

# Universidad de las Ciencias Informáticas



**Título:** Componente de control de conexiones entrada/salida de las cámaras IP en el sistema SURIA Vision.

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

**Autora:** Eniley Albuerne Díaz

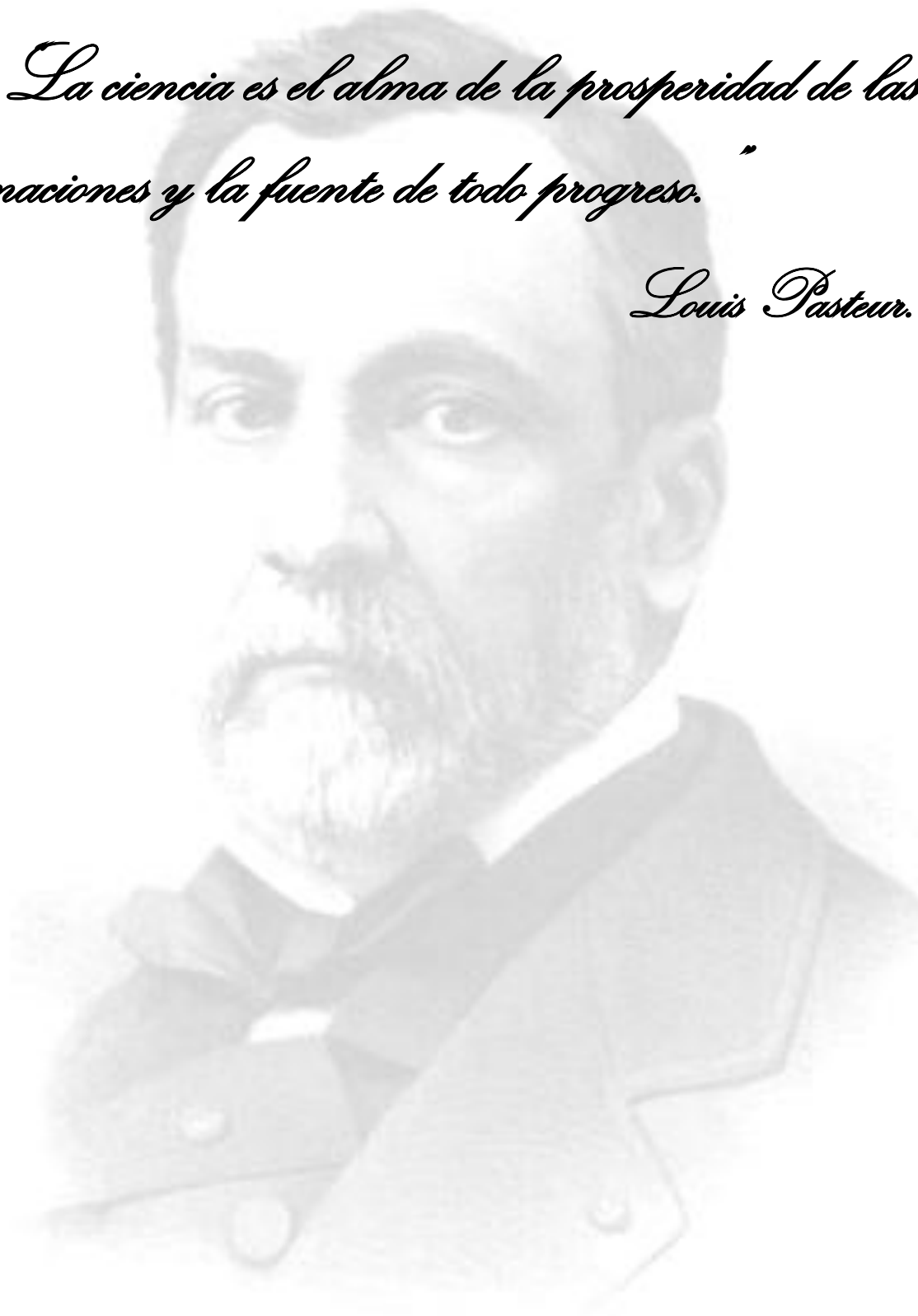
**Tutor:** Ing. Edmis Deivis Semanat Aldana

La Habana, 25 de junio del 2012

“Año 54 de la Revolución”

*La ciencia es el alma de la prosperidad de las naciones y la fuente de todo progreso. ”*

*Louis Pasteur.*



## **DECLARACIÓN DE AUTORÍA**

Declaro que soy la única autora de este trabajo y autorizo a la Facultad 6 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los \_\_\_\_ días del mes de \_\_\_\_\_ del año \_\_\_\_\_.

Eniley Albuerne Díaz

Ing. Edmis Devis Semanat Aldana

---

---

## DATOS DE CONTACTO

### ➤ Diplomante

**Nombre y apellidos:** Eniley Albuerne Díaz

**Sexo:** F    **Grupo:** 6505

**Correo electrónico:** [enalbuerne@estudiantes.uci.cu](mailto:enalbuerne@estudiantes.uci.cu)

### ➤ Tutor

**Nombre y apellidos:** Edmis Deivis Semanat Aldana

**Sexo:** M    **Institución:** Universidad de las Ciencias Informáticas (UCI)

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

**Reparto:** Torrens, **Municipio:** Boyeros, **Provincia:** Ciudad de La Habana.

**Teléfono del trabajo:** 07-837-3780    **Cargo del trabajador:** Profesor

**Título de la especialidad de graduado:** Ingeniero en Ciencias Informáticas

**Año de graduación:** 2009    **Correo electrónico:** [edsemanat@uci.cu](mailto:edsemanat@uci.cu)

**Institución donde se graduó:** UCI

### ➤ Oponente

**Nombre y apellidos:** Félix Iván Romero Rodríguez

**Sexo:** M    **Institución:** UCI    **Cargo del trabajador:** Recién Graduado  
en Adiestramiento

**Título de la especialidad de graduado:** Ingeniero en Ciencias Informáticas

**Año de graduación:** 2011    **Correo electrónico:** [firomero@uci.cu](mailto:firomero@uci.cu)

**Institución donde se graduó:** UCI

## Agradecimientos

---

*Gracias especialmente a mi madre, "mi vieja" como yo le digo, por guiarme siempre por el camino correcto, por su ejemplo intachable y su apoyo incondicional, por confiar en mí y darme fuerzas para continuar, aún cuando yo misma perdía la esperanza.*

*A mi abuelita Orte, por sus consejos, por todo el cariño que me ha dado y por hacerme sentir tan querida.*

*A mi padrastro Pedro, por ayudarme siempre en todo lo que ha estado a su alcance.*

*A Fher, por todo el cariño, por la paciencia, por su apoyo y ayuda incondicional para seguir adelante, por no permitirme renunciar bajo ningún concepto... gracias por hacer tuya también la responsabilidad de esta tesis.*

*A mis amigas, Tay, Emy, Day, Miry y Erlenys, por todo lo que hemos compartido juntas, el cariño, los momentos buenos y otros no tan buenos, las risas y el apoyo mutuo.*

*A todos y cada uno de los amigos que he hecho durante estos años de universidad, no mencionaré nombres, pues no me perdonaría dejar a nadie fuera, les estoy agradecida por el tiempo que hemos compartido y espero que la vida nos depare más encuentros.*

*A mi tutor Deivis por su ayuda, gracias por su guía y los consejos para terminar este trabajo.*

*A mi tribunal, por las recomendaciones tan oportunas, las críticas constructivas y por contar con su ayuda siempre que los necesité.*

*A todos los profesores que durante este tiempo me han ayudado a formarme no solo como profesional, sino como mejor persona.*

*Gracias a todos....*

*A mi mamá, por inspirarme con su ejemplo y esa fuerza de voluntad tan grande que la caracteriza.*

*A mi papá, aunque en ocasiones distante, me hizo sentir que estaba presente.*

*A mi abuelita Orte, tan especial para mí y que tanto la quiero.*

*A mis abuelos Ide y Juanita, quienes se sentirían orgullosos de verme graduada, aunque ya no se encuentren físicamente, siempre los tengo presentes.*

*A mi tía Martha y a Keilita mi prima, que son para mí más que eso.*

*A Fher, que llegó a mi vida en el momento indicado y que significa tanto para mí.*

*A mis amigos, los de tantos momentos, los que están a mi lado y todos los que por un motivo u otro se encuentran lejos.*

## **RESUMEN**

Actualmente la video vigilancia es considerada como una de las soluciones más eficientes cuando de temas de seguridad se trata. Los sistemas de video vigilancia permiten mantener asegurada tanto áreas interiores como los alrededores de empresas, aeropuertos, centros comerciales, bancos, calles e incluso viviendas, donde ya es muy común encontrar sistemas de seguridad basados en video vigilancia. Estos sistemas permiten mantener un seguimiento visual en tiempo de real de bienes, inmuebles, personas, entre otros y además mantener registros de los eventos grabados que pueden ser accedidos posteriormente. Las cámaras pueden ser controladas desde una estación de monitorización física o estaciones móviles y la grabación puede ser activada por horarios y/o mediante eventos recibidos a través de las conexiones de entrada/salida de las cámaras. Todas éstas funcionalidades y las ventajas que traen consigo hacen que los sistemas de video vigilancia tengan cada vez mayor demanda.

El presente trabajo tiene como objetivo fundamental el desarrollo de una aplicación que permita controlar los eventos que tienen lugar a través de las conexiones de entrada/salida de las cámaras IP, utilizando la comunicación a través de los protocolos TCP, FTP, HTTP y SMTP.

Como resultado se obtendrá un producto que permitirá mejorar los niveles de seguridad provistos por el sistema actual, aportando mayores prestaciones y elevando de esta forma la calidad del producto. Por tanto facilitará el control y la vigilancia en las áreas de las entidades donde el sistema sea desplegado.

**Palabras clave:** cámara, conexiones de entrada/salida, producto, protocolo, video vigilancia.

## **ABSTRACT**

Nowadays video surveillance has become the most efficient solutions when it comes to security issues. Video surveillance systems help for maintaining secured areas as the interior and surrounding of businesses, airports, shopping centers, banks, streets and even homes, where it is very common to find security systems based on video surveillance. These systems allow you to keep a visual track of real time of buildings, people, main streets, etc. and also keep records of recorded events that can be accessed later. The cameras can be controlled from a physical monitoring station and mobile stations and the recording can be activated by schedule or by events received through the inlet / outlet connections. All these features and the advantages they bring make video surveillance systems are increasingly in demand.

This work has as main objective the development of an application that can handle the events that take place through the inlet / outlet connections of the IP cameras using communication over the protocols TCP, FTP, HTTP and SMTP.

The result will be an application to improve the security levels provided by the current system, providing higher performance and thus raising the quality of the product. Therefore facilitate the control and surveillance in the areas of the entities where the system is deployed.

**Keywords:** camera, inlet/outlet connections, product, protocol, video surveillance.



## **Índice de Figuras.**

Figura 1: Vista general de RUP.....	13
Figura 2: Modelo de Dominio. ....	22
Figura 3: Diagrama de Casos de Uso. ....	25
Figura 4: Arquitectura en pizarra. ....	39
Figura 5: Patrón Modelo Vista Controlador (MVC). ....	39
Figura 6: Diagrama de análisis. Caso de Uso Escuchar evento por FTP. ....	42
Figura 7: Diagrama de análisis. Caso de Uso Escuchar evento por HTTP.....	42
Figura 8: Diagrama de análisis. Caso de Uso Escuchar evento por SMTP. ....	42
Figura 9: Diagrama de análisis. Caso de Uso Escuchar evento por TCP.....	42
Figura 10: Diagrama de Clases del Diseño. ....	44
Figura 11: Diagrama de secuencia. Caso de Uso Autenticar Usuario. ....	46
Figura 12: Diagrama de secuencia. Caso de Uso Configuran notificación de evento.....	46
Figura 13: Diagrama de secuencia. Caso de Uso Escuchar evento por protocolo FTP.....	46
Figura 14: Diagrama de secuencia. Caso de Uso Escuchar evento por protocolo HTTP. ....	47
Figura 15: Diagrama de secuencia. Caso de Uso Escuchar evento por protocolo TCP. ....	47
Figura 16: Diagrama de secuencia. Caso de Uso Escuchar evento por protocolo SMTP.....	47
Figura 17: Diagrama de secuencia. Caso de Uso Enviar evento al Gestor. ....	48
Figura 18: Diagrama de despliegue. ....	50
Figura 19: Diagrama de componentes. ....	50

## **Índice de Tablas.**

Tabla 1: Definición de los actores. ....	24
Tabla 2: Descripción del Caso de Uso Configurar notificación de evento.....	31
Tabla 3: Descripción del Caso de Uso Escuchar evento por protocolo FTP.....	32
Tabla 4: Descripción del Caso de Uso Escuchar evento por protocolo HTTP. ....	33
Tabla 5: Descripción del Caso de Uso Escuchar evento por protocolo TCP. ....	34
Tabla 6: Descripción del Caso de Uso Escuchar evento por protocolo SMTP.....	35
Tabla 7: Descripción del Caso de Uso Enviar evento al Gestor. ....	36

**Contenido**

<b>Introducción</b> .....	1
<b>Capítulo 1: Fundamentación Teórica</b> .....	6
1.1. Introducción.....	6
1.2. Términos asociados.....	6
1.2.1 Video Vigilancia. La Video Vigilancia IP.....	7
1.2.2 Las Cámaras IP y los sensores físicos. ....	7
1.3. Objeto de estudio.....	8
1.3.1 Descripción general. ....	8
1.3.2 Descripción actual del dominio del problema. ....	8
1.4. Análisis de soluciones existentes.....	9
1.4.1. Axis Camera Station .....	9
1.4.2. Suite de Gestión Remota VisionSurfer .....	10
1.4.3. Xyma Save Vision .....	10
1.5 Fundamentación de la metodología de desarrollo de software seleccionada.....	11
1.5.1 Metodologías Ágiles o Ligeras. ....	11
1.6 Herramientas y tecnologías para el desarrollo de la solución propuesta.....	14
1.6.1 Tecnología para la comunicación.....	14
1.6.2 Lenguajes para el desarrollo de la solución. ....	15
1.6.3 Tecnologías para el desarrollo de la solución.....	17
1.6.4 Herramientas a utilizar en el desarrollo de la aplicación.....	17
1.7. Conclusiones Parciales.....	20
<b>Capítulo 2: Características de la solución propuesta.</b> .....	21
2.1. Introducción.....	21
2.2. Modelo de Dominio.....	21
2.2.1. Conceptos Fundamentales. ....	21
2.2.2. Diagrama de Modelo de Dominio. ....	22

2.3 Propuesta de sistema.....	22
2.4 Requisitos funcionales del sistema.....	22
2.5 Requisitos no funcionales del sistema.....	23
2.6 Definición de los casos de uso.....	24
2.6.1 Definición de los actores.....	24
2.6.2 Listado de los casos de uso.....	25
2.7 Modelo de casos de uso del sistema.....	25
2.8 Descripción de los casos de uso del sistema.....	26
2.9 Conclusiones Parciales.....	37
<b>Capítulo 3: Análisis y diseño de la solución propuesta.....</b>	<b>38</b>
3.1 Introducción.....	38
3.2. Arquitectura.....	38
3.2.1. Arquitectura del componente de control de conexiones entrada/salida.....	39
3.3 Patrones de diseño de software.....	40
3.4 Modelo de análisis.....	41
3.4.1 Diagramas de Clases del Análisis.....	42
3.5 Modelo de Diseño.....	43
3.5.1 Diagrama de Clases del Diseño.....	43
3.6 Diagramas de Interacción.....	45
3.6.1 Diagramas de Colaboración.....	45
3.6.2 Diagramas de Secuencia.....	45
3.7 Conclusiones Parciales.....	48
<b>Capítulo 4: Implementación y Prueba.....</b>	<b>49</b>
4.1 Introducción.....	49
4.2 Modelo de Implementación.....	49
4.2.1 Diagrama de Despliegue.....	49
4.2.2 Diagrama de Componentes.....	50
4.3 Flujo de trabajo de pruebas.....	51

4.3.1 Pruebas realizadas. Resultados .....	52
4.4 Conclusiones Parciales.....	55
<b>CONCLUSIONES GENERALES .....</b>	<b>56</b>
<b>RECOMENDACIONES.....</b>	<b>57</b>
<b>BIBLIOGRAFÍA.....</b>	<b>58</b>
<b>REFERENCIAS BIBLIOGRÁFICAS.....</b>	<b>61</b>
<b>GLOSARIO DE TÉRMINOS.....</b>	<b>64</b>

## Introducción

En este acelerado mundo moderno, todos desean sentirse seguros, tanto en los hogares como en las grandes oficinas empresariales se instalan sistemas que detecten aquello que esté definido como un incidente de seguridad. Los sistemas de video vigilancia ofrecen beneficios para todos pues permiten al menos identificar a los autores en caso de que no sea posible evitar el incidente.

Desde hace más de 25 años el surgimiento de la video vigilancia revolucionó todo lo referente a los sistemas de seguridad pues el uso de las cámaras permitió un gran avance para la seguridad integral, debido a que permite ejercer una vigilancia preventiva mediante el registro visual de eventos. La video vigilancia ha demostrado su valía y beneficios al mantener un seguimiento en tiempo real de personas o activos de una empresa, grabar los sucesos para una posterior revisión, y mantener un bajo margen de errores. La evolución de los sistemas de seguridad basados en video puede dividirse en dos grandes eras, la era analógica y la era digital.

