Universidad de las Ciencias Informáticas
FACULTAD 6

Título: Estación de Monitorización v2.0 en QT para el Sistema de Video Vigilancia Suria.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Enrique Jesús Expósito Aquino.

Tutor: Ing. Reynier Pupo Gómez.

Junio, 2012

Año 54 de la Revolución.

FRASE

"Nunca consideres el estudio como una obligación, sino como una oportunidad para penetrar en el bello y maravilloso mundo del saber."

Albert Einstein

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis y reconozco 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 25 días del mes de Junio del año 2012.

Enrique Jesús Expósito Aquino

_____
Firma del Autor

Ing. Reynier Pupo Gómez

_____
Firma del Tutor

DATOS DE CONTACTO

Tutor: Ing. Reynier Pupo Gómez (rgomez@uci.cu)

Graduado de la Universidad de las Ciencias Informáticas (UCI) en el año 2010-2011. Es Instructor Recién Graduado. Pertenece al Departamento Señales Digitales del Centro de Geoinformática y Señales Digitales (GEYSED) de la Facultad 6. Actualmente se encuentra de jefe del módulo Video Vigilancia en Qt.

AGRADECIMIENTOS

A mi mamá, por ser mi confidente, mi amiga, mi todo, por estar ahí para mí cuando me ha hecho falta.

A mi papá, por hacerme la persona que soy, por confiar en mí y por darme los cocotazos cuando hicieron falta.

A mi hermano, por ser mi guía, un ejemplo a seguir y al que siempre he querido parecerme.

A mis abuelos…

A mi abuelo, Jesús (papichu) que aunque ya no lo tengamos entre nosotros siempre va a estar aquí a mi lado.

A mi abuela Graciela(ita) por todo su amor y cariño y por ser tan especial para mí.

A mi pareja Yurisdelvys, por haberme ayudado, soportado, por estar a mi lado, por darme su apoyo incondicional durante estos 5 años.

A mis tios, a todos, en especial a la gordita linda (Isabel) que ha sido como una segunda madre para mí.

A mis primos, a todos, en especial a Luisito que la vida le deparó una mala pasada.

A mis otros 2 hermanos , que a pesar de no tener la misma sangre nos queremos como tal, Denis y Reisel.

A todos mis amigas, Ale, Grethell, Yanet, Magela, Olivia, en fin a todas por soportar mis pesadeces..

A mis amigos, a los del cuarto, Jose, Lambert, a las tropas Alex, Triana, Leandro, Julio, Elier..., a otros que por último no dejan de ser importantes Guillermo Rubisel... y a todos en general y a los del grupo de primer año.

Al tutor, tribunal y profesores gracias a los cuales salió esta tesis con la calidad requerida.

DEDICATORIA

Le dedico este trabajo de diploma a mi mamá y a mi papá, por ser los mejores padres del mundo.

A mi hermano, por ser el mejor.

En especial, a mis abuelos Jesús y Graciela, por

brindarnos tanto amor, por haber construido una familia tan linda y unida.

RESUMEN

Los sistemas de video vigilancia han tenido un gran auge en todo el mundo, por las prestaciones que suelen brindar en los lugares donde son instalados. En Cuba, estos avances tecnológicos han sido percibidos y tomados en cuenta, ya que existen diferentes áreas que necesitan de estos servicios. Por tal motivo en la Universidad de las Ciencias Informáticas (UCI), existe un proyecto llamado Video Vigilancia que ha creado un sistema de este tipo que lleva por nombre Suria Vision. El mismo cuenta con un módulo llamado Estación de Monitorización, desarrollado completamente con tecnologías libres y que implementa las funcionalidades básicas de un visor de video vigilancia. Actualmente, en el módulo existe un usuario único definido de manera estática con el rol de administrador y con permisos determinados de la misma forma. Además, no existe la forma de visualizar los sucesos ocurridos en todas las cámaras al mismo tiempo y se hace difícil llevar un seguimiento de las acciones realizadas por los usuarios. Para darle solución a estos problemas se implementó la segunda versión de la Estación de Monitorización, siendo este el objetivo de la presente investigación. La misma cuenta con las funcionalidades necesarias para realizar la gestión de usuarios, roles y la configuración de permisos. Además, permite notificar al usuario y al administrador si ha ocurrido algún problema en una cámara, a través de alarmas sonoras, visuales y de correo electrónico.

Palabras Claves: Cámara, Estación de Monitorización, Video Vigilancia

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

En el transcurso de la historia el hombre ha tenido la necesidad de ver lo que sucede a su alrededor y velar por los lugares y recursos que le son importantes. Ha evolucionado la forma de realizar esta actividad con la ayuda de los avances informáticos, dando lugar al surgimiento de una nueva tecnología llamada video vigilancia, la cual permite colocar cámaras en diferentes zonas y poder observar a través de la señal de video los bienes personales desde cualquier parte.

En la actualidad se emplean diversos sistemas de video vigilancia que han gozado de una gran popularidad y efectividad en cuanto a controladores de acceso y guardias de seguridad se refiere. Los primeros pasos fueron a partir de los años 60 en Inglaterra, donde se utilizaron dos cámaras en 1960 para controlar las multitudes atraídas por la familia real tailandesa. Luego, al darse cuenta del potencial que tenían se decidió en 1961 instalar un sistema de video vigilancia en una estación de trenes de Londres. Inmediatamente fueron ampliando más el círculo, instalando sistemas de video vigilancia en cuatro de sus principales estaciones de trenes subterráneo (1) . Al mismo tiempo, se comenzó a monitorear el flujo del tráfico en las carreteras principales. Por otra parte, informes de prensa indican que en 1965, la policía de los Estados Unidos (EEUU) empezó a utilizar esta tecnología para controlar el comportamiento social en lugares públicos, no pasó mucho tiempo antes de que la práctica se extendiera, los agentes de policía vigilaban de cerca en áreas claves, creando así los primeros Sistemas de Circuito Cerrado de Televisión (CCTV). EEUU en la década de 1980, se dio a la tarea de recuperar el tiempo perdido y comenzaron a desplegar ampliamente sistemas de video vigilancia en las zonas públicas (2).

Los acontecimientos del 11 de septiembre de 2001 cambiaron la percepción del público sobre la vigilancia de video. Los desarrolladores de software comenzaron a crear programas que aumentaran la vigilancia por videocámara. Los programas de reconocimiento facial son un ejemplo de estos, con el uso de los puntos clave de las características faciales, se registran las caras en comparación con las fotografías de los terroristas y delincuentes, para darle seguimiento de forma automática a través de las cámaras vigilantes (2).

En Cuba, estos avances tecnológicos han sido percibidos y tomados en cuenta, ya que también se cuenta con diferentes áreas que necesitan de estos servicios. Después de ser víctimas de los atentados terroristas contra los hoteles cubanos en la década de los 90, el país ha insistido para establecer esta tecnología, haciendo inversiones con empresas extranjeras para implantar sistemas de este tipo. En estos momentos se cuenta con varios lugares públicos como hoteles, centros comerciales, plazas y avenidas que tienen instalado estos servicios para mantener el control y orden con mayor eficiencia, detectando por esta vía cualquier infracción cometida.

En la Universidad de Ciencias Informáticas (UCI), en la facultad 6, se encuentra el Centro de Desarrollo de Geoinformática y Señales Digitales (GEySED). Específicamente en el Departamento de

Señales Digitales existe un proyecto productivo llamado Video Vigilancia, en el cual se realizó una aplicación de este tipo, tiene como nombre Suria Vision y cuenta actualmente con una versión de escritorio desarrollada sobre el marco de trabajo .Net.

Con la nueva política del país de migrar todos sus sistemas informáticos hacia software libre, este proyecto se ha dado a la tarea de crear otra versión del componente Estación de Monitorización conocido también como Visor, este se desarrolló totalmente con tecnologías libres. Se creó inicialmente una versión funcional, que cuenta con las funcionalidades básicas de un visor de video vigilancia.

Para poder autenticarse en la aplicación, sólo puede ser mediante un usuario único definido estáticamente en el gestor, con un rol de administrador y sus permisos determinados de la misma forma, trayendo consigo que todos los usuarios se autentiquen bajo una misma identidad y con un grado de privilegios elevado. Otro problema identificado en la aplicación, es que el usuario no tiene forma de visualizar los sucesos ocurridos en todas las cámaras al mismo tiempo, como detección de movimiento o interrupción de la conexión, por lo que se hace engorroso este proceso. Por otra parte, a los administradores del sistema les es difícil llevar un seguimiento de las acciones realizadas por los usuarios, ya sea con fines de mantenimiento o análisis de incidencias que han pasado en un determinado período de tiempo.

La situación descrita anteriormente lleva a plantearse el siguiente **problema a resolver:** ¿cómo lograr un mejor funcionamiento de la estación de monitorización de Suria Vision en Qt? Para darle solución al problema antes planteado se define como **objeto de estudio:** el proceso de monitorización desde cámaras IP para Suria Vision, enmarcado en el **campo de acción:** nuevas funcionalidades para la Estación de Monitorización del sistema Suria Vision.

Para limitar el alcance de la investigación y darle solución al problema planteado se concreta como **objetivo general**: implementar nuevas funcionalidades para la estación de monitorización del Sistema Suria Vision. Del mismo se desprenden los siguientes **objetivos específicos:**

1. Implementar funcionalidades que permitan un fácil manejo de los usuarios en la Estación de Monitorización.
2. Implementar funcionalidades que permitan activar un grupo de alarmas según el estado de las cámaras.
3. Implementar funcionalidades que permitan brindar un servicio de reporte.
4. Validar el módulo desarrollado mediante pruebas.

Teniendo como **idea a defender** que: “la implementación de nuevas funcionalidades permitirá un mejor desempeño de la Estación de Monitorización para el Sistema Suria Vision”**.**

Tareas de la Investigación:

1. Realización de un análisis de las tecnologías más novedosas que se usarán para la implementación del sistema.
2. Implementación de nuevas funcionalidades para la Estación de Monitorización.
3. Realización de las pruebas.

Los métodos de la investigación científica utilizados en el presente trabajo se explican a continuación:

Métodos Teóricos.

Para el análisis de las estaciones de monitorización existentes se emplea el análisis **histórico – lógico** realizando un estudio sobre su evolución y desarrollo. Se utiliza el método **analítico – sintético** para el estudio de bibliografías de diferentes autores para poder realizar una amplia investigación sobre los elementos que se relacionan con el objeto de estudio.

Métodos Empíricos.

La **observación** para la realización de valoraciones y obtención de información sobre el funcionamiento de sistemas similares al que se desarrolla, para tener una idea de cómo tiene que ser el sistema.

A continuación se presenta la estructura que seguirá la investigación por capítulos:

CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA: Se realiza un análisis del estado del arte de los sistemas existentes tanto a nivel nacional como internacional, percibiendo las características de las diferentes estaciones, así como sus ventajas y desventajas. Además, se realiza la selección de la metodología de desarrollo así como las tecnologías y herramientas que se van a utilizar en el desarrollo del sistema.

CAPÍTULO II. DISEÑO E IMPLEMENTACIÓN: Se realiza la especificación de los requisitos funcionales y de los casos de uso del sistema, mediante el diagrama de casos de uso. Además para dar solución a la presente investigación, se realizan los diagramas de despliegue y componentes.

CAPÍTULO III. PRUEBAS AL SISTEMA: Se describen las principales pruebas realizadas al sistema con el objetivo de validar que el mismo funcione de manera correcta y cumpla con las especificaciones planteadas en las descripciones de casos de uso.

CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA.

1.1 Introducción.

En el presente capítulo se exponen conceptos que resultan esenciales en la investigación y que contribuyen al esclarecimiento de su objeto de estudio. Se realiza un análisis más profundo de la problemática antes planteada con el objetivo de lograr una mayor comprensión de la necesidad actual. Se realiza un estudio sobre algunas de las soluciones de video vigilancia existentes, analizando sus principales ventajas y desventajas. Además, se justifica la elección de las herramientas y tecnologías que sustentarán el desarrollo de la solución.

1.2 Principales conceptos asociados al problema.

La Real Academia Española hace referencia a varios conceptos del término **video**. La investigación se centra en que: un video es un sistema de grabación y reproducción de imágenes, acompañadas o no de sonidos, mediante cinta magnética (3). Esta es la definición que más se adecua a la situación tratada en la investigación. Además se puede añadir que un video es una secuencia de imágenes en transición que representan un conjunto de escenas en movimiento.

Con el objetivo de capturar las secuencias de imágenes que conforman un video se crearon las **cámaras de video**. Uno de los tipos existentes son las **cámaras IP**, definidas como cámaras que se conectan directamente a la red empleando el protocolo IP[1] y un servidor web[2] (4).

