Universidad de las Ciencias Informáticas

Facultad 6

Desarrollo del Módulo Web de Monitorización y Administración del Sistema de Video Vigilancia

Trabajo de Diploma para optar por el Título de

Ingeniero en Ciencias Informáticas

Autores: Danieyis Santiago Marrero

Pedro Orlando Acosta Pereira

Tutor: Ing. Heliodoro Rodríguez Milián

La Habana, 12 de junio de 2011

"Año 53 de la Revolución"

"La inteligencia humana tiene como leyes la investigación y el análisis."

"La inteligencia no es la facultad de imponerse; es el deber de ser útil a los demás."

José Martí

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año 2011.

Autores:

_____ _____

Pedro Orlando Acosta Pereira Danieyis Santiago Marrero

Tutor:

_____

Heliodoro Rodríguez Milián

DATOS DE CONTACTO

TUTOR: Ing. Heliodoro Rodríguez Milián (hrodriguezm@uci.cu)

Profesor instructor graduado de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas (UCI). Imparte la asignatura de Historia de la Informática. Actualmente se desempeña como Líder del proyecto de Video Vigilancia.

DEDICATORIA

A mis padres, por todo el apoyo, el amor y el espíritu de lucha que me entregaron.

A mi hermana, por estar ahí y darme su ayuda incondicional en todo momento.

A Jessica, por entregarme su fuerza y no permitirme caer ni renunciar bajo ninguna condición.

Pedro Orlando Acosta Pereira

A mi hermano Ángel Omar y a mi primita Shanik que son las personitas que más quiero en este mundo.

A mi mamá y mi papá por apoyarme siempre en todo.

A toda mi familia por hacerme tan feliz.

Danieyis Santiago Marrero

AGRADECIMIENTOS

A mis padres, por las noches de desvelo, los buenos y malos momentos y por darme fuerzas para seguir y no desfallecer.

A mi hermana y su familia, por cuidarme, apoyarme y comprenderme siempre.

A la Revolución y al Comandante en Jefe Fidel Castro, porque sin su obra mis pasos por la UCI no hubieran sido posibles.

A la UCI y todo el Universo que engloba porque, entre tragos dulces y amargos, me ha convertido en un ser humano mejor y me ha entregado momentos inolvidables.

A todos los que de una forma u otra, permitieron que este trabajo fuera posible y quienes, con mayor o menor fuerza, aportaron su granito arena por ayudar.

A nuestro tutor Heliodoro por confiar en nosotros hasta el último momento y demostrarnos como ser mejores.

A los profesores que entregaron su tiempo y su conocimiento y quienes con sus enseñanzas permitieron que este trabajo se lograse con calidad.

A mis amigos y compañeros, por apoyarme cuando los he necesitado y darme razones para ser mejor.

A todos gracias,

Pedro Orlando Acosta Pereira

Mis más sinceros agradecimientos a todas aquellas personas que me han ayudado a lo largo de mi vida como estudiante. A mis maestros, que se han esforzado por darme una buena formación profesional. A mi familia que siempre me ha apoyado y motivado al estudio y los más especiales a mi mamá y mi papá que se han esforzado junto a mí para lograr este sueño.

Me siento feliz por haber escogido estudiar en esta universidad y haberlo logrado, en este tiempo aquí he conocido a muchas personas especiales que quisiera agradecerles por formar parte de mi vida universitaria.

A Manuel Alejandro, por ser mi amigo en todas, por regañarme, aconsejarme y aguantar mis pesadeces de vez en cuando y por ayudarme a ser una mejor persona.

A Pavel, que hicimos muy buena amistad y me apoyó mucho en el estudio para la Prueba de Nivel.

A Pedro, que además de ser mi compañero de tesis ha sido mi amigo incondicional porque ha estado muy cerca en momentos malos de mi vida este año.

A Yasnary y José Ángel, que han sido más que amigos, siempre han estado ahí para cualquier problema, hemos estudiado juntos, nos hemos peleados, hemos discutidos, pero ahí estamos. Son y seguirán siendo muy importantes en mi vida.

A Yiyi, que aunque solo hace pelearme nos queremos mucho y estas vacaciones sí voy a Granma.

A Dayana que este año me ha dado su apoyo incondicional y me ha sido de mucha ayuda.

A Elizabetha que he tenido que luchar con ella estos 5 años, pero es una excelente amiga.

A Rafael que en poco tiempo ha llegado a ser alguien muy importante para mí y que me ha ayudado mucho.

También muchas otras personas que me han ayudado a Susana, Lisbet, Ibelin, Alejandro, Yoendris, a todos los de mi proyecto que me he sentido muy bien este tiempo que he sido parte de ese colectivo, a mi compañeras de apartamento y a los Killers.

No puedo hablar de mi vida universitaria sin mencionar a mi familia de aquí de La Habana que los quiero mucho. A mi tía, que me ha tratado como a una hija, a mis primos Keniel y Keyli que también me quieren mucho y a mi tío. A Dennis, que también conté con su apoyo en una etapa de mi vida, a su abuelita que me tiene mucho cariño y a su mamá.

En la realización de este Trabajo de Diploma nos han apoyado mucho todos los profesores del Proyecto Video Vigilancia, el tribunal con sus revisiones en cada corte y a nuestro tutor que ha luchado con nosotros desde el principio.

Danieyis Santiago Marrero

RESUMEN

En la actualidad, la video vigilancia se ha convertido en un componente clave de los sistemas de seguridad. Con los Sistemas de Video Vigilancia es posible dar un seguimiento visual, en tiempo real, a un determinado lugar o a varios lugares simultáneamente, crear un registro grabado de eventos sucedidos y proteger así: bienes, inmuebles, instituciones y personas. Estos sistemas tienen una gran demanda en el mundo y a medida que avanza la tecnología se amplían sus funcionalidades y aumenta su complejidad. De modo que, actualmente, se puede controlar un conjunto de cámaras desde de una estación de monitorización física o incluso desde estaciones móviles.

El presente trabajo tiene como objetivo implementar una estación de monitorización de video vigilancia en plataforma web, para manipular la visualización de los flujos de videos de un Sistema de Video Vigilancia desde cualquier sitio con acceso a la Web.

Como resultado se obtendrá una aplicación que permita mejorar la seguridad en las instituciones del país. Esto facilitaría en gran medida el control y vigilancia de las principales áreas de las entidades donde se despliegue dicho sistema.

ABSTRACT

Today, video surveillance has become a key component of security systems. Video surveillance systems provides a visual tracking, at real time, to a place or several places simultaneously, creating a record of success and protecting assets, properties, institutions and others. Today, these systems are in great demand and as technology advances will extend their functionality and increases the complexity of security systems. So, nowadays, is possible to control a set of cameras from a physical monitoring station or even from mobile stations.

The present work aims to implement a monitoring station for video surveillance at Web platform, to manipulate the display of video streams from a video surveillance system from anywhere with Web access.

The result is an application that will improve security in the country's institutions. This would greatly facilitate the control and monitoring of the main areas of the institutions where the system is deployed.

Índice

Introducción *1*

CAPÍTULO 1. Fundamentación Teórica. Tendencias y tecnologías para el desarrollo del software *5*

1.1. Términos asociados **5**

1.2. La Tecnología IP y la Video Vigilancia **6**

1.2.1. Ventajas de la tecnología IP en los Sistemas de Video Vigilancia *6*

1.3. Soluciones existentes. Estado del Arte **7**

1.4. Metodologías, herramientas y tecnologías a utilizar para el desarrollo de la solución **10**

1.4.1. Tecnologías a usar para el desarrollo del sistema *10*

1.4.2. Herramientas *13*

1.4.3. Tecnologías a usar para la comunicación y desarrollo de la interfaz de usuario *17*

1.4.4. Tecnologías a emplear para la visualización de los flujos de video *20*

1.4.5. Metodologías *22*

1.5. Conclusiones **26**

CAPÍTULO 2. Presentación de la solución propuesta *27*

2.1. Modelo de dominio **27**

2.1.1. Conceptos Fundamentales *27*

2.2. Propuesta del Sistema **29**

2.3. Requerimientos Funcionales del Sistema **31**

2.4. Requerimientos No Funcionales del Sistema **33**

2.4.1. Requerimientos de rendimiento *33*

2.4.2. Requisitos de software *33*

2.4.3. Requisitos de hardware *33*

2.4.4. Restricciones de diseño. *33*

2.4.5. Requisitos de apariencia o interfaz externa *33*

2.4.6. Requerimientos de seguridad 34
2.4.7. Requerimientos de usabilidad 34
2.4.8. Requerimientos de soporte 34
2.5. Definición de los Casos de Uso 35
2.5.1. Definición de los actores 35
2.5.2. Listado de los Casos de Uso 35
2.6. Diagrama de Casos de Uso 36
2.7. Especificación de los Casos de Uso 37
2.8. Conclusiones 49
CAPÍTULO 3. Construcción de la solución propuesta 50
3.1. Arquitectura 50
3.2. Patrones de Diseño de Software 52
3.3. Flujo de Trabajo Análisis y Diseño 53
3.3.1. Diagramas del análisis 53
3.3.2. Diagramas del diseño 55
3.4. Flujo de trabajo Implementación 62
3.5. Modelo de datos 62
3.6. Modelo de implementación 65
3.6.1. Diagrama de despliegue 65
3.6.2. Diagrama de componentes. 66
3.7. Flujo de trabajo de Pruebas 67
3.7.1. Pruebas de Caja Blanca o Pruebas estructurales 67
3.7.2. Pruebas de Caja Negra 70
3.8. Conclusiones 73
Conclusiones 74

Recomendaciones *75*

Referencias Bibliográficas *76*

Bibliografía *77*

Índice de Tablas

Tabla 1. Actores definidos del sistema..... 35
Tabla 2. Prueba de Caja Negra CU Gestionar Usuario..... 72

Índice de Imágenes

Figura 1. Propuesta de un Sistema de Video Vigilancia IP 6
Figura 2. Diagrama del Modelo de dominio 29
Figura 3. Diagrama de Casos de Uso del Sistema 36
Figura 4. Arquitectura en Pizarra 50
Figura 5. Patrón Modelo-Vista-Controlador (MVC) 51
Figura 6. Diagrama de análisis. CU Autenticar Usuario. 54
Figura 7. Diagrama de análisis. CU Visualizar Flujo de Video 54
Figura 8. Diagrama de análisis. CU Gestionar Cámara 55
Figura 9. Diagrama de clases de diseño. CU Autenticar Usuario 56
Figura 10. Diagrama de clases de diseño. Adicionar cámara 57
Figura 11. Diagrama de clases de diseño. Configurar cámara 58
Figura 12. Diagrama de secuencia. CU Autenticar Usuario 59
Figura 13. Diagrama de secuencia. Adicionar cámara 60
Figura 14. Diagrama de secuencia. Configurar cámara 61
Figura 15. Modelo de datos. Tablas para los registros del sistema 62
Figura 16. Modelo de datos. Tablas del sistema de seguridad, tabla "tbl_eventlog" para los registros de eventos y tabla "tbl_camara" 63
Figura 17. Modelo de datos. Tablas para el sistema de permisos y tabla "tbl_video" 64
Figura 18. Diagrama de despliegue 65
Figura 19. Diagrama de componentes 66

INTRODUCCIÓN

Desde la antigüedad, el hombre se ha preocupado por su seguridad y la de sus bienes, esto le permitía un mayor desarrollo como ser social. En la década de los 70, con el surgimiento de los Sistemas de Video Vigilancia, se logra un avance decisivo que apoya la seguridad de determinadas áreas importantes para las grandes empresas y negocios existentes. Estos sistemas en sus inicios, fueron pensados solamente con fines policiales y para la seguridad de propiedades estatales. Comenzaron empleando tecnología analógica pero su implantación era algo costosa y además eran basados en la fiabilidad humana, por lo que su uso no era muy común en estos años.

La actividad de video vigilancia sigue siendo hoy un componente clave de la seguridad y de la organización de muchas entidades. Ha demostrado su valor y beneficios al brindar un seguimiento en tiempo real del entorno de una instalación, las personas y los activos, también al posibilitar eventos de grabación para una posterior investigación y al disminuir notablemente los riesgos de hechos delictivos. En 1996, Axis[1] muestra al mundo la primera solución de video vigilancia utilizando una cámara de red. Esta solución rompió con las barreras de la video vigilancia tradicional e hizo que estos sistemas se hicieran mucho más comunes y populares en la sociedad. La demanda social produjo un enriquecimiento del mercado con potentes soluciones basadas en esta tecnología, las cuales además de ofrecer más funcionalidades y mayor calidad en la visualización de video, eran de costo asequible, por lo que su uso se extendió a lugares como: escuelas, oficinas, lugares públicos y hogares.

Nuestro país también ha mejorado su seguridad con las ventajas de la video vigilancia, y existen sistemas como Xyma Safe Vision que están instalados actualmente en lugares públicos, hoteles, tiendas y otros entornos. Este sistema realiza la vigilancia a un área determinada a través de cámaras de video IP[2] o analógicas conectadas a servidores IP. Para estos fines, el proyecto Video Vigilancia, del Centro de desarrollo GEYSED[3], de la facultad 6, de la Universidad de las Ciencias Informáticas (UCI), desarrolló el producto Suria Vision, el cual tiene un módulo Visor que es una aplicación de escritorio y permite visualizar, administrar y monitorear los flujos de video de las

[1] **Axis**: Compañía líder a nivel mundial en la producción de sistemas de video vigilancia integrados.

[2] **IP (*Internet Protocol*):** Protocolo empleado para la comunicación a través de una red.

[3] **GEYSED:** Centro de desarrollo de Geoinformática y Señales Digitales.

cámaras de un Sistema de Video Vigilancia, incluyendo además: eventos, reportes y configuraciones relacionadas con el sistema.

Esta aplicación ofrece varias funcionalidades útiles para el cliente como la gestión de usuario, roles y privilegios, además de la gestión y visualización de cámaras IP, que pueden ser explotadas desde una estación de monitorización de un sistema de video vigilancia, pero para emplearlas es necesaria la instalación del software y las configuraciones pertinentes al mismo de acuerdo con la institución en que se instaure el sistema. Conociendo esto, surgen las siguientes preguntas:

¿Qué sucedería si se quiere acceder a los flujos de video que ofrece la aplicación desde una computadora que no se encuentre contemplada dentro del Sistema de Video Vigilancia? ¿Cómo se podrían visualizar y monitorear los flujos de video, tal como lo logra el Suria Vision, si la computadora está ubicada fuera de la red de la institución que controla el Sistema de Video Vigilancia? Para responder a la primera pregunta, el procedimiento a seguir sería instalar el Suria Vision, y configurar el sistema según la estructura y los recursos tecnológicos de la institución. En el caso de la segunda pregunta, con las funcionalidades implementadas hasta la actualidad en el Suria Vision, no se puede acceder a la estación de monitorización desde fuera de la red local.

La situación descrita anteriormente lleva a plantearse el siguiente **problema a resolver**: ¿Cómo facilitar el acceso a los flujos de video, eventos y configuraciones del sistema Suria Vision desplegado, para obtener un Sistema de Video Vigilancia funcional desde cualquier lugar con acceso a la Web, sin ser necesaria la instalación del software? Para llevar un estudio profundo de la problemática y desarrollar la posible investigación se define como **objeto de estudio:** la informatización de la video vigilancia y específicamente como **campo de acción:** los procesos de visualización de los flujos de video, manipulación de eventos y administración de usuarios como parte de un sistema de video vigilancia sobre plataforma web.

Para limitar el alcance de la investigación y darle solución al problema planteado se concreta como **objetivo general:** “Desarrollar una aplicación sobre plataforma web para la monitorización de los flujos de video, la visualización de eventos, la gestión de cámaras y administración de usuarios del Sistema de Video Vigilancia Suria Vision.” Como **idea a defender** se plantea que: “El desarrollo de una aplicación web como módulo integrado del Sistema Suria Vision, permite la monitorización de los flujos de video, la visualización de eventos, así como la gestión de cámaras y usuarios del sistema, desde cualquier sitio con acceso a la Web.”

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

1. Realizar un estudio del arte de los sistemas web de video vigilancia existentes en la actualidad que permitan la visión de múltiples flujos de video en tiempo real.
2. Investigar sobre las herramientas y tecnologías existentes para la realización de la aplicación sobre plataforma web.
3. Seleccionar las mejores herramientas y tecnologías para el diseño y la implementación del sistema.
4. Efectuar el levantamiento de requisitos del sistema.
5. Realizar el Modelo de dominio, de Casos de Uso, análisis y diseño del sistema.
6. Realizar el diseño de la aplicación web con una interfaz amigable y entendible para el usuario.
7. Implementar los casos de uso definidos.
8. Realizar las pruebas de unidad al sistema.

Para dar cumplimiento al grupo de tareas planteadas servirán de ayuda varios **métodos científicos de investigación**, tanto teóricos como empíricos.

Métodos teóricos:

1. Analítico-Sintético: este método permitirá estudiar y llegar a una conclusión o un tipo de resumen de lo estudiado, se utilizará para estudiar 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: este método permitirá estudiar los sistemas existentes de video vigilancia, sus antecedentes y su evolución a lo largo de la historia.
3. Modelación: mediante este método se realizarán los modelos que especifican como se va a desarrollar la aplicación y que permitirán un mejor entendimiento a los desarrolladores para una posterior implementación.

Métodos empíricos:

1. Entrevistas: se realizan entrevistas con el objetivo de entender el funcionamiento del sistema que se va a desarrollar y para sacar los requisitos funcionales que debe cumplir la aplicación. Se toma como muestra los desarrolladores del Módulo Visor y Gestor del Suria Vision, 2 profesores y 1 estudiante, los cuales tienen conocimiento de las necesidades del cliente desde el inicio del desarrollo del proyecto.

El siguiente trabajo de diploma estará dividido en 3 capítulos fundamentales en los cuales se abordará todo el contenido necesario para concluir la investigación:

Capítulo 1: Fundamentación teórica. Tendencias y tecnologías para el desarrollo del software. En este capítulo, se fundamentan términos técnicos de importancia para la investigación. Se realiza un estudio del arte sobre software existente con características similares para ver si estos pueden solucionar el problema. Se seleccionan tecnologías, lenguajes de programación, herramientas y metodología necesaria para la construcción de la idea planteada.

Capítulo 2: Presentación de la solución propuesta. En este capítulo se describen los procesos actuales a través del modelado del negocio, se seleccionan los requisitos funcionales y no funcionales, los casos de uso del sistema, así como, los diagramas correspondientes que lo modelan.

Capítulo 3: Construcción de la solución propuesta. En este capítulo se plantea la construcción propuesta en el capítulo anterior, en función de diagramas de clases y estándares del diseño, diseño de la base de datos, Modelo de despliegue y Modelo de implementación, además se realizan pruebas a la aplicación entregable.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA. TENDENCIAS Y TECNOLOGÍAS PARA EL DESARROLLO DEL SOFTWARE

El objetivo de este capítulo es una introducción al tema de la investigación, mostrando una explicación detallada de todo el contenido teórico que se necesita para un entendimiento del funcionamiento del software. El estudio realizado está centralizado en las características específicas del cliente actual del proyecto.

1.1. Términos asociados

Los términos incluidos en este epígrafe fueron seleccionados por los autores por su gran importancia en el estudio del campo de la video vigilancia. Es necesario conocer su significado para llegar a entender sobre el tema tratado en el documento.

Cámara IP: es una cámara que se conecta directamente a la red empleando el protocolo IP. Se le asigna una dirección IP de forma tal que se pueda acceder a sus funciones desde la red, para ser empleadas por aplicaciones capaces de detectarla y emplearla.

Vigilancia IP: efectiva solución de seguridad que ofrece monitorización y control avanzado en sistemas de seguridad, emplea la tecnología IP para efectuar la vigilancia.

Tecnología IP: emplea el protocolo IP para la comunicación entre las unidades que componen el sistema.

Tecnología Analógica: requiere una infraestructura separada que utiliza un sistema de cableado coaxial. Este cable fue diseñado, en el campo de la video vigilancia, para transmisiones punto a punto de video desde una cámara hasta una grabadora en el mismo sitio.

Plugin: es un complemento que se integra a una aplicación para aportar una función nueva o muy específica.

1.2. La Tecnología IP y la Video Vigilancia

Las soluciones de vigilancia IP garantizan que se pueda aprovechar la infraestructura de los sistemas de seguridad actuales y optimizarla reduciendo los costes de mantenimiento y gestión asociados con la vigilancia. Por esta razón esta técnica es muy utilizada actualmente como una efectiva solución, además de los numerosos beneficios que brinda. Se basa en cámaras o servidores de imágenes que disponen en su interior de un servidor web, dispositivo que permite operar las mismas sin necesidad de estar conectadas a ningún otro elemento. En la figura que se muestra a continuación, se esquematiza una solución basada en cámaras IP.

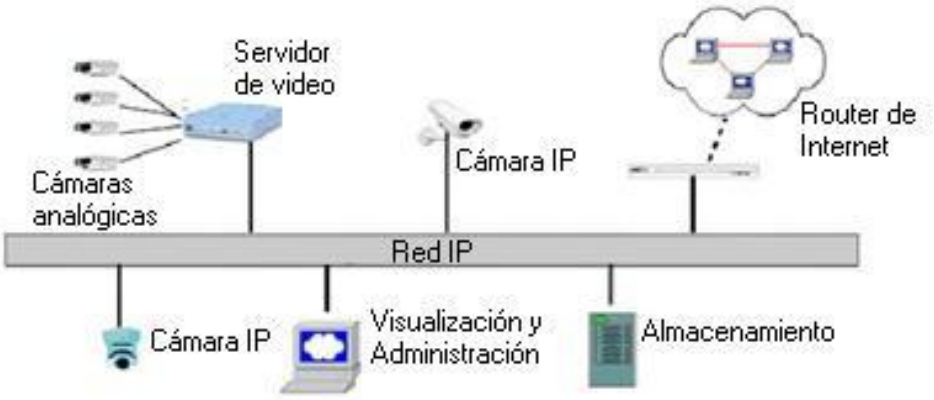


Figura 1. Propuesta de un Sistema de Video Vigilancia IP

(Tomada de Dlink : www.dlink.es/vigilanciaIP)

1.2.1. Ventajas de la tecnología IP en los Sistemas de Video Vigilancia

La tecnología IP es la más reciente y poderosa tecnología en este campo. Su uso proporciona algunas ventajas, entre ellas:

1. Captura digital de imagen, mejor calidad y mayor resolución: Esto permite poder almacenar el flujo de video de forma digital en dispositivos como discos duros o soportes magnético-ópticos, seguir de cerca los detalles y los cambios de las imágenes para tomar decisiones rápidas y eficaces tanto en la visualización como en el envío de alarmas.