Los sistemas analógicos comenzaron a ser implementados en los años 50 y son conocidos como sistemas CCTV<sup>1</sup> y su funcionalidad consiste en monitorear imágenes con las cámaras y la grabación de estas en cintas de casetes (1). Ya desde los años 70 en países como Estados Unidos e Inglaterra la policía ha usado las cámaras de vigilancia en lugares públicos, tales como calles principales y estaciones de trenes.

Las cintas de casetes fueron responsables en gran medida del auge de la vigilancia por cámaras, producto de que posibilitaban preservar las pruebas permitiendo de esta forma que pudieran ser accedidas para una posterior revisión. Aunque esta tecnología tenía como inconveniente que las cintas debían ser cambiadas diariamente, por lo que estos sistemas CCTV analógicos presentaban muchas debilidades, entre ellas su alto costo de mantención.

Los sistemas digitales, jóvenes aún, marcaron un hito, en el tema de la seguridad y la vigilancia al utilizar la tecnología DVR<sup>2</sup>, por lo que las dificultades de las tecnologías analógicas son eliminadas, las imágenes grabadas digitalmente poseen mayor calidad, además de permitir recopilar una mayor cantidad de información en los dispositivos de almacenamiento. La tecnología digital reconoce entre

---

<sup>1</sup> **CCTV:** Circuito Cerrado de Televisión.

<sup>2</sup> **DVR (Digital Video Recorder o Grabador de Video Digital):** Permite grabar las imágenes en formato digital.

otras cosas activar la grabación por eventos como el movimiento, sensores y alarmas o mediante horarios programados y archivarlos en dispositivos de almacenamiento como los discos duros. Posee una avanzada capacidad de búsqueda, que elimina la necesidad de estar rebobinando cintas, se puede grabar y revisar los archivos de forma simultánea, la calidad de la imagen no se deteriora con el tiempo, permiten la conexión remota desde otros equipos como PCs<sup>3</sup> de escritorio y laptops tanto para la visualización de imágenes en tiempo real como para la configuración de los sistemas DVR<sup>4</sup>. (1)

En el año 1996 Axis lanzó al mercado la primera cámara de red del mundo que se podía conectar a una red IP<sup>5</sup> (2). Esta solución quebró el marco en el que se encontraban los sistemas de vigilancia del momento y creó un nuevo paradigma. Las evidentes ventajas de esta nueva solución dispararon la demanda y por tanto el mercado se llenó de nuevas soluciones potentes y fiables basadas en esta tecnología.

Estas cámaras permiten la incorporación de otros dispositivos de seguridad como los sensores físicos, a través de las conexiones de entrada/salida que poseen. Esto incrementa la posibilidad de detectar incidentes, permitiendo de esta forma tener un mayor control sobre el área que se desea mantener segura, sin que sea necesaria la presencia del factor humano.

En Cuba, esta tecnología se ha ido introduciendo poco a poco, aplicada mayormente en centros comerciales, áreas turísticas y calles donde circulen una gran cantidad de autos y personas. El uso del sistema Xyma Safe Vision ha posibilitado mejorar en gran medida el concepto de seguridad. Este sistema de video vigilancia desarrollado por especialistas de la Empresa de Desarrollo de Aplicaciones, Tecnologías y Sistemas (Datys), del Ministerio del Interior, es capaz de monitorizar y controlar en tiempo real los movimientos que tengan lugar en las áreas identificadas, está basado en tecnología IP y permite además el uso de tecnologías analógicas. (3)

En la Universidad de las Ciencias Informáticas (UCI), en el centro de desarrollo Geoinformática y Señales Digitales (GEySED), específicamente en el departamento de Señales Digitales, el proyecto Video Vigilancia, desarrolla un sistema de video vigilancia basado en tecnología digital llamado SURIA Vision. De este sistema se realizó un despliegue piloto en la terminal de cruceros de La Habana, obteniendo como resultado una buena aceptación por parte de los clientes y cuenta además como posibles clientes para su despliegue, al Ministerio del Turismo (MINTUR) y la Empresa de Telecomunicaciones de Cuba (ETECSA), entre otros.

---

<sup>3</sup> **PC (Personal Computer):** Computadora Personal.

<sup>4</sup> **DVR (Digital Video Recording)**

<sup>5</sup> **IP (Internet Protocol):** Número que identifica un dispositivo en una red.

Actualmente el sistema SURIA Vision no es capaz de detectar notificación alguna proveniente de las cámaras que se encuentren conectadas a sensores físicos, desaprovechando de esta forma funcionalidades que ofrecen las cámaras IP en beneficio de realizar una vigilancia más eficaz. Siendo así, que uno de los problemas que presenta este sistema para garantizar un mejor desempeño en las actividades de monitorización, es la incapacidad de controlar los eventos que notifican las cámaras de vigilancia a través de las conexiones de entrada/salida.

La posibilidad de controlar esta notificación de eventos realizada por las cámaras, otorgará al sistema mayor versatilidad en el proceso de monitorización y permitirá reducir los errores humanos. De esta forma, una vez desplegado, el cliente podrá disfrutar de un sistema más completo, capaz de satisfacer sus exigencias, quedando satisfechos con el producto entregado.

De lo tratado anteriormente surge como **problema a resolver: ¿Cómo controlar los eventos que generan las cámaras de video-vigilancia por las conexiones de entrada/salida?**

Para dar solución a este problema se plantea como objeto de estudio **el control de las conexiones de entrada/salida mediante los protocolos FTP, HTTP, TCP y SMTP en las cámaras de vigilancia**, delimitando como campo de acción **la creación de componentes de escucha a través de protocolos FTP, HTTP, TCP y SMTP para las conexiones de entrada/salida de las cámaras de vigilancia.**

Siendo la idea a defender; **si se implementa un componente que permita controlar las conexiones entrada/salida de las cámaras de vigilancia utilizando plugins, se satisfacen las necesidades actuales del Sistema SURIA Vision de controlar las conexiones de entrada/salida a través de los protocolos FTP, HTTP TCP y SMTP.**

Para solucionar el problema planteado se propone como **objetivo general desarrollar un componente para el Sistema SURIA Vision capaz de controlar las conexiones entrada/salida de las cámaras de vigilancia.**

Las **tareas de investigación** a cumplir son:

1. Caracterizar los sistemas de video vigilancia existentes en la actualidad que permitan controlar las conexiones entrada/salida de las cámaras de vigilancia.
2. Definir las herramientas y tecnologías para el diseño y la implementación del componente de control de conexiones de entrada/salida.



3. Identificar los requisitos del componente de control de conexiones de entrada/salida.
4. Realizar el Modelo de Dominio, Análisis y Diseño del componente de control de conexiones de entrada/salida.
5. Implementar los casos de uso definidos.
6. Validar las funcionalidades implementadas.

Los **métodos científicos de investigación** son aquellos que sirven de apoyo para dar cumplimiento a las tareas de investigación. Los mismos pueden ser teóricos, empíricos o matemáticos. Para cumplir con el grupo de tareas planteadas servirán de ayuda varios **métodos científicos de investigación**, específicamente **métodos teóricos**.

#### **Métodos teóricos:**

1. **Analítico-Sintético:** Este método permitirá estudiar y llegar a una conclusión o un tipo de resumen de lo investigado, se utilizará para profundizar acerca de la tecnología de video vigilancia que se está desarrollando actualmente y algunas características que deben tener los sistemas de video vigilancia.
2. **Análisis histórico-lógico:** Con este método se estudiarán los sistemas existentes de video vigilancia, sus antecedentes y su evolución a lo largo de la historia.
3. **Modelación:** Este método permitirá la realización de los modelos que especifican las características que va a desarrollar la aplicación y que permitirán un mejor entendimiento con los desarrolladores para una posterior implementación.

Este trabajo está organizado de la siguiente manera:

**Capítulo 1:** Fundamentación teórica. En este capítulo se tratan los términos y conceptos de importancia relevante para la investigación. Se realiza un estudio del estado del arte para analizar otras herramientas existentes con el objetivo de saber si estas representan una solución viable. Se seleccionan las herramientas, lenguaje de desarrollo, tecnologías y metodología para desarrollar la propuesta de solución.

**Capítulo 2:** Características de la solución propuesta. En este capítulo se tratan las principales características del sistema. Se muestra el modelo de dominio correspondiente, así como la especificación de los requisitos funcionales y no funcionales, se muestra además el modelo de casos de uso del sistema.

**Capítulo 3:** Análisis y diseño de la solución propuesta. En este capítulo se exponen las particularidades de la aplicación en cuanto a análisis y diseño, además de los correspondientes diagramas que la componen.

**Capítulo 4:** Implementación y prueba. Expone el modelo de implementación derivado del análisis y diseño realizado. Se muestran los diagramas de despliegue y componentes además de los resultados obtenidos a través de las pruebas realizadas.

# Capítulo 1: Fundamentación Teórica

## 1.1. Introducción.

Este capítulo está dirigido a abordar los elementos teóricos que sustentan el objetivo de la investigación científica, se tratan todos aquellos conceptos considerados de importancia para lograr un buen entendimiento de lo que se plantea como problema a resolver. Los sistemas de video vigilancia que existen actualmente serán analizados haciendo énfasis en sus características y funcionalidades a fin de identificar si estos constituyen una posible solución, esto ofrecerá una medida de la importancia y aporte científico de la solución propuesta. Adicionalmente se examinan y definen las herramientas y tecnologías a utilizar en la confección del componente a desarrollar.

## 1.2. Términos asociados.

Los siguientes términos fueron seleccionados por su vital importancia en el campo de la video vigilancia. Para una mejor comprensión del tema tratado en este documento es necesario conocer el significado de los mismos.

**Cámara IP:** Es una cámara que se conecta directamente a la red. Se le asigna una dirección IP para ser accedida desde la red.

**Dirección IP:** Es una serie de números que identifica inequívocamente un dispositivo de red. Las direcciones IP son el estándar utilizado para asignar direcciones a los dispositivos en prácticamente todos los tipos de redes de computadoras, las mismas pueden ser asignadas por ejemplo a computadoras, servidores, periféricos y dispositivos de una red. (4)

**Protocolo IP:** Conocido como el protocolo de internet. Tiene como propósito transmitir paquetes, llamados datagramas IP a través de la red hasta que cada datagrama alcanza su destino. (5)

**Tecnología IP:** Es aquella donde se utiliza el protocolo IP para establecer la comunicación entre las unidades que componen el sistema en general.

**Tecnología Analógica:** La infraestructura del sistema se comunica mediante el uso de un cable coaxial, mediante este cable se realizan las transmisiones punto a punto de video desde una cámara hasta una grabadora.

**Plugin:** Es una aplicación que añade funcionalidades específicas a un programa principal. (6)

### **1.2.1 Video Vigilancia. La Video Vigilancia IP.**

La video vigilancia no es más que el control que se ejerce sobre determinadas zonas mediante la utilización de registros visuales captados por cámaras de video. La video vigilancia IP hace uso de las ventajas de las redes de comunicación IP para realizar la monitorización. En la actualidad son muy utilizados debido a la sencillez de su instalación pues al aprovechar los beneficios de la red no es necesario desplegar toda una estructura de cableado coaxial específica para el sistema de video vigilancia. (7)

### **1.2.2 Las Cámaras IP y los sensores físicos.**

Una cámara IP tiene su propia dirección IP y características propias para tramitar la comunicación en la red. Todo lo que se necesita para la visualización de las imágenes a través de la red se encuentra dentro de la propia cámara. Estas cámaras se conectan directamente a la red como cualquier otro dispositivo e incluyen software propio para servidor Web, cliente FTP y cliente de correo electrónico. (8)

Adicionalmente incorporan puertos conectores I/O (del inglés in/out) o entrada/salida para la conexión de sensores físicos de alertas por ejemplo sensores de movimiento, sensores de humo, entre otros. Cuando un suceso es detectado por uno de estos sensores, una señal es emitida a la cámara, en la cual el propio software incluye una serie de parámetros configurables sobre las acciones a tomar cuando se recibe una notificación de este tipo.

### **1.3. Objeto de estudio.**

#### **1.3.1 Descripción general.**

Dentro del campo de la seguridad los sistemas de video vigilancia como soluciones informáticas entran a jugar un papel fundamental producto de las ventajas que ofrecen tanto a empresas como a pequeños negocios e incluso viviendas. La masiva incorporación de la tecnología a la vida actual ha posibilitado la aparición de dispositivos cada vez más perfeccionados que eliminan los inconvenientes que traía consigo el uso de los aparatos más tradicionales.

Uno de los componentes fundamentales de un sistema de video vigilancia lo constituyen las cámaras. Actualmente las cámaras IP por todas las facilidades que ofrecen disfrutan de gran popularidad y demanda en el mercado, estas vienen equipadas con conexiones I/O del inglés *IN/OUT* o conexiones de entrada/salida mediante las cuales una serie de dispositivos externos como los sensores físicos pueden ser conectados a estas cámaras para recibir las señales emitidas por estos, y a su vez realizar una determinada acción gestionada por el sistema. Esta posibilidad de conexión entre las cámaras IP y diferentes tipos de sensores, por ejemplo sensores de movimiento, humo, calor, entre otros, ofrece numerosas ventajas que enriquecen la calidad en la prestación de servicios de un sistema de video vigilancia pues confiere al sistema mayor automatización y control sobre un área determinada. El problema del control de las conexiones de entrada/salida de las cámaras IP desde el área de la informática, está principalmente en diseñar y desarrollar una solución que sea capaz de controlar los eventos que se generan a través de estas conexiones, utilizando como medio de comunicación los protocolos FTP, HTTP, TCP y SMTP.

#### **1.3.2 Descripción actual del dominio del problema.**

Actualmente los sistemas de video vigilancia gozan de tanta aceptación y demanda producto de las numerosas ventajas que ofrecen a la hora de detectar o prevenir la ocurrencia de un acto delictivo. Con el objetivo de asegurar la permanencia y elevar la demanda de un producto en el mercado y mantenerse a la altura de la competencia, los desarrolladores de estas soluciones añaden cada vez más funcionalidades que hacen posible la aparición de un producto más completo y eficaz.

En la Universidad de las Ciencias Informáticas los Centros de Desarrollo son los encargados de investigar y desarrollar una determinada área o tema de investigación. Los Proyectos Productivos que

componen dichos Centros se especializan en desarrollar las soluciones especificadas de acuerdo a la temática que sigue su Centro.

En el Centro de Desarrollo GEySED, específicamente el proyecto Video Vigilancia, es el encargado de desarrollar un sistema llamado SURIA Vision que hasta el momento no es capaz de gestionar los eventos generados a través de las conexiones de entrada/salida que poseen las cámaras IP. Debido a esto se decidió desarrollar para este sistema un componente que sea capaz de controlar la notificación de los eventos generados a través de las conexiones de entrada/salida de las cámaras, esta solución dotará a dicho sistema de una mayor capacidad de detección, además de favorecer la calidad del producto.

#### **1.4. Análisis de soluciones existentes.**

Hoy día la seguridad sigue siendo un aspecto crucial en la sociedad. Anteriormente era muy común encontrar cámaras solo en los bancos, centros comerciales, instituciones del gobierno y aeropuertos, por solo mencionar algunos. Actualmente, sin embargo vivimos bajo los “ojos” de cientos de cámaras que nos siguen prácticamente a todas partes cuidando calles, negocios e incluso viviendas, donde cada vez es más habitual encontrar cámaras de vigilancia. Con el avance de las tecnologías surgen nuevas formas de realizar los sistemas de video vigilancia y con estos aparecen otras posibilidades que ofrecen solución a los inconvenientes que presentaban las tecnologías que se habían utilizado hasta el momento.

Los sistemas de video vigilancia alcanzan cada vez mayor demanda en el mercado, ante aquellos que cometen actos delictivos más precisos y especializados para burlar los sistemas aparecen otros cada vez con funcionalidades más potentes y automatizadas. Axis fue la primera empresa que utilizó la tecnología de video en red, pero actualmente varias empresas hacen uso de los beneficios de esta tecnología para desarrollar sus propias soluciones.

##### **1.4.1. Axis Camera Station**

Este software está diseñado para los productos de video en red de Axis, proporciona funciones de supervisión de vídeo, grabación y gestión de eventos, grabación de vídeo continua, programada, activada por alarma y/o por detección de movimiento, control avanzado de las conexiones entrada/salida de las cámaras. Adicionalmente esta solución dispone de múltiples funciones de búsqueda de eventos grabados. Además de permitir la visualización y reproducción remota mediante

el cliente para Windows de AXIS Camera Station. Una de sus principales características es su diseño sencillo e intuitivo y el fácil control de las cámaras de red PTZ<sup>6</sup> y PTZ domo. (9)

#### **1.4.2. Suite de Gestión Remota VisionSurfer**

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 que unidos a otras aplicaciones de terceros como control de accesos, alarmas de incendios...etc. y cámaras Megapíxel<sup>7</sup> es capaz de dar una respuesta integral y simplificada a cualquier proyecto de CCTV. (10) Ningún módulo de esta solución se comercializa de forma independiente.