Para garantizar la supervisión local y/o remota de imágenes de video captadas por cámaras se utiliza la tecnología de **video vigilancia**. La constitución española define un sistema de video vigilancia como "todo software instalado en un espacio público o privado, para el cuidado y protección de personas y bienes contra intrusión, agresión, robo o hurto. Permitiendo obtener imágenes y sonidos que puedan establecer pautas de comportamiento que supongan intervención policial o afecten a la seguridad ciudadana" (5). La **video vigilancia IP** es una efectiva solución de seguridad que ofrece monitorización y control avanzado en sistemas de seguridad, emplea tecnología analógica para efectuar la vigilancia (6).

La **monitorización** hace referencia a la habilidad de controlar y observar el curso de un parámetro determinado para descubrir alguna anomalía existente (3).

[1] **Protocolo IP**: Protocolo que forma parte de la capa de Internet del conjunto de protocolos TCP/IP. Permite el envío de paquetes de datos a través de la red.

[2] **Servidor web**: programa informático que procesa una aplicación del lado del servidor.

1.3 Evolución de los sistemas de video vigilancia.

Los primeros pasos fueron a partir de los años 60 en Inglaterra donde se comenzó a monitorear el flujo del tráfico en las carreteras principales. Por otra parte en 1965, la policía de los EEUU empezó a utilizar esta tecnología para controlar el comportamiento social en lugares públicos, los agentes de policía vigilaban de cerca en áreas claves, creando así los primeros Sistemas de Circuito Cerrado de Televisión (CCTV). Empezaron siendo sistemas analógicos[3] al ciento por ciento y paulatinamente se fueron digitalizando. Los sistemas de hoy día han avanzado mucho desde la aparición de las primeras cámaras analógicas.

En la actualidad, estos sistemas utilizan cámaras y servidores de PC para la grabación de video en un sistema completamente digitalizado. Sin embargo, entre los sistemas completamente analógicos y los sistemas completamente digitales existen diversas soluciones que son parcialmente digitales. Inicialmente surgieron los Sistemas de Circuito Cerrado de TV (CCTV) los cuales eran completamente analógicos, donde se utilizaba un Grabador de Video (VCR, por sus siglas en inglés) el cual guardaba la información en una caja de cinta magnética (video casete), sustituyéndose posteriormente por un Grabador de Video Digital (DVR, por sus siglas en inglés) donde se utilizaba un disco duro para guardar las grabaciones. Estos sistemas estaban formados por cámaras del mismo tipo, con salida coaxial (7) (8). *(Ver Figura 1y 2)*

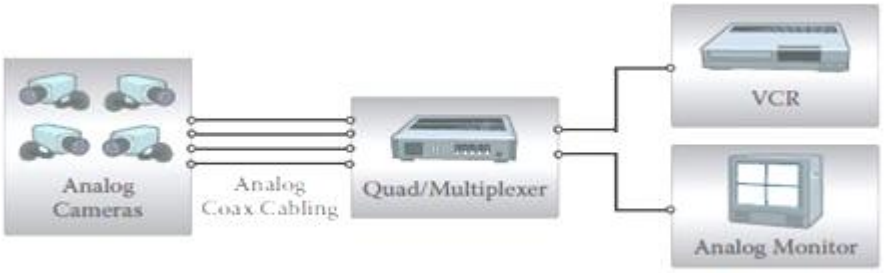


Figura 1. Sistemas de circuito cerrado de TV analógicos usando VCR.

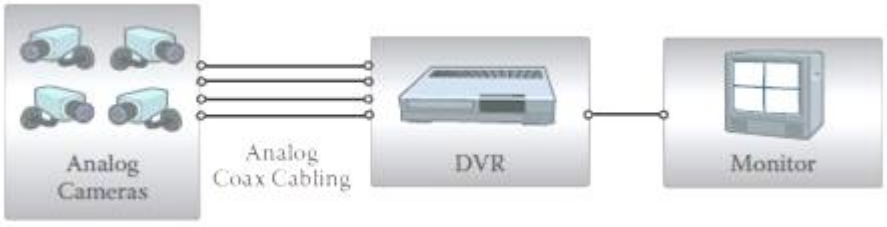


Figura 2. Sistemas de circuito cerrado de TV analógicos usando DVR.

[3] **Sistemas Analógicos:** las magnitudes de la señal se representan mediante variables continuas, las cantidades varían sobre un intervalo continuo de valores.

Con el surgimiento de la señal digital se crean los sistemas híbridos naciendo así una segunda etapa. Donde se usaba un CCTV analógico junto a un DVR con tecnología IP, sistema parcialmente digital que incluye un DVR equipado con un puerto Ethernet para conectividad de red. Como el video se digitaliza y comprime en el DVR, se puede transmitir a través de una red informática para que se monitorice en una PC desde una ubicación remota (7) (8). (*Ver Figura3*)

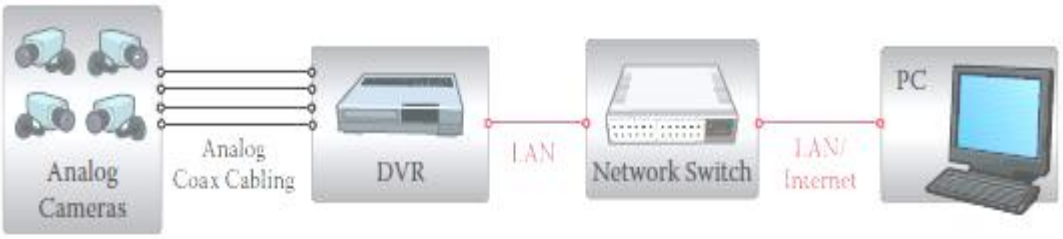


Figura 3. Sistemas de circuito cerrado de TV analógicos usando DVR de red.

En la actualidad se cuenta con los sistemas de tercera generación completamente digitales, en el cual se utiliza una cámara IP, un conmutador y un ordenador. El video se transmite a través de una red IP, mediante los conmutadores de red (Switch) y se graba en una estación de trabajo con un software de gestión de video, lo que incluye la digitalización y la compresión del video. Esto representa un verdadero sistema de video IP donde no se utilizan componentes analógicos (7) (8). (*Ver Figura 4*)

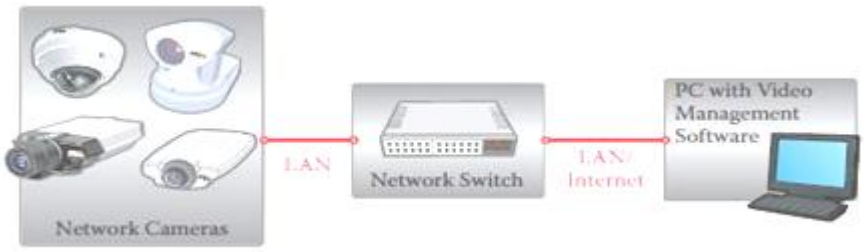


Figura 4. Sistemas de video IP que utilizan cámaras IP.

1.4 Descripción de la situación problemática.

Actualmente en Cuba se hacen grandes esfuerzos para aumentar el desarrollo de la informática y las comunicaciones. Siendo la Universidad de las Ciencias Informáticas (UCI) un ejemplo de estos avances. En esta universidad existe un proyecto productivo que creó un sistema de video vigilancia, llamado Suria Vision.

Suria Vision es un sistema integral de video vigilancia inteligente de tercera generación, compuesto por varios módulos. Tiene como ventaja que no se limita a un proveedor de cámaras específico para su

funcionamiento, por el contrario, el sistema soporta equipos de diferentes modelos. A diferencia de la mayoría de los sistemas de video vigilancia existentes en el mundo, pues no son compatibles con cualquier tipo de fabricante. Este sistema está realizado sobre la plataforma .Net, utilizando tecnologías privativas para su desarrollo, teniendo como desventaja su acoplamiento al sistema operativo Windows, por lo que no es posible su instalación en lugares donde se usen diferentes sistemas operativos, y no cumple con las nuevas políticas del país de migrar a software libre.

Por tal motivo este proyecto se dio a la tarea de crear una versión con tecnologías libres. Inicialmente se realizó el módulo Estación de Monitorización, al cual sólo se puede acceder mediante un sistema de autenticación, existiendo un único usuario creado de manera estática en el gestor, donde se le definió un conjunto de permisos pertenecientes a un rol de administración. Trayendo como consecuencia que todos se autentiquen con una misma identidad y por tal motivo no hay forma de distinguir y restringir a los usuarios que acceden a la aplicación. Además, los usuarios no tienen forma de conocer los eventos que ocurren al mismo tiempo en todas las cámaras, como detección de movimiento o interrupción de la conexión, siendo este procedimiento complejo para su identificación. Por otra parte a los administradores del sistema les es engorroso tener control de las acciones realizadas por los usuarios, por ejemplo: la eliminación de una grabación, modificación de los datos de una cámara o la autenticación en horarios indebidos, de lo cual es necesario mantener un control estricto sobre dichas actividades, pues son de suma importancia para los sistemas de video vigilancia.

1.5 Análisis de soluciones existentes.

En las versiones de la Estación de Monitorización se realizó un estudio de las soluciones existentes, tanto en Cuba como en el mundo, llegando a la conclusión de que presentan varias limitantes, lo que impidió su utilización en la creación de las aplicaciones anteriores. Por tal motivo se hace necesario realizar la implementación de una segunda versión, pues no se puede utilizar ningún componente para manipular las funcionalidades de visualización, no son multiplataforma o tienen un alto costo de adquisición y el hardware que necesitan depende completamente de su fabricante. A continuación se presenta un resumen de las características de las principales empresas analizadas que argumentan la afirmación anterior.

Axis.

Axis es una compañía sueca fundada en 1984 que ofrece soluciones de video IP dirigidas al mercado profesional. La compañía es líder del mercado del video IP, conduciendo el cambio de video vigilancia analógica hacia las soluciones digitales. Axis ofrece una gama completa de soluciones para un amplio abanico de segmentos y aplicaciones industriales. Las soluciones que desarrolla no brindan una

interfaz de comunicación que permita acceder a funcionalidades necesarias para el desarrollo de la aplicación (9).

Datys.

Datys, creada desde el año 2005, especializada en la creación de software de video vigilancia profesional, cuenta con muchas soluciones en diferentes áreas, destacándose Xyma Safe Vision, la cual no es multiplataforma y tiene un alto costo de adquisición (10).

Vivotek.

Vivotek establecido en el año 2000, ha tomado rápidamente su lugar como uno de los principales fabricantes en la industria de vigilancia IP. Vivotek se especializa en la integración de componentes audio-visuales en redes computacionales. Usando tecnologías sofisticadas de códec[4], el equipo de ingenieros innovador de Vivotek desarrolla y dirige una amplia gama de productos de vigilancia IP. Cuenta con una solución llamada Vast, la cual presenta un alto costo de adquisición y las cámaras que utiliza dependen completamente de su fabricante (11).

1.6 Metodologías, herramientas y tecnologías a utilizar para el desarrollo de la solución.

Para desarrollar un software con calidad, es necesario escoger las herramientas y tecnologías que permitan crear una aplicación que cumpla con las características necesarias para su correcto funcionamiento. En este caso se utilizarán las mismas que se emplearon en la realización de la versión anterior de la Estación de Monitorización, con el objetivo de garantizar una mejor integración con lo anteriormente implementado.

1.6.1 Características del lenguaje de programación a utilizar.

Un lenguaje de programación es una técnica estándar de comunicación que permite expresar las instrucciones que han de ser ejecutadas en una computadora. Consiste en un conjunto de reglas sintácticas y semánticas que definen un lenguaje informático. Un lenguaje de programación permite a un programador especificar de manera precisa sobre qué datos una computadora debe operar, cómo deben ser estos almacenados y transmitidos y qué acciones debe tomar bajo una variada gama de circunstancias (12).

El lenguaje C++ fue diseñado a mediados de los años 1980, por Bjarne Stroustrup, como extensión del lenguaje de programación C; abarca tres paradigmas de la programación: la programación estructurada, la programación genérica y la programación orientada a objetos, es uno de los más

[4] **Códec:** es la abreviatura de codificador-decodificador, capaz de transformar un archivo con un flujo de datos o una señal.

empleados en la actualidad. Algunas de las características que presenta, por las cuales se escogió como lenguaje de programación a utilizar son:

- Está estandarizado y un mismo código fuente puede ser compilado en diversas plataformas.
- Es muy potente en lo que se refiere a creación de sistemas complejos, un lenguaje muy robusto.
- Actualmente, puede compilar y ejecutar código de C, ya que viene con librerías para realizar esta labor.
- Es un lenguaje muy empleado, existen varios tutoriales en línea, libros y códigos fuentes abiertos.
- Existe una gran cantidad de compiladores, depuradores y librerías.
- Posee tratamiento de memoria (13).