2. Accesibilidad, flexibilidad y escalabilidad: Con el video en red puede visualizar en tiempo real desde cualquier computadora lo que sucede en un sitio controlado por el sistema. El flujo de video puede almacenarse en ubicaciones remotas, por motivos de comodidad o seguridad. El sistema se puede ampliar añadiendo más cámaras, además se pueden integrar nuevas tecnologías y ampliar capacidad de almacenamiento adicional según se precise.
3. Procesamiento distribuido e inteligencia a nivel de cámara: Permite que la cámara disponga de detección de movimiento, gestión de alarmas, u otras funciones para que, sin necesidad de la ayuda humana, se detecten situaciones sospechosas.
4. Visualización de video, audio, control de cámaras PTZ[4]: Permite no tan solo la captura de las imágenes de la cámara sino también el audio, y con el control a cámaras PTZ se puede realizar una monitorización más eficaz, pues el área de visión se adapta en la dirección que se requiera.
5. Opciones para la integración con software: Se puede adaptar al software que cumpla con sus necesidades específicas, y así brindarle un mejor resultado.
6. Posibilita la expansión del equipamiento asociado: Permite que existan sistemas interconectados de gran extensión.

1.3. Soluciones existentes. Estado del Arte

La seguridad física sigue siendo el primer y más visible aspecto de protección. Con el incremento en el procesamiento y almacenamiento de datos, las empresas se han percatado de la necesidad de proteger datos y también sus recursos humanos e inmuebles. De nada serviría tener una red infranqueable si luego cualquier persona puede acceder desde los propios ordenadores de la compañía, penetrando sin problemas desde el exterior. El mercado de la video vigilancia va prosperando, impulsado por el aumento de hechos delictivos y, en consecuencia, la mayor conciencia de seguridad pública y privada. Esto es posible gracias al rápido cambio tecnológico, en el cual los sistemas analógicos de Circuitos Cerrado de Televisión (CCTV) son remplazados por el video en red, una tecnología que empezó en la empresa Axis, pero ya varias empresas explotan las

[4] **PTZ:** Acrónimo de **P**an **T**ilt **Z**oom (Encuadre, Giro, Aumento). Describe el movimiento que realiza una cámara y la posibilidad que ofrece de ampliar una imagen.

ventajas de esta técnica y desarrollan soluciones que hacen más fácil la vigilancia de cualquier negocio, empresa y vivienda.

Las soluciones basadas en Web también son un acontecimiento en el desarrollo de la video vigilancia por las facilidades que estas pueden proporcionar. Entre las compañías envueltas en el desarrollo de soluciones de este tipo está Cisco, que además cuenta con un amplio catálogo de equipos físicos. Cisco Video Surveillance Stream Manager Software es unos de los sistemas que brinda esta compañía, basado en una página web sencilla. El sistema sigue el modelo cliente-servidor, donde el cliente es una PC con una cámara conectada al servidor, que es donde se desean controlar las cosas. El servidor es un alojamiento web habitual y la "secuencia de video" se puede visualizar por medio de cualquier PC conectada a Internet**. Este sistema está diseñado para manejar sólo los equipos físicos que brinda esta compañía.**

El software libre, por su parte, sigue ganando fuerza y evolucionando rápidamente, por lo que también presenta software para efectuar la video vigilancia. El proyecto Fedora, patrocinado por Red Hat, es un proyecto de código abierto apoyado por la comunidad de software libre. Este desarrolló el sistema de vigilancia de video completo ZoneMinder, **que solamente funciona para el sistema operativo Linux**, del cual se han desarrollado varias versiones. Esta aplicación web se basa en un sistema de detección de movimientos para reducir la cantidad de datos de video que deben enviarse por la red. Permite, la captura de video y análisis independientes de diferentes fenómenos, un gran número de opciones de configuración que ofrece un buen rendimiento en cualquier hardware, interfaz de usuario amigable para el control del sistema o de las cámaras, así como puntos de vista en vivo y repeticiones de eventos.

Actualmente, la Web es una comodidad para los usuarios, pues en la red se puede encontrar la mayor parte de las soluciones a los problemas, bajo una forma dinámica y sin muchas complicaciones. En esto basa su trabajo el equipo de **Web Gestión**, compañía que funciona desde la Web y realiza aplicaciones sobre la red. Una de las aplicaciones realizada por este equipo de desarrollo es Vidium, que es una plataforma de video vigilancia que ofrece servicios de video en tiempo real, a través de un canal de comunicación seguro y sin que el usuario tenga que instalar ningún programa en su ordenador. Este sistema es capaz de generar alertas por mensajes de texto

(SMS) o correo electrónico en función de eventos propios (cortes de corriente o señal) o propios de las cámaras (detección de movimiento). Además ofrece un sistema de cuatro ventanas para crear un entorno de trabajo más práctico pero **está pensado para instalaciones de hasta 12 cámaras porque con esto satisfacen hasta el 80% de las necesidades del mercado, algo que resulta una limitante para un sistema de video vigilancia de mayor envergadura.**

Luego de un estudio detallado del arte se puede notar que las aplicaciones web conocidas empleadas para la video vigilancia, presentan ciertas limitantes para su desempeño. Una aplicación que limite su uso a un determinado fabricante, como es el caso de Cisco Video Surveillance Stream Manager Software, no podrá funcionar correctamente en un entorno donde existan recursos de diferentes marcas, pues inutilizará una parte del sistema de seguridad. Por otro lado, el hecho de que una aplicación web solo pueda ser ejecutada en un determinado sistema operativo, como sucede con ZoneMinder, obliga al usuario a modificar su entorno de trabajo a expensas del uso de la aplicación. También, el hecho de que una aplicación limite la cantidad de recursos a emplear, tal como lo hace Vidium, acota las posibilidades de los sistemas de seguridad y no permite expansiones de los mismos. Por tanto, se hace necesaria la implementación de una aplicación web multiplataforma, capaz de soportar recursos de diferentes fabricantes, y que no limite la cantidad de recursos a emplear ni las facilidades técnicas de los sistemas de video vigilancia.

Una aplicación web para la video vigilancia debe ser capaz de permitir visualizaciones simultáneas de los flujos de video emitidos por sus cámaras bajo un ambiente confortable y entendible para el usuario. También, para el correcto acceso a los medios, las aplicaciones de este tipo deben contener un sistema de administración para evitar que personas no autorizadas accedan al sistema o a funcionalidades que permitan cambios sensibles en el sistema de seguridad.

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

1.4.1. Tecnologías a usar para el desarrollo del sistema

Para desarrollar una aplicación es necesario contar con tecnologías que permitan que el ciclo de desarrollo del software sea un proceso exitoso. Las mismas deben, fundamentalmente, facilitar la implementación, permitir compatibilidad, y ofrecer posibilidades de generar un software escalable y potente.

Un factor principal para crear una aplicación es el lenguaje con que se implementa. En el caso de la Web, la mayoría de las tecnologías existentes se integran para lograr aplicaciones de alto nivel en esta plataforma; de ahí que lenguajes diferentes se agrupen para, de una forma bien estructurada, generar sitios más dinámicos. Actualmente, se pueden encontrar este tipo de integraciones en los conocidos framework. Un framework, es una estructura de software integrada por componentes personalizables e intercambiables, en la cual otro proyecto de software puede ser organizado y desarrollado. (1) (2)

Para el desarrollo de la aplicación se tienen los siguientes lenguajes candidatos:

- **JavaScript**: lenguaje de programación interpretado, basado en prototipos, que permite a los desarrolladores crear acciones en sus páginas web. No requiere de compilación ya que el lenguaje funciona del lado del cliente, los navegadores son los encargados de interpretar el código. JavaScript es soportado por la mayoría de los navegadores como Internet Explorer, Netscape, Opera, Mozilla Firefox, entre otros.

- **HTML (*Lenguaje de Marcado de Hipertexto*):** es el lenguaje que se utiliza para crear las páginas web. Este lenguaje indica a los navegadores cómo deben mostrar el contenido de una página web.

- **PHP**: lenguaje de código abierto, utilizado principalmente para el contenido de páginas web dinámicas y aplicaciones del lado del servidor. Es libre y muy usado, aunque presenta limitantes pues no es totalmente orientado a objetos, no permite el desarrollo conjunto con lenguajes como C, C++, C#, y presenta manejo de errores poco eficiente.

- **JSP (*Java Server Pages*):** es una tecnología web, del lado del servidor, que se usa generalmente para generar documentos XHTML[5] y XML[6] dinámicos, y su funcionamiento se basa en scripts. Esta tecnología obliga a los programadores a emplear lenguaje Java y estar familiarizado con el mismo. Asimismo, requiere de una gran cantidad de espacio para el almacenamiento de sus archivos y las páginas deben ser recompiladas en el servidor en el primer acceso, lo que provoca un notable retraso durante el acceso a las mismas por primera vez y resta escalabilidad.

Además de los lenguajes antes mencionados, resulta importante mencionar un framework que resalta por sus ventajas en el desarrollo de aplicaciones web. Este es, ASP.NET:

- **ASP.NET (*Active Server Page*)**: es un framework para aplicaciones web desarrollado y comercializado por Microsoft. Es usado por los programadores para construir sitios web dinámicos, aplicaciones web y servicios web XML. Para su desarrollo puede utilizar los lenguajes de programación C#, VB.NET (Visual Basic.NET) o J#. Este framework es completamente orientado a objetos, y entre las principales ventajas de su uso se tienen:

  1. Reduce la cantidad de código requerido para crear grandes y potentes aplicaciones web.

  2. Simplifica y facilita el desarrollo, debido al modelo de programación que emplea del lado del servidor.

  3. El código fuente se ejecuta en el servidor, lo que hace que las páginas web tengan una alta flexibilidad y potencia.

[5] **XHTML**: Lenguaje extensible de marcado de hipertexto.

[6] **XML:** Metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). (2)

4. El código HTML producido por la página es enviado al navegador, lo que permite que el código fuente de la aplicación web no pueda ser robado fácilmente.

5. Posee un gran soporte para XML, CSS u otros estándares web establecidos.

6. Permite la actualización de aplicaciones web desplegadas sin necesidad de reiniciar el servidor, lo que facilita el mantenimiento.

7. Valida la información, a través de controles de validación, ahorrando trabajo a los desarrolladores en la creación de mecanismos de validación de datos.

8. Ofrece un entorno de trabajo organizado, con una división entre la capa de diseño y el código.

9. Las aplicaciones desarrolladas con ASP.NET pueden trabajar con grandes volúmenes de usuarios manteniendo una gran velocidad y alto rendimiento.

Por las características que tiene el sistema que dará solución al problema a resolver, los autores consideraron que el desarrollo de la aplicación se realice empleando los lenguajes HTML y JavaScript, así como emplear ASP.NET usando el lenguaje C#, pues permiten la creación de aplicaciones web de alta calidad, dinámicas, escalables y funcionales. Además, esta decisión se basa en las características anteriormente expuestas de estos lenguajes.

Dado que entre las tecnologías a emplear se encuentra ASP.NET usando el lenguaje C#, a continuación se describe este último:

- **C#:** es un lenguaje de alto nivel, orientado a objetos, potente y fácil de aprender. Este lenguaje facilita la vida a los programadores por su gran capacidad para manipular errores. C#, incorpora las ventajas o mejoras de lenguajes ya existentes tales como C, Java, Visual Basic y C++, lográndose un lenguaje flexible y poderoso. También contiene una librería de clases muy completa y bien diseñada, lo que hace que sea uno de los lenguajes predilectos y más utilizados.

1.4.2. Herramientas

Para el desarrollo de un sistema de vigilancia se necesitan herramientas que se pueden agrupar en grandes grupos de acuerdo con su función, a continuación se mencionan dichos grupos y se realiza una valoración de los principales exponentes:

- **Ingeniería y documentación:** Son conocidas por herramientas CASE[7]. Actualmente entre las más reconocidas en esta clasificación son: Rational Rose Enterprise Edition Suite, Visual Paradigm y Enterprise Architect 7.0.

  - **Rational Rose Enterprise Edition Suite:** es la herramienta CASE que comercializan los desarrolladores de UML[8] y que soporta de forma completa la especificación del UML en su campo. La Suite está compuesta por varias herramientas que cubren todos los aspectos de la ingeniería y documentación de cualquier tipo de proyecto utilizando UML. Presenta altos precios y es poco intuitiva de trabajar.

  - **Visual Paradigm:** es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software. Requiere bastantes recursos de memoria RAM[9] debido a estar desarrollada en Java[10] y presenta problemas de integración con otras herramientas de desarrollo.

  - **Microsoft Office Visio 2010:** es un software de creación de dibujos y diagramas que lleva la creación de diagramas a un nuevo nivel de desafío con herramientas y plantillas de visualización dinámicas y, controladas por datos, características eficaces de administración de procesos y capacidades avanzadas de uso compartido a través de la Web. La simplicidad, las formas basadas en datos y el uso compartido en la Web

[7] **CASE (*Computer Aided Software Engineering*):** Ingeniería de Software Asistida por Ordenador.

[8] **UML (*Unified Modeling Language*):** Lenguaje Unificado de Modelado): Lenguaje utilizado para modelar software.

[9] **RAM (*Random Access Memory. Memoria de Acceso Aleatorio*):** Tipo de memoria utilizada por las computadoras.

[10] **Java:** Lenguaje de desarrollo multiplataforma.

convierten a Visio 2010 en una de las formas más eficaces de ver y comprender información importante. Sus proyectos son fáciles de actualizar y pueden hacer la diferencia en cuanto a la productividad del equipo de desarrollo.

- **Enterprise Architect 7.0:** es una herramienta comprensible de diseño y análisis UML, cubriendo el desarrollo de software desde el paso de los requerimientos a través de las etapas del análisis, modelos de diseño, pruebas y mantenimiento. "Combina el poder de la última especificación UML 2.1 con alto rendimiento, interfaz intuitiva, para traer modelado avanzado al escritorio, y para el equipo completo de desarrollo e implementación." (3) Además es una herramienta multiusuario, de bajos costos de licencia, diseñada para ayudar a construir software potente y fácil de mantener. Permite salida de documentación flexible y de alta calidad. También, "provee trazabilidad completa desde el análisis de requerimientos hasta los artefactos de análisis y diseño, a través de la implementación y el despliegue." (3)
  Soporta generación e ingeniería inversa de código fuente para muchos lenguajes populares, incluyendo C++, C#, Java, Delphi, VB.Net, Visual Basic y PHP.

➢ **Control de versiones:** Para ofrecer seguridad al código fuente y los archivos con que se trabaja en el ciclo de desarrollo del software se hace necesario emplear tecnologías para el control de versiones. Entre las candidatas tenemos:

- **SubVersion (SVN):** es un software para el control de versiones, en este no se actualizan los archivos por separados, todo el repositorio tiene un único número de revisión que indica el estado de todos los archivos. Puede ser usado por varias personas aunque estén en distintos ordenadores, esto permite que se pueda progresar más rápidamente. Los cambios que se envían por la red solo son actualizados si se logra enviar el paquete completo con éxito para eliminar el problema de que pueda afectarse la información ya guardada. SubVersion funciona idénticamente con ficheros de texto y ficheros binarios. Ambos tipos de ficheros son

almacenados igualmente y comprimidos en el repositorio. Es fácil de mantener y adaptable a cualquier lenguaje.

- **Microsoft Visual Source Safe 2008 (VSS):** es una herramienta para los desarrolladores en el sistema operativo Windows. Este software permite que el código fuente, de la aplicación controlada, pueda ser modificado por un solo programador o varios, pero esto último no es aconsejable ya que las herramientas para reunificar el código fuente que proporciona no son muy buenas comparadas con otras herramientas gestoras de código fuente. No es estable para guardar archivos binarios, los toma como si fueran de texto, por lo que no es confiable para almacenar documentación, solo código fuente.

- **Concurrent Versions System (CVS):** está desarrollado para controlar versiones concurrentes y mantiene el registro de todo el trabajo principalmente de los ficheros de código fuente. Presenta como desventaja que los archivos en el repositorio sobre la plataforma CVS no pueden ser renombrados, estos deben ser agregados con otro nombre y luego eliminados. CVS solamente cuenta con un historial de ficheros individuales, y mantiene el registro de la historia de las versiones solamente para desarrolladores locales.

- **IDE de desarrollo:** Los IDEs[11], son aplicaciones claves para una implementación rápida y eficaz del software, los mismos se deben seleccionar en dependencia de los lenguajes en que se piensa desarrollar, puesto que no todos soportan los lenguajes requeridos.

  Teniendo en cuenta que se seleccionaron HTML, JavaScript y C#, como lenguajes para el desarrollo del sistema, se tienen los siguientes IDEs candidatos:

[11] **IDE (*Integrated Development Environment*):** Entorno de Desarrollo Integrado.

- **Aptana Studio:** es un entorno para desarrollo web que permite trabajar con diferentes lenguajes y tecnologías de programación web, tales como HTML, DOM, JavaScript y CSS. Este IDE es gratuito, de código abierto, multiplataforma, altamente extensible, posee diferentes modos de conexión y herramientas para el trabajo con bases de datos. Además, ayuda a aumentar la productividad de sus proyectos pues contiene librerías de JavaScript y AJAX integradas, entre ellas ExtJS, jQuery, Prototype, Scriptaculous, YUI entre otras, así como, ofrece un soporte para JavaScript de depuración interna con la integración de Firefox e Internet Explorer.

- **Microsoft Visual Studio 2010:** es un reconocido entorno de desarrollo, lanzado por Microsoft, enfocado hacia las necesidades y perspectivas de los desarrolladores de software de la actualidad. Este IDE, está desarrollado, fundamentalmente, para el uso de C# y contiene nuevas formas de trabajo para Windows Presentation Foundation (WPF) y aplicaciones Microsoft Silverlight a modo de lograr aplicaciones enriquecidas de Internet (RIA) de mayor calidad y rapidez. Además, presenta herramientas integradas para el desarrollo de Windows 7, posee mecanismos, como IntelliTrace, para una mejor detección de errores y permite un “seguimiento rápido del flujo de ejecución de un programa sin necesidad de llamar al depurador.” (4)

- **Mono Tools for Visual Studio:** es una tecnología desarrollada por el proyecto de código abierto Mono, para permitir que las aplicaciones construidas en Microsoft Visual Studio puedan funcionar en otros sistemas operativos, y no solamente en Windows. Funciona como un complemento del Microsoft Visual Studio y abre posibilidades a los desarrolladores para que sin abandonar el ambiente del IDE, escriban aplicaciones .NET para Linux, UNIX y Mac OS X.

1.4.3. Tecnologías a usar para la comunicación y desarrollo de la interfaz de usuario

- **Comunicación:** Para la comunicación, se necesitan tecnologías que permitan la transmisión de datos sobre la red condicionadas por el lenguaje de desarrollo a emplear, así como también protocolos de comunicación para poder enviar y recibir datos.

  Entre estos protocolos se tienen los siguientes:

  - **TCP/IP:** es un conjunto de protocolos empleados para la transmisión de datos en plataforma web, sus siglas TCP significan *"Protocolo de Control de Transmisión".*

  - **HTTP** *(Protocolo de Transferencia de Hipertextos)***:** se emplea en las transacciones que se realizan en la Web y permite la interacción entre las estaciones que fungen como cliente y servidor.
  - **HTTPS:** se emplea para la transmisión de datos sensibles y para establecer comunicaciones seguras. Sigue los mismos principios de funcionamiento del protocolo HTTP, salvo que cuenta con certificados y métodos de cifrado para establecer la seguridad a través del cifrado de los datos.

  - **RTSP** *(Real Time Stream Protocol)*: se emplea para controlar los flujos de multimedia (stream) que transitan por la red.

  - **Sockets:** se emplea como un método para la comunicación entre un programa cliente y un programa servidor en una red. Esta tecnología crea y utiliza un sistema de peticiones para lograr la comunicación y emplea la mayoría de los protocolos de comunicación de datos, sobre todo, TCP/IP y UDP. Es una tecnología muy eficiente en cuanto al rendimiento y la velocidad de transmisión de datos. Posee una alta flexibilidad para su uso en el tipo de aplicaciones que pueden emplearla y es altamente compatible con los sistemas operativos actuales.

- **Microsoft Net Remoting:** es una interfaz de programación de aplicaciones (API[12]) para la comunicación. Es muy utilizada en la creación de aplicaciones distribuidas, debido a las facilidades que ofrece para el desarrollo, estableciendo enlaces entre los componentes de la aplicación sin importar la ubicación de los mismos.

  .NET Remoting proporciona un enfoque abstracto en la comunicación entre procesos que separa el objeto utilizado de forma remota de un dominio de aplicación de cliente o servidor específico y de un mecanismo específico de comunicación. Por lo tanto, se trata de un sistema flexible y fácilmente personalizable. Se puede reemplazar un protocolo de comunicación con otro o un formato de serialización con otro sin tener que recompilar el cliente ni el servidor. Además, 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 (5).

Los protocolos anteriormente mencionados permitirán la comunicación del sistema. Por otra parte, como tecnología para la comunicación, aunque no es su principal objetivo, se empleará **AJAX** *(Asynchronous JavaScript And XML)* que es una técnica de desarrollo web para crear aplicaciones interactivas y además permite interactuar dinámicamente con los datos, empleando XML, JSON[13] y XSLT[14], para intercambiar y manipular datos de manera asíncrona con un servidor web. Además, **se utilizará Microsoft NET Remoting 2.0 para establecer la comunicación del Módulo Web con otros componentes y módulos asociados funcionalmente a la aplicación.**

[12] **API (*Application Programming Interface*)**: Interfaz para programación de aplicaciones.

