías de DVR: Grabadora Digital de Video*. Disponible en: http://www.voxdata.com.ar/tiposdvr.html.
7. Axis. *La evolución de los sistemas de vigilancia por video, Sistemas de video IP que utilizan cámaras IP*. [Página Web]; Disponible en: http://www.axis.com/es/products/video/about_networkvideo/evolution.htm.
8. Axis. *Axis Communication*. [Página Web]; Disponible en: http://www.axis.com/products/video/software/index.htm.
9. Labs., S. *About Scati*. [Página Web]; Disponible en: http://www.scati.com/empresa/index.php?id=1.
10. Inc., A.S. *About us*. [Página Web]; Disponible en: http://www.alnetsystems.com/page/en/o_firmie.
11. Inc., V. *Overview.* [Página Web]; Disponible en: http://www.vivotek.com/aboutus/overview.php.
12. Datys. *Datys, tegnologías & Sistemas*. [Página Web]; Disponible en: http://www.datys.cu/.
13. SOURCEFORGE.NET. *wxCam*. [Página Web]; Disponible en: http://wxcam.sourceforge.net/.
14. *Domótica en Linux!, Videovigilancia, Devolution security*. [Página Web]; Disponible en: http://www.ubuntu-es.org/index.php?q=node/14035.
15. [Página Web]; Disponible en: http://www.zoneminder.com/.
16. *What is Motion?* . [Página Web]; Disponible en: http://www.lavrsen.dk/foswiki/bin/view/Motion/WebHome.
17. Marble. *Metodologías de desarrollo*. [Página Web] 2008; Disponible en: http://www.marblestation.com/?p=644.

18. I. Jacobson, G.B., J. Rumbaugh, *Cap. 1*, in *"El Proceso Unificado de Desarrollo del Software"*. 2000, Addison Wesley: Madrid.
19. *UML*. [Página Web]; Disponible en: http://www.ecured.cu/index.php/UML.
20. *Ingeniería De Software I*. Disponible en: http://www.scribd.com/doc/3062020/Capitulo-I-HERRAMIENTAS-CASE.
21. Systems, S. *Características de Enterprise Architect* Disponible en: http://www.sparxsystems.com.ar/products/ea_features.html.
22. *Lenguaje de Programación*. Disponible en: http://www.scribd.com/doc/13719562/Lenguaje-de-Programacion.
23. Codebox. *FRAMEWORK*. [Página Web]; Disponible en: http://www.codebox.es/glosario.
24. Ecured. *Qt*. [Página Web]; Disponible en: http://www.ecured.cu/index.php/Qt.
25. *Entornos de Desarrollo Integrado*. Disponible en: http://petra.euitio.uniovi.es/~i1667065/HD/documentos/Entornos%20de%20Desarrollo%20Integrado.pdf.
26. Ecured. *Qt Creator*. [Página Web]; Disponible en: http://www.ecured.cu/index.php/Qt_Creator.
27. *XML-RPC*. [Página Web]; Disponible en: http://www.multilingualarchive.com/ma/enwiki/es/XML-RPC.
28. EcuRed. *Sistemas de control de versiones*. [Página Web]; Disponible en: http://www.ecured.cu/index.php/Sistemas_de_control_de_versiones.
29. Kalinovskaya, D.D.C. and R.O. Milán. *Sistema de Gestión de la Configuración para Repositorios*. [Tesis] 2010; Disponible en: http://bibliodoc.uci.cu/TD/TD_02863_10.pdf.
30. Collins-Sussman, B., B.W. Fitzpatrick, and C.M. Pilato. *Control de versiones con Subversion*. Disponible en: http://www.assembla.com/spaces/project_ADM/documents/dzEreK0q0r3zFYab7jnrAJ/download/svn-book.pdf.
31. Carlos Reynoso and N. Kicillof. *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft*. Estilos Centrados en Datos; Disponible en: http://es.scribd.com/Francia_Mascar_6388/d/56600616/8-Arquitecturas-de-Pizarra-o-Repositorio.
32. *Patrones de Diseño*. [Página Web]; Disponible en: http://es.scribd.com/doc/71964170/Patron-de-diseno.
33. Ecured. *Flujo de Trabajo Análisis y Diseño*. Disponible en: http://www.ecured.cu/index.php/Flujo_de_Trabajo_An%C3%A1lisis_y_Dise%C3%B1o.