1.6.2 Framework a utilizar.

Qt v4.7.2, es una biblioteca multiplataforma para desarrollar interfaces gráficas de usuario. Qt utiliza el lenguaje de programación C++ de forma nativa, adicionalmente puede ser utilizado en otros lenguajes de programación. La API[5] de la biblioteca cuenta con métodos para acceder a bases de datos mediante SQL[6], así como uso de XML[7], gestión de hilos, soporte de red, una API multiplataforma unificada para la manipulación de archivos y otros para el manejo de ficheros, además de estructuras de datos tradicionales (14).

1.6.3 Entorno de Desarrollo Integrado (IDE) a utilizar.

Un entorno de desarrollo integrado (IDE) es un programa compuesto por un conjunto de herramientas para un programador. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien, puede utilizarse para varios. Los IDE han sido empaquetados como programas de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica GUI[8]. Los IDEs proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación (15).

Qt Creator v2.1.0, es una plataforma de entorno de desarrollo integrado disponible junto con las bibliotecas Qt. Es distribuido bajo tres tipos de licencias: Qt Commercial Developer License, Qt GNU LGPL v. 2.1, Qt GNU GPL v. 3.0, está disponible para las plataformas: Linux, Mac OSX y Windows.

Principales características de Qt Creator:

[5] **API (Application Programming Interface)**: Interfaz para programación de aplicaciones.
[6] **SQL**: (Structured Query Language): Lenguaje de Consulta Estructurado.
[7] **XML:** Metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C).
[8] **GUI**: Interfaz Gráfica de Usuario.

- Posee un avanzado editor de código C++.
- Posee también una GUI integrada y diseñador de formularios.
- Ayuda sensible al contexto integrado.
- Depurador visual.
- Resaltado y auto-completado de código (16).

Las características expuestas anteriormente permitieron escoger QT Creator como IDE de desarrollo.

1.6.4 Herramienta CASE.

Las herramientas CASE[9] son numerosas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero.

Visual Paradigm v5.0, constituye una herramienta CASE que utiliza UML v8.0 (Lenguaje Unificado de Modelado), como lenguaje de modelado. Soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite realizar ingeniería tanto directa como inversa, tiene la capacidad de crear el esquema de clases a partir de una base de datos y crear la base de datos a partir del esquema de clases. Genera la documentación del proyecto automáticamente en varios formatos como Web y PDF[10]. Permite el control de versiones, exportar las imágenes, es multiplataforma, además es muy sencillo de usar, instalar y actualizar. Visual Paradigm admite compatibilidad con las demás versiones (17).

1.6.5 Metodología de desarrollo.

Las metodologías de desarrollo de software se dividen en dos grupos estos son Metodologías Ágiles y Robustas, dentro de la Robustas se encuentra el Proceso Unificado de Desarrollo de Software (RUP por sus siglas en inglés Rational Unified Process) esta metodología cuenta con una infraestructura flexible y adaptable al contexto y necesidades del proyecto.

RUP es una metodología robusta que cuenta ya con una vasta experiencia. Dentro de la ingeniería de software, es un proceso planteado por Kruchten en 1996, cuyo objetivo es producir software de alta calidad, es decir, que cumpla con los requerimientos de los usuarios dentro de la planificación y presupuesto establecido.

[9] **CASE**: (Computer Aided Software Engineering): Ingeniería de Software Asistida por Computadoras.
[10] **PDF:** formato de documento portátil.

RUP, es un modelo de software que permite el desarrollo de software a gran escala, mediante un proceso continuo de pruebas y retroalimentación, garantizando el cumplimiento de ciertos estándares de calidad. Aunque con el inconveniente de generar mayor complejidad en los controles de administración del mismo. Sin embargo, los beneficios obtenidos recompensan el esfuerzo invertido en este aspecto (18).

El proyecto Video Vigilancia utiliza la metodología antes mencionada. Esta sin dudas, se adapta a las condiciones del mismo y ofrece una mayor organización en el desarrollo de la Estación de Monitorización. Además, puede ser utilizada por proyectos que sean de gran envergadura, pues presenta una alta flexibilidad, lo que permite que se puedan realizar diferentes cambios en distintos ambientes de desarrollo. También es una metodología para entornos de desarrollo orientado a objetos. Se tiene en cuenta también la selección de esta metodología, pues los integrantes del proyecto tienen al menos un conocimiento básico sobre el desarrollo de aplicaciones con la misma, no siendo así con las otras existentes. Esto proporciona una ventaja respecto al tiempo de desarrollo del sistema.

1.6.6 Fundamentación de las herramientas seleccionadas.

El framework Qt ofrece una plataforma para el desarrollo de la comunicación entre los componentes que integran la Estación de Monitorización. Este framework es multiplataforma y se acopla con los componentes que se encuentran desarrollados. C++ se escoge por ser un lenguaje potente, multiplataforma y libre, brinda facilidades a la hora de realizar una programación a bajo nivel, este lenguaje posee un tratamiento de memoria que permite que el tiempo de ejecución sea menor que los programas escritos en Java o C#, lo que es conveniente para establecer una comunicación eficiente entre cada cámara y la Estación de Monitorización. Como IDE se elige Qt Creator, por ser multiplataforma y tener un amplio soporte y documentación. Cuenta con un potente editor de código C++, lenguaje escogido para el desarrollo, y es por excelencia el IDE que se utiliza para el trabajo con el framework Qt. Esto crea una perfecta unión entre lenguaje, framework e IDE. Además permite la integración de manera sencilla con la actual versión de la Estación de Monitorización, pues la misma se encuentra desarrollada sobre estas mismas tecnologías. Como herramienta case se selecciona Visual Paradigm, por ser multiplataforma y permitir el ciclo de vida completo del desarrollo de software, unido al lenguaje de modelado UML. Como guía del proceso de desarrollo se elige la metodología RUP.

1.6 Conclusiones.

Durante la elaboración de este capítulo se logró dar cumplimiento a las tareas de la investigación propuestas para el mismo, logrando obtener una descripción detallada sobre la evolución que han

tenido los sistemas de video vigilancia, y las principales compañías dedicadas a la creación de estos sistemas. Además se definieron las herramientas y tecnologías a utilizar para el desarrollo del software, lo cual permitió seleccionar en base a sus potencialidades, el lenguaje de programación C++ unido al framework QT ya que guarda una estrecha relación con el mismo por ser su lenguaje nativo, como IDE de desarrollo QT Creator por ser la herramienta por defecto del framework seleccionado y ser las mismas que se seleccionaron en la versión anterior lo cual permite un fácil acoplamiento a la hora de agregarle las nuevas funcionalidades al visor. Para guiar el desarrollo del software se utilizará la metodología RUP y como herramienta CASE Visual Paradigm lo que permitirá obtener un software que cumpla con la calidad y eficiencia que este requiere.

CAPÍTULO II: DISEÑO E IMPLEMENTACIÓN.

2.1 Introducción.

En el presente capítulo se realizará el diseño e implementación de la segunda versión de la Estación de Monitorización, siguiendo el flujo de trabajo de la metodología RUP. Se describen los requisitos funcionales y los casos de uso y actores del sistema. También, se refinan los diagramas de clases del diseño incluyendo las nuevas funcionalidades que se van a implementar, de manera que se muestra la utilización de los patrones de diseño y el patrón arquitectónico seleccionado. Además se presenta el diagrama de despliegue y componentes pertenecientes a la aplicación.

2.2 Requisitos Funcionales del Sistema.

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir (19). A continuación se describen los requisitos funcionales de la versión 2.0 de la Estación de Monitorización, los mismos guiarán la implementación de la aplicación y permitirán obtener un software que brinde mayores prestaciones en cuanto a la visualización de los flujos de video que se obtienen a través de las cámaras y a la gestión de los usuarios, roles y permisos.
*Nota: Se usa el prefijo **RF** en su nomenclatura.*

RF 1 Gestionar Usuario.
- **RF 1.1 Adicionar Usuario.**
- **RF 1.2 Eliminar Usuario.**
- **RF 1.3 Modificar Usuario.**

Descripción: El sistema debe permitir adicionar, eliminar o modificar un usuario según las necesidades de este.

RF 2 Gestionar Roles.
- **RF 2.1 Adicionar Rol.**
- **RF 2.2 Eliminar Rol.**
- **RF 2.3 Modificar Rol.**

Descripción: El sistema debe permitir que el usuario adicione, elimine o modifique un determinado rol, según este lo desee.

RF 3 Generar Reporte.

Descripción: El sistema debe permitir que el usuario visualice los reportes que este desee.

RF 4 Generar Alarma Visual.

Descripción: El sistema debe permitir mostrar al usuario una señal visual para indicarle que existe problema con una determinada cámara.

RF 5 Generar Alarma Sonora.

Descripción: El sistema debe permitir reproducir un sonido para indicarle al usuario que existe problema en una cámara.

RF 6 Configurar Permisos.

Descripción: El sistema debe permitir que el administrador configure los permisos de acceso de cada uno de los usuarios.

RF 7 Chequear Cámaras Offline.

Descripción: El sistema debe permitir que se chequen las cámaras offline[11] que se encuentren en ese momento.

RF 8 Generar Alarma de correo.

Descripción: El sistema debe permitir que se envíe al administrador una alarma mediante un correo electrónico para indicar que existe problema en una cámara.

2.3 Definición de los casos de uso del sistema.

En este epígrafe se describen los actores que interactúan con la aplicación. Los mismos representan a terceros fuera del sistema que colaboran con el mismo (19). Es decir, pueden ser personas u otro tipo de sistema que interactúe con la aplicación y pueden tener diferentes roles en dependencia del papel que ocupen en la misma.

[11] **Offline**: SE refiere a las cámaras que están apagadas o fuera de servicio.

2.3.1. Definición de los actores del sistema.

Tabla 1. Definición de los actores del sistema.

Actor	Descripción
Administrador	Rol responsable de velar por el funcionamiento apropiado del sistema, configurarlo y realizar todas las demás funcionalidades que permite el sistema.
Operador	Rol que se beneficia de diversas funcionalidades del sistema pero no tiene acceso a las configuraciones del mismo.
Estación de Monitorización	Rol que representa la Estación de Monitorización que controla y visualiza el flujo de video obtenidos de las cámaras IP.
Orquestador	Rol que representa el orquestador del sistema, controlador del tráfico de información entre el resto de los agentes autónomos del mismo.

2.3.2. Listado de los casos de uso del sistema.

Luego de identificar los requisitos funcionales que constituyen las funcionalidades finales con que contará el software, se procede a definir los casos de uso del sistema según dichos requisitos. Estos permiten guiar el proceso de desarrollo en su totalidad, ya que son la entrada para realizar el diseño, implementación y la validación del sistema (19). En el presente epígrafe se listan los casos de uso definidos para la implementación de la versión 2.0 de la Estación de Monitorización, resultado de agrupar los requisitos funcionales antes identificados.

CU1-Gestionar Usuario.
CU2-Gestionar Roles.
CU3-Generar Reporte.
CU4-Generar Alarma Visual.
CU5-Generar Alarma Sonora.
CU6-Configurar Permisos.

CU7-Chequear Cámaras Offline.

CU8-Generar Alarma de Correo.