[13] **JSON (*JavaScript Object Notation*):** formato ligero de intercambio de datos.

[14] **XLST (*Extensible Stylesheet Language Transformations*):** lenguaje para transformar documentos XML.

- **Interfaz de Usuario:** Para que el usuario interactúe a gusto con un sistema necesita, como aspecto primordial, de una interfaz de usuario atractiva que lo motive y le sea entendible e intuitiva en el trabajo que realiza. Para el desarrollo de la misma en tecnología web, se emplean librerías o frameworks que facilitan este trabajo, aunque existen también métodos tradicionales de diseño que emplean imágenes estáticas o animadas y textos decorados.

  Entre las tecnologías más populares se encuentran:

  - **Dojo Toolkit:** es un framework que contiene APIs y controles(widgets) para facilitar el desarrollo de aplicaciones web que utilicen tecnología AJAX. Las aplicaciones desarrolladas dependen del soporte de los navegadores para Dojo, y el código fuente es imposible de ocultar en caso de productos comerciales.

  - **JQuery:** es una biblioteca o framework de JavaScript que permite simplificar la manera de interactuar con los documentos HTML, manejar eventos, desarrollar animaciones y emplear la tecnología AJAX en páginas web. JQuery impone, para poder desarrollar con esta tecnología, el conocimiento obligatorio de CSS[15]. Además, posee un lenguaje poco legible y el código fuente generado es difícil de proteger.

  - **ExtJS:** es una librería JavaScript que permite construir aplicaciones complejas en Internet o RIAs[16]. Esta librería incluye:
    - Componentes de Interfaz de Usuario del alto nivel y personalizables.
    - Modelo de componentes extensibles.
    - Un API fácil de usar.
    - Licencias Open Source y comerciales.

[15] **CSS (*Cascading Style Sheets*):** Hojas de estilo en cascada. Es el lenguaje empleado para definir la presentación de un documento estructurado escrito en HTML.

[16] **RIA** (***Rich Internet Applications***)*:* Aplicaciones de Internet Enriquecidas.

ExtJS permite crear aplicaciones complejas utilizando componentes predefinidos así como un manejador de *layouts* (diseños), esto provee una experiencia consistente sobre cualquier navegador, evitando el problema de validar que el código escrito funcione bien en cada uno (Firefox, Internet Explorer, Safari, etc.). También provee un balance entre Cliente – Servidor, donde la carga de procesamiento se distribuye, permitiendo que el servidor pueda manejar más clientes al mismo tiempo. Además emplea comunicación asíncrona y el *motor de render*[17] puede comunicarse con el servidor sin necesidad de estar sujeta a la aplicación, a un evento o una acción del usuario, dándole la libertad de cargar información sin que el cliente se dé cuenta; por tanto permite la eficiencia de la red, pues el tráfico de red puede disminuir al permitir que la aplicación elija que información desea transmitir al servidor y viceversa.

- **Ext.NET:** es una librería de código abierto diseñada para el framework ASP.NET que integra la librería de JavaScript ExtJS para la creación de aplicaciones enriquecidas de Internet en dicho framework. Ext.NET ofrece más de 100 controles de alto rendimiento que pueden ser empleados en ASP.NET para el desarrollo de interfaces de usuario de alta calidad y funcionalidad empleando las bondades de ExtJS, AJAX y JSON. Además, permite la fusión de estas tecnologías con la plataforma .NET, así como el trabajo con las mismas tanto desde el lado del cliente como del servidor.

1.4.4. Tecnologías a emplear para la visualización de los flujos de video

Para visualizar los flujos de video en la aplicación es necesario contar con tecnologías que permitan la visualización de video a través de la red. Como principales exponentes se tienen:

- **OpenCV (*Open Source Computer Vision*):** es una librería empleada en el tratamiento de imágenes, preparada fundamentalmente para la visión en tiempo real. OpenCV tiene una gran gama de funcionalidades pero presenta problemas para reconocer los dispositivos

[17] **Motor de render:** es el encargado de generar o dibujar los componentes de la librería y generar las imágenes a partir de un modelo dado.

conectados a un ciclo cerrado de televisión (CCTV), en un sistema operativo que no sea Linux.

- **HTML5:** es la quinta versión del lenguaje HTML, la misma ofrece elementos y atributos que proporcionan nuevas funcionalidades para la creación de sitios web. HTML5 permite la reproducción de audio y video desde el navegador sin necesidad de complementos (plugins[18]) o reproductores embebidos. Esta novedosa tecnología presenta varias limitantes pues solo soporta dos formatos de video (H264, Theora), solo es soportado por las versiones más recientes de los navegadores y limita el formato a reproducir en dependencia del navegador.

- **VLC Mozilla Plugin:** es el complemento desarrollado por VideoLAN para lograr extender las funciones del reproductor de multimedia VLC a la Web, fue creado inicialmente para Mozilla pero funciona en casi todos los navegadores empleados en la actualidad. Este complemento permite la reproducción de una gran cantidad de formatos de audio y video, ofrece muchas ventajas para el trabajo con flujos de audio y video a través de la red y no consume muchos recursos durante su funcionamiento.

Teniendo en cuenta las características de las tecnologías seleccionadas como candidatas, descritas anteriormente, y las del software a desarrollar:

Se emplearán para realizar la Ingeniería del Software y la Documentación: el Enterprise Architect 7.0, y el Microsoft Office Visio 2010. Para efectuar el control de versiones, el Subversion. Como entornos de desarrollo integrado (IDE), los software Aptana Studio, por las facilidades que ofrece para el trabajo con JavaScript y Hojas de estilo (CSS) y Microsoft Visual Studio 2010 para el desarrollo con ASP.NET y C#. Además, se propone emplear *Mono Tools for Visual Studio* para la construcción del sistema para otras plataformas. Por otra

[18] **Plugin:** "Pequeño programa que añade alguna función a otro programa, habitualmente de mayor tamaño. Un programa puede tener uno o más conectores. Son muy utilizados en los programas navegadores para ampliar sus funcionalidades." (16)

parte, para realizar la interfaz de usuario se empleará la librería de JavaScript ExtJS asociada a la librería Ext.NET para poder emplear ExtJS integrado al framework ASP.NET. Para la visualización de los flujos de video, se utilizará el plugin de VLC para Mozilla (*VLC Mozilla Plugin*).

1.4.5. Metodologías

Con el tiempo, la informática fue madurando y algunos profesionales de las tecnologías de la información se dieron cuenta que se hace necesario seguir ciertas pautas predefinidas en el desarrollo del software de calidad, es decir, seguir una metodología. Una metodología de desarrollo de software o Proceso de Desarrollo de Software es la definición del conjunto de actividades y reglas que guían los esfuerzos de las personas que trabajan en el proyecto, explicando los pasos necesarios para terminar el producto con buena calidad. Su objetivo es aumentar y garantizar la calidad del software en todas las fases por las que transita. Su planificación ayuda a producir lo esperado, en el tiempo esperado y con el coste esperado. No tiene sentido ajustarse a un proceso sino que hay que adaptarlo a las necesidades y características de cada equipo de trabajo, además de elegir con un análisis previo al desarrollo del software, el que se adapte mejor a la situación para obtener los resultados esperados.

Se podría decir que en los últimos años se han desarrollado dos corrientes en lo referente a los procesos de desarrollo, los llamados métodos pesados y los métodos ligeros (también conocidos como métodos ágiles). Estos últimos se basan en la comunicación inmediata y directa, mientras que las metodologías pesadas o tradicionales proponen que sea a través del orden y la documentación. Existen muchas diferencias entre estas metodologías, por lo que no se debe escoger una de ellas por conveniencia sino la que sea más útil después de un estudio donde se defina el alcance del proyecto, la complejidad y tamaño del mismo, y se estudie el entorno de aplicación del software.

¿Qué tipo de metodología se podría aplicar para guiar el desarrollo del software?

Para responder esta pregunta, es necesario analizar cómo influyen las metodologías en el ambiente de desarrollo del producto.

Metodologías Ágiles

Las metodologías ágiles, están especialmente preparadas para cambios en el proyecto, lo que no sería de mucha ayuda en el software que se propone como solución en este documento, ya que el producto a desarrollar debe estar guiado desde el principio por los requisitos funcionales que se definan. Además, brindan un proceso poco controlado con los documentos lo que no asegura que el producto se vaya a entregar en el tiempo establecido ni que se vaya a cumplir con las necesidades planteadas por el cliente. En estas metodologías, el cliente forma parte del equipo de desarrollo para estar al tanto de las nuevas funcionalidades que se le puedan insertar y presentan menos énfasis en la arquitectura de software.

Metodologías Tradicionales

Las metodologías tradicionales tienen cierta resistencia al cambio por lo que, por un mal entendimiento entre el cliente y los desarrolladores, no fracasará el desarrollo del software. Asimismo, ayudan a mantener un proceso muy controlado con numerosas políticas y normas. El cliente no tiene que ser parte del equipo de desarrollo solo debe mantener el contacto para las entrevistas que se le realicen y otras dudas. En ellas la arquitectura de software es esencial y se expresa mediante modelos.

Con el análisis anterior y teniendo en cuenta las diferencias mostradas se concluye que sería más adaptable aplicar una metodología tradicional, ya que esta propone la realización de una detallada documentación y organización de: documentos, modelos, diagramas y otros artefactos generados en el desarrollo. Además, el cliente actual no forma parte del equipo de trabajo, el software a desarrollar gestiona una gran cantidad de información y los requisitos son estables. Las

metodologías tradicionales más utilizadas son RUP[19] y MSF[20], estas son adaptables a cualquier proyecto y llevan una documentación exhaustiva del mismo. Se escoge RUP por ser la más moderna de las metodologías tradicionales, y con mucho auge en el desarrollo de software, siendo la metodología más utilizada en el mundo entero en este ámbito, y es soportada por la herramienta Enterprise Architect que es la herramienta a utilizar para el desarrollo.

Algunas de las ventajas de RUP sobre las otras metodologías tradicionales son:

1. Realiza la evaluación en cada fase, permitiendo cambios de objetivos: entre más temprano se detecten los cambios menos costoso serán para el proyecto.

2. Sigue los pasos intuitivos necesarios a la hora de desarrollar el software y guía detalladamente el desarrollo del software.

3. Permite un seguimiento detallado en cada una de las fases: hace posible que se detecten los errores lo más temprano posible.

RUP:

El Proceso Unificado Racional (en inglés, *Rational Unified Process*) es un proceso de desarrollo de software que utiliza UML (*Unified Modeling Language*, en inglés) como lenguaje de modelado de procesos y constituye la metodología más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. Es dirigido por casos de uso, estos reflejan lo que los usuarios futuros desean y necesitan, se captan cuando se está iniciando el proceso y en esta fase se representan como requerimientos, a partir de ahí guían todo el proceso de desarrollo. También, este proceso es totalmente centrado en la arquitectura porque muestra una visión común del sistema completo, con la que el equipo de desarrollo y los usuarios deben estar de acuerdo. Además, RUP es un proceso iterativo e incremental pues propone que cada fase se desarrolle en iteraciones, y cada iteración tiene que proponerse un incremento en el proceso de desarrollo del

[19] **RUP (*Rational Unified Process*):** Proceso Unificado Racional.

[20] **MSF (*Microsoft Solutions Framework*):** Marco de Soluciones de Microsoft.

software. Los principales elementos de esta metodología son los trabajadores, sus actividades, los artefactos y el flujo de actividades que es el que muestra el resultado observable.

Según la guía de *Rational Unified Process*, por IBM (6):

RUP tiene como **principales características:**

1. Forma disciplinada de asignar tareas y responsabilidades (quién hace qué, cuándo y cómo).
2. Pretende implementar las mejores prácticas en Ingeniería de Software.
3. Desarrollo iterativo.
4. Administración de requisitos.
5. Uso de arquitectura basada en componentes.
6. Control de cambios.
7. Modelado visual del software.
8. Verificación de la calidad del software.

Fases de RUP:

1. Conceptualización (Concepción o Inicio): Se describe el negocio y se identifican los casos de usos del sistema.
2. Elaboración: se define la arquitectura del sistema y se obtiene una aplicación ejecutable que responde a los casos de uso que la comprometen.
3. Construcción: se obtiene un producto listo para su utilización que está documentado y tiene un manual de usuario.
4. Transición: se realizan pruebas y reparación de errores.

Al finalizar cada fase se le presentan al cliente los artefactos definidos, es decir, el avance del proyecto para una evaluación de la calidad del mismo desde el punto de vista del cumplimiento o no con las necesidades planteadas por él.

1.5. Conclusiones

En este capítulo se muestra el avance que significó el desarrollo de la tecnología IP para el esparcimiento de los Sistemas de Video Vigilancia principalmente por su bajo costo de implantación y la flexibilidad que le brinda a estos sistemas. Además se hizo un estudio del arte, tomando como centro del mismo los Sistemas de Video Vigilancia para plataforma web existentes en la actualidad, analizando detalladamente sus funcionalidades y definiéndose sus limitantes como aspectos a superar por la solución que propone este trabajo, las aplicaciones de este tipo en el mundo fundamentalmente, están dirigidas a problemas concretos de un cliente, por lo que existen diversidad de soluciones. Luego de asegurar que los productos existentes en el mercado mundial no cumplen con las necesidades planteadas y con el objetivo de desarrollar el software propuesto por el Proyecto Video Vigilancia, se realizó un estudio de las tendencias y tecnologías a emplear en el desarrollo del software, seleccionándose las más adecuadas para lograr un software potente, escalable y altamente funcional, además en la selección de los lenguajes y tecnologías intervino mucho la necesidad de integración con el Sistema de Video Vigilancia que es una aplicación de escritorio.

CAPÍTULO 2. **PRESENTACIÓN DE LA SOLUCIÓN PROPUESTA**

Para comenzar con el modelado del negocio, se realizó un estudio del mismo con el objetivo de lograr un entendimiento de los conceptos que utiliza el cliente y con los cuales tendrá que trabajar la aplicación. Debido a que no se lograron determinar bien los procesos del negocio, con sus responsabilidades delimitadas, los actores que lo inician, los que se benefician de estos procesos y los que desarrollan cada actividad relacionada con cualquier parte de los procesos, se establece el Modelo de dominio.

2.1. Modelo de dominio

"El Modelo de Dominio es una representación visual estática del entorno real objeto del proyecto." (7)

Un Modelo de dominio representa clases conceptuales o de objetos en un dominio de interés en el mundo real. Captura los tipos más importantes de objetos que existen o los eventos que suceden en el entorno donde estará el sistema y no incluyen las responsabilidades de las personas que ejecutan las actividades. Aprovechando las posibilidades de los diagramas UML para la representación de conceptos, el Modelo de dominio se representa en forma de diagrama de clases en el que se muestran los conceptos u objetos del dominio del sistema en cuestión, y las asociaciones entre ellos.

2.1.1. Conceptos Fundamentales

Para una mejor comprensión del Diagrama de Modelo de dominio que se presentará en el siguiente epígrafe, a continuación se proporciona un marco conceptual con las definiciones identificadas.

- **Cámaras:** hardware que brinda la posibilidad de obtener los flujos de video de un área determinada.
- **Flujos de Video:** representan la secuencia de imágenes que transmiten las cámaras, los cuales se pueden visualizar o realizar otras operaciones con ellos.

- **Gestor de Información:** sistema que está funcionando actualmente, el cual es el encargado de realizar cualquier operación con los flujos de videos que envían las cámaras del Sistema de Video Vigilancia.
- **Visor:** aplicación de escritorio que permite visualizar los flujos de video. La aplicación tiene que estar instalada en la estación de trabajo y conectada a una red local para que pueda completarse el proceso de visualización de los flujos de video a través de ella.
- **PC:** máquinas dedicadas en la institución al proceso de vigilancia, estas tendrán el Visor instalado y acceso a la red local.
- **Estación de monitorización:** área física donde estarán situadas las máquinas y los guardias de seguridad encargados de la video vigilancia.
- **Personal de seguridad:** trabajadores encargados de la seguridad.

2.1.2. Diagrama del Modelo de dominio

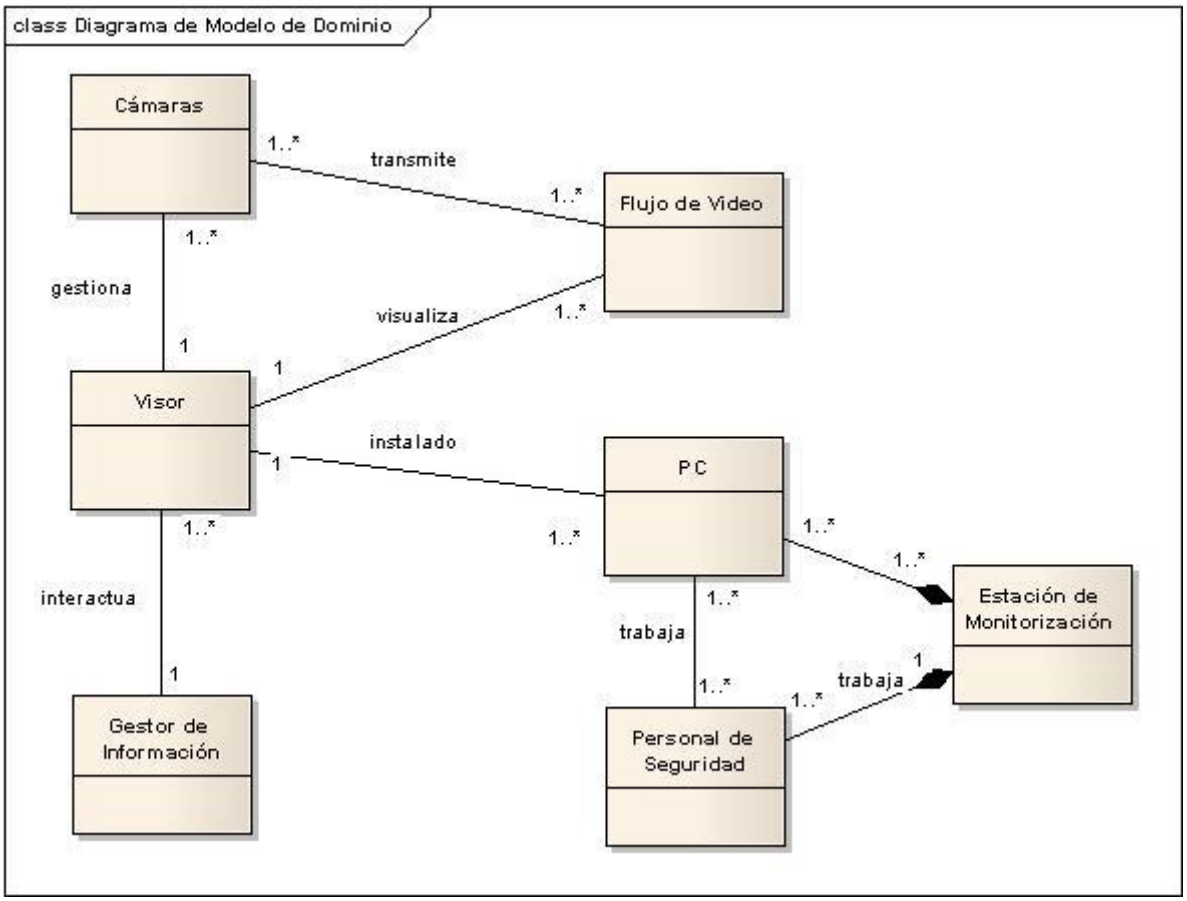


Figura 2. Diagrama del Modelo de dominio

2.2. Propuesta del Sistema

La solución que se plantea, es la de desarrollar un sistema modular que complemente el Sistema de Video Vigilancia Suria Vision, a fin de poder efectuar la actividad, empleando los recursos de dicho sistema, desde la Web. El sistema proveerá de funcionalidades que permitan acceder a los flujos de video de las cámaras IP que compongan el sistema de seguridad, así como permitirá la manipulación de las mismas según las capacidades

particulares de cada una. Con la solución propuesta se ofrece una alternativa que permita, desde cualquier lugar con acceso a la Web, efectuar la video vigilancia de una localidad, centro o institución donde esté desplegado el Sistema Suria Vision, esto no sustituye al personal de seguridad, ni a los software de video vigilancia de las estaciones principales de monitorización del lugar.

El sistema propuesto, es un sistema abierto que tendría como entradas los flujos de video provenientes de las cámaras IP y otros datos obtenidos del Gestor Central de Información (de manera abreviada, Gestor). El Gestor es un sistema externo, que constituye el corazón del Sistema de Video Vigilancia Suria Vision y es el que provee la información a cada instancia del Módulo Web de Monitorización y Administración del Sistema de Video Vigilancia (para abreviar, Módulo Web), este último constituye la solución a desarrollar. (8)

El Módulo Web, constituirá una aplicación que se vinculará al Gestor de un Sistema de Video Vigilancia ya implantado, para poder trabajar con los recursos de dicho sistema. Para acceder a la aplicación, cada usuario tendrá un nombre de usuario y contraseña, así como, un rol con un conjunto de permisos asignados que habilitarán las funciones que podrá emplear en el sistema. La aplicación, además de las funcionalidades antes mencionadas permitirá agregar una nueva cámara al Sistema de Video Vigilancia, y configurar o eliminar las existentes. Para la mayoría de las operaciones, se efectuará la comunicación con el Gestor, ya que es el responsable de gestionar el flujo de información del Sistema de Video Vigilancia.

Por otra parte, la aplicación tiene como funcionalidad principal, la visualización simultánea de varios flujos de video, empleando diferentes estilos de visualización, ya sea de 4, 10 o 13 pantallas, lográndose así la visión del video de varias cámaras desde una misma interfaz de usuario. Por esto, las interfaces de usuario conformarán un ambiente sencillo e intuitivo, con una combinación de colores agradables, a fin de lograr un diseño atractivo y fácil de entender.

