UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 7

Sistema de Video Vigilancia

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

Autores: Edmis Deivis Semanat Aldana
Leonor Verdecia Four

Tutores: Ing. Heliodoro Rodríguez Milian
Ing. José Alejandro Segura Roque

Ciudad de La Habana, marzo de 2009
"Año del 50 aniversario del Triunfo de la Revolución"

DECLARACIÓN DE AUTORÍA

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

Para que así conste firmamos la presente a los 13 días del mes de marzo del año 2009.

Edmis Deivis Semanat Aldana

_____

Leonor Verdecia Four

_____

Ing. Heliodoro Rodríguez Milian

_____

Ing. José Alejandro Segura Roque

_____

DATOS DE CONTACTO

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

Profesor instructor graduado de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas (UCI). Imparte la asignatura de Práctica Profesional 2. Actualmente se desempeña como Líder del proyecto de video vigilancia.

TUTOR: Ing. José Alejandro Segura Roque (jsegura@uci.cu)

Profesor graduado de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas (UCI). Imparte la asignatura de Segundo Perfil. Actualmente se desempeña como arquitecto del proyecto de video vigilancia.

AGRADECIMIENTOS

A mis padres por su apoyo incondicional y por todo su amor.

A mi hermano por haberlo sentido a mi lado en cada momento.

A mi Familia por todo el apoyo que me han brindado a lo largo de estos años de estudio.

A mis Profesores por haberme no solo formado como profesional, sino también haber influenciado en mi formación como persona

A mis amigos.

Edmis Deivis Semanat Aldana

A mi mamá, por estar siempre a mi lado, por darme todo su amor y cariño y por haber confiado siempre en mí.

A mi padre Jorge, por darme toda la ternura y cariño de un padre y apoyarme en todo lo que he necesitado.

A Emir, por brindarme todo su cariño y su apoyo incondicionalmente.

A todos mis profesores, que de una forma u otra han ayudado a que este sueño se me hiciera realidad.

A todos mis amigos.

Leonor Verdecia Four

DEDICATORIA

A mi Mamá por su amor, su dedicación y sacrificio, por apoyarme en todo momento, por haber estado siempre a mi lado en cada problema.

A mi Papá por su constante presencia donde quiera que me llevara la vida, y por enseñarme a ser una mejor persona.

A mi Hermano, por su apoyo, por ser el faro que me guio en toda mi vida hasta aquí, y que me mostró los valores que todo hombre debe tener.

A Toda mi familia, porque todos han dejado su huella en mi, y me han ayudado cuando los necesité.

A mis amigos por su compañía y apoyo.

Edmis Deivis Semanat Aldana

A mi mamita, que es mi luz, mi guía, por la que son mis razones de seguir adelante, y por ser lo que el tesoro más preciado en mi corazón.

A mi padre Jorge, una persona muy importante en mi vida, al cual quiero y aprecio mucho.

A mis hermanos, que siempre me han protegido y apoyado.

A mi querido Emir, que ocupa también un espacio especial en mi corazón.

A mi novio Yoani, por todo el amor que me ha dado y por estar en todos estos momentos a mi lado.

A mi familia, que de una forma u otra me han dado la mano-

Leonor Verdecia Four

RESUMEN

Los sistemas de video vigilancia tienen cada vez mayor demanda en la actualidad. Especialmente para garantizar la seguridad de interiores y alrededores de las empresas e instituciones. La presencia de numerosas cámaras de seguridad en cualquier entorno urbano es un hecho. La evolución tecnológica posibilita que la instalación de cámaras de captura de video no precise altas inversiones económicas. Por tanto, la mayoría de las organizaciones como bancos, estaciones, aeropuertos, tiendas, parkings, entre otras; incorporan en sus instalaciones algún sistema, más o menos complejo, de seguridad basado en video vigilancia.

El presente trabajo tiene como objetivo implementar un sistema de video vigilancia, desarrollado para el uso de Cámaras IP[1]. Para su desarrollo se propone la utilización de librerías gratuitas y de código abierto compatible con los estándares elegidos, y el marco de trabajo (Framework) .Net de Microsoft.

Como resultado se pretende crear una herramienta 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.

Palabras claves: sistemas, video vigilancia, seguridad.