#### **1.4.3. Xyma Save Vision**

Es un sistema de video vigilancia creado por la Empresa de Desarrollo de Aplicaciones, Tecnologías y Sistemas (Datys), es basado en la tecnología IP, aunque admite además cámaras analógicas mediante servidores de videos. Está compuesto por varios módulos interconectados, que se comunican y comparten la base de datos donde se almacena el sistema. Entre sus prestaciones tiene el monitoreo y grabación de video con audio, revisión y manejo de grabaciones de videos, grabaciones de forma manual, programada, continua y por detección de movimiento, sistema de alertas, de análisis inteligente de video y de reportes, con posibilidades de configuración. Adicionalmente este sistema de video vigilancia soporta varios modelos de equipamiento provistos por diferentes fabricantes, entre los que se encuentran Axis, Panasonic, Sony, Planet, entre otros. (11) Tiene como inconveniente que no soporta múltiples plataformas, además del alto costo para adquirir una licencia para su uso.

Luego de analizar algunas de las soluciones existentes actualmente se puede apreciar que presentan varias condiciones restrictivas para su desempeño, por ejemplo la plataforma soportada. Aunque una de ellas provee el control de conexiones entrada/salida de las cámaras, es una solución provista por el fabricante de hardware, por lo tanto presenta una alta dependencia del hardware del fabricante en cuestión. Es válido destacar que son soluciones propietarias, por lo que su adquisición incurriría en una inversión de alto costo para nuestra universidad.

---

<sup>6</sup> **PTZ(Pan Tilt Zoom):** Cámaras que permiten el movimiento de forma vertical, horizontal y realizar zoom.

<sup>7</sup> **Cámaras Megapíxel:** Cámaras que poseen más de un millón de píxeles de resolución. Producen imágenes de mayor calidad que las cámaras tradicionales.

## **1.5 Fundamentación de la metodología de desarrollo de software seleccionada.**

En sus inicios el desarrollo de software estaba enfocado en completar el trabajo en el menor tiempo posible y lograr que este cumpliera satisfactoriamente sus funcionalidades. Con el paso del tiempo el crear software fue volviéndose cada vez más complicado, hoy en día la complejidad de las soluciones de software aumenta velozmente pues cada vez se exigen más funcionalidades que tributen a productos cada vez más completos.

Muchos profesionales notaron que el proceso para crear el producto era tan importante como la solución en sí misma. Ante la necesidad de utilizar determinadas pautas y procedimientos para desarrollar una solución surgen las metodologías de desarrollo de software. Una metodología de desarrollo de software no es más que un conjunto de pasos que permiten controlar y planificar el proceso de desarrollo de un software. La correcta aplicación de estas reglas garantiza la calidad del software y que el mismo sea terminado en el tiempo y con el coste establecido.

Actualmente existen dos vertientes principales en cuanto a los procesos de desarrollo de software, los métodos pesados o tradicionales y los métodos ágiles o ligeros. Las metodologías pesadas hacen mayor énfasis en la planificación y control del proyecto, estas metodologías no son muy flexibles a cambios y generan una rigurosa y detallada documentación de todo el proceso de construcción del software. Por otro lado las metodologías ágiles no siguen un diseño estable ni siguen un proceso planificado. Ambas metodologías son en extremo diferentes por lo que su uso debe estar condicionado por las necesidades que presente el proyecto.

Para fundamentar el tipo de metodología que se seleccionará para el desarrollo de la solución propuesta en este trabajo, se realizará un breve análisis de ambos tipos de metodología.

### **1.5.1 Metodologías Ágiles o Ligeras.**

Estas metodologías están especialmente diseñadas para soportar cambios en el proceso de desarrollo, por tanto no prestan la atención adecuada a realizar una detallada recolección de los requisitos. El análisis y el diseño tampoco es un punto fuerte en estas metodologías, lo que puede traer como consecuencia que el producto no quede con la calidad requerida y deba ser sometido a



constantes pruebas a fin de comprobar su correcta evolución. Una de las metodologías ágiles con más aceptación actualmente es XP<sup>8</sup>.

## **XP**

Surgió como respuesta a los problemas derivados de los cambios en los requisitos, y se plantea como una metodología a emplear en los proyectos de riesgo, esta metodología se basa en la simplicidad, comunicación y retroalimentación. Es una metodología de desarrollo de software que se enmarca en cuatro variables fundamentales, coste, tiempo, calidad y alcance. Esta metodología permite que el control de recursos y la planificación del proyecto se adapten fácilmente a los cambios, adicionalmente la intervención del cliente juega un papel fundamental. Cuenta con 4 fases de desarrollo, Planificación, Diseño, Desarrollo y Pruebas. (12) (13)

### **1.5.2 Metodologías Tradicionales o Pesadas.**

Son resistentes a los cambios pues tratan de prever a priori todos los eventos posibles con una rigurosa planificación del proceso de desarrollo realizando un fuerte análisis y diseño antes de enfrascarse en la construcción del producto. Estas metodologías generan una minuciosa documentación de cada etapa planificada.

Entre las metodologías tradicionales más utilizadas actualmente se encuentra RUP<sup>9</sup>. Una de las principales ventajas de esta metodología es que es adaptable a cualquier proyecto y su popularidad puede estar dada en gran medida a que puede ser utilizada en su estilo tradicional o puede ser ajustada para ser usada de una forma ágil.

## **RUP**

Es una metodología de desarrollo de software que hace uso del UML<sup>10</sup> como lenguaje de modelado. Actualmente está considerada como la más utilizada para el análisis, diseño y documentación de aplicaciones orientadas a objetos.

RUP tiene tres características esenciales, dirigido por casos de uso, donde los casos de uso están directamente relacionados con los requisitos pues definen la secuencia de pasos que conlleva a la realización en implementación de los requisitos definidos por el cliente. Es un proceso iterativo-

---

<sup>8</sup> **XP** (Extreme Programan o Programación Extrema).

<sup>9</sup> **RUP** (Rational Unified Procesos o Proceso Unificado Racional).

<sup>10</sup> **UML** (Unified Modeling Language o Lenguaje Unificado de Modelado): Lenguaje de modelado de sistemas de software.

Incremental donde para cada iteración se definen objetivos a cumplir pues cada iteración se propone un incremento en el proceso de desarrollo, y por último es un proceso centrado en la arquitectura pues define los elementos más significativos y relevantes del sistema y la misma debe ser lo suficientemente completa como para que los involucrados en el desarrollo sepan lo que están construyendo. (14) (15)

**Fases de RUP** (16)

**Inicio:** Se describe el negocio y se identifican todas las entidades externas que interactúan con el sistema.

**Elaboración:** Se establece un marco de trabajo arquitectónico para el sistema y se identifican los riesgos claves del proyecto. Al terminar esta fase se debe tener un modelo de los requisitos del sistema, una descripción arquitectónica y un plan de desarrollo del software.

**Construcción:** Comprende el diseño del sistema, la programación y las pruebas. En esta fase se desarrollan e integran las partes del sistema.

**Transición:** Se entrega el sistema al usuario para hacerlo trabajar en un entorno real. Al terminar esta fase se debe tener un software que funcione correctamente en su entorno operativo.

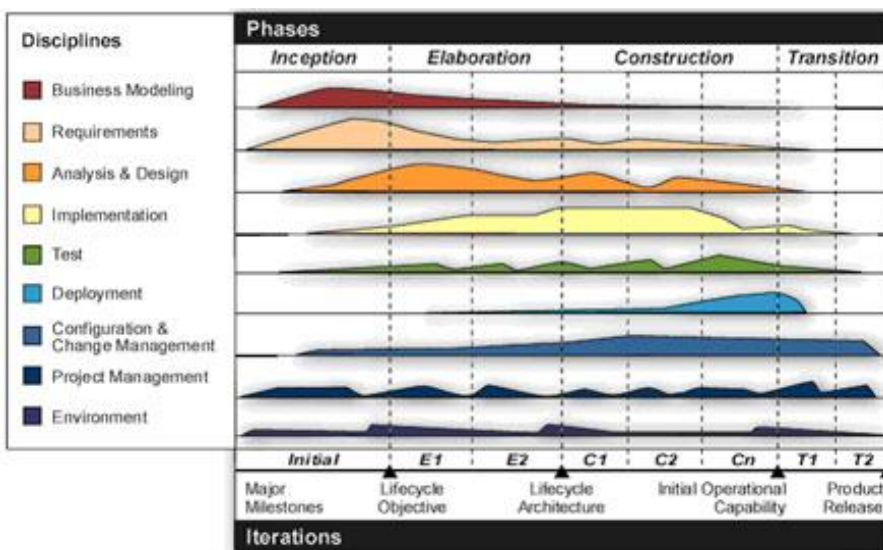


Figura 1: Vista general de RUP.

Teniendo en cuenta el estudio realizado y las características que presentan ambas metodologías analizadas se concluye que es RUP la más recomendable a utilizar pues aporta todos los elementos necesarios para construir aplicaciones de calidad. Los requisitos de la solución a desarrollar son estables, además de que no es necesaria la intervención del cliente durante el proceso de desarrollo. Su uso es conveniente además producto de que el proceso de desarrollo del sistema actual fue soportado por esta metodología.

## 1.6 Herramientas y tecnologías para el desarrollo de la solución propuesta.

### 1.6.1 Tecnología para la comunicación.

**Microsoft Net Remoting:** Es una interfaz de programación de aplicaciones (API) para la comunicación entre procesos. Esta tecnología ofrece un modelo de programación simple y un soporte en tiempo de ejecución para realizar estas interacciones de forma transparente (17). El sistema de interacción remota no presupone ningún modelo de aplicación en particular. Se puede comunicar desde una aplicación web, una aplicación de consola, un servicio de Windows, desde casi cualquier aplicación que se desee utilizar. Los servidores de interacción remota también pueden ser cualquier tipo de dominio de aplicación. Cualquier aplicación puede albergar objetos de interacción remota y proporcionar sus servicios a cualquier cliente en su equipo o red. (18)

**XML-RPC:** Es un protocolo de llamada a procedimiento remoto que usa XML<sup>11</sup> para codificar los datos y HTTP<sup>12</sup> como protocolo de transmisión de mensajes (19). Está diseñado para ser lo más sencillo posible posibilitando siempre que las estructuras de datos sean transmitidas, procesadas y devueltas. XML-RPC funciona mediante intercambio de mensajes entre cliente y servidor, utilizando el protocolo HTTP para el transporte de los mensajes (20).

XML RPC define de manera sencilla como enviar el nombre de un método y una lista de argumentos de un sistema a otro. La idea fundamental de XML RPC es que un documento XML es utilizado para indicar el nombre de un método y la lista de argumentos que este necesita. Este documento XML es enviado a un servidor web utilizando HTTP POST. El servidor procesa el documento XML y ejecuta el método solicitado, retornando luego el resultado en otro documento XML. Una característica de XML RPC es que no se encuentra orientado a un lenguaje particular, sino que simplemente define el formato de mensaje. Esta característica hace que el protocolo sea independiente del lenguaje. (21)

---

<sup>11</sup> XML (extensible Markup Language o Lenguaje Extensible de Marcado).

<sup>12</sup> HTTP (Hyper Text Transfer Protocol o Protocolo de Transferencia de Hipertexto).

Luego de la investigación realizada sobre las tecnologías para la comunicación se selecciona XML-RPC para establecer la comunicación del componente de control de conexiones entrada/salida de las cámaras IP con otros componentes asociados. Este protocolo permitirá el intercambio de información de forma sencilla y eficaz.

### 1.6.2 Lenguajes para el desarrollo de la solución.

**C++:** Es un lenguaje de programación de alto nivel, multiparadigma, elaborado conservando la mayoría de los conceptos del lenguaje C. C++ está considerado como un lenguaje potente y general y ha influido en el diseño de varios lenguajes como Perl, Java, Ada 95 y C#.

Principales ventajas de C++ (22)

**Difusión:** Al ser uno de los lenguajes más empleados en la actualidad, posee un gran número de usuarios y existe una gran cantidad de libros, cursos, páginas web, etc. dedicados a él.

**Versatilidad:** Es un lenguaje de propósito general, por lo que se puede emplear para resolver cualquier tipo de problema.

**Portabilidad:** El lenguaje está estandarizado y un mismo código fuente se puede compilar en distintas plataformas.

**Eficiencia:** Es uno de los lenguajes más rápidos en cuanto a ejecución.

**Herramientas:** Existe una gran cantidad de compiladores, depuradores, librerías, etc.

**C#:** Es un lenguaje de programación de alto nivel orientado a objetos. Fue desarrollado por Microsoft como parte de su plataforma .Net. Este lenguaje está entre los favoritos y más utilizados por los desarrolladores por su sencillez de uso y facilidad de aprendizaje, además de incorporar las ventajas o mejoras de otros lenguajes como C/C++ (del cual deriva su sintaxis básica) y Java, el mismo “se creó como un lenguaje orientado a objetos que recoge las características más avanzadas de Java, así como las más potentes de C++.” (23)

Entre las principales características de este lenguaje se citan las siguientes (23):

**Orientación a objetos:** C# es un lenguaje de programación orientado a objetos puro, es decir, no admite funciones ni variables globales, todo el código debe definirse dentro de tipos de datos (clases, estructuras...). Esto reduce los conflictos de nombres y facilita la legibilidad del código.

C# soporta las características principales del paradigma orientado a objetos:

**Encapsulación:** además de los modificadores public, private y protected se añade uno nuevo: internal. Internal indica que el elemento sólo podrá ser accedido desde su propio ensamblado.

**Herencia:** C#, al igual que Java, sólo soporta herencia simple debido a las ambigüedades que produce la herencia múltiple. La diferencia con Java, está en que para C# todos los métodos serán sellados y para poder redefinir alguno habrá que usar el modificador virtual (como en C++). Con esto las llamadas a los métodos serán más rápidas, ya que si no está presente el modificador virtual se tomará la llamada al método de la clase base.

### **Polimorfismo.**

**Seguridad de tipos:** C# es un lenguaje fuertemente tipado, esto evita cometer errores que son difíciles de detectar.

Sólo se pueden realizar conversiones entre tipos que sean compatibles.

No se pueden usar variables sin inicializar, ya que no tendrían un estado inicial seguro.

En los accesos a elementos de tablas (arreglos, matrices,...) se comprueba que los índices no se salgan de rango.

Se informa con excepciones cuando se producen desbordamientos en operaciones aritméticas.

Aunque un método posea un número indeterminado de parámetros de un tipo, se comprueban.

**Gestión automática de memoria:** C# dispone de recolector de basura, lo cual evita al programador el tener que preocuparse de destruir objetos, liberar memoria.

**Modernidad:** en C# se han incluido elementos nuevos que a lo largo de los años se ha comprobado que son muy útiles para ciertas aplicaciones y que en otros lenguajes como Java y C++ hay que simularlos. Por ejemplo:

- Creación del tipo básico decimal para permitir realizar operaciones de alta precisión con reales de 128 *bits*.
- Creación de la instrucción *foreach* para recorrer colecciones con facilidad y que además se puede ampliar para tipos definidos por el usuario.
- Creación del tipo básico *string* para la representación de cadenas.

### **1.6.3 Tecnologías para el desarrollo de la solución.**

**Qt:** Qt es un framework multiplataforma que incluye una biblioteca de clase, herramientas de desarrollo integrado y un IDE multiplataforma. Utilizando Qt, se puede escribir aplicaciones web y desplegarlas a través de escritorio y sistemas operativos embebidos sin tener que reescribir el código fuente. Qt proporciona los elementos básicos para construir interfaces de usuario modernas, incorporar gráficos 3D, efectos visuales y animaciones. Está licenciado bajo dos licencias de código abierto, GPL y LGPL, así como una licencia comercial (24).

Fue desarrollado por la compañía noruega Trolltech en el año 1995 y en 1996 entró al mundo de los negocios convirtiéndose en la base de un éxito a nivel mundial de miles de aplicaciones. Qt es completamente orientado a objeto y entre sus principales características se encuentran sus excelentes características multiplataforma, representación gráfica 2D/3D, OpenGL, XML y una gran cantidad de documentación de desarrollo. (25)

Para el desarrollo de la solución propuesta se seleccionó el uso del framework Qt con lenguaje de desarrollo C++. Este framework se caracteriza por su sencillez y eficacia. Sus ventajas en cuanto a licenciamiento no impone restricciones para su uso ni para licenciar productos desarrollados bajo él, además de fomentar la política de soberanía tecnológica por la que aboga el país.

### **1.6.4 Herramientas a utilizar en el desarrollo de la aplicación.**

Para el desarrollo de una aplicación es necesario hacer uso de diferentes herramientas que posibiliten desarrollar una solución de calidad. A continuación se realiza un análisis de los principales exponentes de las herramientas existentes de acuerdo a la categoría en la que fueron agrupados.

## ➤ Ingeniería y documentación

Son las llamadas herramientas CASE<sup>13</sup>. Las herramientas CASE nos asisten en la planificación, gestión de requisitos, análisis y diseño de la aplicación e incluso pueden llegar a semi-automatizar la generación de código. El uso de estas herramientas proporciona una mejora en la planificación del proyecto así como la productividad en el desarrollo del mismo. Algunas de las herramientas CASE más conocidas y utilizadas actualmente son las siguientes:

**Rational Rose Enterprise Edition Suite:** Esta es una poderosa herramienta de diseño que permite modelar todo el ciclo de vida de un software. Permite realizar ingeniería inversa, adicionalmente hace posible generar código en diferentes lenguajes a partir de un determinado diseño en UML. Entre sus características está ser una aplicación multi-usuario, generar documentación del sistema, además de admitir múltiples plataformas. Es una herramienta propietaria desarrollada para ser usada sobre el sistema operativo Windows por lo que presenta problemas de compatibilidad con otras plataformas, adicionalmente tiene como inconveniente su elevado costo de licencia.