2.3.3. Diagrama de Casos de Uso del sistema.

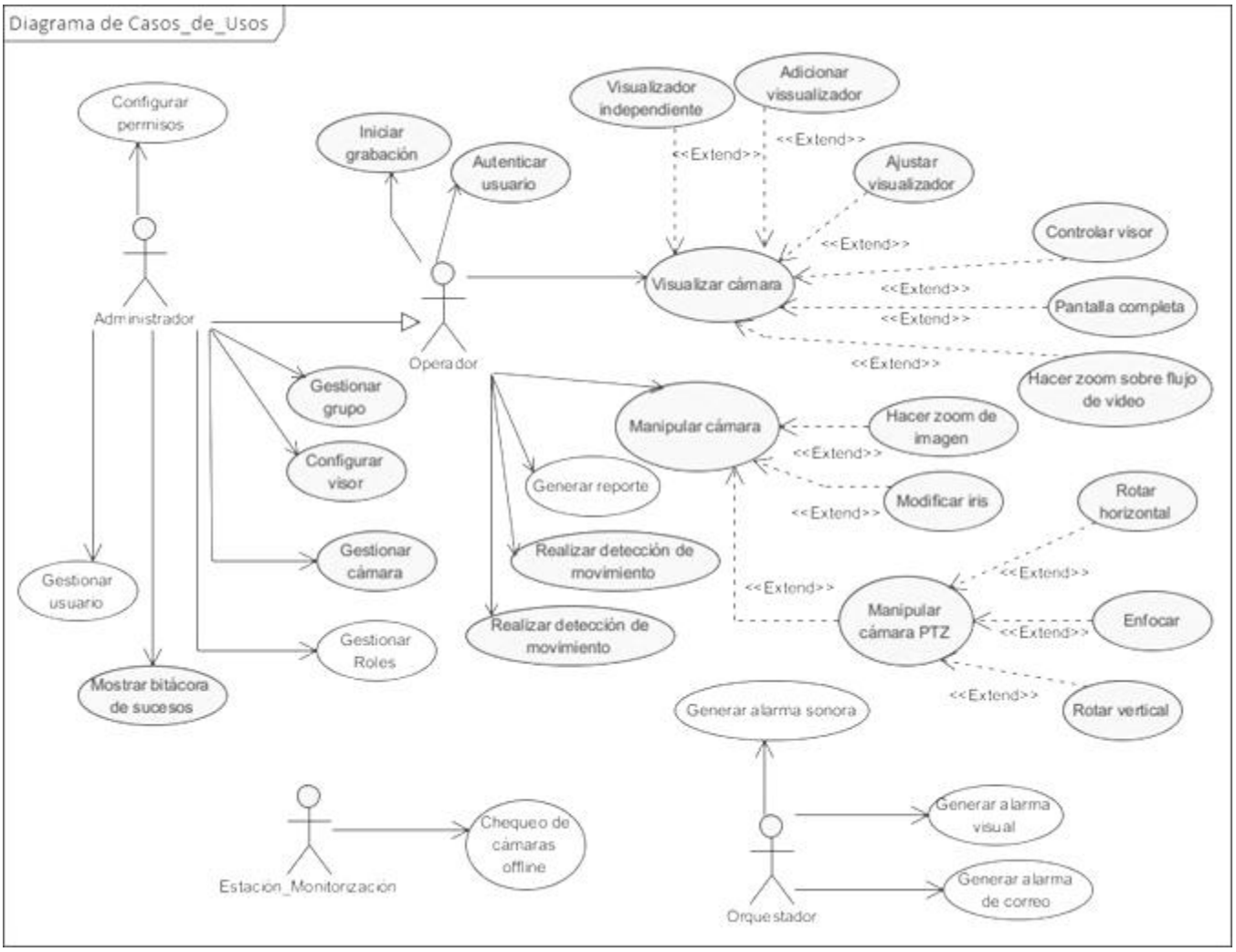


Figura 5. Diagrama de Casos de Uso del sistema.

Los casos de uso que se implementarán se muestran con el color verde. Los mismos se encuentran descritos desde los flujos de trabajo de Requisitos y Análisis que se realizaron en el proyecto Video Vigilancia. La funcionalidad correspondiente al CUS Generar alarma de correo ha surgido al refinar el sistema y su descripción puede consultarse en el anexo 1.

2.4 Arquitectura del Módulo en QT.

Luego de finalizar la fase de requisitos, se hizo necesaria la elección de un patrón arquitectónico que rigiera o modelara la estructura del sistema. El patrón arquitectónico que se utiliza para la construcción de la aplicación es el Modelo-Vista-Controlador (MVC), 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. Se separan los módulos en 3 componentes fundamentales, el Modelo, la Vista y el Controlador como se muestra en la *Figura 6*. 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 la Estación de Monitorización es capaz de utilizar diferentes interfaces de usuario manteniendo la misma lógica de negocio (20).

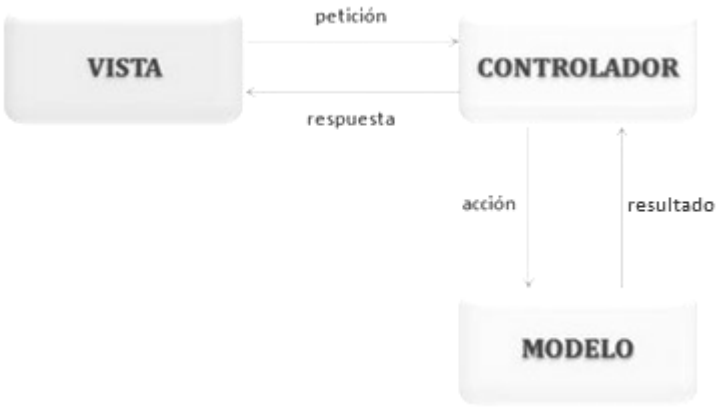


Figura 6. Patrón Modelo- Vista- Controlador (MCV).

2.5 Diagramas de Clases del Diseño.

A partir del patrón arquitectónico definido anteriormente se refinaron los diagramas de clases ya diseñados en la versión anterior de la Estación de Monitorización. Un diagrama de clases es un diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Los diagramas de clases son utilizados durante el proceso de análisis y diseño para lograr una mejor interpretación de la implementación del sistema (19). Los diagramas que se muestran a continuación pertenecen a los casos de uso que se van a implementar, estos han sido refinados a partir de las nuevas funcionalidades que se le agregarán a la Estación de Monitorización.

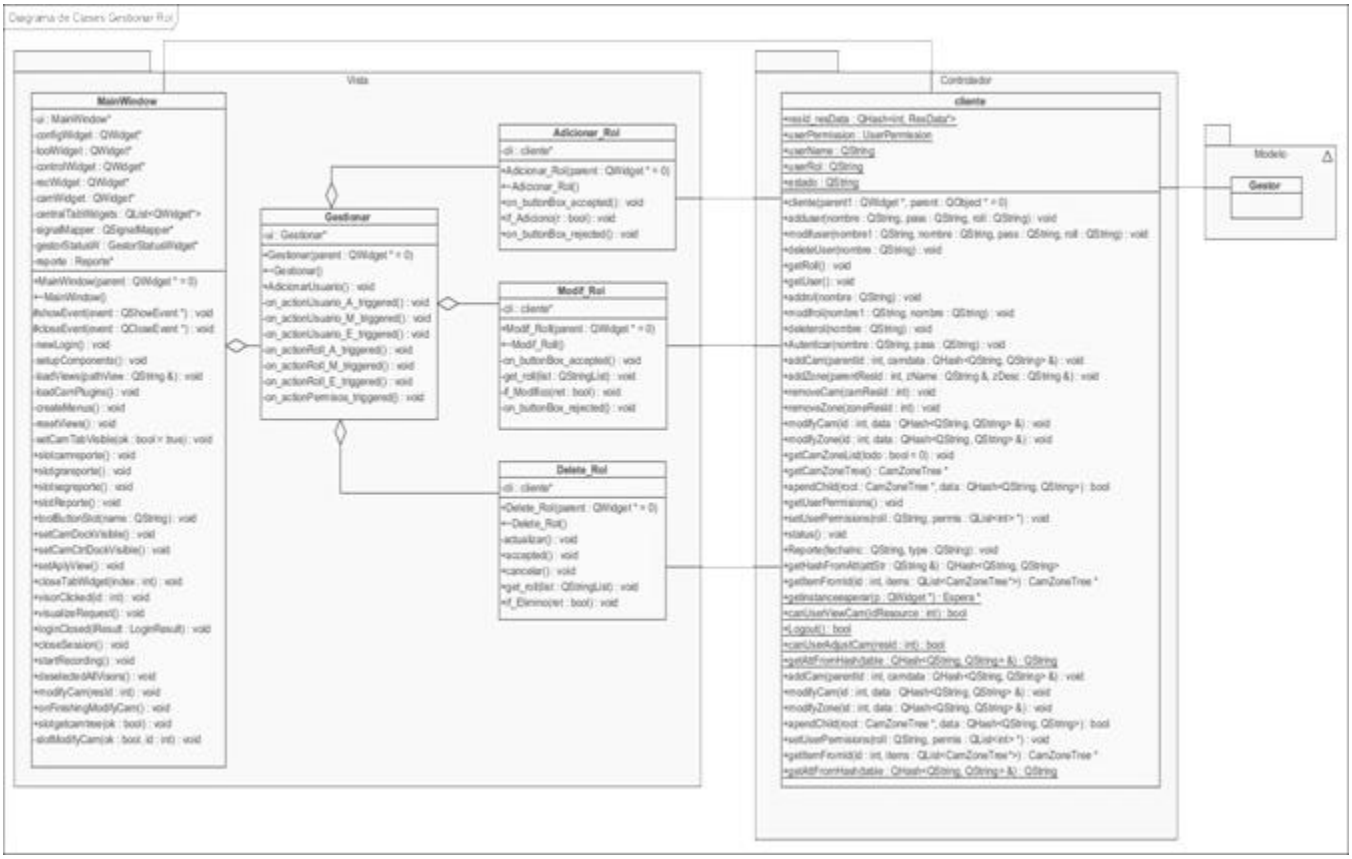


Figura 7. Diagrama de Clases del Diseño. CU Gestionar Rol.

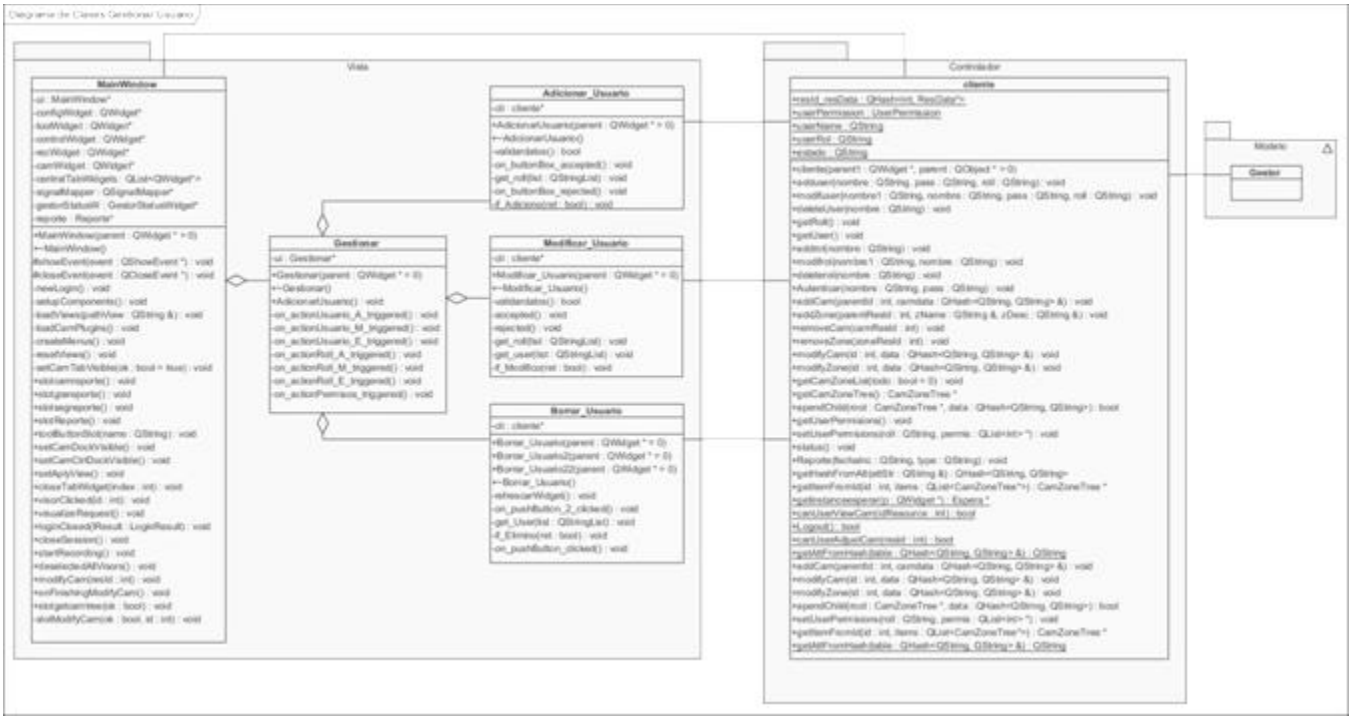


Figura 8. Diagrama de Clases del Diseño. CU Gestionar Usuario.