[1] IP: Internet Protocol (Protocolo de internet).

ÍNDICE

INTRODUCCIÓN

La necesidad de proteger los medios y recursos ha existido en todas las épocas. Históricamente se han destinado muchos esfuerzos y recursos a las actividades relacionadas con la vigilancia. Durante mucho tiempo, esta actividad fue llevada a cabo casi exclusivamente por personas. Aunque ello aumenta el riesgo de errores: es posible que un custodio pueda quedarse dormido, ausentarse de su puesto de trabajo entre otras posibles fallas humanas. Además, que si es limitado el espacio a cubrir porque mientras más grande es el área a vigilar, más personal se requiere, más difícil controlar su trabajo y pueden ocurrir más errores como los planteados.

La problemática anterior y el desarrollo de las tecnologías que lo sustentan propiciaron el surgimiento de la video vigilancia, como un paso más en la evolución en los métodos de seguridad de las empresas e instituciones. Esta nueva herramienta reduce la necesidad de personal de seguridad y mantiene el control sobre todas las áreas requeridas de manera centralizada. Además, permite grabar en diferentes medios de almacenamientos los videos obtenidos para así tener referencia de todo lo que ha pasado en la institución. Esta información sirve incluso para peritajes posteriores a un determinado evento, ya sea por la policía o por la misma dirección de la empresa o institución.

Desde sus inicios, la video vigilancia no ha tenido como meta la sustitución de las personas; sino la de asistirlas, y en combinación, mejorar la efectividad de los métodos aplicados. Es común en un entorno donde se apliquen estos sistemas que solo existan algunos guardias de seguridad. Estos son ubicados en lugares claves, donde es necesaria la respuesta humana a determinadas situaciones (puntos de control, recepciones etc.). Lo que se complementa con un grupo que debe estar rotando por otras áreas o esperando una alerta para dirigirse a cualquier zona necesaria ante un problema determinado. El cual puede ser localizado desde la estación de visualización del sistema, y desde ahí, asistir al personal de seguridad con información actualizada de la posición de los perpetradores del hecho.

Con el crecimiento y avance del desarrollo de las tecnologías en la actualidad, cada vez es más frecuente que las grandes empresas e instituciones cuenten con una elevada infraestructura tecnológica de alta escala. Se han ido informatizando muchas labores que antes se hacían de forma manual, posibilitando cambios en todos los ámbitos de esta actividad.

En Cuba se han realizado grandes inversiones en nuevas tecnologías para instituciones como hospitales, bancos, escuelas, empresas, entre otras. Estas cuentan con equipos tecnológicos de altos costos: los centros de salud han sido dotados con equipamientos que mejoran calidad en los exámenes médicos o

en dar sus resultados. Además centros como la Universidad de las Ciencias Informáticas tiene gran cantidad de equipos como computadoras, refrigeradores, televisores, aires acondicionados y ventiladores por cada apartamento. Además, tiene almacenes de equipos y piezas de reparación para estos medios, todos potenciales motivadores de actos delictivos.

Esta situación convierte a estas entidades en un entorno muy complejo para ser asegurado solamente con custodios las 24 horas. Actualmente el cuidado de estas zonas la realiza el personal de seguridad, que se encuentra distribuido por las principales áreas según el lugar lo requiera. Su trabajo es proteger los recursos de la institución donde se encuentre e impedir que se cometan actos delictivos. Existen zonas en las que no es suficiente con tener al personal de seguridad frente a toda el área. Son áreas que tienen gran movimiento de personas y el guardia no tendría el control total de lo que ocurre.

En general, la seguridad es un tema de vital importancia para el país y las instituciones tienen que preservar los recursos con que cuentan por su importancia, alto costo y valor social de los mismos.

La situación planteada anteriormente lleva a formular el siguiente problema científico ¿Cómo mejorar la seguridad utilizando la tecnología IP en instituciones cubanas? Para darle solución a este problema el objeto de estudio se enmarca en la seguridad de las instituciones del país. Dicho estudio, delimita como campo de acción la informatización de la video vigilancia de las instituciones del país.