**Visual Paradigm:** Es una herramienta profesional que utiliza el lenguaje de modelado UML. Soporta el ciclo completo de desarrollo del software, análisis, diseño, construcción, pruebas y despliegue. Adicionalmente permite además generar códigos desde diagramas. Permite aplicar la ingeniería tanto directa como inversa, además de ser una herramienta colaborativa por lo que es posible que varios usuarios trabajen sobre el mismo proyecto. Su intuitivo diseño la convierte en una herramienta muy fácil de utilizar.

**Enterprise Architect 7.0:** Es una plataforma de modelado basada en UML. “Es una herramienta progresiva que soporta todos los aspectos del ciclo de desarrollo, proporcionando una trazabilidad completa desde la fase inicial del diseño a través del despliegue y mantenimiento. También provee soporte para pruebas, mantenimiento y control de cambio.” (26)

Esta herramienta posee bajos costos de licencia, cuenta con una interfaz intuitiva, es multi-usuario y está diseñada para construir un software sólido y fácil de mantener.

En cuanto a Ingeniería y Documentación se selecciona la herramienta Visual Paradigm, su diseño es intuitivo además de ser fácil de usar. Esta herramienta soporta todo el ciclo de desarrollo que propone

---

<sup>13</sup> CASE (Computer Aided Software Engineering o Ingeniería de Software Asistida por Ordenador).

RUP, la cual es la metodología seleccionada. Adicionalmente permite optimizar el trabajo tanto individual como colectivo de los desarrolladores.

### ➤ IDE de Desarrollo

Los IDE's<sup>14</sup> de desarrollo son herramientas que proveen un marco amigable para el trabajo con los lenguajes de programación. Están compuestos por un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. La selección de un determinado IDE debe estar condicionada por el lenguaje que se decida utilizar en el desarrollo de la solución puesto que los IDE's soportan diferentes tipos de lenguajes.

**Microsoft Visual Studio 2010:** Creado por Microsoft para el sistema operativo Windows, este entorno de desarrollo está enfocado fundamentalmente para el uso del lenguaje de programación C#. Este IDE permite a los desarrolladores crear aplicaciones, sitios y servicios web y contiene nuevas formas de trabajo para aplicaciones Microsoft Silverlight a modo de lograr aplicaciones enriquecidas de Internet (RIA) de más calidad. Entre sus principales características se encuentra la incorporación de herramientas para el desarrollo de aplicaciones para el sistema operativo Windows 7.

**Qt Creator:** Es un IDE multiplataforma creado por Nokia para el desarrollo de aplicaciones mediante el uso de librerías Qt. Este IDE soporta varios sistemas operativos incluyendo Linux (32 y 64 bit). De acuerdo a la descripción oficial, Qt Creator está diseñado para permitir a los desarrolladores utilizar el framework Qt de forma más eficiente y completar las tareas de desarrollo fácilmente. Posee integrada la herramienta Qt Designer que permite diseñar y construir GUIs<sup>15</sup> de forma rápida y sencilla.

Se selecciona como IDE de desarrollo Qt Creator. El mismo es la plataforma por excelencia para el trabajo con el framework de desarrollo Qt, siendo este ultimo parte de la tecnología seleccionada para el desarrollo de la solución propuesta. El uso de esta tecnología permitirá desarrollar el componente de forma más rápida y sencilla.

---

<sup>14</sup> IDE (Integrated Development Environment o Entorno de Desarrollo Integrado).

<sup>15</sup> GUI (Graphic User Interface o Interfaz Gráfica de Usuario).



## **1.7. Conclusiones Parciales.**

En este capítulo se ofrece una visión general de los elementos teóricos de la investigación asociados al dominio del problema, lo que posibilita un mayor entendimiento de lo abordado en la situación problemática. Del estudio realizado del actual estado del arte se concluye que los sistemas de video vigilancia analizados tanto en entorno web como de escritorio presentan restricciones que hacen imposible su uso para satisfacer las necesidades planteadas. Producto de esto se realizó un análisis de las principales metodologías, herramientas y tecnologías que se podrían utilizar para desarrollar una solución eficaz y de alta calidad, seleccionando aquellas que se consideraron las más adecuadas de acuerdo a las características del componente a desarrollar y que permitieran la integración de la solución al actual sistema de video vigilancia SURIA Vision.

## Capítulo 2: Características de la solución propuesta.

### 2.1. Introducción

Este capítulo comprende un estudio del modelo de negocio. Se desarrolla un Modelo de Dominio producto de la poca estructuración de los procesos de negocio. Se tratan los requisitos funcionales y no funcionales, identificando los Casos de Uso a partir de los requisitos funcionales, estructurando de esta forma los Diagramas de Casos de Uso del Sistema.

### 2.2. Modelo de Dominio.

Producto de la inexistente estructura de un proceso de negocio se decidió desarrollar un Modelo de Dominio o Modelo Conceptual para comprender el ambiente en el cual se desarrolla el sistema. El Modelo de Dominio permite mostrar a los usuarios los conceptos fundamentales referentes al dominio del sistema propuesto. Aprovechando la notación UML para la representación de conceptos el Modelo de Dominio se representa en forma de diagramas de clases y aunque algunos objetos del Modelo de Dominio puedan terminar siendo clases del propio software, este no representa objetos de software sino aquellos objetos relacionados con el sistema y las relaciones entre ellos.

#### 2.2.1. Conceptos Fundamentales.

**Cámara:** Dispositivo mediante el cual son recibidas las señales emitidas por los dispositivos externos.

**Administrador:** Personal encargado de trabajar con las cámaras y el gestor.

**Dispositivo Externo:** Hardware capaz de detectar ciertos eventos ocurridos en el área donde se encuentre instalados, por ejemplo sensores físicos de detección de humo, movimiento, calor, presión, etc.

**Gestor:** Es uno de los módulos del sistema actual, como su nombre lo indica se encarga de gestionar las peticiones y distribuir las a los otros módulos del sistema de acuerdo a su funcionalidad.

## 2.2.2. Diagrama de Modelo de Dominio.

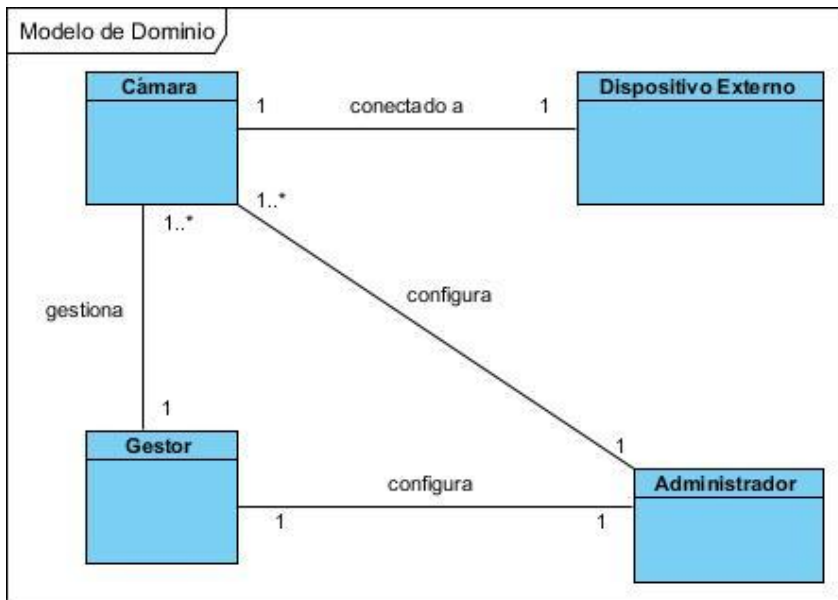


Figura 2: Modelo de Dominio.

## 2.3 Propuesta de sistema.

La solución a desarrollar proveerá al sistema actual SURIA Vision de las funcionalidades de escucha de los eventos generados por las cámaras a través de sus conexiones de entrada/salida. El sistema trabajará obteniendo del Gestor los datos necesarios para su correcto funcionamiento. Al usuario seleccionar una cámara para configurar un plugin de escucha determinado podrá visualizar una ventana con las opciones de configuración para ese plugin y el administrador podrá modificar la configuración existente. En caso de que la cámara no tenga una configuración previa, el administrador podrá introducir una configuración básica que permita a la aplicación funcionar de forma adecuada. De acuerdo al modelo de la cámara y el protocolo de comunicación seleccionado se cargará el plugin de escucha adecuado, siempre y cuando este se encuentre implementado.

## 2.4 Requisitos funcionales del sistema.

Los requisitos funcionales son aquellos que describen o especifican lo que el software debe realizar. Declaran de forma explícita los servicios que el sistema debe brindar.

**RF1** Autenticar Usuario.

**RF2** Configurar notificación de eventos.

**RF3** Escuchar eventos por protocolo FTP.

**RF4** Escuchar eventos por protocolo HTTP.

**RF5** Escuchar eventos por protocolo TCP.

**RF6** Escuchar eventos por protocolo SMTP.

**RF7** Enviar evento al Gestor

## **2.5 Requisitos no funcionales del sistema.**

Los requisitos no funcionales de un sistema son características específicas que el software debe tener y que deben existir en el entorno en el que será desplegado el producto para que este pueda funcionar correctamente.

Los requisitos no funcionales consisten en restricciones impuestas por el entorno y la tecnología, especificaciones sobre el tiempo de respuesta o volumen de información tratado por unidad de tiempo, requisitos en cuanto a interfaces, extensibilidad, facilidad de mantenimiento, etc. (27)

### ➤ **Restricciones de diseño.**

El sistema será implementado siguiendo el paradigma de la Programación Orientada a Objetos, se utilizará el framework Qt 4.x.

### ➤ **Usabilidad**

Está concebido para ser operado por usuarios con conocimientos informáticos.

### ➤ **Eficiencia**

Eficiencia: El sistema debe ser eficiente a las peticiones realizadas en cada momento.

### ➤ **Soporte**

El sistema debe ser sencillo y práctico para los usuarios. Se requiere de la instalación de un servidor FTP, instalación de Windows XP o superior en las estaciones de trabajo. Además de la posibilidad de conexión a un servidor de correo.

➤ **Software**

Sistema Operativo: GNU Linux y Windows XP Service Pack 2 o superior.

➤ **Hardware**

1 GB de Memoria RAM (mínimo), 2GB Memoria RAM (recomendado).

Microprocesador Intel Pentium IV a 3.0 GHz o superior (recomendado).

Tarjeta de Red Gigabit Ethernet (recomendado).

➤ **Seguridad**

Se garantizará la seguridad mediante la autenticación de usuarios.

## 2.6 Definición de los casos de uso.

Los diagramas de caso de uso sirven para mostrar las funciones de un sistema de software desde el punto de vista de sus interacciones con el exterior y sin entrar ni en la descripción detallada ni en la implementación de estas funciones. (27)

### 2.6.1 Definición de los actores.

Actor	Descripción
Administrador	Rol responsable de velar por el correcto funcionamiento del sistema así como su configuración.
Componente	Rol ficticio para representar al componente de control de conexiones de entrada /salida como iniciador de algún caso de uso.

Tabla 1: Definición de los actores.

## 2.6.2 Listado de los casos de uso.

**CU-1** Autenticar usuario

**CU-2** Configurar notificación de eventos.

**CU-3** Escuchar eventos por protocolo FTP.

**CU-4** Escuchar eventos por protocolo HTTP.

**CU-5** Escuchar eventos por protocolo TCP.

**CU-6** Escuchar eventos por protocolo SMTP.

**CU-7** Enviar evento al Gestor.

## 2.7 Modelo de casos de uso del sistema.

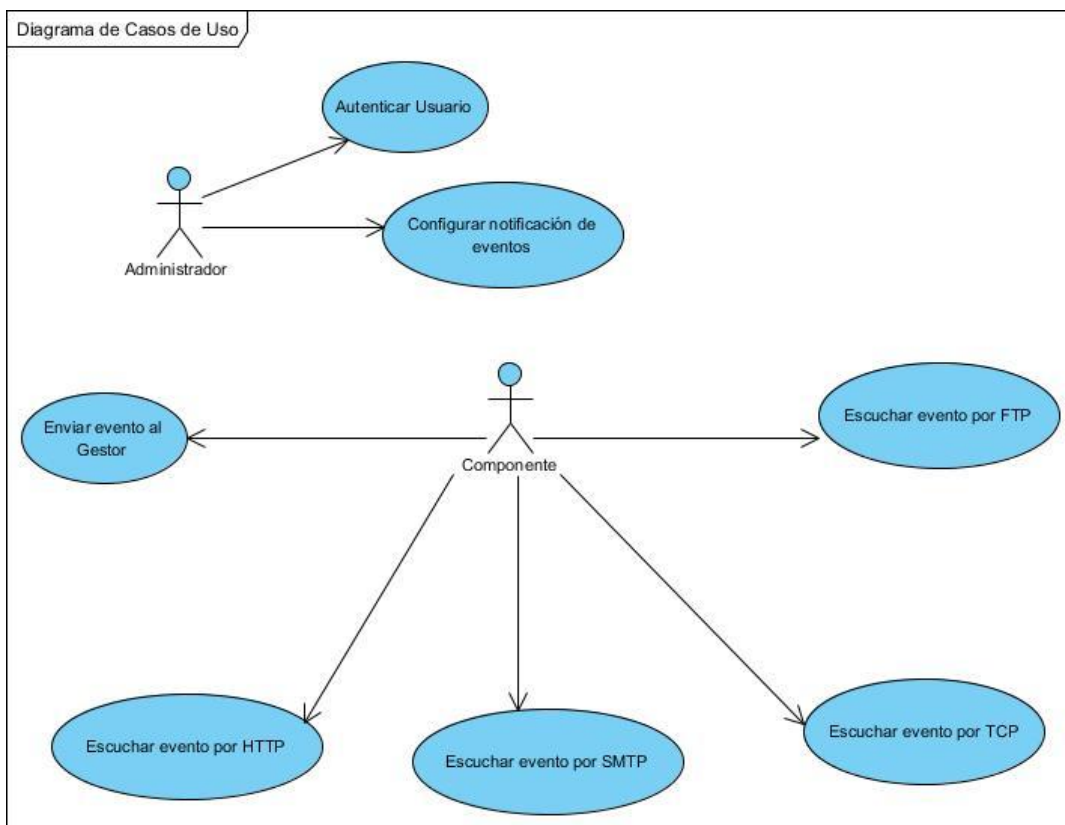


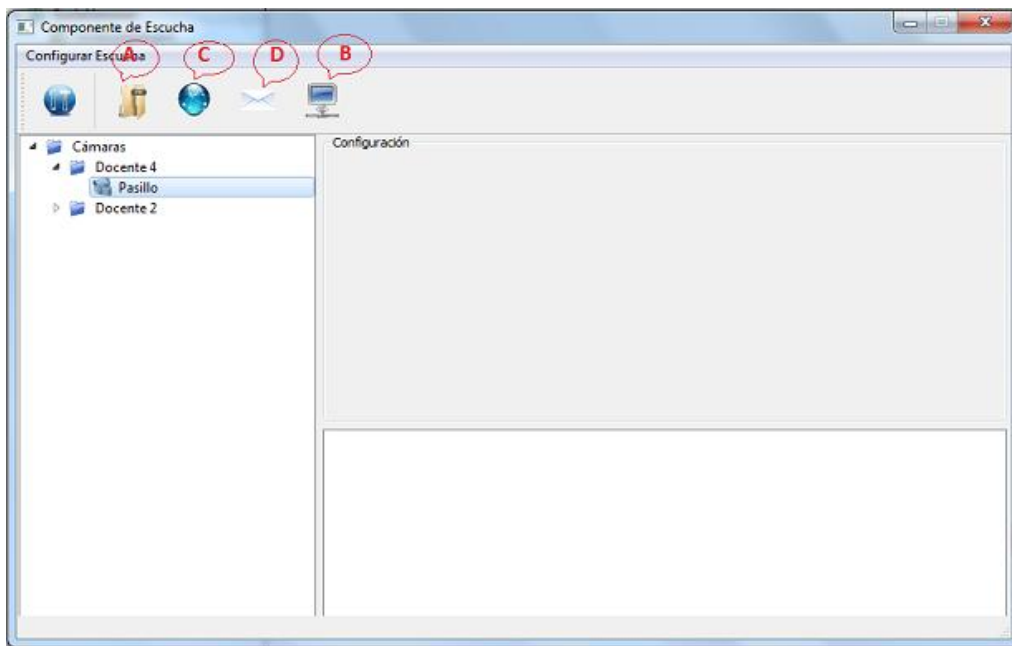
Figura 3: Diagrama de Casos de Uso.

## 2.8 Descripción de los casos de uso del sistema.

<b>Caso de Uso:</b>	Configurar notificación de eventos.
<b>Actores:</b>	Administrador
<b>Resumen:</b>	El caso de uso comienza cuando el Administrador accede a la aplicación para ejecutar las configuraciones para la notificación del proceso de escucha de eventos. Finaliza cuando se hayan realizado satisfactoriamente o no alguna de las operaciones realizadas.
<b>Precondiciones:</b>	Esta acción solo puede ser realizada por el administrador del sistema siempre y cuando haya sido autenticado previamente.
<b>Referencias:</b>	RF2
<b>Prioridad:</b>	
<b>Flujo Normal de Eventos</b>	
<b>Acción del Actor</b>	<b>Respuesta del Sistema</b>
1. Accede a la aplicación para configurar la notificación de escucha de eventos.	2. Muestra una interfaz donde brinda la opción para configurar la notificación de escucha de eventos:  A. FTP B. TCP C. HTTP D. SMTP
2.1 Si el administrador selecciona la opción A).	2.1.1 Ver sección (A) "Configurar notificación FTP".
2.2 Si el administrador selecciona la opción B).	2.2.1 Ver sección (B) "Configurar notificación TCP".
2.3 Si el administrador selecciona la opción C).	2.3.1 Ver sección (C) "Configurar notificación HTTP".