2.3. Requerimientos Funcionales del Sistema

El sistema debe permitir:

RF1 Visualizar flujos de videos de las cámaras.

RF1.1 Visualizar varios flujos de videos de las cámaras, simultáneamente.

RF1.2 Visualizar empleando diferentes estilos de visualización (Modo pantalla única, Modo 4x4, Modo 2x8 y Modo 1x12).

El usuario debe poder visualizar varios flujos de video simultáneamente, en diferentes estilos de visualización, según su conveniencia.

RF2 Gestionar Cámara.

RF 2.1 Adicionar cámara.

RF 2.2 Explorar cámara.

RF 2.3 Eliminar cámara.

RF 2.4 Configurar cámara.

El requerimiento permitirá adicionar, eliminar y configurar las cámaras para poder visualizar sus flujos de video y ver las características de las mismas.

RF3 Autenticar Usuario.

Para acceder a la aplicación, el usuario debe estar registrado en la base de datos del sistema, en la cual junto a sus credenciales (nombre de usuario y contraseña), tendrá asignado un rol al que se asocian diferentes permisos que son asignados por el administrador del sistema. Igualmente, el usuario puede tener otros permisos además de los del rol. De esta forma cada usuario, bajo los principios de mínimo privilegio, realiza las acciones que son imprescindibles para su trabajo.

RF4 Gestionar Usuario.

RF 4.1 Adicionar usuario.

RF 4.2 Actualizar datos de usuario.

RF 4.3 Eliminar usuario.

Mediante este requisito el Administrador podrá adicionar un nuevo usuario, eliminarlo y actualizar sus datos, rol y permisos.

RF5 Gestionar Rol.

RF 5.1 Adicionar rol.

RF 5.2 Editar rol.

RF 5.3 Eliminar rol.

Con este requisito el Administrador puede crear un rol para asignarlo a un usuario o grupo de usuarios, así como, editar los permisos que conlleva el rol o eliminar el rol del sistema.

RF6 Asignar permisos.

Según el rol, el usuario podrá realizar las acciones que habiliten los permisos asociados a dicho rol. El Administrador podrá asignar o quitar permisos a un rol dado.

RF7 Manipular Cámara.

RF 7.1 Manipular Cámaras PTZ.

El sistema permite la manipulación de las cámaras, según las particularidades de la misma. En el caso de las cámaras PTZ, se pueden efectuar las operaciones fundamentales como son: rotar la cámara, enfocar el lente y aumentar la imagen.

2.4. Requerimientos No Funcionales del Sistema

2.4.1. Requerimientos de rendimiento

RNF1. Permitir la visualización simultánea de al menos 13 cámaras.

2.4.2. Requisitos de software

RNF2. Sistema Operativo Windows XP o superior.

RNF3. NET Framework 3.5 o superior.

RNF4. DirectX 9.0 o superior.

RNF5. Navegador Mozilla Firefox 3.6 o superior.

RNF6. Plugin de VLC (Video LAN), para Mozilla, versión 0.8.6 o superior.

2.4.3. Requisitos de hardware

RNF7. 256 MB de Memoria RAM (mínimo), 2GB Memoria RAM (recomendado).

RNF8. Microprocesador Intel Pentium IV a 3.0 GHz o superior (recomendado).

RNF9. Tarjeta de Red Gigabit Ethernet NIC (recomendado).

Descripción: Las recomendaciones anteriores permiten un funcionamiento eficiente del sistema a desarrollar, pues se necesita de una gran capacidad de memoria y un microprocesador potente para poder manipular todos los flujos de video y realizar las operaciones, sin que se presenten retrasos en las respuestas del sistema.

2.4.4. Restricciones de diseño

RNF10. Resolución de pantalla 1024 x 768.

Descripción:

La aplicación está optimizada para una resolución de 1024x768.

2.4.5. Requisitos de apariencia o interfaz externa

RNF10. La interfaz gráfica de la aplicación debe concebirse con un ambiente sencillo y de navegación fácil e intuitiva para el usuario.

RNF11. Los colores serán convenientemente utilizados dada la funcionalidad y objetivos del sistema, siendo claros y sobrios en la mayor parte de la aplicación, logrando una vista agradable a los usuarios y resaltando con otras tonalidades los mensajes de interacción de los que dependen las funcionalidades críticas.

RNF12. Los usuarios deben ser capaces de modificar los estilos o temas de colores que emplee el sistema a fin de trabajar en un entorno según su gusto.

2.4.6. Requerimientos de seguridad

RNF13. Garantizar la integridad, disponibilidad y confidencialidad de la información manipulada por el sistema, empleando la autenticación de usuarios con diferentes permisos para trabajar en el sistema.

RNF14. Garantizar validación de los datos que son entrados por los usuarios al sistema.

RNF15. Solicitar confirmación ante acciones irreversibles en el sistema, dígase eliminación de cualquier información.

2.4.7. Requerimientos de usabilidad

RNF16. Debe poder ser operado por usuarios sin grandes conocimientos informáticos, por lo que los enlaces y botones serán fáciles de asociar con las operaciones que realizan.

RNF17. El sistema deberá estar bajo los estándares establecidos de la norma ISO 9241-11, la cual define los parámetros internacionales de usabilidad de cualquier software.

2.4.8. Requerimientos de soporte

RNF18. Los requisitos para el despliegue y mantenimiento del sistema no deben ser complejos.

RNF19. Se requiere de la instalación del plugin de VLC (Video LAN), para Mozilla, en su versión 0.8.6 o superior.

RNF20. Se requiere de la instalación de Windows XP o superior en las estaciones de trabajo.

RNF21. Se requiere de la instalación de NET Framework 3.5 o superior en las estaciones de trabajo.

RNF22. Se requiere de la instalación de DirectX 9.0 o superior en las estaciones de trabajo.

2.5. Definición de los Casos de Uso

2.5.1. Definición de los actores

Actor	Descripción
Administrador	Actor de más altos privilegios en la aplicación, responsable de configurar el sistema, de determinar las acciones que realizarán los demás usuarios, y capaz de realizar todas las funcionalidades.
Usuario	Actor condicionado por los permisos asignados al rol que presente en el sistema.

Tabla 1. Actores definidos del sistema

2.5.2. Listado de los Casos de Uso

CU1. Autenticar Usuario.

CU2. Visualizar Flujo de Video.

CU3. Gestionar Cámara

CU4. Gestionar Usuario.

CU5. Gestionar Rol.

CU6. Asignar Permisos.

CU7. Manipular Cámara.

CU8. Manipular Cámara PTZ (Caso de Uso Extendido).

2.6. Diagrama de Casos de Uso

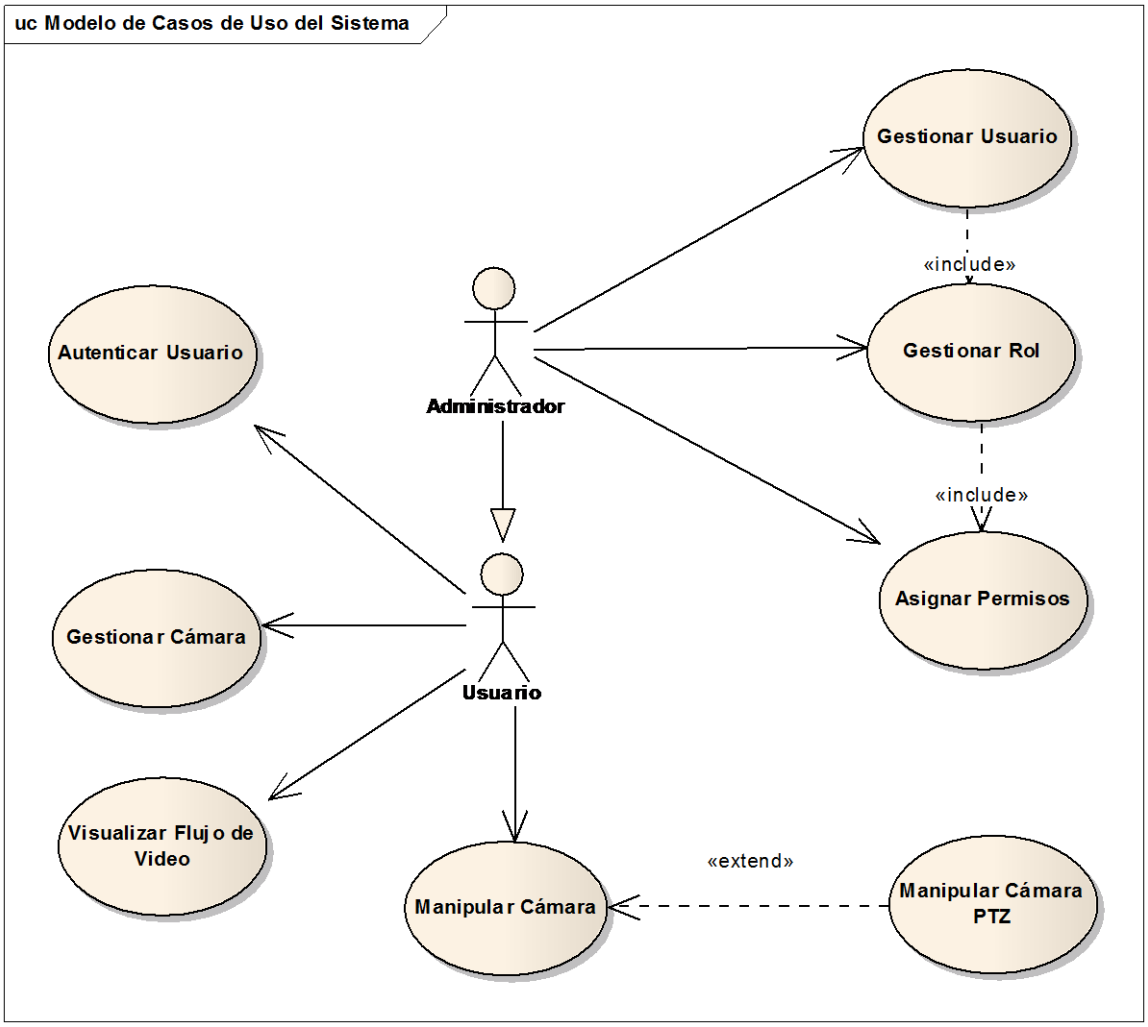


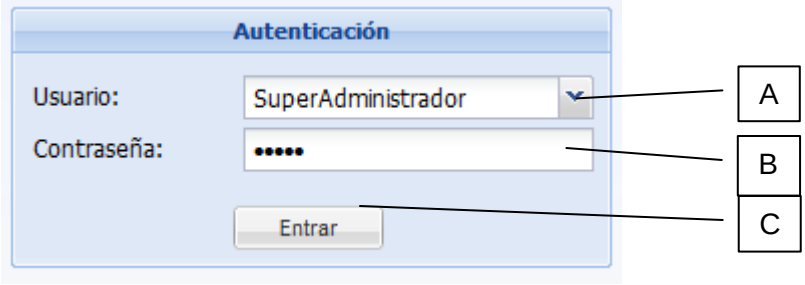
Figura 3. Diagrama de Casos de Uso del Sistema

2.7. Especificación de los Casos de Uso

CU1. Autenticar Usuario.

Caso de Uso:	**Autenticar Usuario.**
Actores:	Usuario, Administrador.
Resumen:	En este caso de uso, el sistema debe permitir que cada trabajador que interactúe con él, se autentique y tenga acceso a las funcionalidades que le corresponden.
Precondiciones:	El usuario tiene que estar registrado en la base de datos que emplea el sistema.
Referencias	RF3, RNF12.
Prioridad	Crítico.

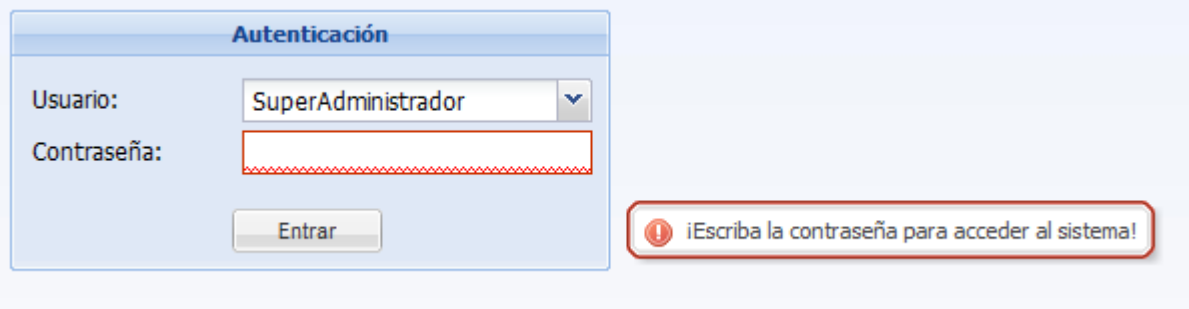
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1- El usuario ingresa su nombre de usuario (A), su contraseña (B) y oprime el botón Entrar (C).	2- El sistema lo autentica, verifica su identidad, emite un mensaje de bienvenida y muestra la interfaz de la aplicación, con las funcionalidades habilitadas según los permisos otorgados al usuario y su rol.
Prototipo de Interfaz	

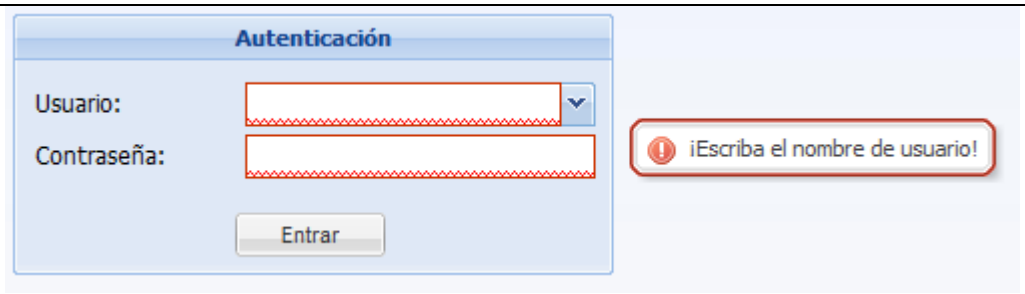


Flujos Alternos

Acción del Actor	Respuesta del Sistema
	2.1- Si el usuario no está autorizado o ingresa algún dato mal, el sistema muestra un cartel con el error, le ordena repetir la acción, y reinicia la interfaz de usuario. El sistema remite al usuario a la acción 1. En caso de errores leves (campos en blancos), se notifican los mismos marcando en rojo el campo con el error y mostrando un comentario.

Prototipo de Interfaz





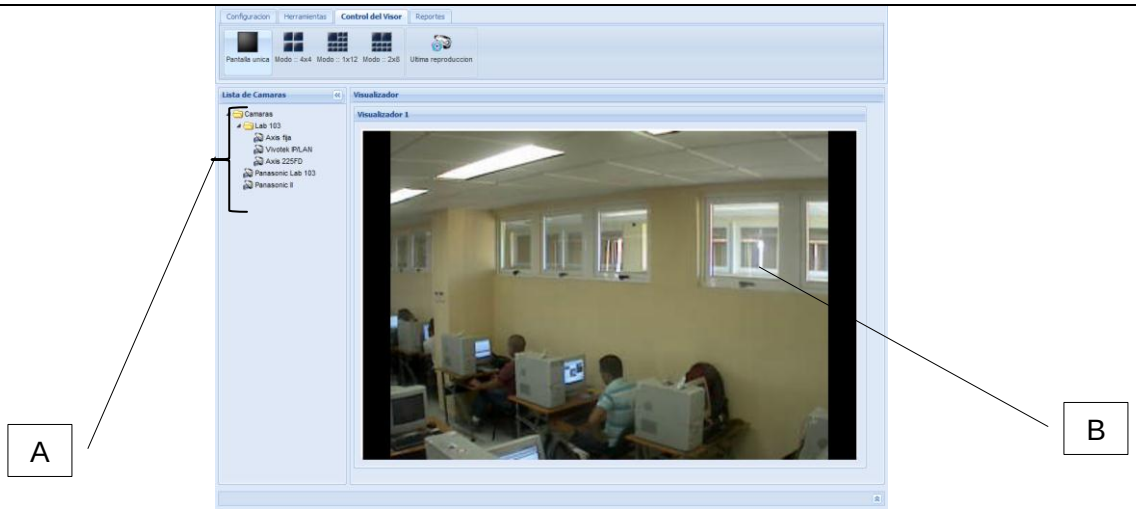
Poscondiciones	Cuando el trabajador esté interactuando con el sistema, ya tiene los privilegios otorgados y solo tiene permiso a las funcionalidades que realiza su rol.

CU2. Visualizar Flujo de Video.

Caso de Uso:	**Visualizar Flujo de Video.**
Actores:	Usuario, Administrador.
Resumen:	En este caso de uso, el sistema permite visualizar los flujos de video que estén tomando las cámaras activas en ese momento.
Precondiciones:	El usuario tiene que estar autenticado en el sistema, y tener permiso para visualizar los flujos de video.
Referencias	RF1, RF 1.1, RF 1.2.
Prioridad	Crítico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1- El usuario selecciona la cámara que desee del árbol de cámaras (A) y la arrastra hacia la pantalla (B) o, en el menú contextual del árbol, selecciona la pantalla (B) en que será	2- El sistema muestra el flujo de video que emite la cámara en ese momento.

reproducido el flujo de video de la cámara.	
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
2.4- El usuario instala el plugin necesario desde el enlace al instalador en la región Sur de la ventana, y reinicia el explorador.	2.1- El sistema detecta que la cámara no está funcionando y emite un mensaje para informar al usuario. 2.2- El flujo de video demora en cargar, por lo que la pantalla mostrará un mensaje expresando el estado del sistema. 2.3- El plugin requerido para la visualización no se encuentra instalado, por lo que la pantalla mostrará un cartel con una alerta del plugin faltante. 2.5- Se vuelve a la operación 1 del flujo normal de eventos.
Prototipo de Interfaz	



Poscondiciones	

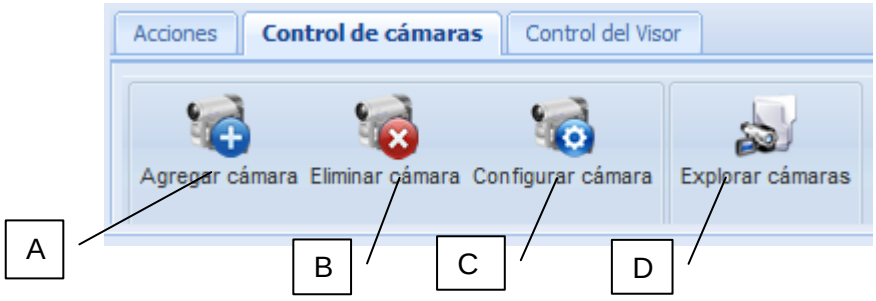
CU3. Gestionar Cámara

Caso de Uso:	**Gestionar Cámara.**
Actores:	Usuario, Administrador.
Resumen:	En este caso de uso, el usuario puede adicionar, eliminar o configurar cualquier cámara implantada en el sistema, así como, explorar las propiedades de las mismas.
Precondiciones:	El usuario tiene que estar autenticado en el sistema, y tener permisos para adicionar, configurar o eliminar cámaras.
Referencias	RF2, RF 2.1, RF 2.2, RF 2.3, RF 2.4.
Prioridad	Crítico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1- El usuario selecciona una de las siguientes	

opciones: a) Adicionar cámara (A). b) Configurar cámara (C). c) Eliminar cámara (B). d) Explorar cámaras (D).	2- El sistema ejecuta las siguientes acciones: a) Si el usuario decide adicionar cámara, ir a la sección Adicionar Cámara. b) Si el usuario decide actualizar cámara, ir a la sección Configurar Cámara. c) Si el usuario decide eliminar cámara, ir a la sección Eliminar Cámara. d) Si el usuario decide explorar cámaras, ir a la sección Explorar Cámaras.