Para darle solución al problema planteado anteriormente, se tiene como objetivo general: Implementar una aplicación de escritorio para mejorar la video vigilancia en instituciones cubanas utilizando la tecnología IP. Como resultado se obtiene un sistema que debe aumentar la vigilancia mediante la instalación de centros de control, que pueden estar atendiendo de manera simultánea los puntos de importancia clave en el lugar donde se utilice.

Para el cumplimiento del objetivo planteado se proponen las siguientes tareas de investigación:

- Realizar un análisis valorativo de los sistemas de video vigilancia existentes en la actualidad.
- Valorar las herramientas y tecnologías existentes para aplicaciones de escritorio y seleccionar las que se utilizarán.
- Realizar un análisis de las tecnologías más novedosas que se usarán para la implementación del sistema.
- Obtener los modelos de negocio, análisis y diseño del sistema.
- Realizar el levantamiento de requisitos del sistema representados en lenguaje natural.

- Diseñar una aplicación que tenga una interfaz amigable e intuitiva para el usuario.
- Diseñar la base de datos a utilizar en función de las necesidades de la solución y los cambios tecnológicos, cambios conceptuales o del software que puedan ocurrir.
- Implementar el Software con todas las funcionalidades requeridas.

Los métodos utilizados en la investigación se explican a continuación:

Métodos Teóricos.

- Analítico-sintético: Es usado para estudiar y analizar documentos y bibliografías de diferentes autores para poder realizar una amplia investigación sobre los elementos que se relacionan con el objeto de estudio.
- Análisis histórico lógico: Se utiliza para hacer un estudio sobre la evolución y desarrollo que han tenido los sistemas de video vigilancia, lo que permite ver en que etapa se encuentran dichos sistemas y cual es la tecnología factible a los mismos.

Métodos Empíricos.

- Observación: Permite realizar valoraciones y obtener informaciones a partir de lo observado. Esto se manifiesta principalmente cuando se realizan observaciones sobre el funcionamiento de sistemas similares al desarrollado, lo que da una visión de cómo tiene que ser el sistema a realizar en su forma externa.
- Entrevista: Permite tener una comunicación con otras personas que pueden servir para apoyar y dar ideas en cuanto al sistema que se quiere construir. En este caso, es de vital importancia ya que se necesita de mucha comunicación con los posibles clientes para que el producto final cumpla con todo lo requerido.

El presente documento consta de cuatro capítulos:

El **Capítulo 1** describe algunas de las características de los Sistemas de Video Vigilancia y hace referencia al estado del arte de los proyectos existentes en el mundo semejantes al que se va a

desarrollar. Además, se realiza un análisis de las tecnologías, metodologías y herramientas actuales, quedando seleccionadas las que se va a utilizar para el desarrollo del sistema.

El **Capítulo 2** da a conocer las principales características del sistema, haciendo comparaciones con otro software de este tipo existentes en el mundo. Además, se ofrece el modelo de dominio de la aplicación conjuntamente con la especificación de los requerimientos funcionales y no funcionales, y el modelo de casos de usos del sistema.

El **Capítulo 3** muestra los resultados obtenidos en el desarrollo de los procesos de análisis y diseño del sistema, así como los diagramas que fueron necesarios para obtener una mayor claridad a la hora de elaborar la solución que se propone.

Finalmente, en el **Capítulo 4** se muestra el modelo de implementación como resultado del análisis y diseño estando compuesto por su respectivo diagrama de despliegue, y por su diagrama de componentes.

.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En este capítulo se abordan aspectos relacionados con la evolución de los sistemas de video vigilancia. También, se da una visión del estado en que se encuentran en el mundo los sistemas similares al desarrollado en el proyecto. Además, se hace un análisis sobre las principales metodologías y herramientas utilizadas en la aplicación.

1.1 Evolución de los sistemas de video vigilancia.

Existen soluciones en el mundo que le han dado respuesta a los problemas de seguridad mediante sistemas de vigilancia. Así pues, cualquier empresa que quiera proteger a sus clientes, empleados o bienes frente a posibles amenazas a la seguridad puede apostar por este tipo de soluciones, pero teniendo en cuenta que, **en función del tipo de empresa que sea, las amenazas a las que tendrá que hacer frente serán diferentes. Y es que suele ser raro que una empresa o negocio no esté expuesta a amenazas a la seguridad**, aunque éstas sean de diferente naturaleza, **por lo que prácticamente cualquiera puede beneficiarse de la implantación y uso de un sistema de video vigilancia**.