2.4 Si el administrador selecciona la opción D).	2.4.1 Ver sección (D) “Configurar notificación SMTP”.
	3. Muestra los parámetros de configuración de acuerdo a la opción seleccionada por el administrador.

**Prototipo de Interfaz**



**Sección (A) “Configurar notificación FTP”**

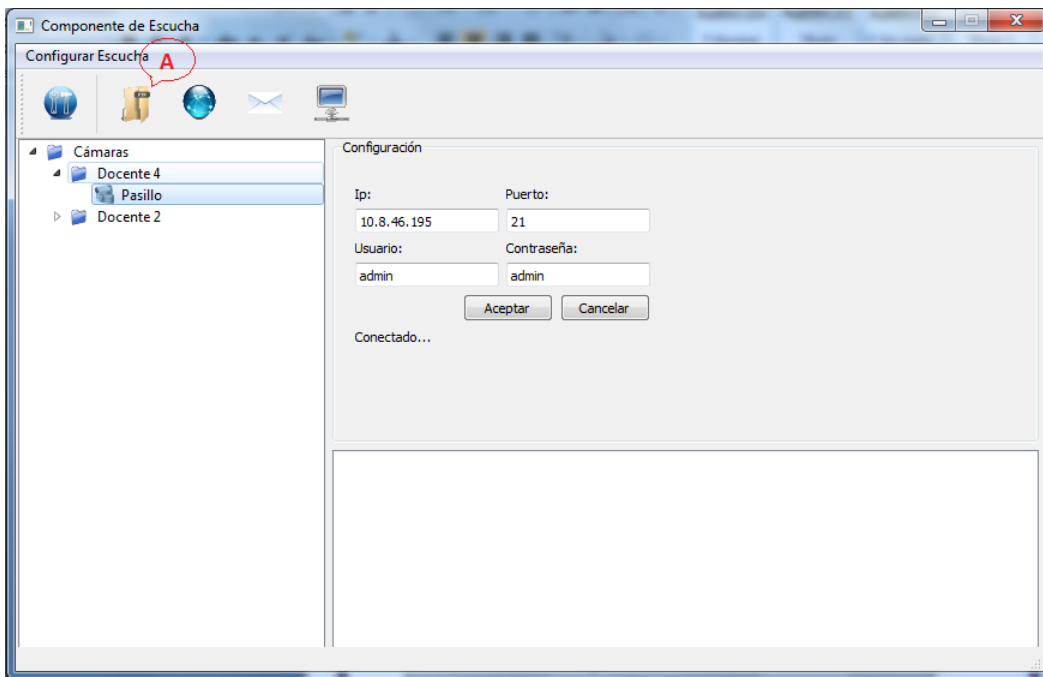
Acción del Actor	Respuesta del Sistema
	2.1 Muestra los parámetros de notificación de escucha de un determinado evento a través del protocolo FTP
3. Modifica las opciones y acepta.	4. Aplica los cambios realizados.

**Flujo Alterno**



Acción del Actor	Respuesta del Sistema
3. No introduce los parámetros de configuración en el formato correcto.	4. Notifica el error y no realiza ningún cambio en la configuración.

**Prototipo de Interfaz**



**Sección (B) “Configurar notificación TCP”**

Acción del Actor	Respuesta del Sistema
	2.2 Muestra los parámetros de notificación de escucha de un determinado evento a través del protocolo TCP.

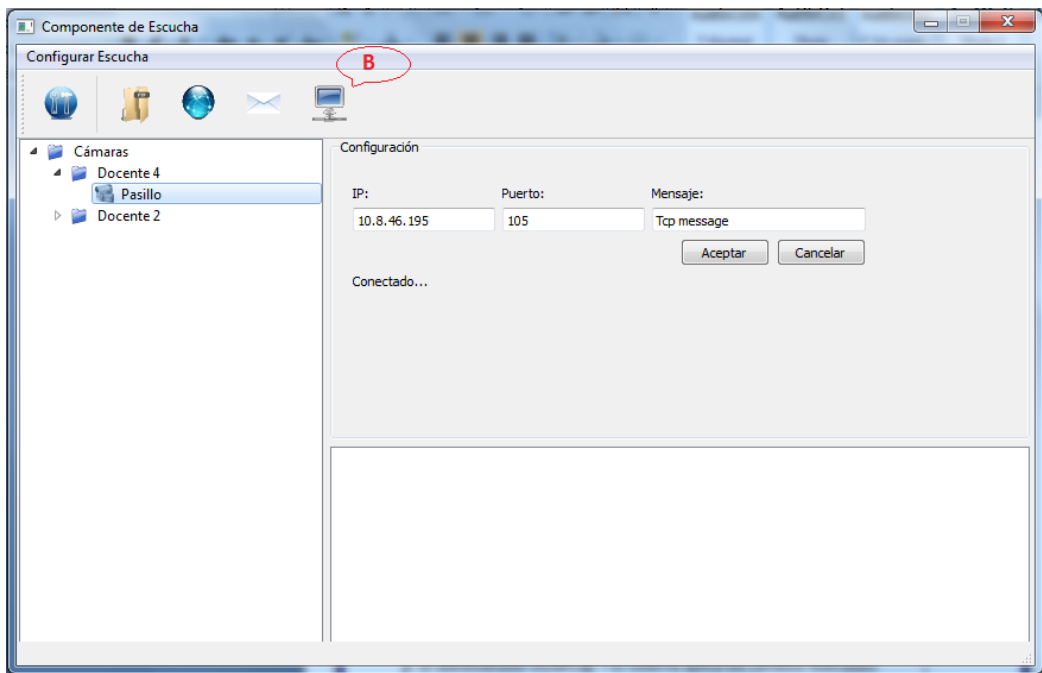
3. Modifica las opciones y acepta.	4. Aplica los cambios realizados.
------------------------------------	-----------------------------------

**Flujo Alterno**

Acción del Actor	Respuesta del Sistema
------------------	-----------------------

3. No introduce los parámetros de configuración en el formato correcto.	4. Notifica el error y no realiza ningún cambio en la configuración.
---	--

**Prototipo de Interfaz**



**Sección (C) “Configurar notificación HTTP”**

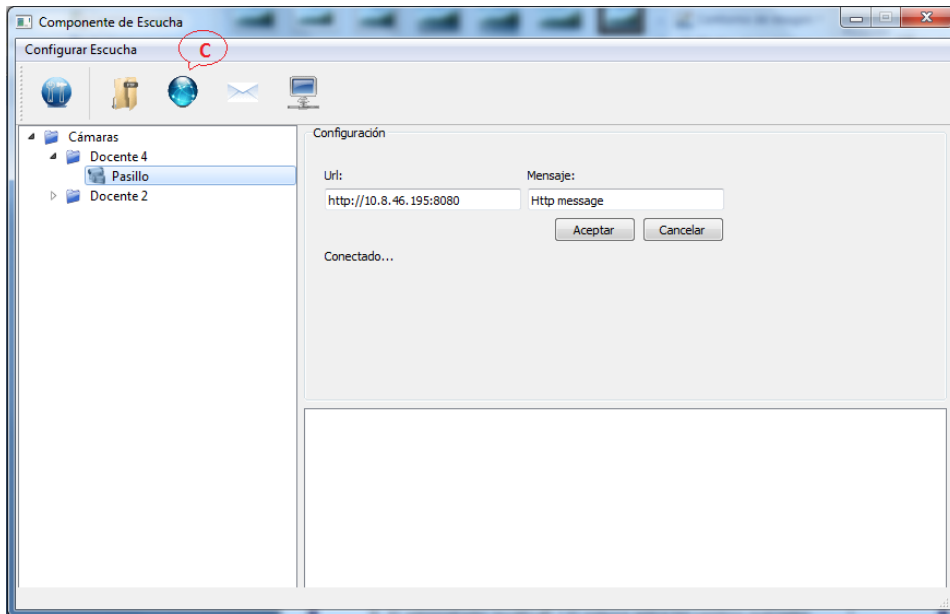
Acción del Actor	Respuesta del Sistema
------------------	-----------------------

	2.3 Muestra los parámetros de notificación de escucha de un determinado evento a través del protocolo HTTP.
3. Modifica las opciones y acepta.	4. Aplica los cambios realizados.

**Flujo Alterno**

<b>Acción del Actor</b>	<b>Respuesta del Sistema</b>
3. No introduce los parámetros de configuración en el formato correcto.	4. Notifica el error y no realiza ningún cambio en la configuración.

**Prototipo de Interfaz**



**Sección (D) “Configurar notificación SMTP”**

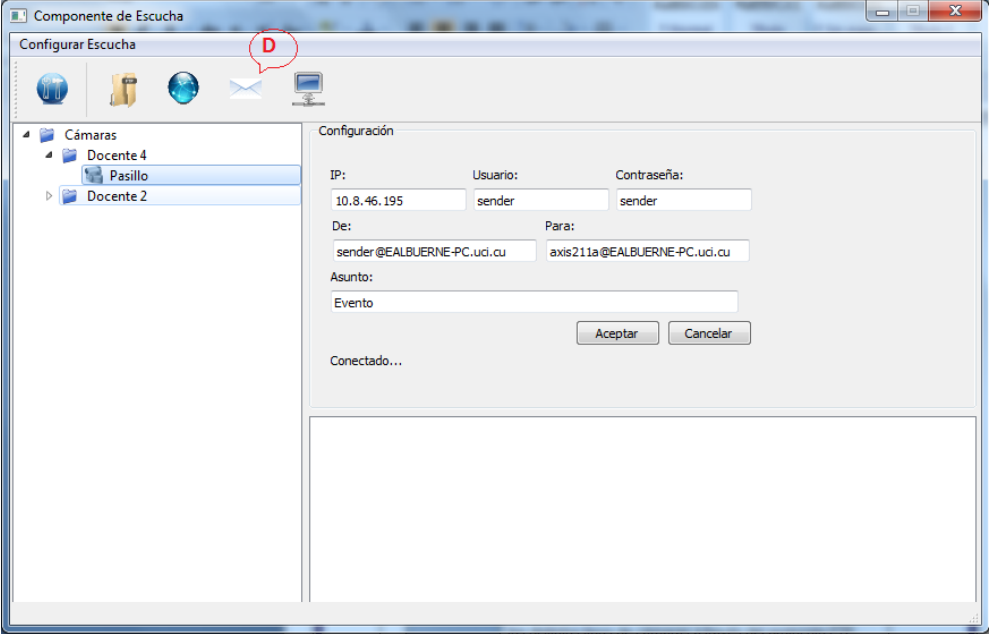
Acción del Actor	Respuesta del Sistema
	2.4 Muestra los parámetros de notificación de escucha de un determinado evento a través del protocolo SMTP.
3. Modifica las opciones y acepta.	4. Aplica los cambios realizados.
Flujo Alternativo	
3. No introduce los parámetros de configuración en el formato correcto.	4. Notifica el error y no realiza ningún cambio en la configuración.
Prototipo de Interfaz	
	
Poscondiciones	Se configura la notificación de escucha de eventos teniendo en cuenta la selección del administrador.

Tabla 2: Descripción del Caso de Uso Configurar notificación de evento.

<b>Caso de Uso:</b>	Escuchar evento por protocolo FTP	
<b>Actor:</b>	Componente	
<b>Resumen:</b>	El caso de uso se inicia cuando se recibe una petición de escucha a través del protocolo FTP para una cámara determinada, y termina cuando se inicia la escucha.	
<b>Precondiciones:</b>	La cámara debe estar en línea y debe existir el plugin para ese modelo de cámara.	
<b>Referencias</b>	RF3	
<b>Prioridad</b>	Crítico	
<b>Flujo Normal de Eventos</b>		
<b>Acción del Actor</b>	<b>Respuesta del Sistema</b>	
	Recibe la petición de escucha.	
	Se mantiene escuchando esperando por el evento emitido por la cámara.	
<b>Flujos Alternos</b>		
<b>Acción del Actor</b>	<b>Respuesta del Sistema</b>	
<b>Prototipo de Interfaz</b>		
<b>Poscondiciones</b>	El sistema notificará al Gestor la escucha de un evento.	

Tabla 3: Descripción del Caso de Uso Escuchar evento por protocolo FTP.

<b>Caso de Uso:</b>	Escuchar evento por protocolo HTTP	
<b>Actor:</b>	Componente	
<b>Resumen:</b>	El caso de uso se inicia cuando se recibe una petición de escucha a través del protocolo HTTP para una cámara determinada, y termina al iniciar la escucha.	
<b>Precondiciones:</b>	La cámara debe estar en línea y debe existir el plugin para ese modelo de cámara.	
<b>Referencias</b>	RF4	
<b>Prioridad</b>	Crítico	
<b>Flujo Normal de Eventos</b>		
<b>Acción del Actor</b>	<b>Respuesta del Sistema</b>	
	Recibe la petición de escucha.	
	Se mantiene escuchando esperando por el evento emitido por la cámara.	
<b>Flujos Alternos</b>		
<b>Acción del Actor</b>	<b>Respuesta del Sistema</b>	
<b>Prototipo de Interfaz</b>		
<b>Poscondiciones</b>	El sistema notificará al Gestor la escucha de un evento.	

Tabla 4: Descripción del Caso de Uso Escuchar evento por protocolo HTTP.

<b>Caso de Uso:</b>	Escuchar evento por protocolo TCP
<b>Actor:</b>	Componente
<b>Resumen:</b>	El caso de uso se inicia cuando se recibe una petición de escucha a través del protocolo TCP para una cámara determinada, y termina cuando se inicia la escucha.
<b>Precondiciones:</b>	La cámara debe estar en línea y debe existir el plugin para ese modelo de cámara.
<b>Referencias</b>	RF5
<b>Prioridad</b>	Crítico
<b>Flujo Normal de Eventos</b>	
<b>Acción del Actor</b>	<b>Respuesta del Sistema</b>
	Recibe la petición de escucha.
	Se mantiene escuchando esperando por el evento emitido por la cámara.
<b>Flujos Alternos</b>	
<b>Acción del Actor</b>	<b>Respuesta del Sistema</b>
<b>Prototipo de Interfaz</b>	
<b>Poscondiciones</b>	El sistema notificará al Gestor la escucha de un evento.

Tabla 5: Descripción del Caso de Uso Escuchar evento por protocolo TCP.

<b>Caso de Uso:</b>	Escuchar evento por protocolo SMTP	
<b>Actor:</b>	Componente	
<b>Resumen:</b>	El caso de uso se inicia cuando se recibe una petición de escucha a través del protocolo SMTP para una cámara determinada, y termina cuando se inicia la escucha.	
<b>Precondiciones:</b>	La cámara debe estar en línea y debe existir el plugin para ese modelo de cámara.	
<b>Referencias</b>	RF6	
<b>Prioridad</b>	Crítico	
<b>Flujo Normal de Eventos</b>		
<b>Acción del Actor</b>	<b>Respuesta del Sistema</b>	
	Recibe la petición de escucha.	
	Se mantiene escuchando en espera del evento emitido por la cámara.	
<b>Flujos Alternos</b>		
<b>Acción del Actor</b>	<b>Respuesta del Sistema</b>	
<b>Prototipo de Interfaz</b>		
<b>Poscondiciones</b>	El sistema notificará al Gestor la escucha de un evento.	

Tabla 6: Descripción del Caso de Uso Escuchar evento por protocolo SMTP.



<b>Caso de Uso:</b>	Enviar evento al Gestor
<b>Actor:</b>	Componente
<b>Resumen:</b>	El caso de uso se inicia cuando se escucha un evento proveniente de una cámara y termina cuando se notifica al Gestor.
<b>Precondiciones:</b>	Debe estar activo el plugin de escucha.
<b>Referencias</b>	RF7
<b>Prioridad</b>	Crítico
<b>Flujo Normal de Eventos</b>	
<b>Acción del Actor</b>	<b>Respuesta del Sistema</b>
	Detecta un evento emitido por una cámara.
	Notifica al Gestor la ocurrencia de dicho evento.
<b>Flujos Alternos</b>	
<b>Acción del Actor</b>	<b>Respuesta del Sistema</b>
<b>Prototipo de Interfaz</b>	

Tabla 7: Descripción del Caso de Uso Enviar evento al Gestor.

## **2.9 Conclusiones Parciales**

Con la descripción realizada de la solución propuesta se logró un acercamiento al funcionamiento básico de la aplicación. Con el modelo de dominio correspondiente se obtiene una mayor comprensión de los conceptos que se manejan en el dominio del sistema. El levantamiento de requisitos realizado permitió desde los requisitos funcionales describir los casos de uso correspondientes y mediante los no funcionales se especifican las cualidades que el producto debe tener así como las condiciones necesarias para su correcto funcionamiento.

## **Capítulo 3: Análisis y diseño de la solución propuesta.**

### **3.1 Introducción**

En este capítulo se describe la arquitectura y los patrones de diseño seleccionados para el desarrollo de la solución propuesta. Se modelan además los diagramas de clases del análisis y los diagramas de interacción correspondientes con el objetivo de proporcionar un mayor entendimiento del funcionamiento de la aplicación.