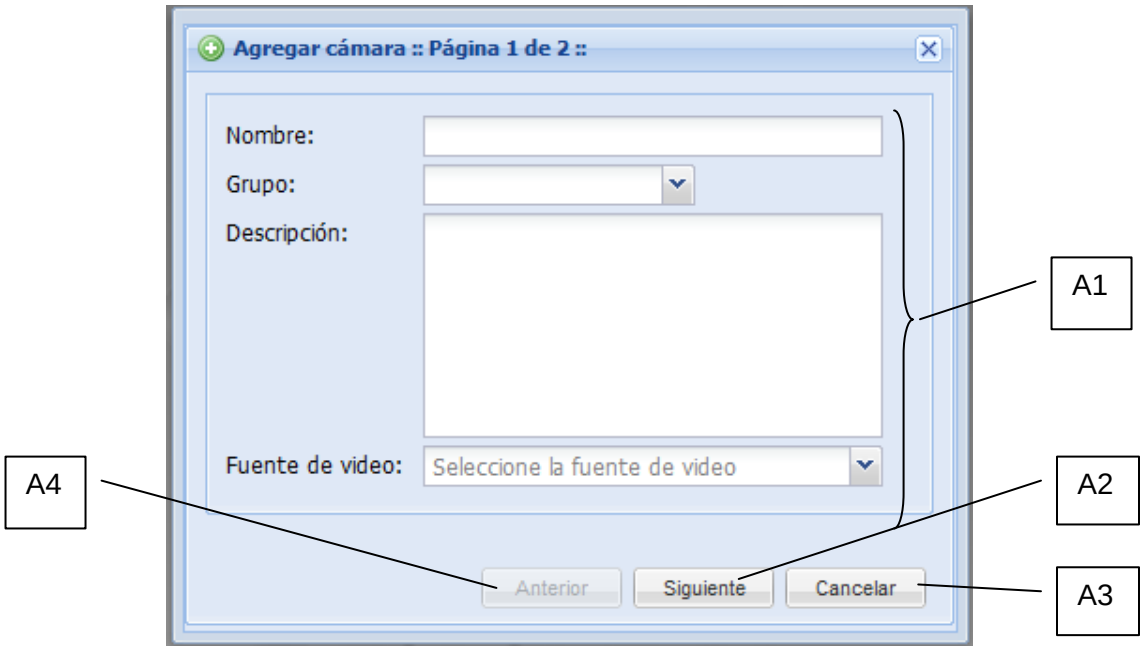
Prototipo de Interfaz



Sección Adicionar cámara	
Acción del Actor	**Respuesta del Sistema**
1- El usuario oprime el botón Adicionar cámara (A) de la pestaña Control de cámaras.	2- El sistema muestra un formulario con todos

	los campos que el usuario debe llenar (A1) correspondientes al primer paso (Formulario 1) para adicionar una nueva cámara.
3- El usuario llena los campos correspondientes para realizar la acción y oprime el botón Siguiente (A2).	4- El sistema valida los datos y, en caso de éxito, muestra otro formulario a completar según la fuente de video seleccionada.
5- El usuario llena los campos correspondientes para realizar la acción y oprime el botón Aplicar del formulario de la fuente de video.	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
3.1- El usuario oprime el botón Cancelar (A3). 5.1- El usuario oprime el botón Anterior (A4).	3.2- Se cierra el formulario (Formulario 1). 5.2- Se muestra el Formulario 1 con los datos insertados. 4.1- Si la validación es fallida el sistema muestra un mensaje informándole sobre el error y lo remite nuevamente a realizar la operación 3 del flujo normal de eventos. En caso de errores leves (campos en blancos), se notifican los mismos marcando en rojo el campo con el error y mostrando un comentario.
Prototipo de Interfaz	

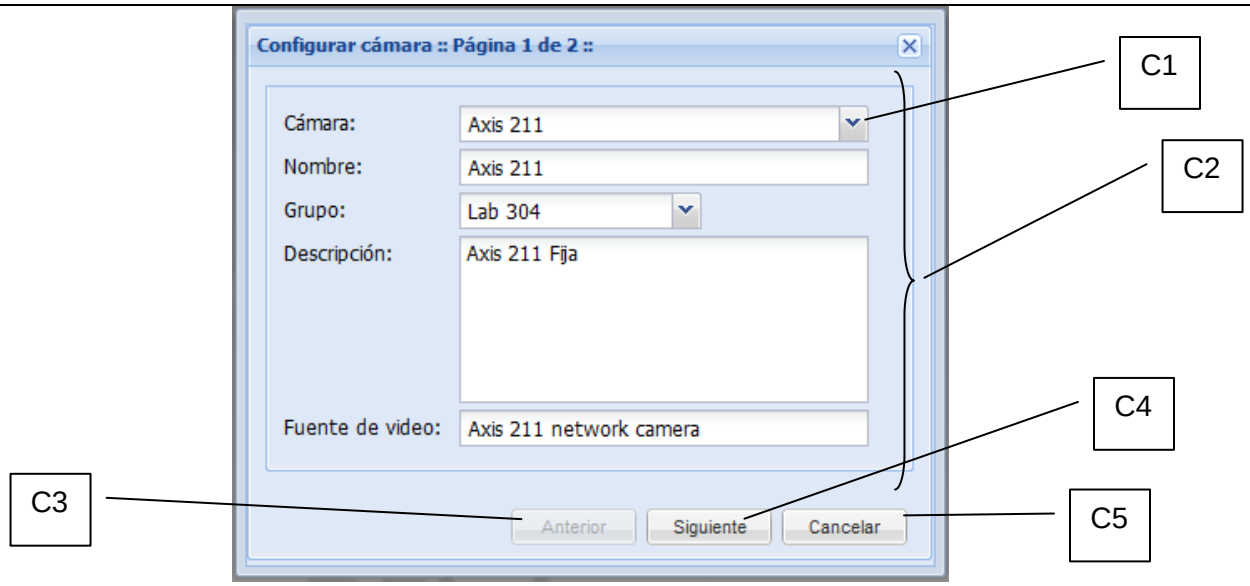
Agregar cámara. (Formulario 1)



Sección Configurar cámara

Acción del Actor	Respuesta del Sistema
1- El usuario oprime el botón Configurar cámara (C) de la pestaña Control de cámaras.	
	2- El sistema muestra un formulario que contiene todas las cámaras registradas en el sistema, para que se seleccione la que se desee configurar (C1).
3- El usuario selecciona la cámara que desea actualizar (C1).	
	4- El sistema muestra todos los datos de la cámara seleccionada (C2).
5- El usuario modifica los campos de su interés y	

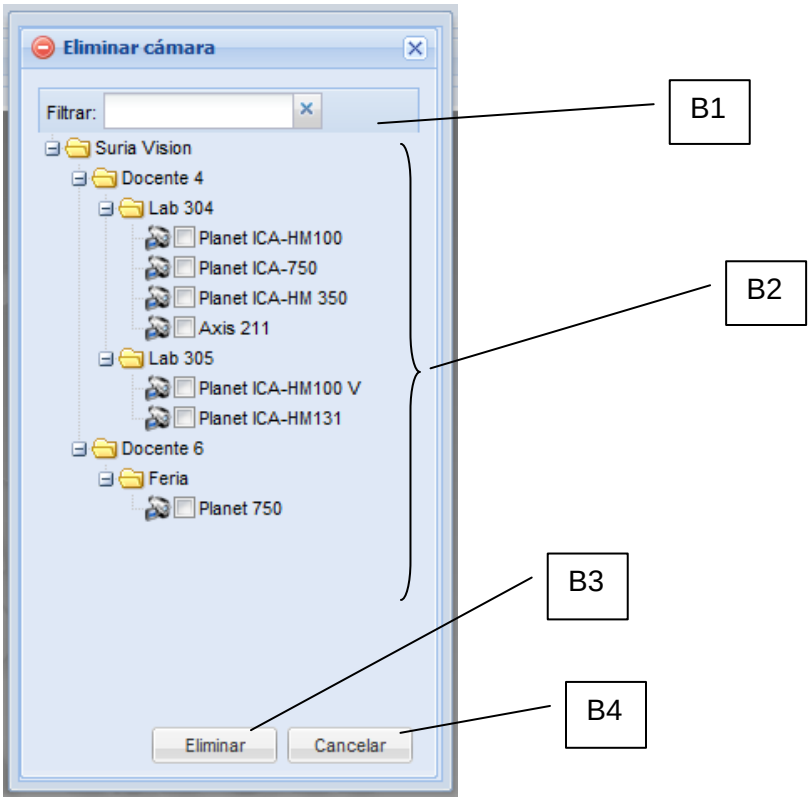
oprime Siguiente (C4). 7- El usuario llena los campos correspondientes para realizar la acción y oprime el botón Aplicar del formulario de la fuente de video.	6- El sistema muestra la configuración de la cámara según la fuente de video de la misma en un formulario nuevo. 8- El sistema valida los campos y en caso de éxito oculta el formulario.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	8.1- Si la validación es fallida, el sistema muestra un mensaje informando sobre el error y lo remitiendo al usuario a realizar la operación 5 del flujo normal de eventos. En caso de errores leves (campos en blancos), se notifican los mismos marcando en rojo el campo con el error y mostrando un comentario.
Prototipo de Interfaz	



Sección Eliminar cámara	
Acción del Actor	**Respuesta del Sistema**
1- El usuario oprime el botón Eliminar cámara (B) de la pestaña Control de cámaras. 3- El usuario selecciona la(s) cámara(s) que desea eliminar (B2) o busca la cámara deseada a través del campo (B1) y la selecciona y oprime el botón Eliminar (B3).	2- El sistema muestra un panel con un árbol conformado por las cámaras a las que puede acceder el usuario (B2). 4- El sistema procede a eliminar la cámara y se muestra un mensaje informando que la operación fue desarrollada con éxito.
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**

3.1- El usuario cancela la acción oprimiendo el botón Cancelar (B3).	3.2- El sistema oculta el formulario Eliminar cámara.

Prototipo de Interfaz

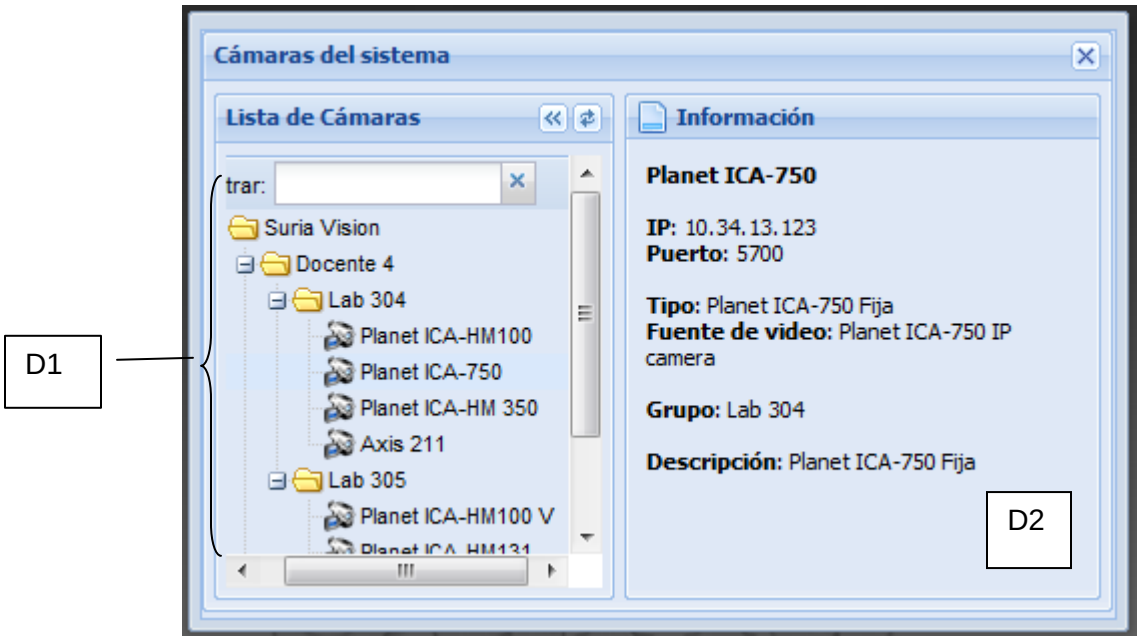


Sección Explorar cámaras

Acción del Actor	Respuesta del Sistema
1- El usuario oprime el botón Explorar cámaras (D) de la pestaña Control de cámaras.	2- El sistema muestra un formulario con todas las cámaras que están ya implantadas para trabajar con el sistema.
3- El usuario selecciona la cámara que desea	

explorar del árbol de cámaras (D1).	4- El sistema muestra al usuario una sección del formulario (D2) con las propiedades principales de la cámara seleccionada.

Prototipo de Interfaz



Poscondiciones	

2.8. Conclusiones

En el desarrollo del capítulo se realizó la presentación de la solución propuesta, se describen detalladamente las funcionalidades que debe cumplir el sistema basadas en las necesidades del cliente, las cuales quedan explícita en la confección de los requisitos funcionales y no funcionales que se redactan a partir de la comunicación con el cliente desde la primera fase del desarrollo de un software. Para que la aplicación quede con calidad debe cumplir en su terminación con todas estas funcionalidades. Con las tablas de descripción y diagramas mostrados se logra un mayor entendimiento entre el equipo de desarrollo y además se obtienen artefactos de gran importancia para la siguiente fase de análisis que propone el ciclo de desarrollo para la aplicación.

CAPÍTULO 3. **CONSTRUCCIÓN DE LA SOLUCIÓN PROPUESTA**

3.1. Arquitectura

La arquitectura de un sistema de software es la organización o estructura de los componentes importantes que interactúan en el mismo. Por tanto, resulta necesario conocer el entorno en el que se desenvuelve la aplicación y antes de definir la arquitectura del sistema modular propuesto, conocer la arquitectura del Sistema de Video Vigilancia al que se relaciona el módulo y el subsistema externo que interactúa con él. En este caso, tal y como se explica en el *Epígrafe 2.2 Propuesta del Sistema,* del capítulo anterior, el Módulo Web toma la información que necesita para su funcionamiento del Gestor Central de Información, con el que se comunica para realizar la mayoría de las operaciones.

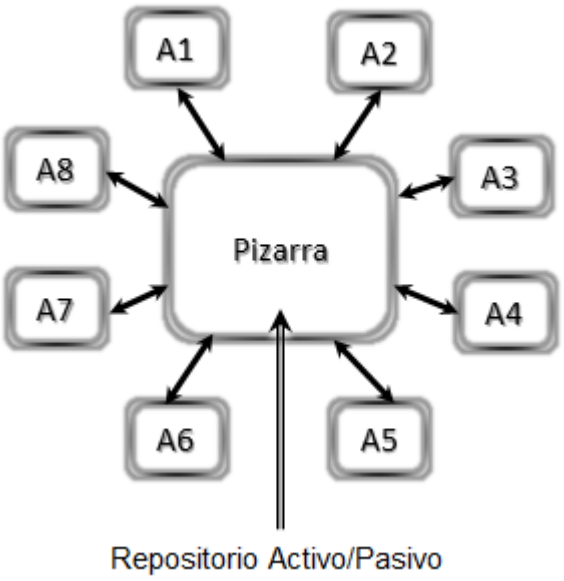
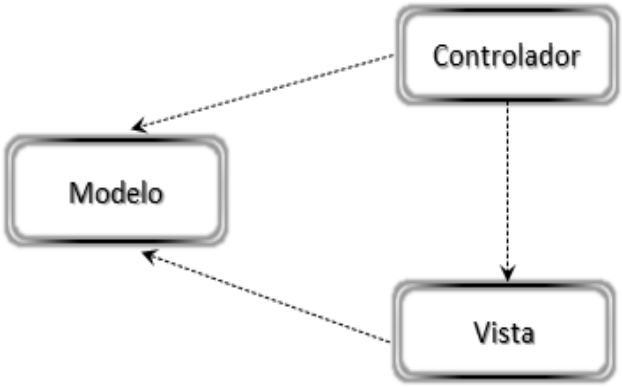


Figura 4. Arquitectura en Pizarra

El Sistema, posee una arquitectura de pizarra en forma de tablero de control que "…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. Este se denomina Repositorio Activo, entrega y recibe información de los agentes y coordina su funcionamiento. El Gestor sería el Repositorio Activo [y el Módulo Web uno de los agentes autónomos que funcionaría con el Sistema]." (8)



Por su parte, el Módulo Web está diseñado bajo el patrón de arquitectura Modelo-Vista-Controlador (MVC) que implementa por defecto las aplicaciones ASP.net .Este es un patrón de diseño que considera dividir una aplicación en tres módulos claramente identificables y con funcionalidades bien definidas: el Modelo, las Vistas y el Controlador. (9)

Figura 5. Patrón Modelo-Vista-Controlador (MVC)

Actualmente, es muy utilizado para el desarrollo de aplicaciones web de gran alcance, permitiendo desarrollar proyectos más extensibles que cualquier aplicación basada en otro patrón.

El uso de este patrón, entre otras ventajas proporciona al usuario vistas siempre actualizadas sin que el programador tenga que estar preocupado por esa tarea, además de facilitar el trabajo a los desarrolladores en caso de cambio, por su independencia de funcionalidades para cada capa. Las capas que lo componen tienen gran interacción entre ellas completando el funcionamiento del patrón, cada una por separada se encarga de su trabajo pero interactúan entre sí para llegar a un resultado final. Con el uso de este patrón, se logra un sistema donde el Modelo contendría todos los datos necesarios para el funcionamiento de la aplicación y se encargaría de la gestión de los datos tomados o entregados al Gestor. Por otra parte, la Vista muestra al usuario la información que contiene el Modelo y se actualiza al ocurrir algún cambio en él y el Controlador controla la interacción en la aplicación de la Vista con el Modelo, responde a las acciones seleccionadas por el usuario en la Vista y manipula el Modelo para solucionar dichas acciones. Para un mejor entendimiento del sistema, se debe relacionar el Gestor como parte del Modelo de datos de la aplicación.

3.2. Patrones de Diseño de Software

Para comenzar la implementación de un software no basta con definir la arquitectura, es necesario además establecer directrices que permitan lograr un sistema bien estructurado, para así construir una solución eficaz. Estas directrices son los llamados Patrones de Diseño de Software. Para satisfacer las necesidades de la sociedad cada día se hace más necesario desarrollar más software de gran alcance y complejidad, en lo que son de gran utilidad los patrones de diseño empleados específicamente como mecanismos de reutilización.

En el presente trabajo se emplearán cinco patrones GRASP[21], estos son:

1. Experto
2. Creador
3. Bajo Acoplamiento
4. Alta Cohesión
5. Controlador

Empleando el patrón Experto se garantiza que cada clase del sistema asuma las responsabilidades que le conciernen, según las funcionalidades que se quieren implementar y a partir de la información que posee; por lo que cada clase contendrá la información necesaria para cumplir su responsabilidad. Al utilizar el patrón Creador cada clase instancia y crea las clases que colaborarán con la misma para cumplir su responsabilidad, tratando de lograr siempre un Bajo acoplamiento y una Alta cohesión. Con el cumplimiento de estos dos últimos patrones se logra que cada clase recurra solamente a las clases que son imprescindibles para su trabajo y tenga asociada las responsabilidades que le corresponden de acuerdo con su comportamiento. Al emplear el patrón Controlador, se estructura el sistema con una clase controladora para cada caso de uso para que sea esta quien, respetando las condiciones del patrón arquitectónico Modelo-Vista-Controlador, se encargue de los eventos y funcionalidades que representan dicho caso de uso.

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

También se apoyó el diseño con la utilización de patrones GoF[22]:

1. Singleton
2. Fachada

Se empleó el patrón de creación Singleton al desarrollar la integración del Módulo Web, en desarrollo, al Sistema de Video Vigilancia Suria Vision. Este sistema utiliza como tecnología de comunicación .NetRemoting para la interacción con múltiples clientes compartiendo con ellos información. Este patrón, en el presente sistema, posibilita resolver el problema del duplicado de información o de discordancia entre los módulos funcionales al proporcionar una sola instancia de las clases que brinda para la comunicación de los módulos secundarios con el orquestador del sistema. Esta clase a la que se accede para la comunicación con el Gestor es un objeto Fachada implementándose par esto el segundo patrón estudiado, que oculta el funcionamiento de todo este subsistema para el agente externo, este solo accedería a su único punto de acceso público sin conocer el funcionamiento de este subsistema.

En el flujo de trabajo donde se encuentra el desarrollo de la aplicación se propone llegar a una especificación de cómo implementar el sistema cumpliendo con la realización de todos los requisitos planteados en la fase anterior.

3.3. Flujo de Trabajo Análisis y Diseño

3.3.1. Diagramas del análisis

El análisis se encarga de describir que hace el sistema apoyándose solamente en los requisitos funcionales y en el análisis de la descripción de cada Caso de Uso. Este facilita la construcción del diseño.

[22] **Patrones GoF (*Gang of Four*):** reconocido grupo de patrones de diseño para la programación orientada a objetos.

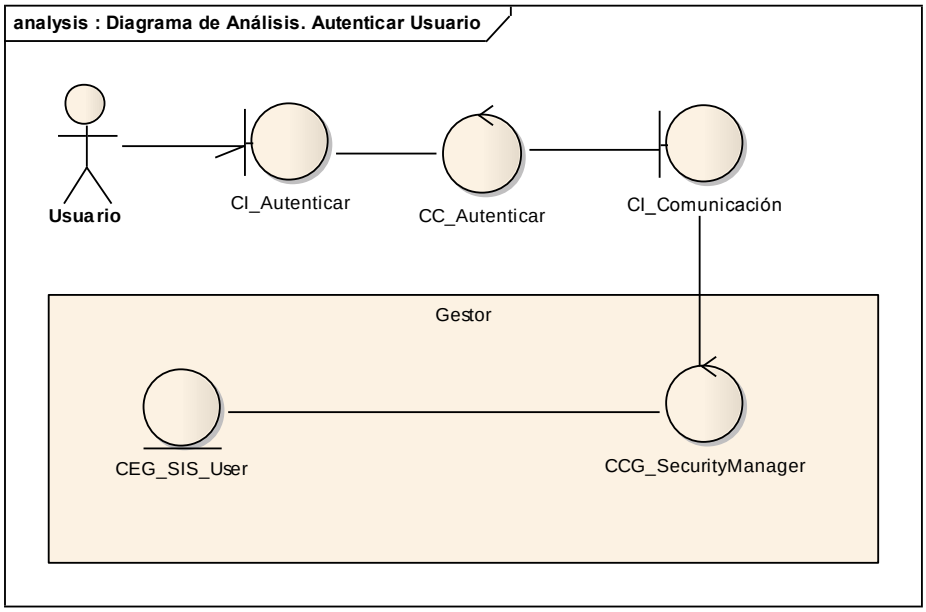


Figura 6. Diagrama de análisis. CU Autenticar Usuario.