La seguridad es un factor de importancia en todas las instituciones del mundo y "en función del tipo de empresa que sea, las amenazas a las que tendrá que hacer frente serán diferentes. Y es que suele ser raro que una empresa o negocio no esté expuesta a amenazas a la seguridad", **por muy pequeña que sea tiene que asegurarse,** por lo que prácticamente cualquiera puede beneficiarse de la implantación y uso de un sistema de video vigilancia.

Desde el punto de vista tecnológico, los sistemas de vigilancia han evolucionado atravesando diferentes etapas en las últimas décadas. La primera generación de sistemas de vigilancia basada en video emplea señales y transmisión analógicas. En la toma de decisiones, siempre es el operador humano el encargado de realizar todas las tareas de análisis de secuencias de video presentadas en varios monitores situados en una sala de control remota, donde las escenas monitorizadas por las distintas cámaras se multiplexan y presentan en un orden periódico y predefinido. Adicionalmente, la vigilancia tradicional precisa gran cantidad de espacio de almacenamiento. Todo lo que captura una cámara de seguridad se graba en cintas que, o bien se sobrescriben periódicamente, o bien se guardan en un archivo de video.

Este procedimiento limita la duración de video que puede guardarse y hace que el tiempo necesario para su revisión sea elevado, después de haberse producido un delito –asalto en una tienda o robo de un auto– los investigadores pueden buscar el suceso en el video y ver qué ocurrió, pero ya es tarde para evitarlo. Sería necesario un seguimiento continuo, durante las 24 horas del día, del video generado para alertar al personal de seguridad mientras el delito está en progreso, cuando aún existen opciones de evitarlo.

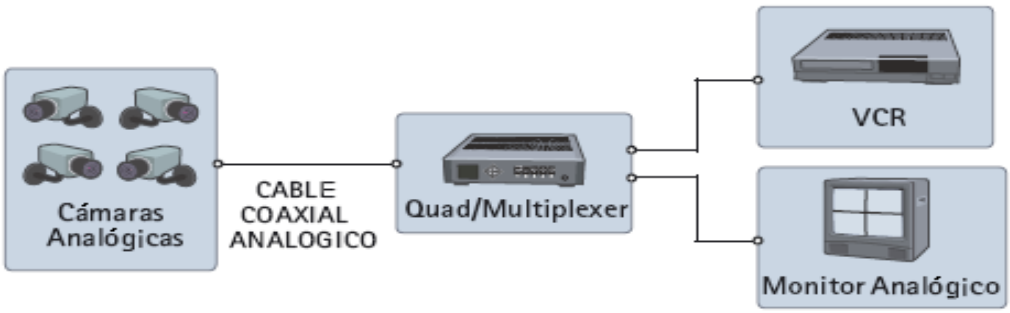

Figura 1 Sistema de Video Vigilancia Analógico.

Los sistemas analógicos emplean amplificadores de baja potencia y cables coaxiales para ver de forma remota las imágenes de las cámaras de seguridad. Las cámaras y los monitores están interconectados mediante una red de conmutación y distribución de video complejo y caro, que direcciona las imágenes de cada cámara hacia un monitor. **Esta tecnología tiene demasiadas limitaciones y defectos**. **El video analógico solo se puede distribuir en una red local de cables coaxiales**.

La segunda generación de sistemas de vigilancia se basa, principalmente, en métodos de procesamiento y comunicación híbridos analógico-digitales. Aprovechan la flexibilidad ofrecida por los primeros algoritmos de procesado de video que permiten centrar la atención del operador humano en un grupo de situaciones de interés y, además, las facilidades proporcionadas por los primeros métodos de compresión digital para aprovechar el ancho de banda de transmisión.