34. Larman, C., *UML Y PATRONES: Introducción al análisis y diseño orientado a objetos*, ed. E.F. Varela. Vol. Tomo 1. 2004. pág.185 - 215.
35. *Definición de modelo de datos*. Disponible en: http://definicion.de/modelo-de-datos/.
36. *ESTÁNDAR PARA CODIFICACIÓN EN LENGUAJE C ++*. Disponible en: http://cursos.iteso.mx/moodle/pluginfile.php/18971/mod_folder/content/1/Programacion_estructurada/estandarcodificacion.pdf.
37. *Fundamentos de Ingeniería de Software*. Disponible en: http://www.inf.utfsm.cl/~visconti/ili236/Documentos/15-Implementacion.pdf.
38. *Prueba de RUP*. Disponible en: http://www.ecured.cu/index.php/Prueba_de_RUP.
39. *Prueba de RUP*. [Página Web]; Disponible en: http://www.ecured.cu/index.php/Prueba_de_RUP.
40. *Prueba de RUP, Tipo de Prueba adecuada al Nivel de Unidad*. [Página Web]; Disponible en: http://www.ecured.cu/index.php/Prueba_de_RUP#Tipo_de_Prueba_adecuada_al_Nivel_de_Unidad.

Bibliografía

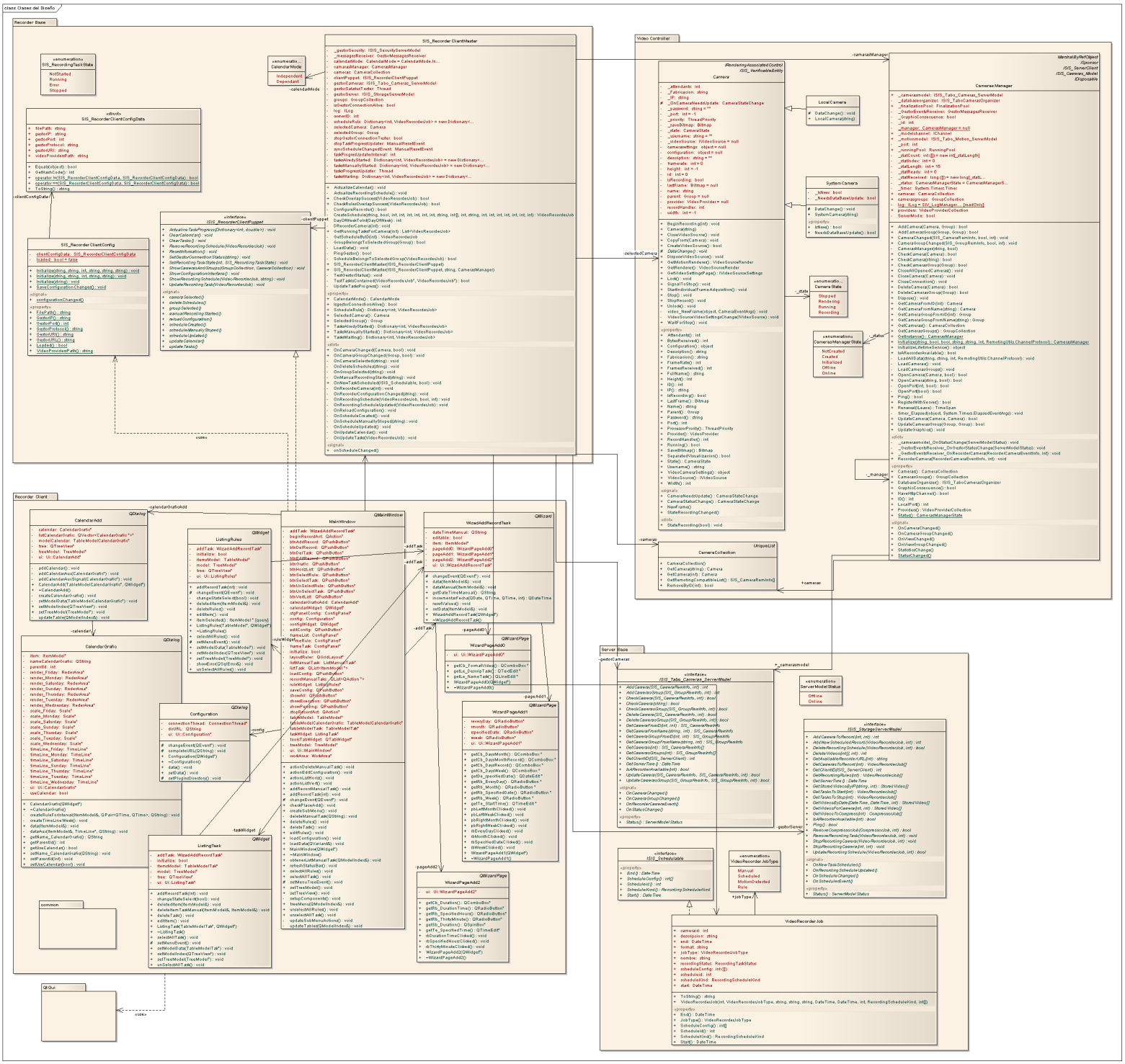
1. Introducción al video digital. [Página Web]; Disponible en: http://es.kioskea.net/contents/video/video.php3.
2. Grabar. [Página Web]; Disponible en: http://www.infoconsumo.es/consa3/ES/seminar/gloss_in_es.htm.
3. Introduccion a las Telecomunicaciones, Señales Electrónicas. [Página Web]; Disponible en: http://telecomunicaciones-ibeth.blogspot.com/2010_10_01_archive.html.
4. Diferencia entre Video Digital y Video Analógico. [Página Web]; Disponible en: http://www.desarrollomultimedia.es/articulos/diferencia-entre-video-digital-y-video-analogico.html.