### **3.2. Arquitectura**

La arquitectura de software es una representación de los subsistemas y componentes de un sistema de software y como se establecen las relaciones entre ellos. De forma más simple podría decirse que establece la organización de los componentes de un sistema y la forma en que estos interactúan. Antes de definir una arquitectura para la solución que se propone en el presente trabajo es necesario conocer las características esenciales de la arquitectura utilizada en el sistema SURIA Vision de forma global.

Este sistema posee una arquitectura en pizarra en la variante tablero de control. Esta desacopla el sistema en componentes denominados agentes autónomos, los cuales son independientes en la realización atómica de su funcionalidad. Pero dependen de una entrada de información externa, que es provista por otros agentes, y a su vez producen un resultado que puede a su vez ser entrada de otros agentes. Cada agente se rige por interfaces estándares que permiten cambios en el ambiente externo al agente sin que este sufra cambios en su funcionamiento interno. Todo el funcionamiento de los agentes autónomos está coordinado por un elemento central denominado Repositorio Activo, el cual entrega y recibe información de los agentes y coordina su funcionamiento. (28) En el sistema SURIA Vision es el Gestor quien funciona como Repositorio Activo mientras que el componente a desarrollar actuará como agente autónomo.

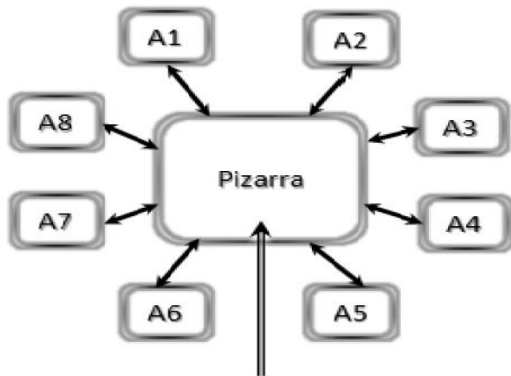


Figura 4: Arquitectura en pizarra.

### 3.2.1. Arquitectura del componente de control de conexiones entrada/salida

Aunque el sistema SURIA Vision de forma general utiliza una arquitectura en pizarra, cada módulo utiliza el estilo Modelo-Vista-Controlador (MVC), el cuál será utilizado para desarrollar el componente propuesto.

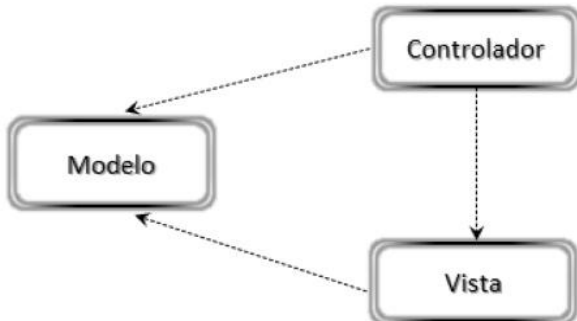


Figura 5: Patrón Modelo Vista Controlador (MVC).

Como su nombre lo indica este patrón permite dividir una aplicación en tres módulos, el Modelo, las Vistas y el Controlador. El Modelo se encarga de gestionar los datos para que la aplicación funcione, la Vista de mostrar los datos al usuario e interactuar con él y se actualiza en caso de que ocurra algún cambio en el Modelo, mientras que el Controlador registrará la comunicación entre la Vista y el Modelo. El uso de este patrón proporciona una mayor flexibilidad ante los cambios realizados en alguna de las capas producto de la independencia de funcionalidades existente entre ellas, por lo que la forma de

trabajo que propone este patrón proporciona un considerable ahorro de tiempo y esfuerzo para los desarrolladores.

Además del uso del MVC se pretende utilizar un estilo basado en Plugins o Extensiones, permitiendo de esta forma que el sistema gane en extensibilidad. La complejidad de este estilo se centra en manejar las particularidades de cada hardware. Esto permitirá al sistema soportar cualquier tipo de hardware siempre y cuando tenga implementado el plugin específico para este.

### 3.3 Patrones de diseño de software

Para desarrollar un producto de forma eficaz no solo es necesario definir una arquitectura adecuada. Los patrones de diseño de software tratan problemas frecuentes del diseño y que se presentan en situaciones particulares, con el fin de proponer soluciones a ellas. Por lo tanto, los patrones de diseño son soluciones exitosas a problemas comunes.

En la programación orientada a objetos es muy importante asignar correctamente las responsabilidades, a fin de lograrlo, se emplearán los siguientes patrones *GRASP*<sup>16</sup>:

- **Experto.**
- **Creador.**
- **Bajo Acoplamiento.**
- **Alta Cohesión.**
- **Controlador.**

Con el patrón Experto se pretende asignar a cada clase las responsabilidades que le competen a partir de la información que esta posee. El patrón Creador permitirá instanciar aquellas clases que se consideran adecuadas para cumplir las funcionalidades, tratando así de mantener un Bajo Acoplamiento y una Alta Cohesión, el uso de estos dos patrones permitirá un diseño más independiente, donde las clases no asuman demasiadas responsabilidades, además de posibilitar la reutilización y mejoras en las funcionalidades. Al emplear el patrón Controlador, se estructura el sistema con una clase controladora para que sea esta quien, de acuerdo con el patrón arquitectónico Modelo-Vista-Controlador, se encargue de los eventos y funcionalidades que representan el caso de uso que implementa.

---

<sup>16</sup> *GRASP (General Responsibility Assignment Software Patterns o Patrones Generales para Asignar Responsabilidades.)*

Como apoyo al diseño también se hace uso de patrones *GoF*<sup>17</sup>:

- ***Singleton***
- ***Abstract Factory***

Al crear una sola instancia de las clases el patrón Singleton permite resolver el problema de duplicado de información o discordancia. Este permite crear un punto de acceso global a la clase mediante un método de clase.

El patrón Abstract Factory en ocasiones es denominado como un patrón “familiar” pues proporciona la forma para crear instancias de familias de objetos. Debido a esto se recomienda su uso cuando se trabaja con “familias” de productos. Para la solución que se propone en este trabajo el uso del abstract factory proporcionará la vía para crear las instancias para las diferentes “familias de plugins”, dígase plugin HTTP, plugin FTP, plugin TCP y plugin SMTP.

### **3.4 Modelo de análisis.**

Durante el análisis los requisitos son analizados de una forma más detallada y refinada. De acuerdo con Jacobson, Booch y Rumbaugh en El Proceso Unificado de Desarrollo de Software el análisis (29):

- Está descrito con el lenguaje del desarrollador.
- Esboza como llevar a cabo una funcionalidad del sistema, sirve como una aproximación del diseño.
- Define realizaciones de casos de uso, y cada una de ellas representa el análisis de un caso de uso del modelo de casos de uso.

---

<sup>17</sup> *GoF (Gang of Four): Patrones de diseño utilizados en la programación orientada a objetos.*

### 3.4.1 Diagramas de Clases del Análisis.

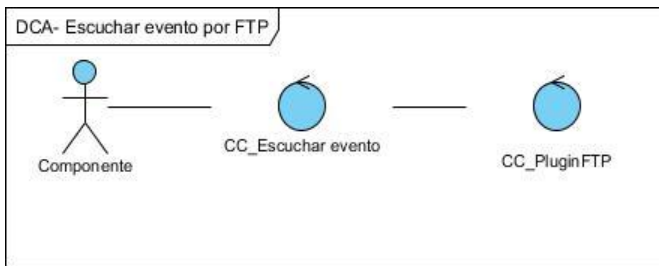


Figura 6: Diagrama de análisis. Caso de Uso Escuchar evento por FTP.



Figura 7: Diagrama de análisis. Caso de Uso Escuchar evento por HTTP.

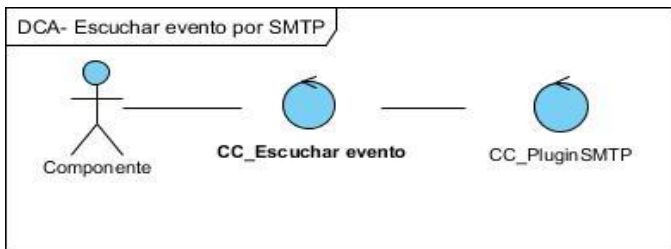


Figura 8: Diagrama de análisis. Caso de Uso Escuchar evento por SMTP.

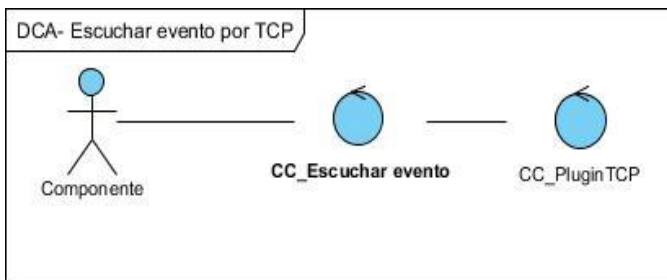


Figura 9: Diagrama de análisis. Caso de Uso Escuchar evento por TCP.

Los diagramas correspondientes a los casos de uso restantes fueron ubicados en los Anexos del presente trabajo, donde pueden encontrarse específicamente en el [Anexo II](#).

### **3.5 Modelo de Diseño.**

El modelo de diseño es un modelo físico porque es un plano de la implementación y figura como realizar un caso de uso en términos de clases de diseño y sus objetos. Este modelo sirve de abstracción de la implementación del sistema, y es utilizada como un punto de partida para las actividades de implementación. Conformar el sistema tratando de mantener todo lo posible la estructura definida por el modelo de análisis. (29)

#### **3.5.1 Diagrama de Clases del Diseño.**

Estos diagramas describen gráficamente las especificaciones de las clases de software. Generalmente contienen la siguiente información (30):

- Clases, asociaciones y atributos.
- Métodos e información sobre los tipos de atributos.



A continuación se muestra el diagrama de clases del diseño correspondiente:

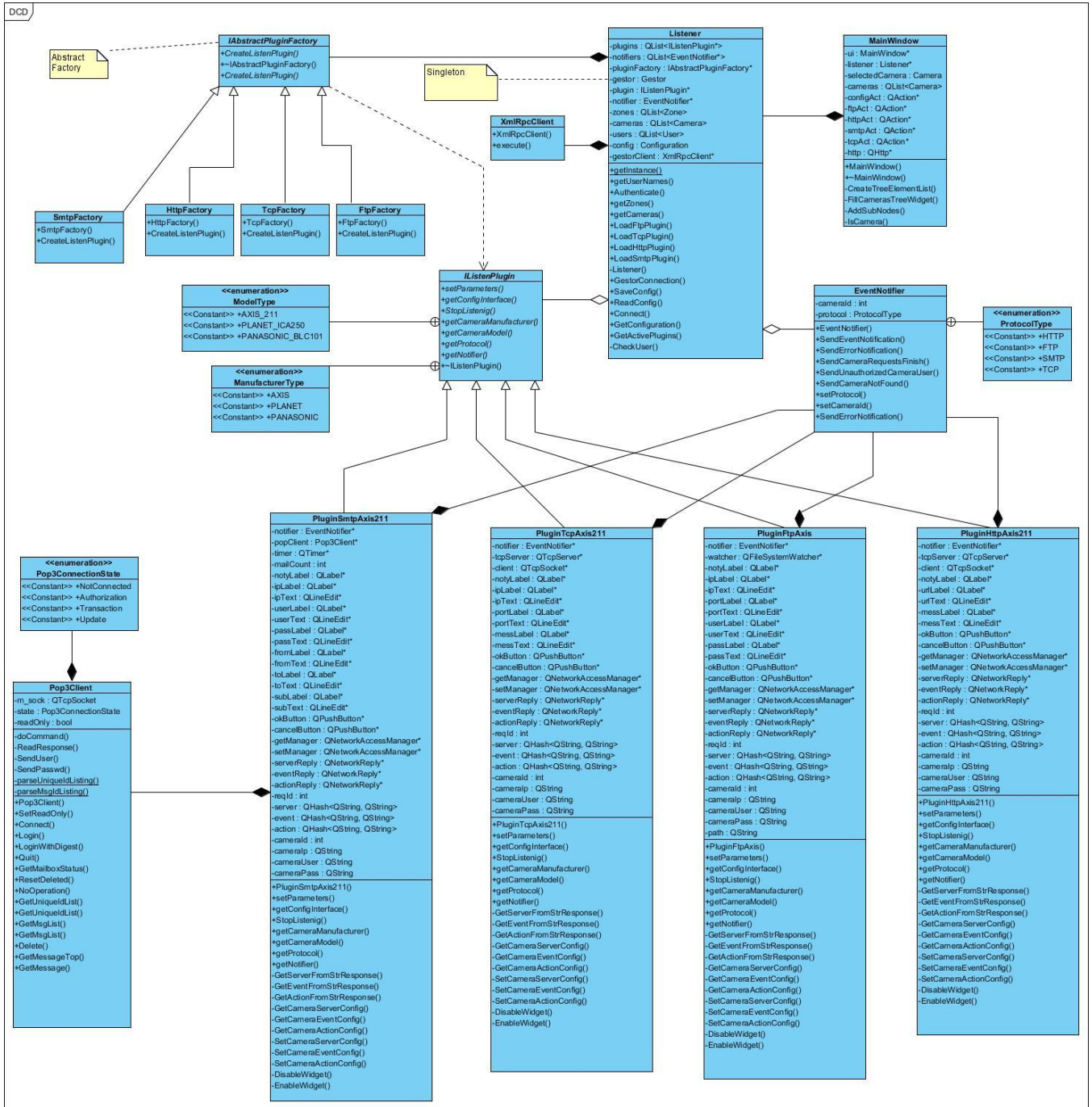


Figura 10: Diagrama de Clases del Diseño.

### **3.6 Diagramas de Interacción.**

Durante la ejecución de un programa orientado a objetos ocurren una serie de transiciones y llamadas a operaciones entre diferentes objetos mediante el envío de señales, así se puede describir de forma simple el funcionamiento de los casos de uso y de operaciones complejas. En la notación UML es mediante los diagramas de interacción que se muestran estas acciones. (27)

Los diagramas de interacción como su nombre lo indica son modelos que figuran la forma en que colaboran grupos de objetos. Usualmente captan el comportamiento de un solo caso de uso. Los diagramas de interacción están conformados por los diagramas de secuencia y los diagramas de colaboración.

#### **3.6.1 Diagramas de Colaboración.**

Los diagramas de colaboración (Anexo III) muestran como se comunican los objetos entre sí para llevar a cabo un proceso. Representan la secuencia de las operaciones a realizar para ilustrar la ejecución de un caso de uso, de forma más simple podría decirse que muestran como los objetos interactúan para lograr el comportamiento de un caso de uso.

Estos diagramas tienen una mayor utilidad para representar las interacciones entre un número no muy grande de objetos, de lo contrario el número de mensajes entre estos aumenta y el diagrama se hace engorroso y difícil de entender.

#### **3.6.2 Diagramas de Secuencia.**

A diferencia de los diagramas de colaboración, los diagramas de secuencia explican gráficamente las interacciones que tienen lugar entre las instancias de las clases del modelo, estos diagramas “representan explícitamente el orden en el tiempo y la duración de los mensajes y las operaciones que ponen en marcha.” (27)

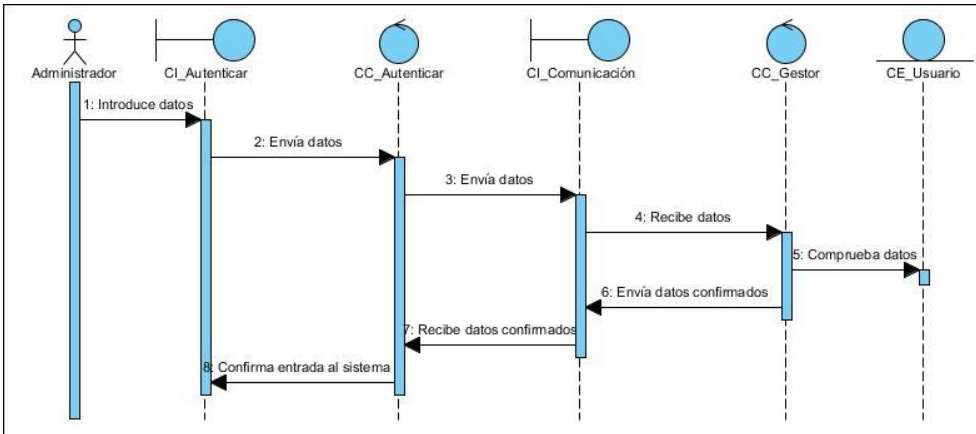


Figura 11: Diagrama de secuencia. Caso de Uso Autenticar Usuario.

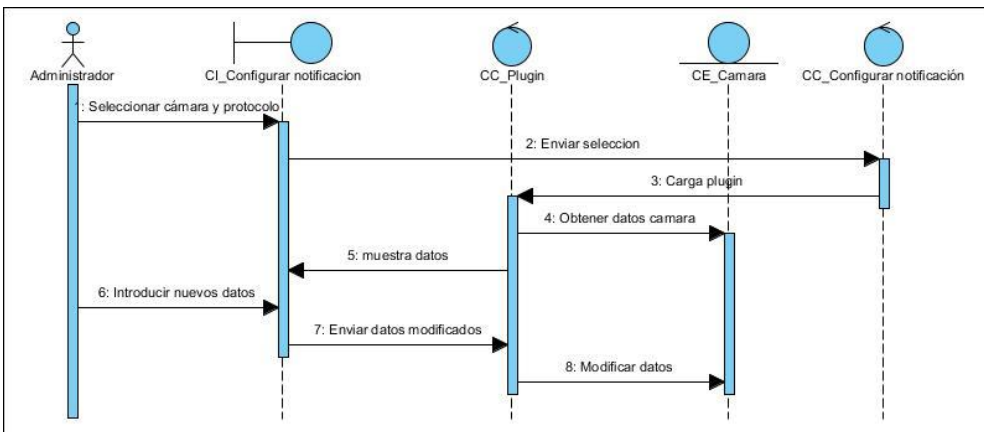


Figura 12: Diagrama de secuencia. Caso de Uso Configuran notificación de evento.