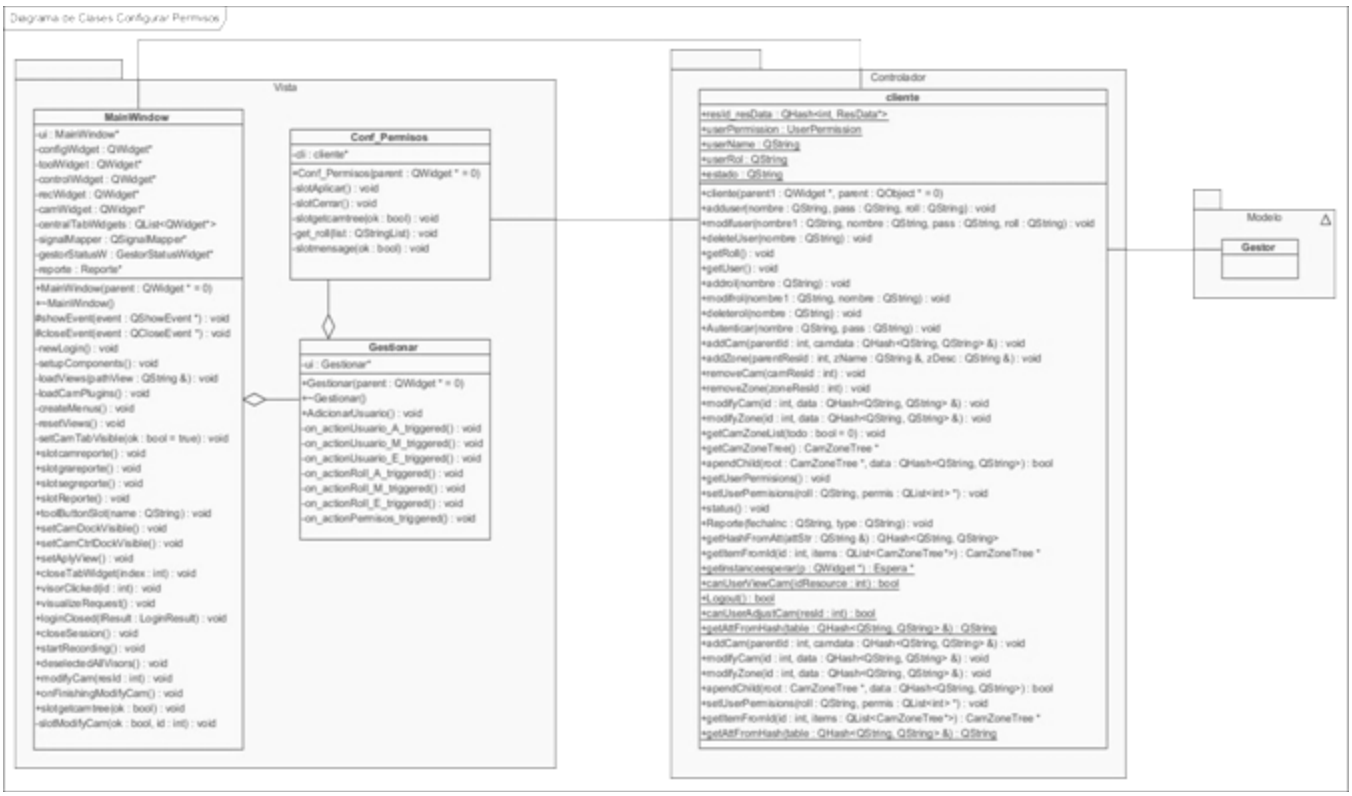


Figura 9. Diagrama de Clases del Diseño. CU Configurar Permisos.

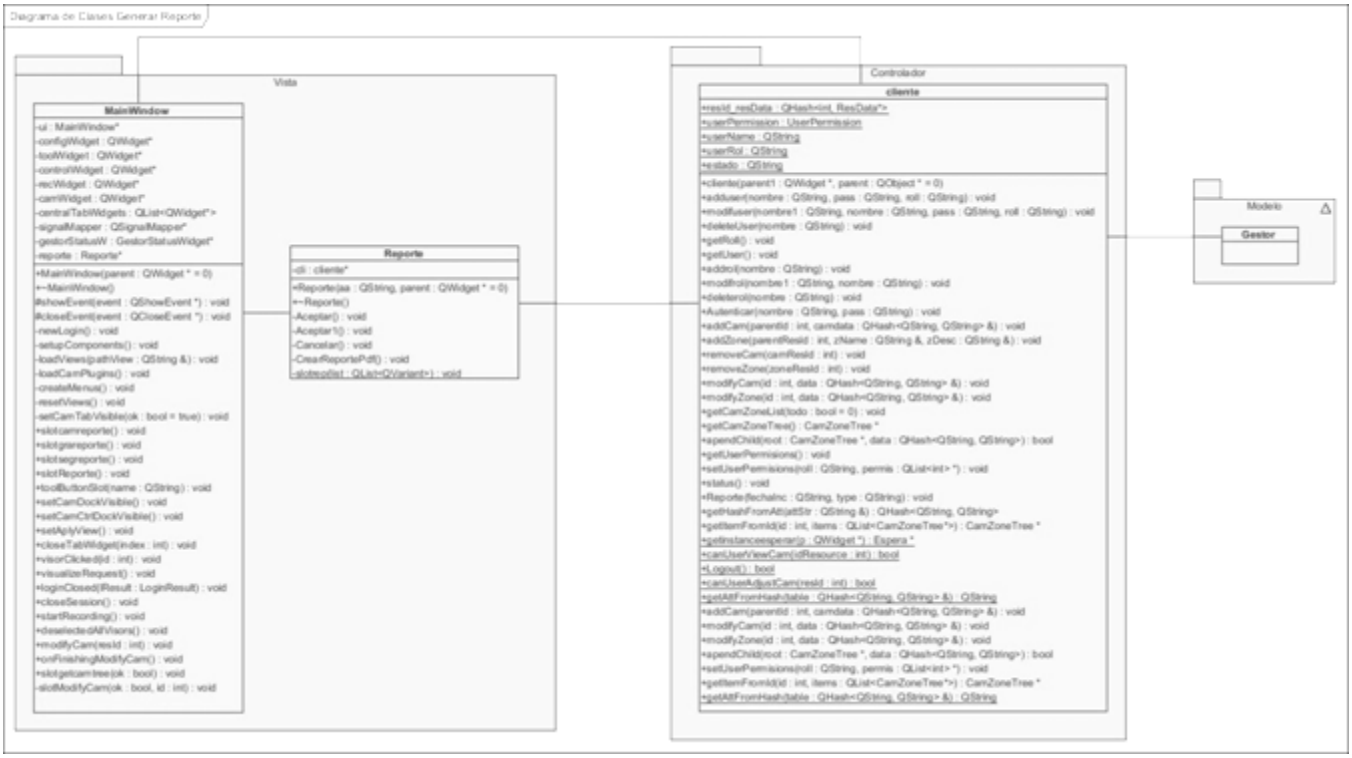


Figura 10. Diagrama de Clases del Diseño. CU Configurar Permisos.

2.6 Patrones de Diseño de Software.

Además de la arquitectura existen otros aspectos a tener en cuenta en la fase de diseño para construir una solución efectiva. Entre estos se encuentran la elección de patrones de diseño que constituyen la base para la solución de problemas que surgen en el desarrollo de una aplicación. Uno de ellos son los patrones de diseño GRASP[12] que describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. De estos se emplearán cinco de ellos, entre los que se encuentran:

Experto: propone la asignación de responsabilidades específicas a las clases creadas; es un principio básico que suele utilizarse en el diseño orientado a objetos. Expresa simplemente la intuición de que cada clase contiene los métodos relacionados con la información que posee. La utilización de este patrón se ve en la clase *Conf_Permisos*, ya que esta se encarga de realizar todas las operaciones que tienen que ver con los permisos.

[12] **GRASP** (**G**eneral **R**esponsibility **A**ssignment **S**oftware **P**atterns): Patrones generales para asignar responsabilidades.

Creador: se utiliza este patrón debido a las propiedades de todo sistema orientado a objetos. Se aplica en todos los casos donde una clase tiene la responsabilidad de crear una nueva instancia de la otra. Un ejemplo donde se utiliza este patrón es en las clases *MainWindows* y *Gestionar*, pues contienen objetos de las clases con las que se comunican.

Bajo Acoplamiento: propone la independencia de las clases del diseño con el fin de reducir el impacto de los cambios y permitir una mayor reutilización del código. Asigna responsabilidades de forma tal que las clases del diseño sean más independientes y que la comunicación sea la menor posible. Este patrón se emplea en todas las clases, ya que están relacionadas de manera tal que se crean sólo las dependencias necesarias para desempeñar sus responsabilidades.

Alta cohesión: propone cuán relacionadas y orientadas están las responsabilidades de una clase. Planteando la contribución entre clases para realizar tareas de elevada complejidad. Soporta un aumento de la capacidad de reutilización, siendo una gran funcionalidad, ya que una clase muy cohesiva puede destinarse a un propósito muy específico. Se utiliza en las clases *Adicionar_Usuario* y *Modificar_Usuario*.

Controlador: define quién deberá encargase de atender un evento del sistema. El controlador es un intermediario entre la interfaz de usuario y el núcleo de las clases donde se encuentra la lógica del sistema, se emplea en la clase *Cliente*, que se encarga de controlar todas las acciones que se realizan en la aplicación (21).

2.7 Modelo de Implementación.

El modelo de implementación describe cómo los elementos del modelo de diseño, como las clases, los ficheros de código fuente y ejecutables se implementan en términos de componentes. Describe también cómo se organizan los componentes de acuerdo a los mecanismos de estructuración y modularización disponibles en el entorno de la implementación (19).

2.7.1. Diagrama de Componentes.

Un diagrama de componentes representa cómo un sistema de software es dividido en componentes y muestra las dependencias entre estos componentes. Los componentes físicos incluyen archivos, cabeceras, bibliotecas compartidas, módulos, ejecutables, o paquetes. Desde el punto de vista del diagrama de componentes se tienen en consideración los requisitos relacionados con la facilidad de

desarrollo, la gestión de software, la reutilización, y las restricciones impuestas por los lenguajes de programación (19). A continuación se muestra la figura correspondiente al diagrama de componentes de la aplicación.

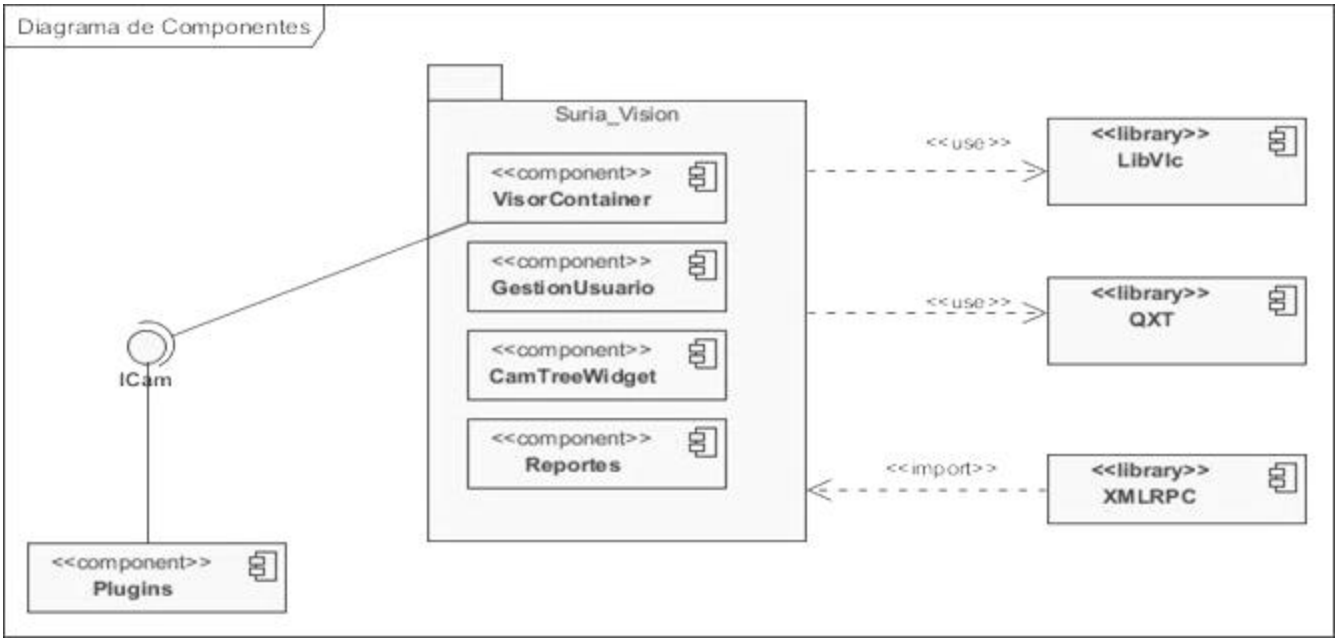


Figura 11. Diagrama de Componentes.

2.7.2. Diagrama de Despliegue.

El diagrama de despliegue muestra la disposición física de los distintos nodos[13] que componen un sistema y el reparto de los componentes sobre dichos nodos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación (19). La figura que se muestra a continuación representa el diagrama de despliegue de la aplicación.