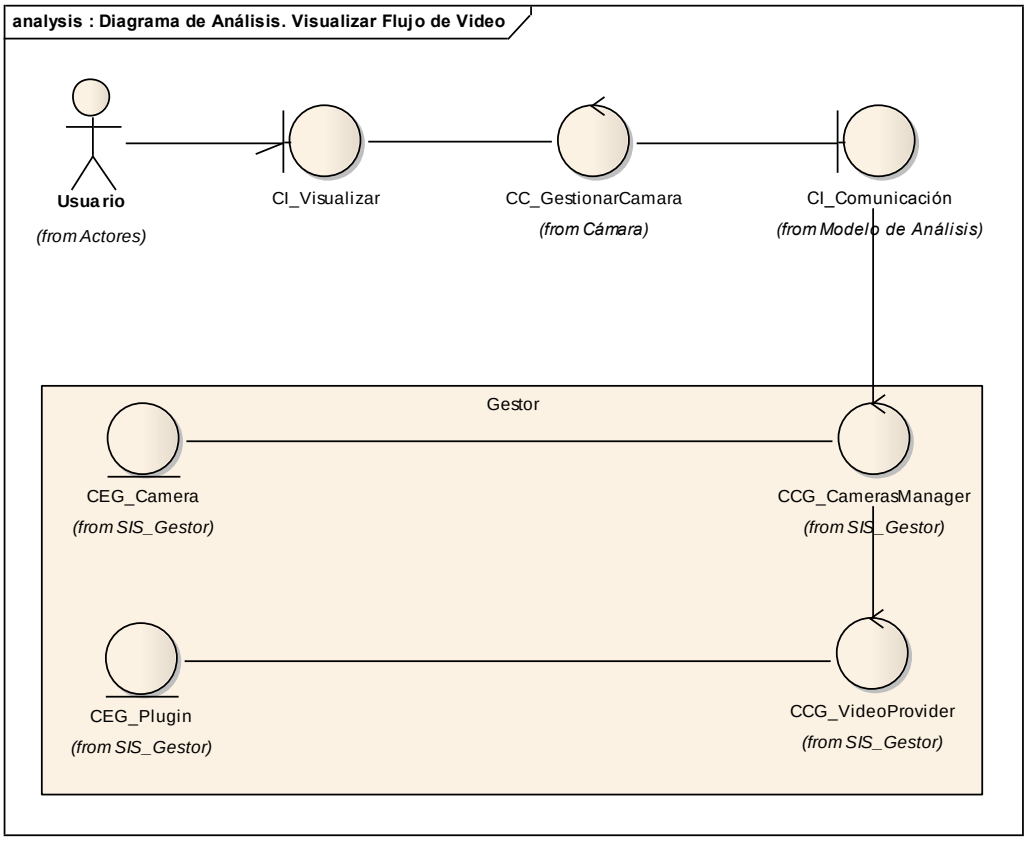


Figura 7. Diagrama de análisis. CU Visualizar Flujo de Video

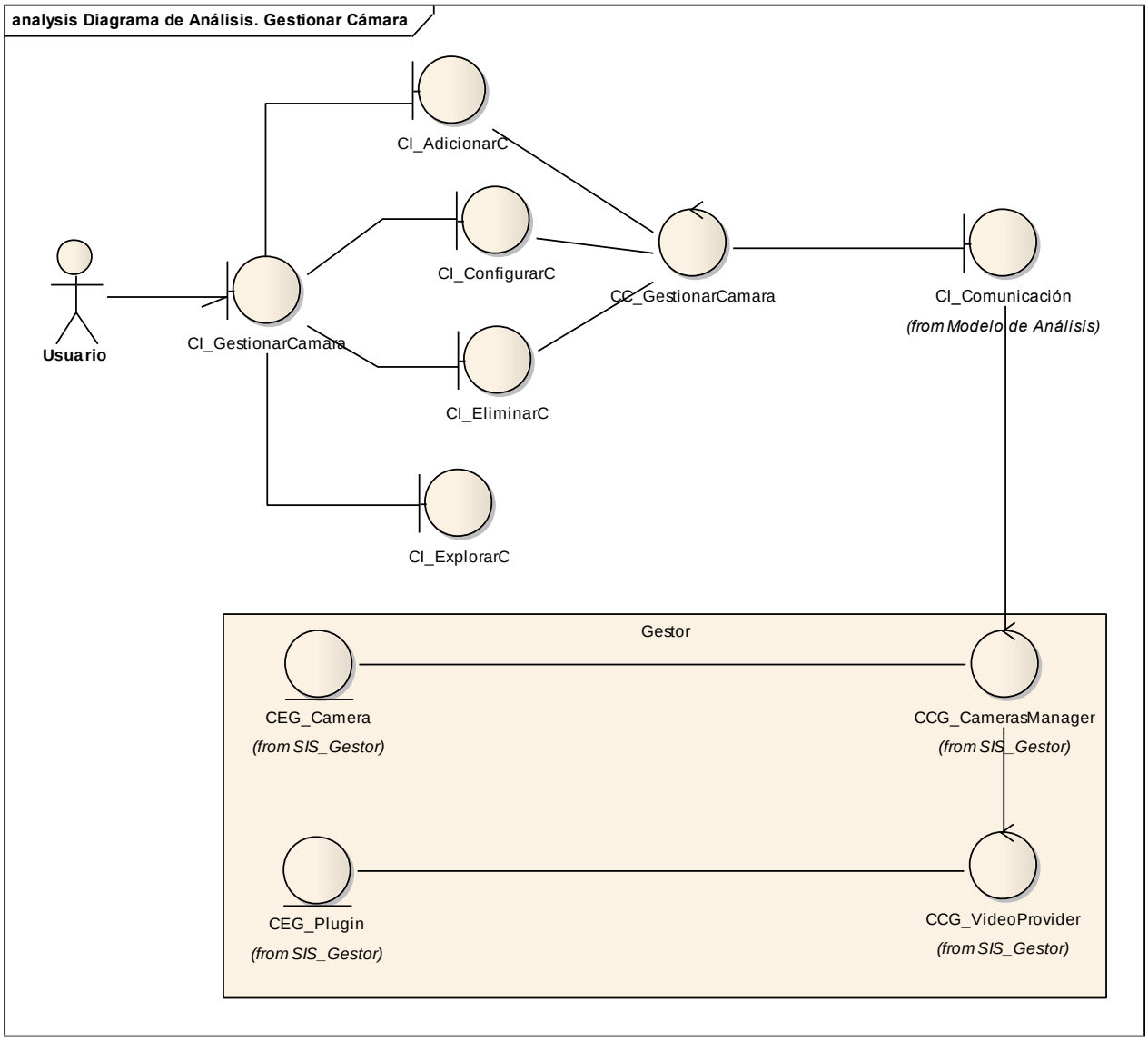


Figura 8. Diagrama de análisis. CU Gestionar Cámara

3.3.2. Diagramas del diseño

El Modelo de diseño describe detalladamente el funcionamiento de los casos de uso centrándose en el cumplimiento de los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación. Este modelo facilita el trabajo de los desarrolladores

ya que es tomado como principal entrada de la fase de implementación porque describe las interfaces y clases que se van a utilizar.