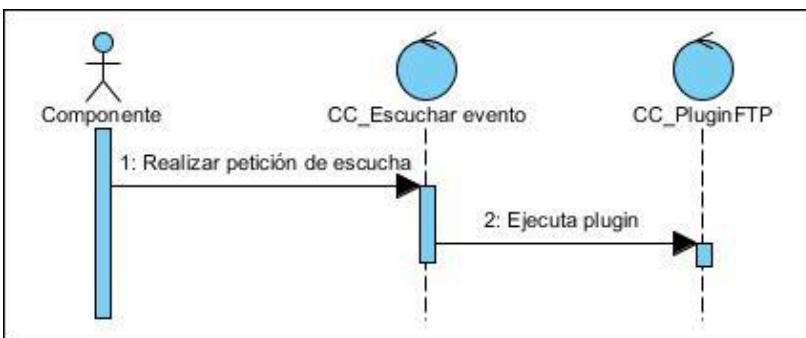


Figura 13: Diagrama de secuencia. Caso de Uso Escuchar evento por protocolo FTP.

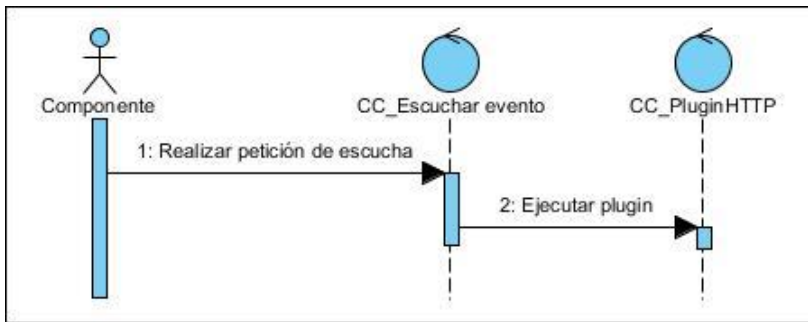


Figura 14: Diagrama de secuencia. Caso de Uso Escuchar evento por protocolo HTTP.

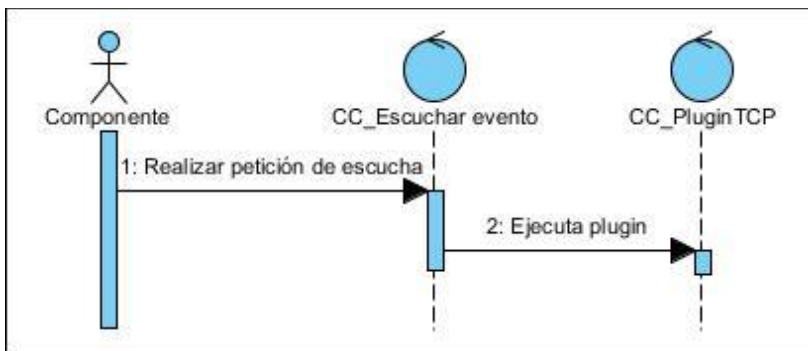


Figura 15: Diagrama de secuencia. Caso de Uso Escuchar evento por protocolo TCP.

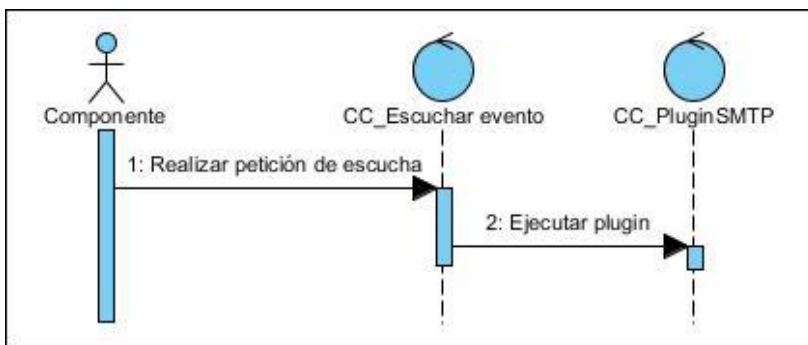


Figura 16: Diagrama de secuencia. Caso de Uso Escuchar evento por protocolo SMTP.

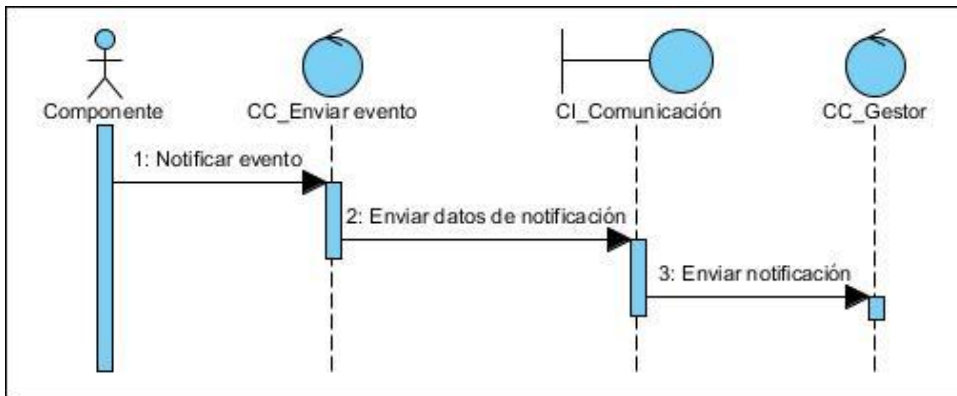


Figura 17: Diagrama de secuencia. Caso de Uso Enviar evento al Gestor.

### 3.7 Conclusiones Parciales.

En este capítulo se recoge el resultado del estudio realizado para definir la arquitectura y los patrones de diseño de software a utilizar en el diseño e implementación de la solución propuesta. Se obtuvo como resultado la selección de la arquitectura y patrones más adecuados, teniendo en cuenta las características de la aplicación a desarrollar. La modelación de los diagramas de interacción permite alcanzar una comprensión más detallada de los flujos de los procesos que tienen lugar en la ejecución de los casos de uso. Una vez cerrado el flujo de Análisis y Diseño se está en condiciones de comenzar la implementación del componente de control de conexiones de entrada/salida de las cámaras IP.

## Capítulo 4: Implementación y Prueba.

### 4.1 Introducción

El presente capítulo aborda los flujos de trabajo de implementación y prueba. Se desarrollan los diagramas correspondientes al modelo de implementación, diagrama de despliegue y de componentes respectivamente. Se describen las pruebas realizadas a la aplicación para corroborar si esta cumple el objetivo para el que fue desarrollada.

### 4.2 Modelo de Implementación.

El modelo de implementación describe como los elementos del modelo de diseño se implementan en términos de componentes, como ficheros de código fuente, ejecutables, etc. Describe también como se organizan los componentes en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados y además como dependen los componentes unos de otros. (29)

#### 4.2.1 Diagrama de Despliegue.

Los diagramas de despliegue representan a los nodos<sup>18</sup> y sus relaciones. Muestran las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. Unidos a los diagramas de componentes proveen la vista de implementación de un sistema. (31)

El diagrama de despliegue ilustra la composición física del sistema, incluyendo el hardware y sus relaciones. El diagrama de despliegue puede mostrar servidores, estaciones de trabajo, impresoras, etcétera. (13)

A continuación se muestra el diagrama de despliegue de la aplicación en cuestión.

---

<sup>18</sup>**Nodo:** Objeto físico que representa un recurso computacional.

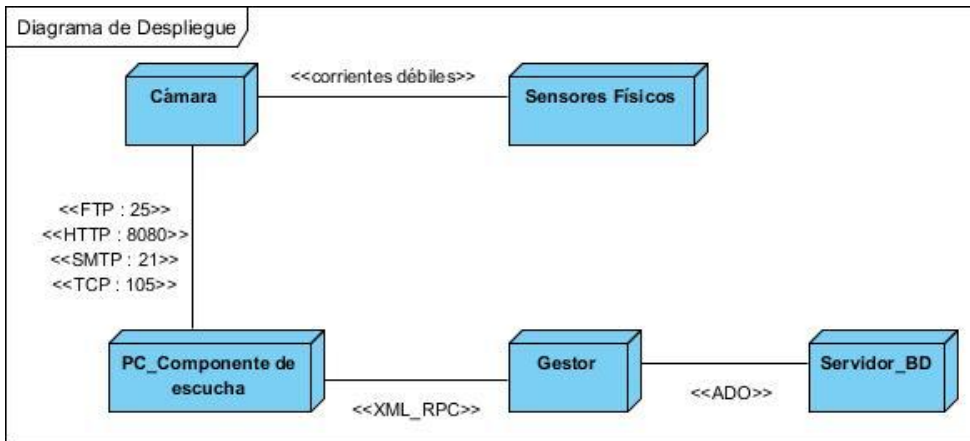


Figura 18: Diagrama de despliegue.

#### 4.2.2 Diagrama de Componentes.

El diagrama de componentes, como su nombre lo indica, muestra los componentes del sistema, como un archivo de clases, una base de datos y como se relacionan entre sí. (13) El diagrama de componentes describe la descomposición física del sistema de software en componentes a efectos de construcción y funcionamiento. La descomposición del diagrama de componentes se realiza en términos de componentes y de relaciones entre los mismos. (27)

La siguiente figura muestra el diagrama de componentes correspondiente a la aplicación.

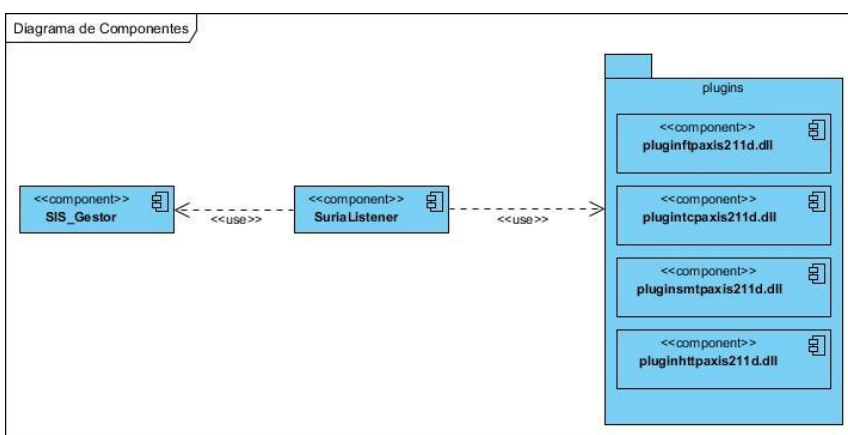


Figura 19: Diagrama de componentes.

### 4.3 Flujo de trabajo de pruebas.

Un aspecto fundamental durante el proceso de desarrollo de un software es la realización de las pruebas, pues mediante estas es posible verificar los resultados de la implementación. El objetivo de las pruebas es descubrir posibles errores en la aplicación antes de que la solución sea entregada a los usuarios finales. Existen dos métodos para la realización de pruebas, las pruebas de caja blanca o pruebas estructurales y las pruebas funcionales o de caja negra.

#### ➤ **Pruebas de caja blanca o pruebas estructurales.**

Con estas pruebas se realiza una cuidadosa evaluación de los detalles procedimentales. Se comprueban los caminos lógicos de la aplicación en base a inspeccionar pedazos específicos de la misma (bucles, sentencias de bifurcación, etc.) (32)

Estas pruebas utilizan el código fuente del programa, en especial su estructura de control, para seleccionar casos de prueba. Existen varias estrategias para casos de prueba a partir del código fuente. Por ejemplo (33):

**Cobertura de código:** Se realiza con el objetivo de que alguna propiedad o característica del código tome el mayor número posible de valores.

**Pruebas de bucles:** Se realizan pruebas en las que un bucle se ejecute ya sea una vez o un número determinado de veces.

**Flujo de datos:** En este caso una determinada variable o una estructura de datos toma diferentes valores.

#### ➤ **Pruebas funcionales o de caja negra.**

Las pruebas de caja negra se realizan sobre la interfaz del software con el objetivo de demostrar que este funciona correctamente. Este tipo de pruebas no toma en cuenta la estructura interna del software. (32)

Estas pruebas se concentran en los requisitos funcionales del software. Permiten al ingeniero de software utilizar conjuntos de condiciones de entrada que probarán por completo todos los requisitos funcionales de un programa. (34)



Estas pruebas tratan de encontrar errores en las siguientes categorías (34):

- Funciones incorrectas o faltantes.
- Errores de interfaz.
- Errores en estructuras de datos o en acceso a bases de datos externas.
- Errores de comportamiento o desempeño.
- Errores de inicialización y término.

#### **4.3.1 Pruebas realizadas. Resultados**

Para los casos de uso Autenticar usuario y Configurar notificación de evento se desarrollaron diseños de casos de pruebas del tipo Pruebas de Caja Negra o Funcionales, utilizando la técnica de partición equivalente. Esta técnica permite examinar los valores válidos e inválidos de las entradas de datos que existen en el software y la respuesta de la aplicación ante estas entradas. De esta forma se puede verificar si las funciones del software operan correctamente. Luego de aplicadas las pruebas a la aplicación se obtuvo como resultado que esta respondió como se esperaba ante las entradas de datos que se utilizaron.

A continuación se muestra el diseño de casos de prueba realizado para el caso de uso Configurar notificación de evento, específicamente para las secciones de configuración de notificación por el protocolo FTP y configuración de notificación por el protocolo HTTP. Para visualizar completamente los diseños de caso de prueba correspondientes a estos casos de uso remitirse a los Anexos del presente documento, específicamente el [Anexo IV](#).

- **Descripción general.**

El caso de uso se inicia cuando el administrador desea modificar los parámetros para notificar la escucha a través de alguno de los protocolos, finalizando el mismo cuando la configuración es aceptada.

- **Condiciones de ejecución.**

El administrador debe estar autenticado.

➤ **Secciones a probar en el Caso de Uso.**

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: FTP	EC 1.1: Configurar notificación por FTP con éxito.	El administrador selecciona la opción "Configurar escucha FTP". El sistema muestra las opciones de configuración. El administrador modifica los parámetros que desee y el sistema aplica los cambios realizados.
	EC 1.2: Configurar notificación por FTP sin éxito.	Si el administrador no introduce los parámetros de configuración en el formato adecuado o deja algún campo vacío, el sistema pondrá en color rojo el campo incorrecto y el botón Aceptar se mantendrá inhabilitado por tanto el sistema no aplica ninguna transformación.
SC 2: HTTP	EC 2.1: Configurar notificación por HTTP con éxito.	El administrador selecciona la opción "Configurar escucha HTTP". El sistema muestra las opciones de configuración. El administrador modifica los parámetros que desee y el sistema aplica los cambios realizados.
	EC 2.2: Configurar notificación por HTTP sin éxito.	Si el administrador no introduce los parámetros de configuración en el formato adecuado o deja algún campo vacío, el sistema pondrá en color rojo el campo incorrecto y el botón Aceptar se mantendrá inhabilitado por tanto el sistema no aplica ninguna transformación.

➤ **Descripción de variable.**

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	IP	Campo de Texto	No	Solo admite números y el carácter "."

2	Puerto	Campo de Texto	No	Solo admite números
3	Usuario	Campo de Texto	No	Solo admite caracteres alfanuméricos
4	Contraseña	Campo de Texto	No	Admite caracteres alfanuméricos y especiales
5	Url	Línea de Texto	No	Solo admite números y los caracteres “.”, “:”, “/”
6	Mensaje	Línea de Texto	No	Solo admite letras y números

➤ **Matriz de Datos**

**SC 1 <FTP >**

ID del escenario	Escenario	Variable 1 IP	Variable 2 Puerto	Variable 3 Usuario	Variable 4 Contraseña	Respuesta del Sistema	Resultado de la Prueba
EC 1.1	Configurar notificación por FTP con éxito.	V/ “10.8.46.190”	V/ “21” “19”	V/ “admin” “admin2”	V/ “adm1n” “admin+”	El administrador modifica los parámetros correctamente. El sistema aplica los cambios.	Satisfactoria
EC 1.2	Configurar notificación por FTP sin éxito.	I/ ”Nulo” “1o.8.46.. 190” “10.8.46.19o”	I/ ”Nulo” “2!”	I/ ”Nulo” “adm n”	I/ ”Nulo”	El administrador no modifica los parámetros correctamente. El	Satisfactoria

## SC 2 <HTTP >

ID del escenario	Escenario	Variable 1 Url	Variable 2 Mensaje	Respuesta del Sistema	Resultado de la Prueba
EC 2.1	Configurar notificación por HTTP con éxito.	V/ "http://10.8.46.190:8080"	V/ "mensaje HTTP" "mensaje HTTP 1"	El administrador modifica los parámetros de configuración correctamente. El sistema aplica los cambios.	Satisfactoria
EC 2.2	Configurar notificación por HTTP sin éxito.	I/"Nulo" "htt://10.8..46.190:8o80"	I/ "Nulo" "Mensaje+*"	El administrador no introduce los parámetros correctamente. El sistema no realiza ningún cambio	Satisfactoria

### 4.4 Conclusiones Parciales.

Este capítulo recoge los resultados arrojados luego de concluidos los flujos de implementación y pruebas. Con la realización del diagrama de componentes se logró una descomposición más detallada del componente desarrollado, ofreciendo una mayor comprensión de la estructura física que tiene la aplicación y las dependencias existentes entre los componentes que la integran. El diagrama de despliegue realizado tiene como resultado la obtención de una mejor visión de la distribución física del sistema. El estudio realizado a los tipos de pruebas permitió seleccionar aquella que se le aplicaría a la solución propuesta y mediante esta validar el cumplimiento de los requisitos funcionales establecidos.