[13]**Nodos:** Un nodo es un recurso de ejecución tal como un computador, un dispositivo o memoria.

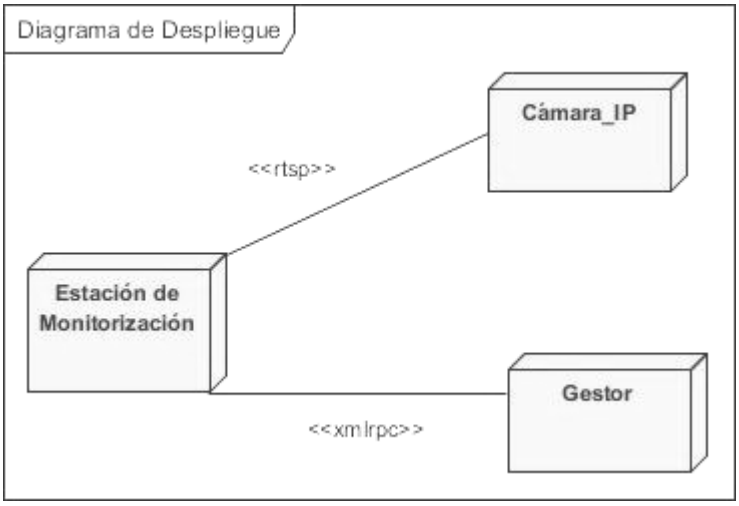


Figura 12. Diagrama de Despliegue.

2.8 Conclusiones.

Durante la realización de este capítulo se le dio cumplimiento a las tareas de la investigación propuestas para el mismo. Se obtuvo como resultado los requisitos funcionales de la aplicación y el diagrama de casos de uso del sistema, mostrando los casos de uso que se van a implementar. Además se refinaron los diagramas de clases del diseño, incorporándole las nuevas funcionalidades de la Estación de Monitorización, lo que permitió mostrar las clases en las cuales se emplean los patrones de diseño y el patrón arquitectónico utilizado. Se mostró también la aplicación en términos de componentes, su ubicación y utilización a la hora de la implementación final.

CAPÍTULO III: PRUEBAS AL SISTEMA.

3.1 Introducción.

En el presente capítulo se realizarán las pruebas al sistema. Lo que permitirá comprobar y validar el software desarrollado, verificando que las nuevas funcionalidades que se agregaron a la Estación de Monitorización realmente cumplen con las características establecidas y con las descripciones de casos de uso anteriormente expuestos.

3.2 Pruebas del Sistema.

Las pruebas se le realizan a un software con la finalidad de descubrir errores, tanto en la lógica interna como en funciones externas del mismo. Las pruebas son un elemento crítico para garantizar la calidad del software, estas no mejoran ni aseguran la calidad del software, pero sí lo hace indirectamente previendo un grupo de debilidades asociadas a la organización y sus riesgos. Existen varios tipos de prueba, unos de los frecuentemente utilizados son las Pruebas de Caja Blanca (CB), y las Pruebas de Caja Negra (CN) (22).

3.2.1. Pruebas de Caja Blanca.

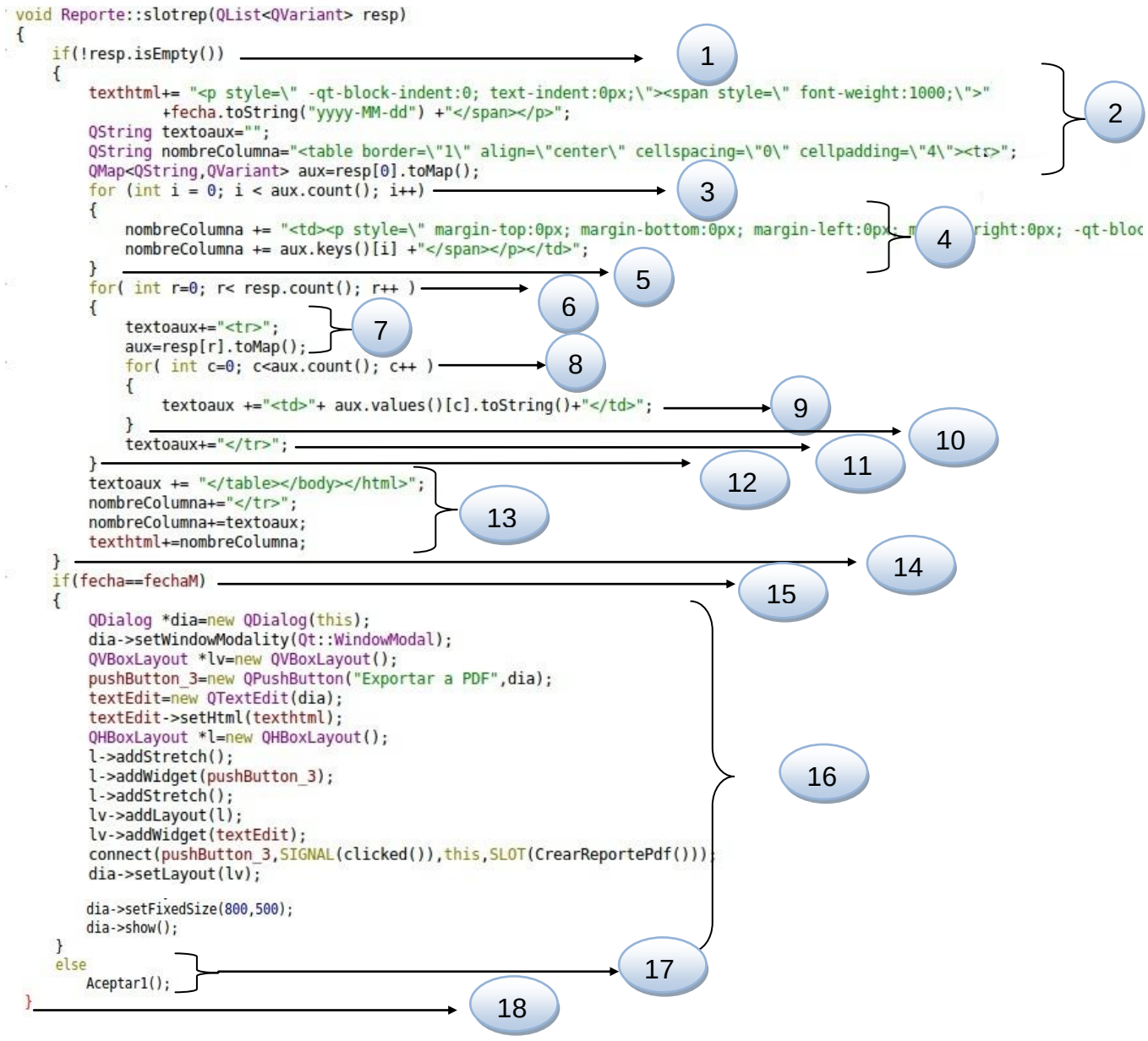
Técnicas de caja blanca o estructural: se basan en un minucioso examen de los detalles procedimentales del código a evaluar, por lo que es necesario conocer la lógica del programa. Es un método de diseño que usa la estructura de control descrita al nivel de componentes para derivar los casos de prueba (22). Se empleó la técnica de camino básico, la cual se describe a continuación.

Técnica de Camino Básico

Esta técnica permite tener una medida de la complejidad lógica de un diseño procedimental y así usar esta medida como guía para definir un conjunto básico de rutas de ejecución. Los casos de prueba derivados para ejercitar el conjunto básico deben garantizar que se ejecuta cada instrucción del programa por lo menos una vez durante la prueba. Los pasos a seguir para realizar esta prueba son:

1. Se realiza el grafo de flujo, a partir del código fuente a probar.
2. Se calcula la complejidad ciclomática del grafo.
3. Se determina un conjunto básico de caminos independientes.
4. Se preparan los casos de prueba que obliguen la ejecución de cada camino del conjunto básico (22).

En la figura que se muestra a continuación se representa el código del método *slotrep* que tiene la función de recibir una lista por parámetro para realizar los reportes.




```
void Reporte::slotrep(QList<QVariant> resp)
{
    if(!resp.isEmpty())
    {
        texthtml+= "<p style=\" -qt-block-indent:0; text-indent:0px;\"><span style=\" font-weight:1000;\">"
                +fecha.toString("yyyy-MM-dd") +"</span></p>";
        QString textoaux="";
        QString nombreColumna="<table border=\"1\" align=\"center\" cellspacing=\"0\" cellpadding=\"4\"><tr>";
        QMap<QString,QVariant> aux=resp[0].toMap();
        for (int i = 0; i < aux.count(); i++)
        {
            nombreColumna += "<td><p style=\" margin-top:0px; margin-bottom:0px; margin-left:0px; m
            nombreColumna += aux.keys()[i] +"</span></p></td>";
        }
        for( int r=0; r< resp.count(); r++ )
        {
            textoaux+="<tr>";
            aux=resp[r].toMap();
            for( int c=0; c<aux.count(); c++ )
            {
                textoaux +="<td>"+ aux.values()[c].toString()+"</td>";
            }
            textoaux+="</tr>";
        }
        textoaux += "</table></body></html>";
        nombreColumna+="</tr>";
        nombreColumna+=textoaux;
        texthtml+=nombreColumna;
    }
    if(fecha==fechaM)
    {
        QDialog *dia=new QDialog(this);
        dia->setWindowModality(Qt::WindowModal);
        QVBoxLayout *lv=new QVBoxLayout();
        pushButton_3=new QPushButton("Exportar a PDF",dia);
        textEdit=new QTextEdit(dia);
        textEdit->setHtml(texthtml);
        QHBoxLayout *l=new QHBoxLayout();
        l->addStretch();
        l->addWidget(pushButton_3);
        l->addStretch();
        lv->addLayout(l);
        lv->addWidget(textEdit);
        connect(pushButton_3,SIGNAL(clicked()),this,SLOT(CrearReportePdf()));
        dia->setLayout(lv);

        dia->setFixedSize(800,500);
        dia->show();
    }
    else
        Aceptar1();
}
```

Figura 13. Código correspondiente al método *slotrep*.

Grafo de Flujo

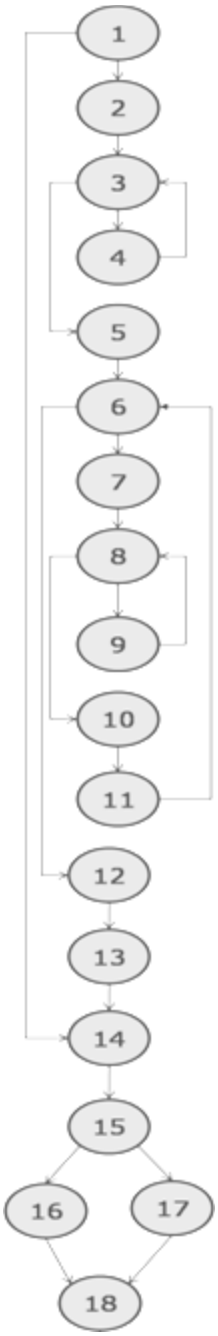


Figura 14. Grafo de flujo asociado al método *slotrep*.

Complejidad ciclomática.

La complejidad ciclomática es una de las métricas de software que proporciona una medida cuantitativa de la complejidad lógica de un programa. El valor calculado mediante la complejidad ciclomática define el número de rutas independientes en el conjunto básico de un programa, y

proporciona un número superior para el número de pruebas que deben aplicarse para asegurar que todas las instrucciones se hayan ejecutado por lo menos una vez (22).

Complejidad ciclomática [V (G)] = Cantidad de Aristas [A] – Cantidad de nodos [N] + 2.

V (G) = A – N + 2

V (G) = 22 – 18 + 2

V (G) = 6

La complejidad ciclomática dio como resultado seis, lo que significa que existen seis posibles caminos por donde el flujo puede circular, lo que indica que el número total de casos de prueba a realizar. En la siguiente tabla se muestran los caminos escogidos para realizar los casos de prueba.

Tabla 2. Caminos básicos.

Números	**Caminos básicos**
1	1-2-3-5-6-12-13-14-15-16-18
2	1-2-3-5-6-12-13-14-15-17-18
3	1-2-3-5-6-7-8-10-11-6-12-13-14-15-17-18
4	1-2-3-5-6-7-8-9-8-10-11-6-12-13-14-15-17-18
5	1-2-3-4-3-5-6-12-13-14-15-17-18
6	1-14-15-17-18

Luego de tener elaborado el grafo de flujos y los caminos a recorrer, se preparan los casos de prueba que forzarán la ejecución de cada uno de esos caminos.

Tabla 3. Caso de Prueba. Camino básico 1.