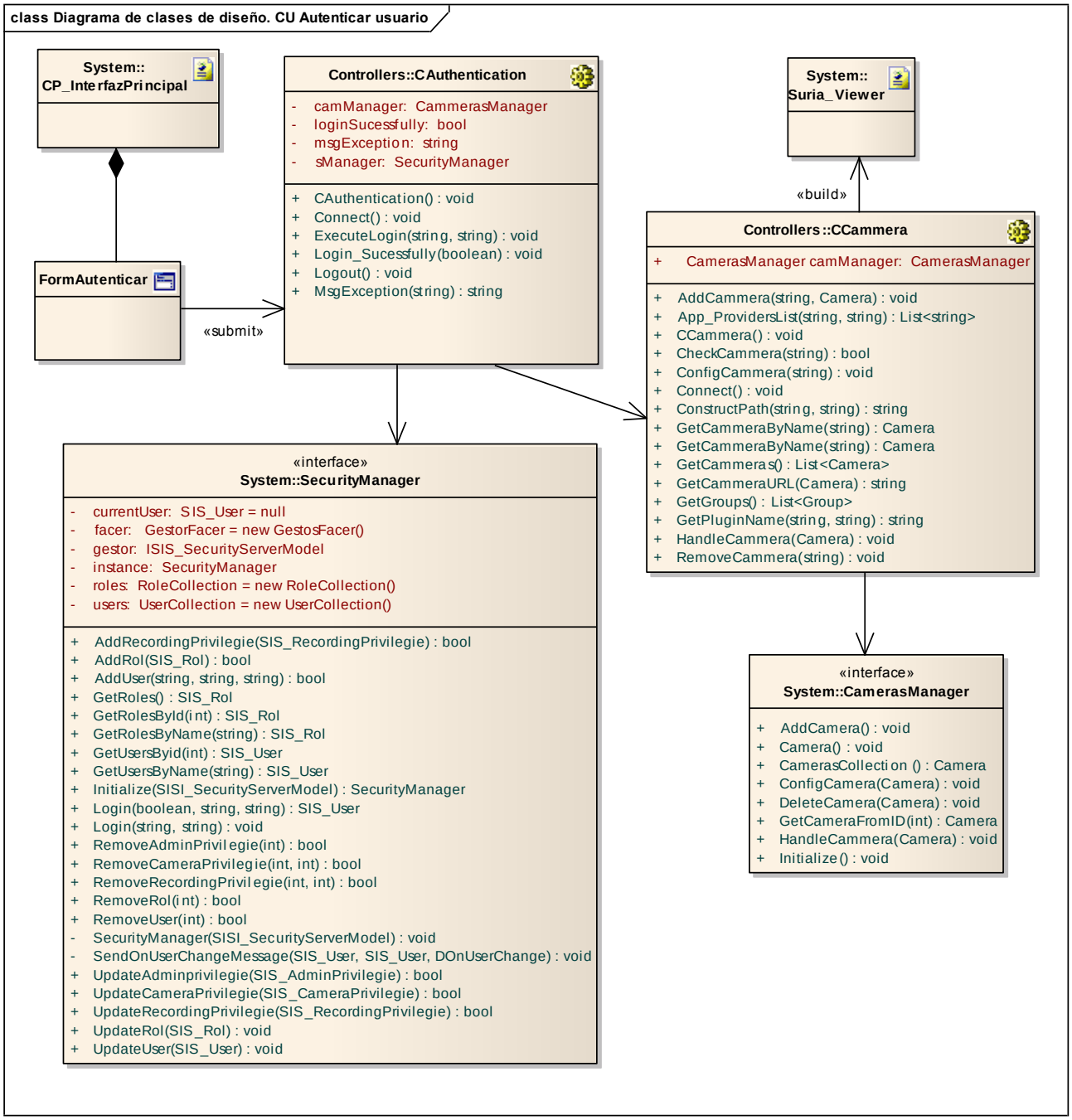


Figura 9. Diagrama de clases de diseño. CU Autenticar Usuario

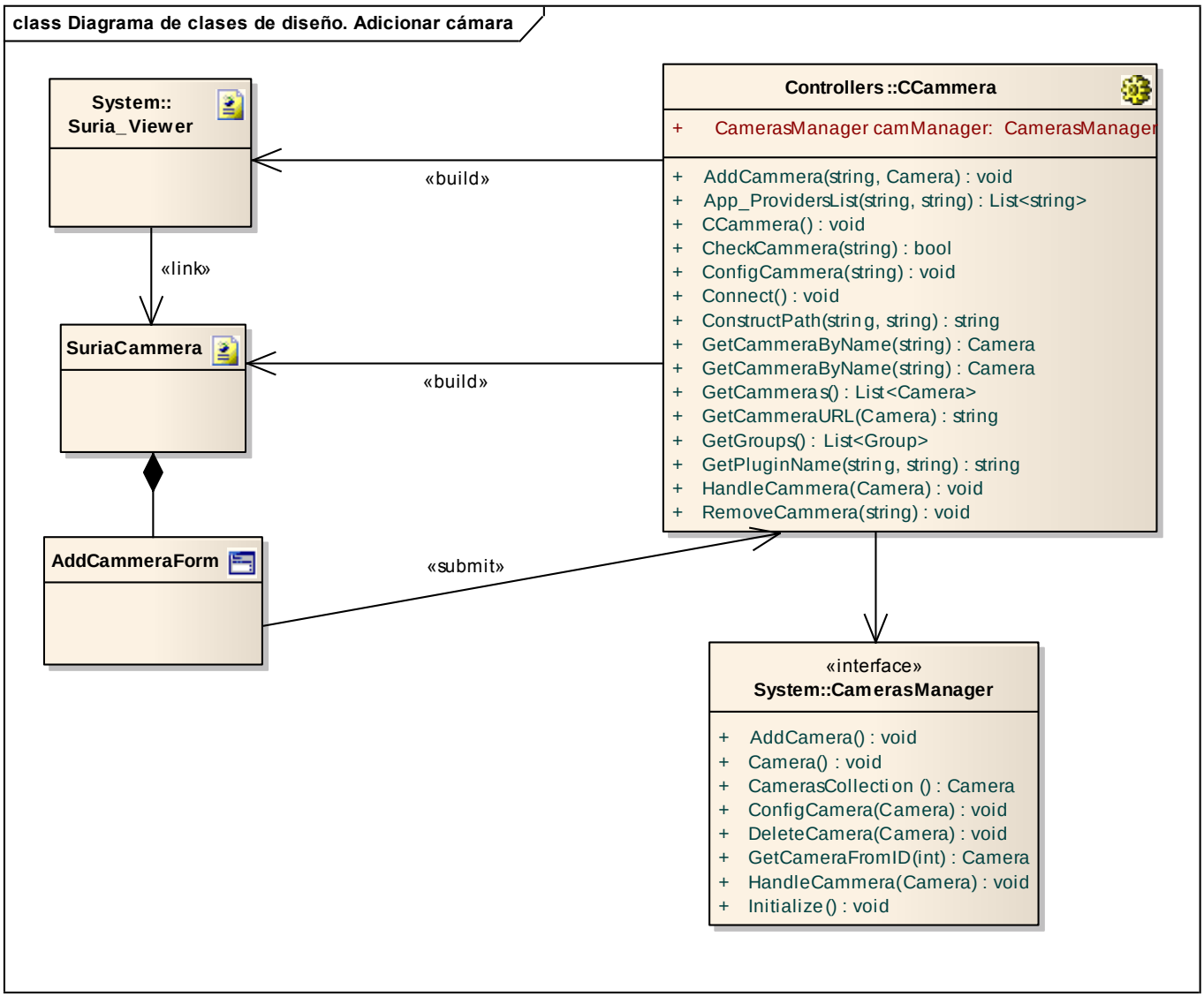


Figura 10. Diagrama de clases de diseño. Adicionar cámara

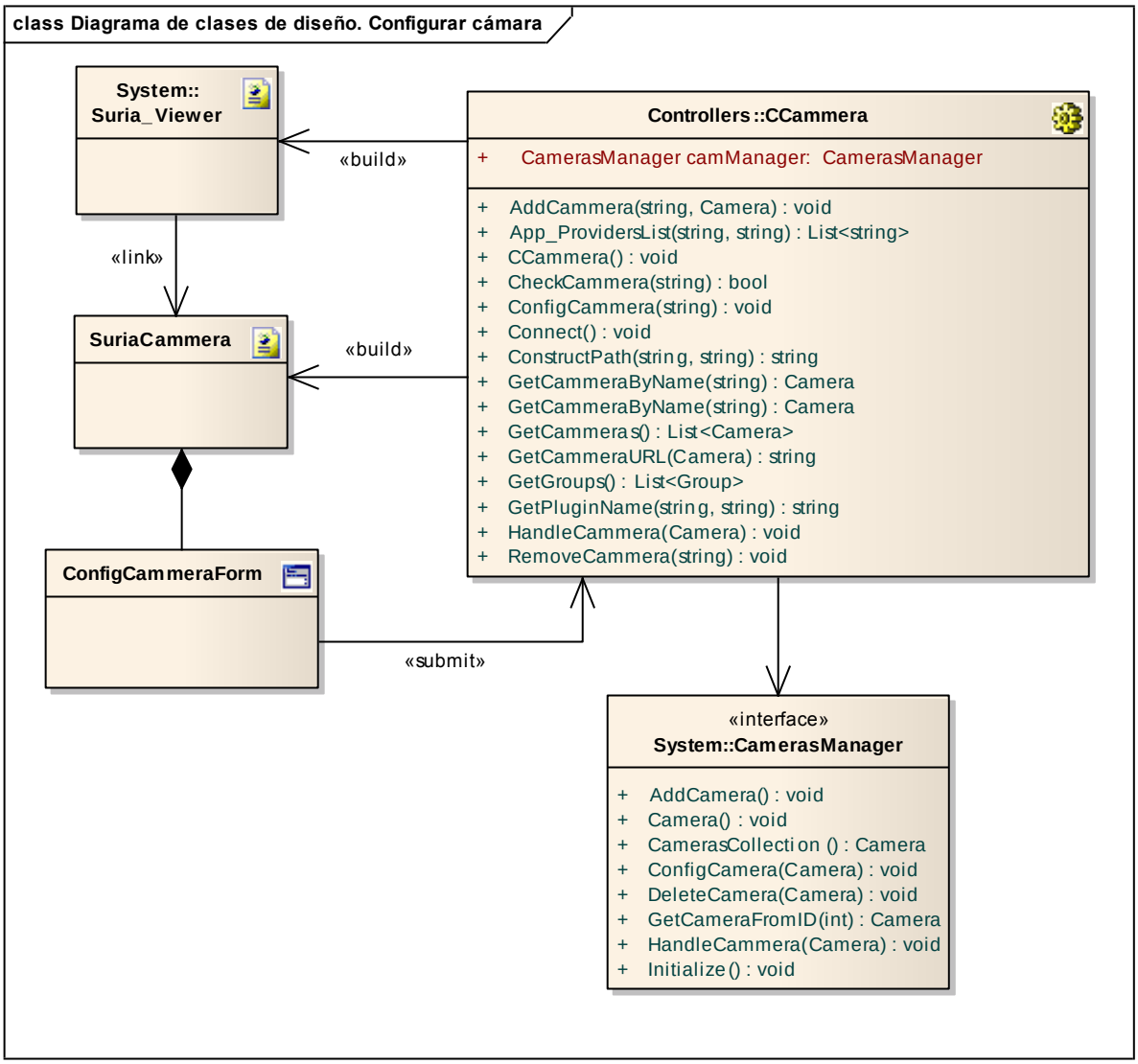


Figura 11. Diagrama de clases de diseño. Configurar cámara

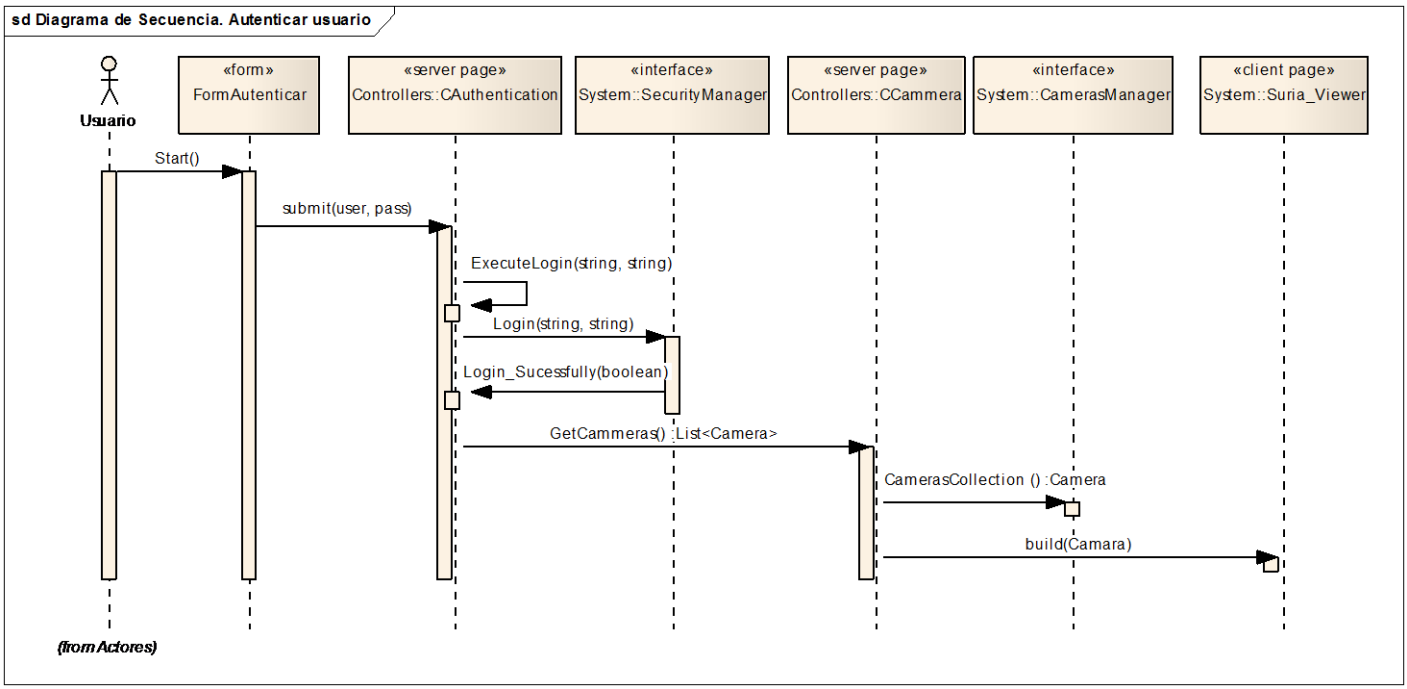


Figura 12. Diagrama de secuencia. CU Autenticar Usuario

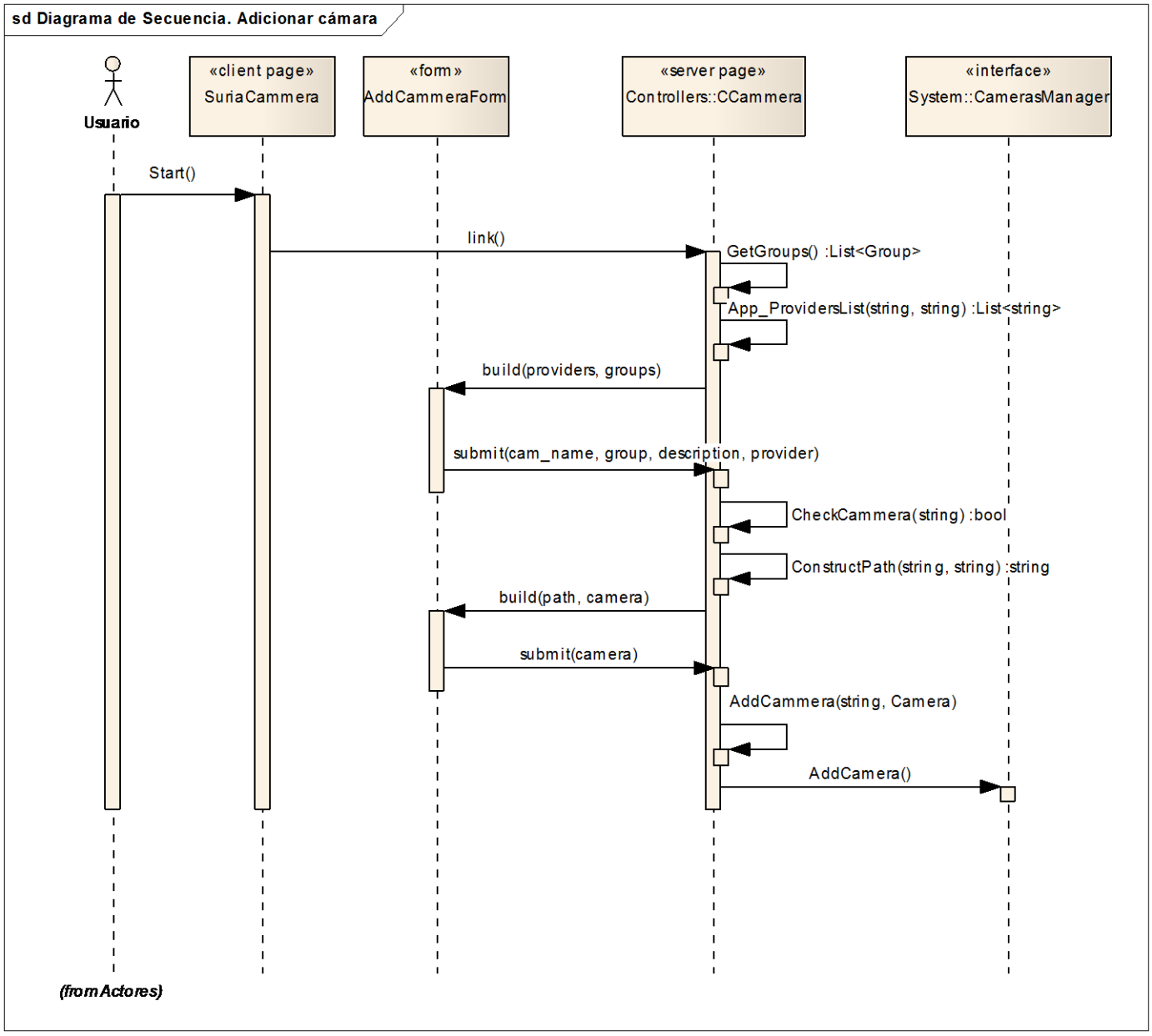


Figura 13. Diagrama de secuencia. Adicionar cámara

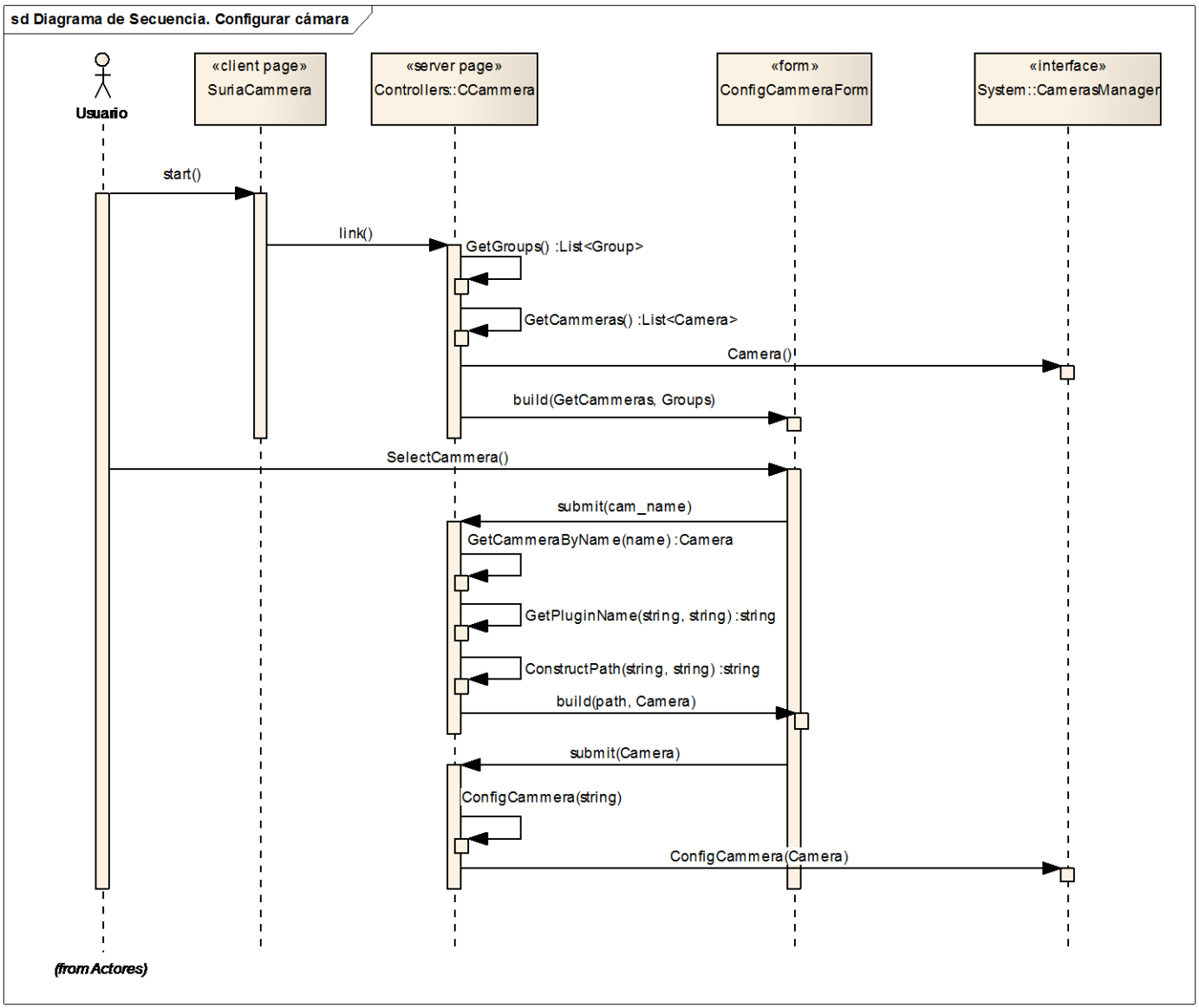


Figura 14. Diagrama de secuencia. Configurar cámara

3.4. Flujo de trabajo Implementación

En la implementación, a partir de los resultados obtenidos en el diseño se describe el sistema en términos de componentes, que son las partes modulares que componen dicho sistema y que junto a las clases y otros artefactos, ya sean ficheros binarios, ejecutables u otros, permiten la implementación. Primeramente, es necesario detallar el Modelo de datos del sistema, que modela la estructura de cómo se almacena toda la información requerida en el sistema. Luego se detalla el Modelo de implementación que está compuesto por el Diagrama de despliegue y el Diagrama de componentes. Ambos diagramas en conjunto, describen los componentes a construir, su organización y dependencias entre los nodos físicos en la que funcionará la aplicación.

3.5. Modelo de datos

El Modelo de datos es el modelo utilizado para el diseño de la base de datos. En el caso de la solución presentada en este documento, la base de datos queda estructurada según las necesidades del Sistema de Video Vigilancia Suria Vision donde los datos contenidos en la misma son accedidos y manipulados a través del anteriormente mencionado Gestor Central de Información. Para poder implementar las funcionalidades de la solución presentada en el capítulo anterior, se hace necesario el uso de la base de datos del sistema, las tablas que intervienen en el proceso se modelan a continuación:

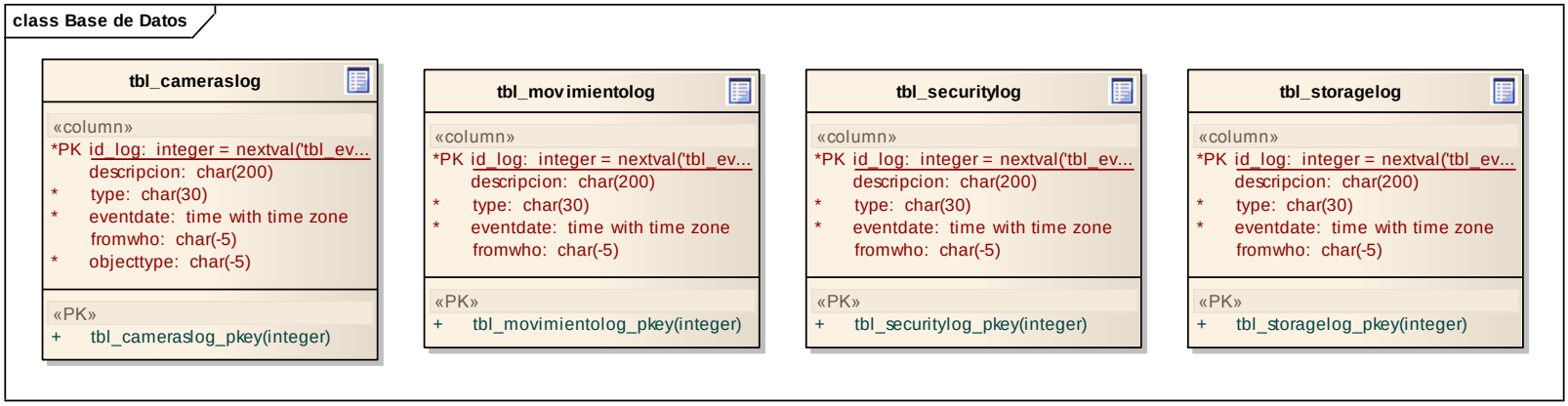


Figura 15. Modelo de datos. Tablas para los registros del sistema

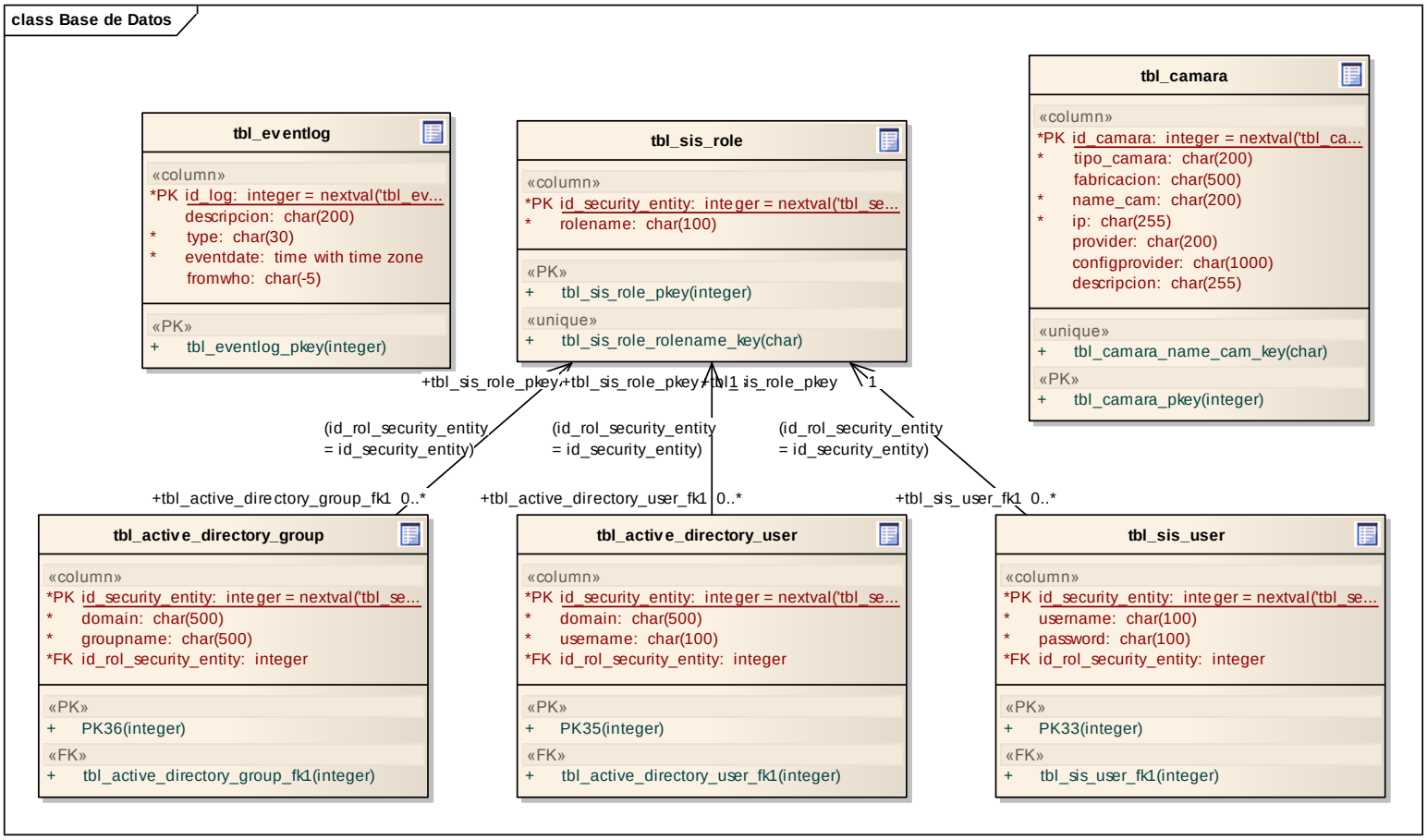


Figura 16. Modelo de datos. Tablas del sistema de seguridad, tabla "tbl_eventlog" para los registros de eventos y tabla "tbl_camara"

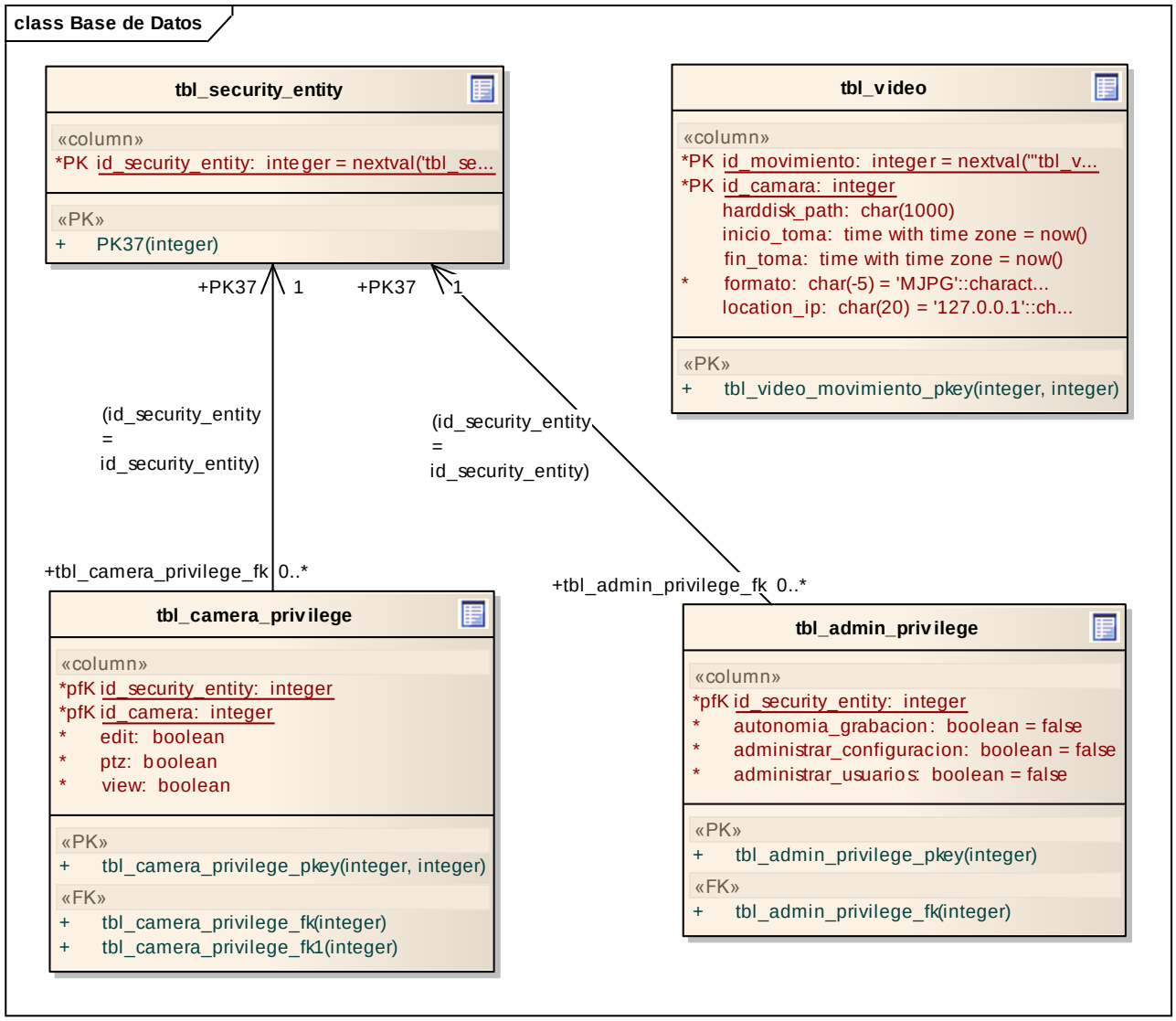


Figura 17. Modelo de datos. Tablas para el sistema de permisos y tabla "tbl_video"

3.6. Modelo de implementación

3.6.1. Diagrama de despliegue

Según la metodología RUP, el diagrama de despliegue tiene como objetivo capturar la configuración de los elementos de proceso y las conexiones entre elementos de proceso, en el sistema. (6) Además, permite comprender el entorno de ejecución física del sistema y para comprender las cuestiones de distribución. (6) Por tanto, constituye el medio principal para conocer el ambiente donde se desarrolla el sistema.

La aplicación planteada cuenta con cinco elementos fundamentales: PC Cliente, Servidor web, Gestor, Servidor de Base de Datos, y Cámara IP.

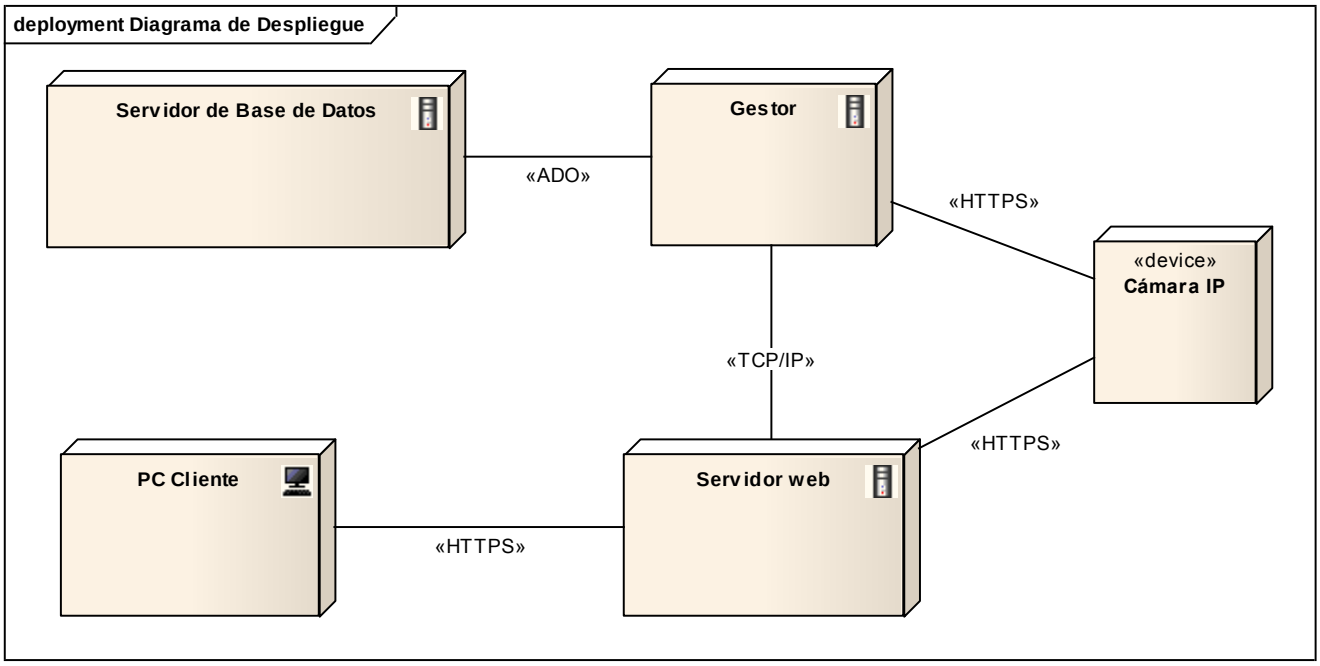


Figura 18. Diagrama de despliegue

3.6.2. Diagrama de componentes.

En el ciclo de desarrollo de una aplicación, el diagrama de componentes resulta de gran importancia para los desarrolladores. Este permite identificar los componentes físicos de la implementación, la estructura que conforman y los artefactos que los implementan. Este diagrama se utiliza para mostrar la estructura de alto nivel del Modelo de implementación en términos de subsistemas de implementación, y las relaciones entre elementos de implementación. (6) RUP enfatiza el uso del diagrama de componentes para modelar los subsistemas de implementación, los elementos de implementación significativos, y sus relaciones. (6)

A continuación se puede observar el Diagrama de componentes de la aplicación que se propone como solución en este trabajo (en el diagrama, SIS Web Viewer), y las dependencias de la misma con otros componentes del Sistema de Video Vigilancia.