5. Tecnología IP para videovigilancia.; Disponible en: http://www.dixita.net/%5Cdocs%5CDixitaseguridadIP.pdf.
6. DVR vs NVR. [Página Web]; Disponible en: http://sites.google.com/site/cctvmoran/in-the-news.
7. Guide, V.S. The history of video surveillance. [Página Web]; Disponible en: http://www.video-surveillance-guide.com/history-of-video-surveillance.htm.
8. IP, V.C. Distintas tecnologías de DVR: Grabadora Digital de Video.; Disponible en: http://www.voxdata.com.ar/tiposdvr.html.
9. Axis. La evolución de los sistemas de vigilancia por video, Sistemas de video IP que utilizan cámaras IP. [Página Web]; Disponible en: http://www.axis.com/es/products/video/about_networkvideo/evolution.htm.
10. Axis. Axis Communication. [Página Web]; Disponible en: http://www.axis.com/products/video/software/index.htm.
11. Labs., S. About Scati. [Página Web]; Disponible en: http://www.scati.com/empresa/index.php?id=1.
12. Inc., A.S. About us. [Página Web]; Disponible en: http://www.alnetsystems.com/page/en/o_firmie.
13. Inc., V. Overview. [Página Web]; Disponible en: http://www.vivotek.com/aboutus/overview.php.
14. Datys. Datys, tegnologías & Sistemas. [Página Web]; Disponible en: http://www.datys.cu/.
15. SOURCEFORGE.NET. wxCam. [Página Web]; Disponible en: http://wxcam.sourceforge.net/.
16. [Página Web]; Disponible en: http://www.zoneminder.com/.
17. Marble. Metodologías de desarrollo. [Página Web] 2008; Disponible en: http://www.marblestation.com/?p=644.
18. Jacobson, G.B., J. Rumbaugh, Cap. 1, in "El Proceso Unificado de Desarrollo del Software". 2000, Addison Wesley: Madrid.