Caso de Prueba para el Camino básico 1	
Caminos	1-2-3-5-6-12-13-14-15-16-18
Entrada	Una lista de *QVariant* no vacía, *fecha* es igual a *fecham*
Resultado de la prueba	Satisfactorio

Tabla 4. Caso de Prueba. Camino básico 2.

Caso de Prueba para el Camino básico 2	
Caminos	1-2-3-5-6-12-13-14-15-17-18
Entrada	Una lista de *QVariant* no vacía, *fecha* es distinto de *fecham*
Resultado de la prueba	Satisfactorio

Tabla 5. Caso de Prueba. Camino básico 3.

Caso de Prueba para el Camino básico 3	
Caminos	1-2-3-5-6-7-8-10-11-6-12-13-14-15-17-18
Entrada	Una lista de *QVariant* no vacía, *fecha* es distinto de *fecham*
Resultado de la prueba	Satisfactorio

Tabla 6. Caso de Prueba. Camino básico 4.

Caso de Prueba para el Camino básico 4	
Caminos	1-2-3-5-6-7-8-9-8-10-11-6-12-13-14-15-17-18
Entrada	Una lista de *QVariant* no vacía, *fecha* es distinto de *fecham*
Resultado de la prueba	Satisfactorio

Tabla 7. Caso de Prueba. Camino básico 5.

Caso de Prueba para el Camino básico 5	
Caminos	1-2-3-4-3-5-6-12-13-14-15-17-18
Entrada	Una lista de *QVariant* no vacía, *fecha* es distinto de *fecham*
Resultado de la prueba	Satisfactorio

Tabla 8. Caso de Prueba. Camino básico 6.

Caso de Prueba para el Camino básico 6	
Caminos	1-14-15-17-18
Entrada	Una lista de *QVariant* vacía, *fecha* es distinto de *fecham*
Resultado de la prueba	Satisfactorio

3.2.2. Pruebas de Caja Negra

Técnicas de caja negra o funcional: realizan pruebas sobre la interfaz del programa a probar. Se define como interfaz a las entradas y salidas de dicho programa. No es necesario conocer la lógica del programa, únicamente la funcionalidad que debe realizar (22). Se empleó la técnica de partición equivalente, la cual se describe a continuación.

Técnica de Partición Equivalente

Esta técnica divide el dominio de entrada en clases de datos a partir de las cuales pueden derivarse casos de prueba. La partición equivalente se esfuerza por definir un caso de prueba que descubra

varias clases de errores, reduciendo así el número de casos de prueba que deben desarrollarse. Se definen un conjunto de estados válidos y no válidos para las condiciones de entrada y se verifica que los datos de salida sean los esperados (22).

Tabla 9. Diseño de caso de prueba. Caso de uso Gestionar Usuario.

Caso de Prueba "Caso de Uso Gestionar Usuario".					
Nombre de la sección	**Escenarios de la sección**	**Descripción de la funcionalidad**	**Flujo Central**	**Resultados Esperados**	**Resultados Obtenidos**
SC 1 *A*dicionar usuario.	EC 1.1: Adicionar usuario correctamente.	El sistema muestra un formulario para entrar los datos del usuario ("Usuario", "contraseña", "confirmar contraseña", y seleccionar el "rol"). Se selecciona la opción Aceptar. El sistema almacena los datos del usuario.	El flujo central debería ser: Módulo Visor/ Clic en ficha "Configuración"/ botón "Administrar Usuarios"/ Menú/Adicionar/Usuario /Aceptar.	El sistema debe adicionar el usuario en la base de datos.	El sistema adicionó el usuario correctamente.
	EC 1.2: Adicionar usuario. Campos vacíos.	En caso de entrar los datos incorrectos, el sistema muestra un mensaje indicando que los datos de entrada no son válidos.	Módulo Visor/ Clic en ficha "Configuración"/ botón "Administrar Usuarios"/ Menú/Adicionar/Usuario /Aceptar.	El sistema no debe permitir adicionar un nuevo usuario.	El sistema no adiciona el usuario.

	EC 1.3: Adicionar usuario. Usuario existente.	En el caso de que al entrar los datos, el usuario exista en la base de datos, se muestra un mensaje indicando que el usuario ya existe en el sistema.	Módulo Visor/ Clic en ficha "Configuración "/ botón "Administrar Usuarios"/ Menú/Adicion ar/Usuario /Aceptar.	El sistema debe verificar los datos entrados e informar en caso de que exista el usuario a través de un mensaje."Verif ique los datos"	El sistema verifica los datos entrados y envía un mensaje indicando que el usuario ya existe y no lo adiciona.
	EC 1.4: Adicionar usuario. Cancelar.	El sistema cancela la operación y vuelve a la interfaz anterior.	Módulo Visor/ Clic en ficha "Configuración"/ Clic en Administrar usuario/ Menú/Adicionar/ Usuario /Cancelar.	El sistema debe cancelar la operación.	El sistema cancela la operación.
SC 2: Actualizar datos de usuario.	EC 2.1: Actualizar usuario correctamente.	El administrador al seleccionar un usuario, el sistema muestra un formulario con la información del usuario que desea modificar ("nuevo nombre", "contraseña", "confirmar contraseña" y "rol"). El administrador modifica datos del usuario y selecciona	Módulo Visor/ Clic en Administrar usuario/ Menú/Actualizar /Usuario /Aceptar.	El sistema debe modificar los datos del usuario en la base de datos.	El sistema modifica los datos del usuario.

		la opción “Aceptar”.			
	EC 2.2: Actualizar usuario. Campos vacios.	El sistema comprueba los datos y al encontrar errores muestra un mensaje indicando que los datos entrados son incorrectos.	Módulo Visor/ Clic en Administrar usuario/ Menú/Actualizar /Usuario /Aceptar.	El sistema debe mostrar un mensaje informando que los datos no han podido ser modificados porque son incorrectos.	El sistema muestra un mensaje informando que los datos son incorrectos.
	EC 2.3: Actualizar usuario. Cancelar	El sistema cancela la operación y vuelve a la interfaz anterior.	Módulo Visor/ Clic en Administrar usuario/ Menú/Actualizar /Usuario /Cancelar.	El sistema debe cancelar la operación.	El sistema cancela la operación
SC 3: Eliminar usuario	EC 3.1: Eliminar usuario correctamente.	El sistema elimina el usuario o los usuarios seleccionados de la Base de Datos.	Módulo Visor/ Clic en Administrar usuario/ Menú/Eliminar/Usuario /Borrar.	El sistema debe eliminar el usuario de la Base de datos.	El sistema elimina el usuario de la Base de datos.
	EC 3.2 Eliminar usuario. Cancelar.	El Administrador selecciona la opción “Cerrar”, el sistema cancela la operación y vuelve a la interfaz anterior.	Módulo Visor/ Clic en Administrar usuario/ Menú/Eliminar/Usuario	El sistema debe cancelar la operación.	El sistema cancela la operación.

			/ Cerrar.		

Tabla 10. Diseño de caso de prueba. Caso de uso Gestionar Rol.

Caso de Prueba "Caso de Uso Gestionar Rol".					
Nombre de la sección	**Escenarios de la sección**	**Descripción de la funcionalidad**	**Flujo Central**	**Resultados Esperados**	**Resultados Obtenidos**
SC 1 Adicionar rol	EC 1.1: Adicionar rol correctamente.	Añadir rol introduciendo el nombre del nuevo rol y pulsando el botón "Aceptar". El sistema adiciona el nuevo rol.	Módulo Visor / Administrar Usuario/Menú/Adicionar/Rol/Aceptar.	El sistema debe adicionar el nuevo rol en la base de datos.	El sistema adiciona el nuevo rol.
	EC 1.2: Adicionar rol. Campos vacíos.	Si no se introdujo ningún valor en el campo nombre del rol el sistema muestra un mensaje de error indicando que verifique los datos.	Módulo Visor / Administrar Usuario/Menú/Adicionar/Rol/Aceptar	El sistema debe mostrar un mensaje indicando que existen campos vacíos.	El sistema muestra un mensaje indicando que existen campos vacíos.
	EC 1.3: Adicionar rol. Rol. Rol existente.	Si existe un rol con el nombre introducido el sistema muestra un mensaje indicando que verifique los datos.	Módulo Visor / Administrar Usuario/Menú/Adicionar/Rol/Aceptar	El sistema debe mostrar un mensaje indicando que ya existe un rol con ese nombre.	El sistema muestra un mensaje indicando que ya existe un rol con ese nombre.
	EC 1.4: Adicionar rol. Cancelar.	El sistema cancela la operación y vuelve a la interfaz	Módulo Visor / Administrar Usuario/Menú/Adicion	El sistema debe cancelar la operación y	El sistema cancela la operación y

		anterior.	ar/Rol/Cancelar.	volver a la interfaz anterior.	vuelve a la interfaz anterior.
SC 2: Actualizar rol.	EC 2.1: Actualizar rol exitosamente.	Editar rol seleccionando la opción de actualizar rol, introduciendo el nombre del nuevo rol y pulsando el botón “Aceptar”. El sistema modifica el rol.	Módulo Visor / Administrar Usuario/Menú/Actualizar/Rol/Aceptar	El sistema debe guardar los cambios, y mostrar un mensaje indicando que la operación fue realizada satisfactoriamente.	El sistema guarda los cambios, y muestra un mensaje indicando que la operación fue realizada satisfactoriamente.
	EC 2.2: Actualizar. Cancelar	El Administrador selecciona la opción “Cancelar”, el sistema cancela la operación y vuelve a la interfaz anterior.	Módulo Visor / Administrar Usuario/Menú/Actualizar/Rol/Cancelar	El sistema debe cancelar la operación y volver a la interfaz anterior.	El sistema cancela la operación y vuelve a la interfaz anterior.
SC 3: Eliminar rol	EC 3.1: Eliminar rol exitosamente.	Eliminar rol, seleccionando la opción “Eliminar rol”, el sistema elimina el rol seleccionado.	Módulo Visor / Administrar Usuario/Menú/Eliminar/Rol/Borrar.	El sistema debe eliminar el rol seleccionado.	El sistema elimina el rol seleccionado.
	EC 3.2 Eliminar rol. Cancelar.	El Administrador selecciona la opción “Cancelar”, el sistema cancela la operación y vuelve a la interfaz anterior.	Módulo Visor / Administrar Usuario/Menú/Eliminar/Rol/Cerrar.	El sistema debe cancelar la operación y volver a la interfaz anterior.	El sistema cancela la operación y vuelve a la interfaz anterior.

SC4: Actualizar permisos de rol	EC 4.1: Actualizar permisos de rol exitosamente.	Actualizar permisos de rol, el Administrador selecciona o deselecciona los permisos que desee asignarle al rol seleccionado y presiona el botón “Aceptar”. El sistema guarda los cambios realizados.	Módulo Visor / Administrar Usuario/Menú/Actualizar/Rol/Aplicar.	El sistema debe guardar los cambios realizados, y mostrar un mensaje que indique que la operación fue realizada satisfactoriamente y regresa al formulario de gestión de roles.	El sistema guarda los cambios realizados, y muestra un mensaje que indique que la operación fue realizada satisfactoriamente y regresa al formulario de gestión de roles.
	EC 4.2: Actualizar permisos de rol existe. Cancelar	El Administrador selecciona la opción “Cancelar”, el sistema cancela la operación y vuelve a la interfaz anterior.	Módulo Visor / Administrar Usuario/Menú/Actualizar/Rol/Cerrar.	El sistema debe cancelar la operación y volver a la interfaz anterior y regresar al formulario de gestión de roles.	El sistema cancela la operación y vuelve a la interfaz anterior y regresa al formulario de gestión de roles.

3.2.3. Plan de pruebas de integración

Una vez comprobado que el módulo Estación de Monitorización funciona correctamente como una unidad y en correspondencia con las descripciones de los casos de uso, se hace necesario verificar que el mismo se siga comportando correctamente aún integrado con otros módulos. Para ello se propone un plan para la realización de pruebas de integración, con el objetivo de validar y evaluar el comportamiento del módulo funcionando en conjunto con otros.

Las pruebas de integración son una técnica sistemática para construir la estructura del programa mientras que, al mismo tiempo, se llevan a cabo pruebas para detectar errores asociados con la interacción. El objetivo es utilizar los módulos ya probados de manera independiente y construir una estructura de programa que esté de acuerdo con lo que dicta el diseño (22).

Para realizar el plan de integración del módulo se utilizará la integración ascendente, lo que significa que se comienza a integrar cada módulo desde el nivel más bajo. Para desarrollar el diagrama es necesario que las pruebas unitarias de cada uno de los módulos hayan sido superadas correctamente.