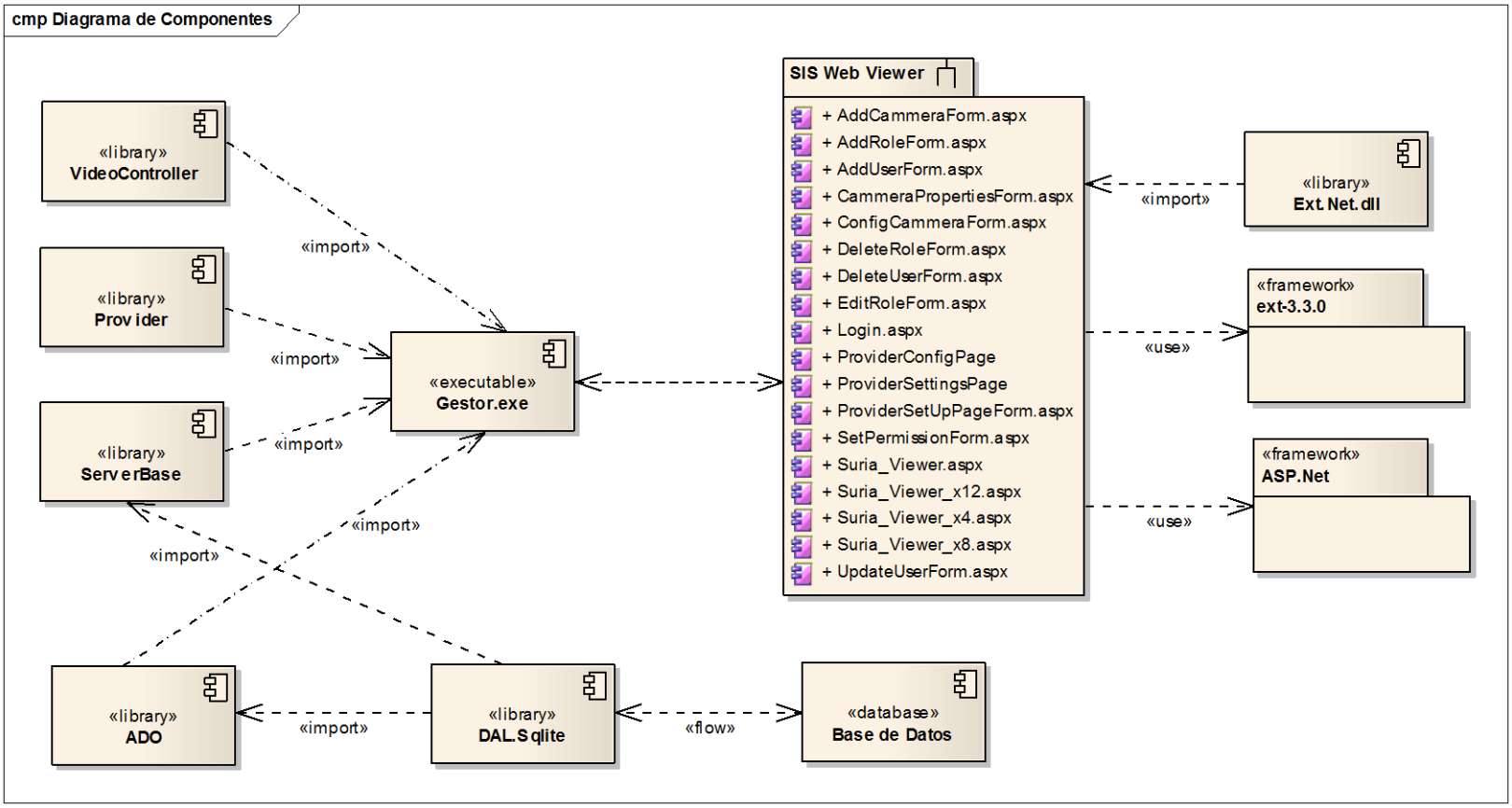


Figura 19. Diagrama de componentes

3.7. Flujo de trabajo de Pruebas

En la etapa de terminación del ciclo de desarrollo se realiza la fase de pruebas, que aunque para muchos no es de gran importancia es muy recomendable para comprobar la calidad del software. El objetivo de esta etapa es encontrar la mayor cantidad de errores en la ejecución de la aplicación. Se realizan a continuación pruebas unitarias a dos casos de uso críticos que se desarrollaron, estas pruebas son: las pruebas de caja blanca (estructurales) y las pruebas de caja negra.

3.7.1. Pruebas de Caja Blanca o Pruebas estructurales

Estas pruebas se realizan en aplicaciones habilitadas para la Web y garantizan que todos los enlaces estén conectados. En estas pruebas se revisa siempre el código. En este caso, se desarrollará esta prueba utilizando la técnica de camino básico. Con esta técnica, por cada caso de prueba es necesaria la construcción de un grafo que represente todas las variantes que tiene el algoritmo para darle solución al caso de uso.

Los pasos del diseño de pruebas a seguir mediante el camino básico son:

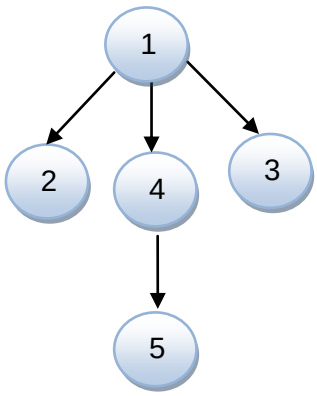
1. A partir del diseño o del código fuente, se dibuja el grafo de flujo asociado.
2. Se calcula la complejidad ciclomática del grafo de flujo.
3. Se determina el conjunto básico de caminos independientes.
4. Se preparan los casos de prueba que obliguen a la ejecución de cada camino del conjunto básico.

Pruebas al Caso de Uso Autenticar Usuario.

```
public void ExecuteLogin(string user, string password)
   {
     try 1
     {
       if (sManager.Login(user, password, true) != null) 1
       {
         LoginSucessfully = true; 2
```

```
                XML_Generator xmlGen = new XML_Generator(path, user); 2
                Alert = "¡Bienvenido al Sistema Suria Vision!"; 2
              }
              else
              {
                Alert = "¡Error, verifique los datos insertados por favor! El usuario o la contraseña es
        incorrecto."; 3
              }
            }
            catch (Exception ex) 4
            {
              LoginSucessfully = false; 5
              Alert = ex.Message; 5
            }
  }
```

Construcción del Grafo



Complejidad Ciclomática

V (G) = Número de Aristas – Número de Nodos + 2

V (G) = 4 – 5 + 2

V (G) = 3

Conjunto Básico de Caminos Independientes

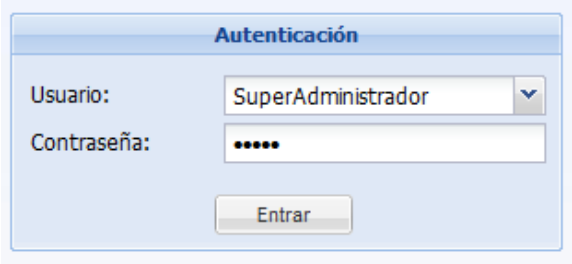
Camino 1: 1-2

Camino 2: 1-3

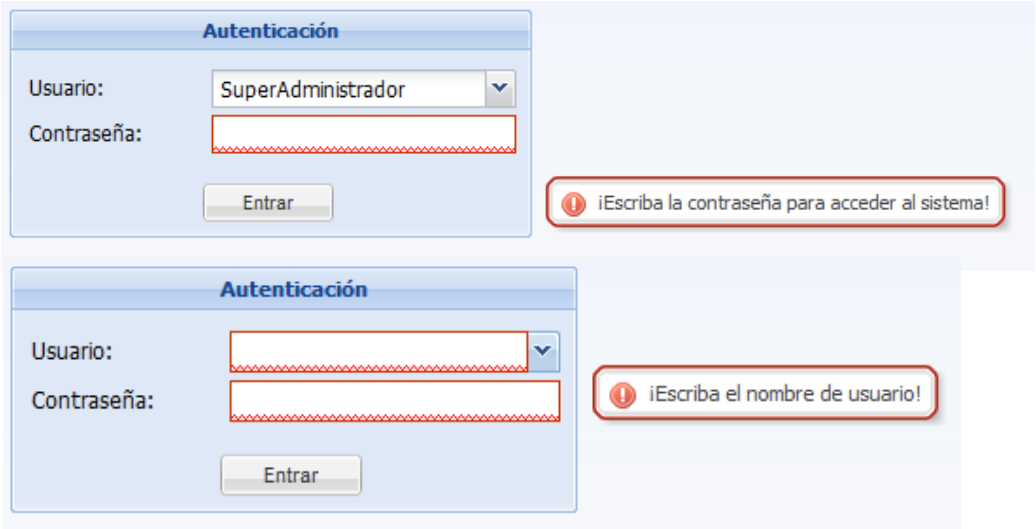
Camino 3: 1-4-5

Caso de pruebas

Para Camino 1: El usuario se autentica correctamente, introduce bien el usuario la contraseña y tiene permiso para acceder al sistema.



Para Camino 2: Se dejan campos en blancos necesarios para acceder al sistema.



Para Camino 3: Son insertados todos los datos correctamente pero no tiene permiso el usuario para entrar en la aplicación, es decir no está en la base de datos de trabajo consultada.

3.7.2. Pruebas de Caja Negra

Estas son las pruebas funcionales. Sólo se analiza la entrada de los datos y los posibles resultados ante las salidas reales de la aplicación.

Pruebas al Caso de Uso Gestionar Usuario.

Descripción General

El caso de uso se inicia cuando el administrador desea adicionar, eliminar o modificar los datos de algún usuario.

Condiciones de Ejecución

El sistema debe mostrar las opciones Agregar usuario, Editar usuario y Eliminar usuario.

Secciones a probar en el Caso de Uso.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo Central
1: Agregar Usuario.	EC 1. Agregar Usuario de forma correcta.	Esta funcionalidad permite agregar un usuario al sistema desde el Módulo Web. Entrando en este escenario todos los datos de forma correcta y adicionándose sin problemas el usuario.	• Página Inicial. • Clic en el botón Administrar sistema. • Clic en la opción Gestionar usuario. • Clic en la opción Agregar usuario. • Formulario Agregar usuario. (Ver CU. Gestionar Usuario. Sección Adicionar usuario). • Llenar los campos en el formulario. • Clic en el botón Aceptar.

	EC 1.2. Faltan datos, el sistema muestra un mensaje de error y alertas en los campos.	El sistema no puede adicionar el usuario porque falta algún dato necesario para la ejecución del método.	• Página Inicial. • Clic en el botón Administrar sistema. • Clic en la opción Gestionar usuario. • Clic en la opción Agregar usuario. • Formulario Agregar usuario. (Ver CU. Gestionar Usuario. Sección Adicionar usuario). • Dejar algún campo vacío. • Clic en el botón Aceptar. • Formulario con errores.
SC 2: Editar Usuario	EC 2 Editar usuario de forma correcta.	Esta funcionalidad consiste en modificar la información de un usuario determinado, el administrador selecciona el usuario, el sistema muestra la información, el administrador cambia los datos que desea y guarda los cambios.	• Clic en la opción Gestionar usuario. • Clic en la opción Editar usuario. • Formulario Agregar usuario. (Ver CU. Gestionar Usuario. Sección Editar usuario). • Llenar los campos en el formulario. • Clic en el botón Aceptar.

	EC 2.1 Modificar Usuarios de forma incorrecta.	El usuario borra los datos que desea modificar y no guarda ninguna información del usuario.	• Clic en la opción Gestionar usuario. • Clic en la opción Editar usuario. • Formulario Agregar usuario. (Ver CU. Gestionar Usuario. Sección Editar usuario). • Llenar los campos en el formulario. • Dejar algún campo vacío. • Clic en el botón Aceptar.
SC 2: Eliminar Usuario	EC 1.1 Eliminar Usuario satisfactoriamente.	Esta funcionalidad consiste en seleccionar el usuario que se desea eliminar.	• Página Inicial. • Clic en el botón Administrar sistema. • Clic en la opción Gestionar usuario. • Clic en la opción Eliminar usuario. • Formulario Agregar usuario. (Ver CU. Gestionar Usuario. Sección Eliminar usuario). • Seleccionar el nombre del usuario. • Clic en el botón Aceptar.

Tabla 2. Prueba de Caja Negra CU Gestionar Usuario

3.8. Conclusiones

Durante este capítulo, se describió la construcción de la solución propuesta, mostrándose la arquitectura escogida para la realización de la aplicación, desarrollándose el Modelo de análisis y diseño del sistema, el Modelo de datos, el Diagrama de despliegue y el Diagrama de componentes. Primeramente se elaboraron los diagramas de clases del análisis donde se exponen los conceptos básicos del sistema, sus partes y relaciones. Después, los diagramas de clases del diseño y los diagramas de interacción por cada uno de los casos de usos, entendiéndose a través de la interpretación de los diagramas la forma en que se desarrolla la aplicación. Posteriormente, ya dentro del Modelo de implementación, se ofrece una visión general de cómo queda finalmente implementada la aplicación mostrándose la misma en términos de componentes, su ubicación y utilización a la hora de la implementación final. También se obtuvo como resultado que todos los componentes se implementaron cumpliendo con esta etapa última de desarrollo. Además se culmina con la fase de pruebas donde se corrigen algunos errores que se detectan.

CONCLUSIONES

Con el presente trabajo se logra desarrollar con calidad un Módulo Web para el proyecto Suria Vision, cumpliendo con los objetivos trazados para la terminación del mismo. Durante su desarrollo se llegaron a las siguientes conclusiones:

- ✓ Los Sistemas de Video Vigilancia actuales están dirigidos a problemas concretos de los clientes y presentan en su funcionamiento gran dependencia del fabricante.
- ✓ El Módulo Web brindaría al proyecto una gran flexibilidad en cuanto a acceso al sistema desde fuera de la red local donde está instalado el mismo.
- ✓ Este Módulo no necesita que se configure ni se instale previamente en la estación de monitorización por lo que resulta muy cómodo para el uso de los clientes y consume pocos recursos de hardware en la estación desde donde se trabaja.
- ✓ El Módulo Web completaría el Sistema de Video Vigilancia Suria Vision, funcionando para situaciones excepcionales y manteniendo una estrecha relación con el sistema a fin de permitir una vigilancia efectiva.

Es importante destacar la capacidad del sistema de soportar equipos de diversos fabricantes, ya que es una de las limitaciones actuales en la mayoría de los sistemas en el mercado.

RECOMENDACIONES

Durante el desarrollo del presente trabajo se han podido recopilar diferentes ideas que pueden permitir, en un desarrollo futuro, una aplicación web para la video vigilancia con mayores funcionalidades. Se recomienda:

1. La implementación de un módulo de grabación accesible desde la Web.
2. La implementación de un módulo para detección de movimientos en la aplicación web.
3. La implementación de un sistema de alarmas y avisos ante sucesos sospechosos.
4. El despliegue piloto del sistema en alguna institución para hacer un seguimiento del rendimiento y enriquecer el mismo según los resultados del seguimiento y las peticiones de los clientes.

REFERENCIAS BIBLIOGRÁFICAS

1. CodeBox: Glosario. [En línea] [Citado el: 03 de Diciembre de 2010.] http://www.codebox.es/glosario.

2. *¿Qué es un framework web?* **Gutiérrez., Javier J.** 2006.

3. Enterprise Architect 7.0 - Modelado avanzado con UML 2.1. [En línea] Sparx Systems. [Citado el: 03 de Diciembre de 2010.] http://sparxsystems.com.es/products/ea_features.html.

4. **Microsoft.** Visual Studio 2010. *Visual Studio 2010.* [En línea] 2010. [Citado el: 20 de Febrero de 2011.] http://www.microsoft.com/business/smb/es-es/servidores-y-herramientas/visual-studio-pro.mspx.

5. —. Información general de .NET Framework Remoting. *Información general de .NET Framework Remoting.* [En línea] Microsoft.com, 2005. [Citado el: 20 de Febrero de 2011.] http://msdn.microsoft.com/es-es/library/kwdt6w2k(v=vs.80).aspx.

6. **IBM.** Rational Method Composer. Classic RUP for SOMA. *Rational Method Composer.* 2007.

7. Modelo de Dominio. Modelo *de Dominio.* [En línea] 17 de Diciembre de 2007. http://migueljaque.com/index.php/tecnicas/tecnicasmodnegocio..

8. **Semanat Aldana, Edmis Deivis y Verdecia Four, Leonor.** *Sistema de Video Vigilancia.* Ciudad de La Habana : s.n., 2009. TD-2136-09.

9. **Pantoja, Ernesto Bascón.** *El patrón de diseño Modelo-Vista-Controlador (MVC) y su implementación en Java Swing.* 2004.

16. **Definicion.org.** Definición de plug in. *Definición de plug in.* [En línea] [Citado el: 20 de Febrero de 2011.] http://www.definicion.org/plug-in.

BIBLIOGRAFÍA

1. CodeBox: Glosario. [En línea] [Citado el: 03 de Diciembre de 2010.] http://www.codebox.es/glosario.

2. *¿Qué es un framework web?* **Gutiérrez., Javier J.** 2006.

3. Enterprise Architect 7.0 - Modelado avanzado con UML 2.1. [En línea] Sparx Systems. [Citado el: 03 de Diciembre de 2010.] http://sparxsystems.com.es/products/ea_features.html.

4. **Microsoft.** Visual Studio 2010. *Visual Studio 2010.* [En línea] 2010. [Citado el: 20 de Febrero de 2011.] http://www.microsoft.com/business/smb/es-es/servidores-y-herramientas/visual-studio-pro.mspx.

5. —. Información general de .NET Framework Remoting. *Información general de .NET Framework Remoting.* [En línea] Microsoft.com, 2005. [Citado el: 20 de Febrero de 2011.] http://msdn.microsoft.com/es-es/library/kwdt6w2k(v=vs.80).aspx.

6. **IBM.** Rational Method Composer. Classic RUP for SOMA. *Rational Method Composer.* 2007.

7. Modelo de Dominio. Modelo *de Dominio.* [En línea] 17 de Diciembre de 2007. http://migueljaque.com/index.php/tecnicas/tecnicasmodnegocio..

8. **Semanat Aldana, Edmis Deivis y Verdecia Four, Leonor.** *Sistema de Video Vigilancia.* Ciudad de La Habana : s.n., 2009. TD-2136-09.

9. **Pantoja, Ernesto Bascón.** *El patrón de diseño Modelo-Vista-Controlador (MVC) y su implementación en Java Swing.* 2004.

10. **Dlink.** ¿Qué es la videovigilancia IP? . [En línea] [Citado el: 02 de Diciembre de 2010.] www.dlink.es/vigilanciaIP.

11. **Hernández Sampieri R., Fernández Collado C., Baptista Lucio P.** *Metodología de la Investigación Científica.* México : McGraw-Hill, 1991.

12. **Zoneminder.com.** Linux Home CCTV and Video Camera Security with Motion Detection. *Linux Home CCTV and Video Camera Security with Motion Detection.* [En línea] [Citado el: 11 de Diciembre de 2010.] http://www.zoneminder.com/.

13. **Cisco Systems.** Cisco Video Surveillance Stream Manager Software. *Cisco Video Surveillance Stream Manager Software.* [En línea] [Citado el: 11 de Diciembre de 2010.] http://www.cisco.com/en/US/products/ps6940/index.html.

14. **Vidium.** Vidium - Videovigilancia Oline. *Vidium - Videovigilancia Oline.* [En línea] [Citado el: 11 de Diciembre de 2010.] http://www.vidium.es/.

15. **Fedora.** Proyecto Fedora. *Proyecto Fedora.* [En línea] [Citado el: 11 de Diciembre de 2010.] http://fedoraproject.org/es/.

16. **Definicion.org.** Definición de plug in. *Definición de plug in.* [En línea] [Citado el: 20 de Febrero de 2011.] http://www.definicion.org/plug-in.

17. ¿Qué es JSP? *¿Qué es JSP?* [En línea] 14 de Enero de 2007. http://casidiablo.net/%c2%bfque-es-jsp/.

18. **Pérez, Javier Eguíluz.** *Introducción a JavaScript.* Madrid : s.n., 2008.

19. **DATYS.** *Xyma Save Vision.Sstema de Video Vigilancia IP. White Paper.* Ciudad de La Habana. : s.n., 2010.

20. **Axis Communications.** *Factores y técnicas a considerar para un correcto uso de las aplicaciones de vigilancia y monitorización remota basadas en IP.* 2009.

21. —. *Vigilancia IP Axis. Infinitas posibilidades de videovigilancia.* 2009.

22. **Mesbah Ahmed, Chris Garrett, Jeremy Faircloth, Chris Payne.** *ASP.NET Web Developer's Guide.* Rockland : Syngress, 2002. 1-928994-51-2.

23. **Pilgrim., Mark.** *HTML5: Up & Running.* s.l. : O'Reilly., 2010.

24. **WhatWG.** HTML Living Standard. *HTML Living Standard.* [En línea] 19 de Enero de 2011. http://www.whatwg.org/specs/web-apps/current-work/.

25. **VideoLan (VLC).** Chapter 7. The Mozilla plugin. [aut. libro] VideoLan (VLC). *VideoLAN Streaming How to.* 2009.

26. **Video LAN (VLC).** Features VLC Controls. *Features VLC Controls.* [En línea] 2010. http://www.videolan.org/vlc/features.html.

27. **Sencha.** Ext JS Cross-Browser Rich Internet Application Framework. *Ext JS Cross-Browser Rich Internet Application Framework.* [En línea] http://www.sencha.com/products/extjs/.

28. **Ferrer Rivas, Juan Carlos y Rivero González, Magdiel.** *Suria Recorder : grabador de flujos de video.* Ciudad de La Habana. : s.n., 2010. TD-03020-10..

29. **Trowbridge, David, y otros, y otros.** *Enterprise Solution Patterns Using Microsoft .NET.* 2003.

30. **Len Bass, Paul Clements, Rick Kazman.** *Software Architecture in Practice, Second Edition.* s.l. : Addison Wesley, 2003. 0-321-15495-9.

31. **Martin Fowler, David Rice, Matthew Foemmel, Edward Hieatt, Robert Mee, Randy Stafford.** *Patterns of Enterprise Application Architecture.* s.l. : Addison Wesley, 2002. 0-321-12742-0.

32. **Proenza Arias, Yuniel Eliades.** *Arquitectura de Seguridad para Aplicaciones Web Empresariales.* Ciudad de La Habana : ISPJAE, 2006. TD-0180-06.

33. **Shklar, Leon y Rosen, Richard.** *Web Applications Architecture. Principles, Protocols and Practices.* San Francisco : Jhon Wiley & Sons, Ltd, 2003. 0-471-48656-6.