Bibliografía

19. Jacobson, G.B., J. Rumbaugh, Breve resumen de UML in "El Lenguaje Unificado de Modelado. Manual de Referencia". 2000, Addison Wesley: Madrid.
20. Ingeniería De Software I.; Disponible en: http://www.scribd.com/doc/3062020/Capitulo-I-HERRAMIENTAS-CASE.
21. Entornos de Desarrollo Integrado.; Disponible en: http://petra.euitio.uniovi.es/~i1667065/HD/documentos/Entornos%20de%20Desarrollo%20Integrado.pdf
22. Larman, C., *UML Y PATRONES: Introducción al análisis y diseño orientado a objetos*, ed. E.F. Varela. Vol. Tomo 1. 2004. pág.185 - 215.

Anexos

Anexo #1. Clases del diseño del Cliente de Grabación



Anexo #2. Forma de organizar el código fuente

```
#define Properties {
 int id() const;
 QString Url() const;
#define Properties}

#define Identificadores }
 TreeModel *treeModel;
 TableModel *tableModel;
 TableModelTak *tableModelTask;
 WizadAddRecordTask *addTask;
 Configuration *config;
#define Public Identificadores}

#define Public Methods {
 void initialize();
 void startConnecting();
 void connectToModel();
#define Public Methods}

#fefine Slot {
 void actionEditConfiguration();
 void deleteRules();
 void deleteTask();
 void refreshStatusBar();
#fefine Slot}
```

Anexo #3. Código fuente de la aplicación

```
85  void MainWindow::onShowRecordingSchedule(VideoRecorderJob *task, QString cameraName)
86  {
87      QList<ItemModel *> r;
88      QList<ItemModel *> t;
89      int id = item_select->getID();
90      if(task->getJobType() ==Rule && task->getJobType()!= MotionDetected)
91      {
92          ruls.append(task);
93      }
94      else if(task->getJobType() ==Manual && task->getJobType() == Scheduled)
95      {
96          tasks.append(task);
97      }
98
99      r = itemsRules(false,id, ruls);
100     t = itemsTasks(false,id, tasks);
101
102     tableModel = new TableModel(r, item_select->getTypeModel());
103     tableModelTask = new TableModelTak(t, item_select->getTypeModel());
104     layoutRuler->update();
105 }
106
```

```
1465 void SIS_RecorderClientMaster::OnCameraGroupChangedEventHandler(Group pGroup, bool Deleted)
1466 {
1467     Q_UNUSED(pGroup);
1468     Q_UNUSED(Deleted);
1469     try
1470     {
1471         clientPuppet->ShowCamerasAndGroups(gestorCameras->GetCamerasGroups(ownerID), gestorCameras->GetCameras(ownerID));
1472     }
1473     catch (...)
1474     {
1475         qCritical("Ha ocurrido un erro al actualizar el listado de camaras");
1476     }
1477 }
1478
1479 void SIS_RecorderClientMaster::OnRecordingScheduleUpdatedHandler(VideoRecorderJob *task)
1480 {
1481     if (scheduleRuls.contains(task->getScheduleid()))
1482     {
1483         scheduleRuls[task->getScheduleid()] = task;
1484     }
1485     if (clientPuppet != NULL)
1486         clientPuppet->UpdateRecordingTask(task);
1487 }
```

```
39  TableModel::TableModel(const TableModel &table, QAbstractTableModel *parent)
40      : QAbstractTableModel(parent)
41  {
42      items = QList<ItemModel *>();
43      for(int i = 0; i < table.items.count(); i++)
44      {
45          if(table.items[i]->getTypeModel() == ItemModel::camera)
46          {
47              QList<ItemModel *> camera = table.items[i]->getChildItems();
48              for(int j(0);  j < camera.size(); ++j)
49              {
50                  if(camera[j]->getTypeModel() == ItemModel::rules)
51                  {
52                      ItemModel *temp=camera[j];
53                      items.append(temp);
54                  }
55
56              }
57          }
58          else if(table.items[i]->getTypeModel() == ItemModel::rules)
59          {
60              ItemModel *temp=table.items[i];
61              items.append(temp);
62          }
63      }
64      type= table.type;
65  }
```