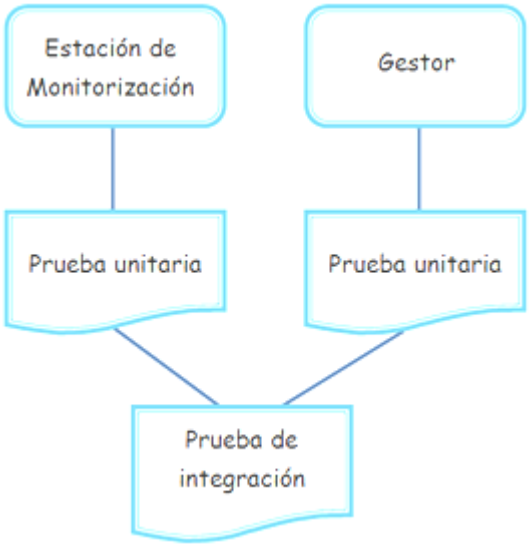


Figura 15. Diagrama de integración entre la Estación de Monitorización y el Gestor.

3.3 Conclusiones.

Durante la realización de este capítulo se logró dar cumplimiento a las tareas de la investigación correspondientes al mismo. Obteniéndose como resultado que las pruebas realizadas al software no arrojaron no conformidades, lo que permitió verificar el cumplimiento de las nuevas funcionalidades, demostrando así la validación y el correcto funcionamiento de la aplicación desarrollada. Además se realizó la propuesta de un plan de integración lo que permitirá validar el comportamiento del sistema al ser integrado con otros módulos.

CONCLUSIONES GENERALES

Con el presente trabajo de diploma se logró dar cumplimiento a las tareas de la investigación propuestas inicialmente, logrando obtener como resultado una segunda versión de la Estación de Monitorización para el proyecto Video Vigilancia Suria Vision. Lo cual permitió llegar a las siguientes conclusiones:

- ✓ Se realizó un análisis sobre los principales sistemas de video vigilancia existentes en la actualidad. Llegando a la conclusión que las principales limitaciones que estos sistemas presentan es que dependen completamente del fabricante.
- ✓ Las herramientas y tecnologías utilizadas permitieron crear una aplicación que cumple con todos los requisitos funcionales y no funcionales identificados, trayendo consigo la obtención de un software que cumple con la calidad que se esperaba.
- ✓ Se agregaron nuevas funcionalidades al módulo Estación de Monitorización las cuales permitieron convertirla en una herramienta de mayor potencia para tareas de video vigilancia.
- ✓ Se probaron las nuevas funcionalidades desarrolladas obteniendo resultados satisfactorios.

RECOMENDACIONES

Durante el desarrollo del presente trabajo surgieron varias ideas, por tanto se recomienda:

- ✓ Implementar nuevas funcionalidades que al agregarlas a la Estación de Monitorización, amplíe las prestaciones que brinda y posibilite el proceso de video vigilancia en las instituciones donde se requieran estos servicios.
- ✓ Integración con los videos sensores pertenecientes al módulo Suria Analytic, para ampliar las prestaciones de la Estación de Monitorización, facilitando la detección de movimiento y el seguimiento de objetos , para detectar la ocurrencia de actos delictivos o de algún evento en específico.

REFERENCIAS BIBLIOGRÁFICAS

1. A history of video surveillance in England. [En línea] [Citado el: 10 de Octubre de 2011.] http://www.notbored.org/england-history.html.

2. **Articles Free Blogs.** La Historia de Video Vigilancia. [En línea] [Citado el: 10 de Octubre de 2011.] http://articulossobredesarrolloweb.blogspot.com/2009/10/la-historia-de-video-vigilancia.html.

3. Real Academia Española. [En línea] [Citado el: 13 de noviembre de 2012.] www.rae.es.

4. **Kioskea .Net.** Kioskea .Net. [En línea] [Citado el: 16 de Septiembre de 2011.] http://es.kioskea.net/contents/internet/protip.php3.

5. **Carrasco, Ignacio.** Borrmart,SA. [En línea] 2005. [Citado el: 5 de Junio de 2012.] http://www.borrmart.es/articulo_seguritecnia.php?id=2275.

6. **Casadomo.** Casadomo. [En línea] [Citado el: 22 de Septiembre de 2011.] http://www.casadomo.com/noticiasDetalle.aspx.

7. **Axis.** La evolución de los sistemas de vigilancia por vídeo. [En línea] [Citado el: 20 de Noviembre de 2011.] http://www.axis.com/es/products/video/about_networkvideo/evolution.htm.

8. **Nilsson, Fredrik and Axis Communications.** The Evolution of Video Surveillance Systems. [En línea] 2009. [Citado el: 20 de Noviembre de 2011.] http://www.infosectoday.com/Articles/Video_Surveillance_Systems.htm.

9. **Axis Communications.** Axis Communications. [En línea] [Citado el: 20 de Noviembre de 2011.] www.axis.com.

10. **DATYS.** DATYS. [En línea] 2011. [Citado el: 25 de Noviembre de 2011.] http://www.datys.cu/WPInfNoticias.aspx?47,N.

11. **VIVOTEK.** VIVOTEK. [En línea] [Citado el: 25 de Noviembre de 2011.] http://www.vivotek.com/products/model.php?soft=vast.

12. **AguilarJoyanes, Luis.** *Fundamentos de programación.Algoritmos,estructuras.* España : s.n., 2003. 8448139860.

13. **Luján Mora, Sergio.** *C++ paso a paso.* Alicante : s.n., 2006. 84-7908-888-5.

14. **NOKIA.** QT Developer Network, NOKIA. [En línea] [Citado el: 5 de Diciembre de 2011.] http://developer.qt.nokia.com/wiki/Category:Tools::QtCreator_Spanish..

15. Programación en castellano. [En línea] 2011. [Citado el: 6 de Diciembre de 2012.] http://www.programacion.com/entornos. B13344544.

16. **Carrero, Angel.** QtCreator, un completo entorno de desarrollo. [En línea] 10 de junio de 2010. [Citado el: 6 de Diciembre de 2011.] http://www.programacion.com/noticia/qt_creator-un_completo_entorno_de_desarrollo_1723..

17. Visual Paradigm. [En línea] International, 2004. [Citado el: 6 de Diciembre de 2011.] http://www.visual-paradigm.com.

18. Desarrollando aplicaciones informáticas. [En línea] [Citado el: 20 de Noviembre de 2011.] http://www.utvm.edu.mx/OrganoInformativo/orgJul07/RUP.htm.

19. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El Proceso Unificado de Desarrollo de Software.* Madrid : Perason Educacion, 2000. 84-7829-063-2.

20. **Semanat Aldana, Edmis Deivis y Verdecia Fou, Leonor.** *Sistema de Video Vigilancia.* La Habana : s.n., 2009.

21. **Larman, Craig.** *UML y Patrones, Introducción al análisis y diseño orientado a objeto.* Mexico : Prentice Hall, 1999.

22. **Pressman, Roger S.** *Ingeniería del Software, Un enfoque práctico.* 2000.

BIBLIOGRAFÍA

1. A history of video surveillance in England. [En línea] [Citado el: Octubre 10, 2011.] http://www.notbored.org/england-history.html.

2. **Articles Free Blogs.** La Historia de Video Vigilancia. [En línea] [Citado el: Octubre 10, 2011.] http://articulossobredesarrolloweb.blogspot.com/2009/10/la-historia-de-video-vigilancia.html.

3. Real Academia Española. [En línea] [Citado el: noviembre 13, 2012.] www.rae.es.

4. **Kioskea .Net.** Kioskea .Net. [En línea] [Citado el: Septiembre 16, 2011.] http://es.kioskea.net/contents/internet/protip.php3.

5. **Carrasco, Ignacio.** Borrmart,SA. [En línea] 2005. [Citado el: Junio 5, 2012.] http://www.borrmart.es/articulo_seguritecnia.php?id=2275.

6. **Casadomo.** Casadomo. [En línea] [Citado el: Septiembre 22, 2011.] http://www.casadomo.com/noticiasDetalle.aspx.

7. **Axis.** La evolución de los sistemas de vigilancia por vídeo. [En línea] [Citado el: Noviembre 20, 2011.] http://www.axis.com/es/products/video/about_networkvideo/evolution.htm.

8. **Nilsson, Fredrik and Axis Communications.** The Evolution of Video Surveillance Systems. [En línea] 2009. [Citado el: Noviembre 20, 2011.] http://www.infosectoday.com/Articles/Video_Surveillance_Systems.htm.

9. **Axis Communications.** Axis Communications. [En línea] [Citado el: Noviembre 20, 2011.] www.axis.com.

10. **DATYS.** DATYS. [En línea] 2011. [Citado el: Noviembre 25, 2011.] http://www.datys.cu/WPInfNoticias.aspx?47,N.

11. **VIVOTEK.** VIVOTEK. [En línea] [Citado el: Noviembre 25, 2011.] http://www.vivotek.com/products/model.php?soft=vast.

12. **Aguilar Joyanes, Luis.** *Fundamentos de programación.Algoritmos,estructuras.* España : s.n., 2003. 8448139860.

13. **Luján Mora, Sergio.** *C++ paso a paso.* Alicante : s.n., 2006. 84-7908-888-5.

14. **NOKIA.** QT Developer Network, NOKIA. [En línea] [Citado el: Diciembre 5, 2011.] http://developer.qt.nokia.com/wiki/Category:Tools::QtCreator_Spanish..

15. Programación en castellano. [En línea] 2011. [Citado el: Diciembre 6, 2012.] http://www.programacion.com/entornos. B13344544.

16. **Carrero, Angel.** QtCreator, un completo entorno de desarrollo. [En línea] junio 10, 2010. [Citado el: Diciembre 6, 2011.] http://www.programacion.com/noticia/qt_creator-un_completo_entorno_de_desarrollo_1723..

17. Visual Paradigm. [En línea] International, 2004. [Citado el: Diciembre 6, 2011.] http://www.visual-paradigm.com.

18. Desarrollando aplicaciones informáticas. [En línea] [Citado el: Noviembre 20, 2011.] http://www.utvm.edu.mx/OrganoInformativo/orgJul07/RUP.htm.

19. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El Proceso Unificado de Desarrollo de Software.* Madrid : Perason Educacion, 2000. 84-7829-063-2.

20. **Semanat Aldana, Edmis Deivis y Verdecia Fou, Leonor.** *Sistema de Video Vigilancia.* La Habana : s.n., 2009.

21. **Larman, Craig.** *UML y Patrones, Introducción al análisis y diseño orientado a objeto.* Mexico : Prentice Hall, 1999.

22. **Pressman, Roger S.** *Ingeniería del Software, Un enfoque práctico.* 2000.

ANEXOS

Anexos 1. Descripción del caso de uso. Gestionar alarma de correo.

Caso de Uso:	Generar alarma de correo.
Actores:	Orquestador.
Resumen:	El caso de uso se inicia cuando el Orquestador detecta algún suceso en una cámara determinada. El caso de uso termina cuando se envía la alarma de correo.
Precondiciones:	La ocurrencia de un evento en una cámara.
Referencias	RF 8
Prioridad	Secundario

Flujo Normal de Eventos "Generar alarma de correo".	
Sección "Generar alarma de correo".	
Acción del Actor	**Respuesta del Sistema**
1. El caso de uso inicia cuando el orquestador detecta algún suceso en una cámara.	2. El Sistema envía una alarma de correo notificando el suceso ocurrido.

Poscondiciones:	Se envía una alarma de correo.

GLOSARIO DE TÉRMINOS

Biblioteca: Es un conjunto de subprogramas utilizados para desarrollar software.

Cámara IP: Es una cámara que emite su contenido directamente a la red.

Cámara Analógica: Es una cámara que emite su contenido mediante un cable coaxial.

IP: Protocolo de Internet (Internet Protocol), es un protocolo usado para la transmisión de datos a través de una red de paquetes conmutados no fiable de mejor entrega posible sin garantías.

Multicast: Es el envío de información a múltiples destinos simultáneamente.

Multiplexor: Dispositivo que puede recibir varias entradas y transmitirlas por un salida de transmisión compartida.

PTZ: Es un acrónimo de pan-tilt-zoom se refiere a las características de las cámaras (giro, inclinación, enfoque).

Plugin: Es un módulo de hardware o software que añade una característica o un servicio específico a un sistema más grande.

Puerto Ethernet: es una interfaz física comúnmente usada para conectar redes de cableado estructurado.

Unicast: Es el envío de información a un único destino.

Video Digital: Es un tipo de sistema de grabación de video que funciona usando una representación digital de la señal de video.

Video Analógico: Es un tipo de sistema de grabación de video que funciona usando una representación digital de la señal de video.