## CONCLUSIONES GENERALES

Con el desarrollo del presente trabajo de diploma, se cumplió con las tareas y objetivos trazados. Lo que permitió llegar a las siguientes conclusiones:

- ✓ Se realizó un análisis sobre los principales sistemas de video vigilancia existentes en la actualidad. Las principales limitaciones estuvieron en que estas soluciones tienen para su desempeño gran dependencia del hardware del fabricante.
- ✓ Se analizaron y valoraron las principales tecnologías y herramientas actuales lo que permitió seleccionar las más adecuadas para el desarrollo de la aplicación garantizando de esta forma que el producto final tenga la calidad requerida.
- ✓ Se implementó un componente que brinda los servicios de escucha mediante los protocolos HTTP, SMTP, TCP y FTP para los eventos que tienen lugar a través de las conexiones de entrada/salida de las cámaras IP. Además de gestionar la configuración de las cámaras IP en cuanto a la notificación de eventos a través de los ya mencionados protocolos, cumpliendo de esta forma con las funcionalidades trazadas.

Se confirma haber dado cumplimiento al objetivo general trazado pues se logró el desarrollo del componente controlador de conexiones de entrada/salida de las cámaras IP en el sistema Suria Vision, garantizando para dicho sistema mayor número de funcionalidades y mejor funcionamiento a la hora de realizar las actividades de vigilancia.

## **RECOMENDACIONES**

Al finalizar el presente trabajo se recomienda:

- ✓ Enriquecer el componente con el desarrollo de plugins de escucha para otros modelos de cámaras.

## BIBLIOGRAFÍA

1 **PROYECTA CONSULTORES ASOCIADOS SAC** *SISTEMAS DE VIDEOVIGILANCIA Y SEGURIDAD REMOTA* Perú

2 **DATYS** *Xyma Safe Vision. Sistema de Video Vigilancia IP.* 2010

3 **AXIS Communications** *Antecedentes Corporativos de Axis* 2010

4. **Jamrich Parsons, June y Oja, Dan.** *Conceptos de Computación. Nuevas Perspectivas.* s.l. : Cengage Learning, 2008. 978-970-686-834-3.

5. **Herrera Pérez, Enrique.** *Tecnologías redes de transmisión de datos.* Mexico : Limusa, 2003. 968-18-6383-6.

6. Sabería. [En línea] [Citado el: 6 de 6 de 2012.] <http://www.saberia.com>.

7. Wikipedia. [En línea] [Citado el: 23 de 5 de 2012.] <http://es.wikipedia.org>.

8. EcuRed. [En línea] [Citado el: 23 de 5 de 2012.] <http://www.ecured.cu>.

9. **Axis.** *Axis Camera Station.* 2010.

10. Scati Video Management Systems. [En línea] [Citado el: 20 de noviembre de 2011.] <http://www.scati.com>.

11. **Datys.** *Xyma Safe Vision. Especificaciones Técnicas V2.8.5.* La Habana : s.n., 2012.

12. **Escribano, Gerardo Fernandez.** *Introducción a Extreme Programming.* 2002.

13. **Kenneth E. Kendall, Julie E. Kendall.** *Systems Analysis and design.* s.l. : Pearson Education, Inc, 2005.

14 *APLICACIÓN DE LA METODOLOGÍA RUP PARA EL DESARROLLO RÁPIDO DE APLICACIONES BASADO EN EL ESTANDAR J2EE* Guatemala 2006

15. **Alejandro Martines, Raul Martinez.** *Guía a Rational Unified Process.* España : s.n.

16. **Sommerville, Ian.** *Ingeniería del Software*. 2005.
17. msdn.microsoft. [En línea] [Citado el: 22 de noviembre de 2011.] <http://msdn.microsoft.com/en-us/library/ms973864.aspx>.
18. **Pedro Orlando Acosta Pereira, Danieyis Santiago Marrero.** *Desarrollo del Módulo Web de Monitorización y Administración del Sistema de Video Vigilancia*. La Habana : s.n., 2011.
19. **Simon St. Laurent, Joe Johnston, Edd Dumbill.** *Programming Web Services with XML-RPC*. s.l. : O'Reilly, 2001.
20. ciberneta. [En línea] [Citado el: 29 de 5 de 2012.] <http://www.ciberneta.com>.
21. IBM. [En línea] [Citado el: 29 de 5 de 2012.] <http://www.ibm.com>.
22. **Mora, Sergio Luján.** *C++ paso a paso*.
23. **Francisco Javier Bueno Martín, M<sup>a</sup> Ángeles Cambrero Sánchez.** *INTRODUCCIÓN AL COMPILADOR Y PREPROCESADOR DE C# EN VISUAL STUDIO.NET*. Mayo, 2003.
24. **Nokia Corporation.** qt.nokia. [En línea] [Citado el: 30 de 1 de 2012.] <http://qt.nokia.com/>.
25. **Zhao, Ming y Shen, YuMing.** The Application of Qt in Liquid Level Detection. [aut. libro] Liangzhong Jiang. *Proceedings of the 2011 international Conference on informatics, Cybernetics and Computer Engineering(ICCE2011) Noviembre 19-20, 2011, Melbourne, Australia*. 2011.
26. **Sparx Systems.** *Guía de Usuario de Enterprise Architect 7.0*. 2007.
27. **Campderrich Falgueras, Benet.** *Ingeniería del software*. Barcelona : Editorial UOC, 2003. 84-8318-997-6.
28. **Semanat Aldana, Edmis Deivis y Verdecia Four, Leonor.** *Sistema de Video Vigilancia*. La Habana : s.n., 2009.
29. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El Proceso Unificado de Desarrollo de Software*. Madrid : Pearson Educación, 2000. 84-7829-036-2.
30. **Visconti, Marcello y Astudillo, Hernán.** *Fundamentos de Ingeniería de Software*.



## Bibliografía y Referencias

---

31. EcuRed. [En línea] [Citado el: 10 de mayo de 2012.] <http://www.ecured.cu>.
32. **Alonso, Fernando, Martinez, Loic y Fco. Javier, Segovia.** *INTRODUCCIÓN A LA INGENIERÍA DEL SOFTWARE. Modelos de desarrollo de programas.* s.l. : Publicaciones DELTA, 2005. 84-96477-00-2.
33. **Tuya, Javier, Ramos Román, Isabel y Dolado Cosín, Javier.** *TÉCNICAS CUANTITATIVAS PARA LA GESTIÓN EN LA INGENIERIA DEL SOFTWARE.* s.l. : NETBIBLIO, S. L., 2007. 978-84-9745-204-5.
34. **Pressman, Roger S.** *Ingenieria del Software. Un enfoque práctico.* s.l. : McGraw-hill Interamericana, 2005. 0072853182.
37. Zone Minder. [En línea] [Citado el: 17 de 1 de 2012.] <http://www.zoneminder.com>.
38. Cisco. [En línea] [Citado el: 17 de 1 de 2012.] <http://www.cisco.com>.
39. **Herranz, Arantxa.** Video Vigilancia IP. *PCWORLD PROFESIONAL.* [En línea] [Citado el: 23 de Noviembre de 2011.] <http://www.idg.es>
41. Scati Labs - Wikipedia, la enciclopedia libre. *Wikipedia, la enciclopedia libre.* [En línea] [Citado el: 25 de Noviembre de 2011.] <http://es.wikipedia.org>
44. Microsoft Visual Studio - Wikipedia, la enciclopedia libre. *Wikipedia, la enciclopedia libre.* [En línea] [Citado el: 25 de Noviembre de 2011.] <http://es.wikipedia.org>
45. Información general de .NET Framework Remoting. *Visual Studio.* [En línea] [Citado el: 25 de Noviembre de 2011.] <http://msdn.microsoft.com>
49. **Wikipedia.** Programación Extrema. *Wikipedia La Enciclopedia Libre.* [En línea] [Citado el: 29 de Noviembre de 2011.] <http://es.wikipedia.org/>
50. **Axis Communications.** *Coste total de propiedad(TCO) Comparativa entre los sistemas de vigilancia con cámaras IP y cámaras analógicas.* s.l. : Axis Communications, 2008.
51. **Axis Communications.** Axis Communications. *Axis Communications.* [En línea] 21 de 11 de 2011. [Citado el: 21 de 11 de 2011.] <http://www.axis.com>.
52. **Wikipedia.** Proceso Unificado de Rational. *Wikipedia.* [En línea] [Citado el: 29 de Noviembre de 2011.] <http://es.wikipedia.org>

## REFERENCIAS BIBLIOGRÁFICAS

1 **PROYECTA CONSULTORES ASOCIADOS SAC SISTEMAS DE VIDEOVIGILANCIA Y SEGURIDAD REMOTA** Perú

2 **DATYS Xyma Safe Vision. Sistema de Video Vigilancia IP.** 2010

3 **AXIS Communications Antecedentes Corporativos de Axis** 2010

4. **Jamrich Parsons, June y Oja, Dan.** *Conceptos de Computación. Nuevas Perspectivas.* s.l. : Cengage Learning, 2008. 978-970-686-834-3.

5. **Herrera Pérez, Enrique.** *Tecnologías redes de transmisión de datos.* Mexico : Limusa, 2003. 968-18-6383-6.

6. Sabería. [En línea] [Citado el: 6 de 6 de 2012.] <http://www.saberia.com>.

7. Wikipedia. [En línea] [Citado el: 23 de 5 de 2012.] <http://es.wikipedia.org>.

8. EcuRed. [En línea] [Citado el: 23 de 5 de 2012.] <http://www.ecured.cu>.

9. **Axis.** *Axis Camera Station.* 2010.

10. Scati Video Management Systems. [En línea] [Citado el: 20 de noviembre de 2011.] <http://www.scati.com>.

11. **Datys.** *Xyma Safe Vision. Especificaciones Técnicas V2.8.5.* La Habana : s.n., 2012.

12. **Escribano, Gerardo Fernandez.** *Introducción a Extreme Programming.* 2002.

13. **Kenneth E. Kendall, Julie E. Kendall.** *Systems Analysis and design.* s.l. : Pearson Education, Inc, 2005.

14 **APLICACIÓN DE LA METODOLOGÍA RUP PARA EL DESARROLLO RÁPIDO DE APLICACIONES BASADO EN EL ESTANDAR J2EE** Guatemala 2006

15. **Alejandro Martines, Raul Martinez.** *Guía a Rational Unified Process.* España : s.n.

16. **Sommerville, Ian.** *Ingeniería del Software*. 2005.
17. msdn.microsoft. [En línea] [Citado el: 22 de noviembre de 2011.] <http://msdn.microsoft.com/en-us/library/ms973864.aspx>.
18. **Pedro Orlando Acosta Pereira, Danieyis Santiago Marrero.** *Desarrollo del Módulo Web de Monitorización y Administración del Sistema de Video Vigilancia*. La Habana : s.n., 2011.
19. **Simon St. Laurent, Joe Johnston, Edd Dumbill.** *Programming Web Services with XML-RPC*. s.l. : O'Reilly, 2001.
20. ciberneta. [En línea] [Citado el: 29 de 5 de 2012.] <http://www.ciberneta.com>.
21. IBM. [En línea] [Citado el: 29 de 5 de 2012.] <http://www.ibm.com>.
22. **Mora, Sergio Luján.** *C++ paso a paso*.
23. **Francisco Javier Bueno Martín, M<sup>a</sup> Ángeles Cambrero Sánchez.** *INTRODUCCIÓN AL COMPILADOR Y PREPROCESADOR DE C# EN VISUAL STUDIO.NET*. Mayo, 2003.
24. **Nokia Corporation.** qt.nokia. [En línea] [Citado el: 30 de 1 de 2012.] <http://qt.nokia.com/>.
25. **Zhao, Ming y Shen, YuMing.** The Application of Qt in Liquid Level Detection. [aut. libro] Liangzhong Jiang. *Proceedings of the 2011 international Conference on informatics, Cybernetics and Computer Engineering(ICCE2011) Noviembre 19-20, 2011, Melbourne, Australia*. 2011.
26. **Sparx Systems.** *Guía de Usuario de Enterprise Architect 7.0*. 2007.
27. **Campderrich Falgueras, Benet.** *Ingeniería del software*. Barcelona : Editorial UOC, 2003. 84-8318-997-6.
28. **Semanat Aldana, Edmis Deivis y Verdecia Four, Leonor.** *Sistema de Video Vigilancia*. La Habana : s.n., 2009.
29. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El Proceso Unificado de Desarrollo de Software*. Madrid : Pearson Educación, 2000. 84-7829-036-2.
30. **Visconti, Marcello y Astudillo, Hernán.** *Fundamentos de Ingeniería de Software*.

## *Bibliografía y Referencias*

---

31. EcuRed. [En línea] [Citado el: 10 de 5 de 2012.] <http://www.ecured.cu>.
32. **Alonso, Fernando, Martinez, Loic y Fco. Javier, Segovia.** *INTRODUCCIÓN A LA INGENIERÍA DEL SOFTWARE. Modelos de desarrollo de programas.* s.l. : Publicaciones DELTA, 2005. 84-96477-00-2.
33. **Tuya, Javier, Ramos Román, Isabel y Dolado Cosín, Javier.** *TÉCNICAS CUANTITATIVAS PARA LA GESTIÓN EN LA INGENIERIA DEL SOFTWARE.* s.l. : NETBIBLIO, S. L., 2007. 978-84-9745-204-5.
34. **Pressman, Roger S.** *Ingenieria del Software. Un enfoque práctico.* s.l. : McGraw-hill Interamericana, 2005. 0072853182.

## **GLOSARIO DE TÉRMINOS**

**Aplicación:** Es un programa informático que provee al usuario una serie de funcionalidades que le facilitan la realización de una determinada actividad.

**CCTV:** Tecnología de video vigilancia donde todos los componentes se encuentran enlazados conformando la infraestructura del sistema. En estos sistemas la información es compartida solo entre los componentes del sistema.

**Componente:** Es un elemento reutilizable que puede interoperar con otros módulos de software por medio de sus interfaces.

**Framework:** Marco de trabajo. Herramienta informática, está compuesto por bibliotecas y un lenguaje interpretado que facilitan el desarrollo de un proyecto informático.

**FTP:** File Transfer Protocol o Protocolo de Transferencia de Archivos. Protocolo para el intercambio de archivos en internet.

**GUI:** Graphic User Interface o Interfaz Gráfica de Usuario. Sistema de interacción entre el usuario y una aplicación. Se distingue por la utilización de iconos y elementos gráficos.

**GPL:** La Licencia Pública General de GNU o más conocida por su nombre en inglés General Public License (GPL).

**Herramienta CASE:** Programas informáticos que facilitan la automatización del ciclo de vida de desarrollo de un software.

**HTTP:** Hyper Text Transfer Protocol o Protocolo de Transferencia de Hipertexto. Protocolo más común de intercambio de información en la red.

**IDE:** Entorno de Desarrollo Integrado. Programa informático compuesto por un conjunto de herramientas para facilitar a los desarrolladores el trabajo con los lenguajes de programación.

**Interfaz:** Medio de conexión entre aplicaciones, o entre un usuario y una aplicación. Apariencia externa de una aplicación informática.

**LGPL:** Licencia Pública General Reducida de GNU, o más conocida por su nombre en inglés GNU Lesser General Public License (antes GNU Library General Public License o Licencia Pública General para Bibliotecas de GNU) creada por la Free Software Foundation.

**Librería:** Colección de subprogramas utilizados para desarrollar soluciones de software.

**Módulo:** Parte de un programa que se encarga de realizar una o varias de las funcionalidades que debe ejecutar el sistema en general.

**Nodo:** Elemento físico con capacidad de cómputo.

**Plataforma:** Hardware o software sobre el cual puede ejecutarse o desarrollarse un software.

**Patrón:** Conjunto de reglas, pasos o acciones que definen la solución a un determinado problema.

**Sistema:** Conjunto de partes que se comunican entre sí de forma tal que cumplan un determinado objetivo.

**SMTP:** Simple Mail Transfer Protocol - Protocolo Simple de Transferencia de Correo. Protocolo utilizado para el intercambio de mensajes de correo electrónico.

**TCP:** Transmission Control Protocol o Protocolo de Control de Transmisión. Estándar muy extendido para la transmisión de datos en la red.

**UML:** Unified Modeling Language o Lenguaje Unificado de Modelado. Lenguaje de modelado para sistemas de software.

**XML:** Extensible Markup Language o lenguaje de marcado ampliable o extensible. Es un metalenguaje extensible de etiquetas que presenta un estándar para el intercambio de información estructurada entre diferentes plataformas. Puede ser utilizado en bases de datos, editores de texto, hojas de cálculo, entre otros.