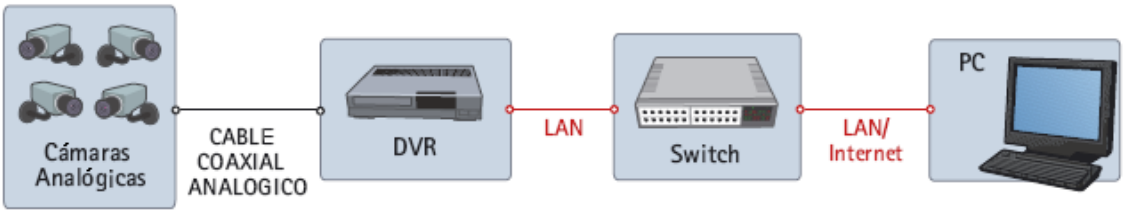

Figura 2 Sistema de Video Vigilancia Mixto Analógico-Digital.

Actualmente, se está produciendo una migración de los sistemas de video clásicos a los sistemas de tercera generación. Los sistemas de tercera generación **aprovechan el progreso de las redes de ordenadores de bajo costo y alto rendimiento, y las comunicaciones multimedia fijas y móviles**. La investigación en este campo trabaja en las siguientes líneas:

- Técnicas distribuidas de procesamiento de video.
- Procesado de secuencias de video en tiempo real.
- Conseguir sistemas robustos de transmisión de imagen.
- Procesamiento de imagen en color.
- Generación de alarmas basada en eventos.
- Reconstrucción de secuencias a partir de modelos.
- Segmentación y análisis en tiempo real de secuencias de imágenes 2D.
- Identificación y seguimiento de múltiples objetos en escenas complejas.
- Reconocimiento de comportamientos humanos.

Se prevé que estas proporcionen a las aplicaciones de vigilancia mejores resultados, gracias a la disponibilidad de una potencia de cálculo muy elevada, a unos precios aceptables. En la actualidad aunque hay muchas empresas que se dedican a la fabricación de software de video vigilancia, estos tienen un alto costo en el mercado internacional. Otro inconveniente de estos sistemas, radica en que la mayoría no son compatibles con cualquier tipo de fabricante; cada uno está hecho para cámaras específicas. Eso trae consigo que haya que comprar cámaras de un mismo fabricante, lo que es ineficiente al perderse la posibilidad de escoger los modelos que sean adecuados a las necesidades con

respecto a precio y funcionalidades del hardware, además dificulta la futura extensión del sistema ya instalado con la adquisición de hardware más moderno. (1)

Figura 3 Sistemas de Video Vigilancia Modernos.

Por la experiencia en este tema, la Axis Comunnications es la compañía líder en el mercado de los sistemas de video vigilancia. Según el estudio realizado sobre la evolución de los sistemas de video vigilancia, se ha podido apreciar que el avance de los mismos es notable. Comenzando por los videos de vigilancia analógicos la empresa Axis Communications plantea que esta tecnología tiene demasiadas limitaciones y defectos. El video analógico solo se puede distribuir en una red local de cables coaxiales. **De esta forma han ido evolucionando estos sistemas en vista de mejoras. Actualmente se encuentran en la tercera generación, los que según Axis Comunnications** aprovecharán el progreso de las redes de ordenadores de bajo coste y alto rendimiento, y las comunicaciones multimedia fijas y móviles. **La elección de los sistemas de video vigilancia de tercera generación sería la mejor solución al problema que se debe resolver, los mismos permiten un mejor aprovechamiento de la infraestructura de la red existente, una mayor calidad en los videos obtenidos, procesamiento inteligente de los flujos de video entre otras características.**

Una de las empresas que ha dado respuesta a los problemas de seguridad mediante sistemas de vigilancia es **Telenorma**. Esta compañía que ofrece soluciones en comunicación IP para voz,

videoconferencia, video vigilancia y redes inalámbricas plantea que “Sin importar que se trate de una corporación, una institución gubernamental, una fábrica o cualquier otro foco de negocios, **la seguridad propia de la infraestructura y del personal son parte de la responsabilidad y preocupación diaria de sus directivas**. La centralización de la administración del video de las instalaciones o de sitios remotos utilizando la infraestructura existente de la red de datos y las cámaras de CCTV, **permite a los responsables de la seguridad, recolectar información en tiempo real reduciendo los costos de operación y respondiendo rápidamente a potenciales amenazas**”. (2)

Los autores consideran que la seguridad propia de la infraestructura y del personal son parte de la responsabilidad y preocupación diaria de sus directivas **y de todos los implicados que de una forma u otra pueden contribuir a elevar su eficiencia ya que permite a los responsables de la seguridad contar con mayor cantidad de soluciones y aportes logrando que sea más ágil** recolectar información en tiempo real reduciendo los costos de operación y respondiendo rápidamente a potenciales amenazas.

1.2 Estado del Arte de los sistemas de video vigilancia.

Las soluciones de video vigilancia, tienen lógicamente como el primer objetivo en el mercado a las grandes empresas, aunque no son las únicas que pueden sacar provecho de sus posibilidades. La experiencia de los fabricantes de este sector posibilita que otras actividades como: gestionar recursos, realizar inventarios, analizar conductas de compradores, controlar procesos industriales. Además de realizar prácticas de tiro a distancia e incluso para controlar y fotografiar las migraciones de la fauna de un parque natural sin tener que molestarla, son ejemplos reales de tareas que pueden realizarse con un uso intensivo de estas herramientas. Las posibilidades que ofrecen los productos de video IP son múltiples, de manera que quizás solo la imaginación sea una limitación a su aplicación a diversos campos.

Eso sí, como en toda tecnología, hay sectores que parecen más interesados y son más proclives a la adopción e incorporación de dichas herramientas. Por citar solo algunos: en la educación (para la detección y análisis de conductas delictivas o control de asistencias), el comercio minorista (conteo de

personas, control de materiales, análisis de conductas de compradores, control del manejo de las cajas), industria (control de procesos) y transporte (acciones delictivas, prevención de actos terroristas); sin pasar por alto las Administraciones Públicas (protección de edificios públicos y del Patrimonio y por el control de tráfico).

Por supuesto, aunque algunos usuarios particulares también apuestan por este tipo de soluciones, y pese a que el objetivo último puede ser similar, los sistemas para usuario final son mucho más sencillos que para aquellos destinados a un uso más profesional, ya que se necesita mejor calidad de imagen, grabación de las imágenes durante un período determinado, infrarrojos, altavoz... (3)

Figura 4 Visualización de varias cámaras

Principales empresas productoras de sistemas de video vigilancia en el ámbito nacional:

Según las investigaciones realizadas no existe un producto de factura nacional que proporcione servicios de vigilancia a instituciones. Actualmente en la UCI[2] el Grupo de Procesamiento de Imágenes (GPI) está

[2] UCI: Universidad de las Ciencias Informáticas.

desarrollando un sistema de video vigilancia en vista de mejorar el proceso de la gestión de la seguridad en las empresas e instituciones del país.

Principales empresas productoras de sistemas de video vigilancia en el ámbito internacional:

Axis: es una compañía de TI (Tecnologías de la Información) que ofrece soluciones de video IP dirigidas al mercado profesional. La compañía es líder del mercado del video IP, conduciendo el cambio del video vigilancia analógica hacia las soluciones digitales. **Los productos y soluciones de Axis están diseñados para los sectores de la vigilancia, la seguridad y la monitorización remota, y están basados en la innovación y en lo estándares abiertos.** Axis es una compañía sueca que tiene oficinas en 18 países, y que coopera con socios comerciales en más de 70 países de todo el mundo. El nombre de su principal software de video vigilancia es Axis Camera Station. (4) **Este software tiene una alta calidad**, y es compatible con la mayoría de los productos de axis(cámaras y grabadores), demanda hardware de calidad, **y requiere para lograr los rendimientos máximos del uso de mpeg4 para la transmisión y almacenamiento de los flujos, lo que es un inconveniente ya que es un formato propietario con licencias muy costosas**. (5)

Figura 5 Axis Camera Station.

Panasonic: también ha sido pionero al lanzar la primera cámara de video vigilancia IP, con una tecnología actualmente conocida como i-Pro, pensada para la monitorización de imágenes on-line y la grabación en alta resolución. Además, su practicidad es máxima ya que su capacidad de comprensión de imágenes es dual, JPEG y MPEG4. Panasonic **ha hecho desde entonces importantes innovaciones en el mundo del hardware de video vigilancia**, este es el caso de la primera cámara con procesador de señal digital y la primera cámara domo, con un sistema novedoso resistente al agua y preparado para soportar temperaturas extremas. Este tipo de cámara profesional tiene capacidad de movimiento de 360° y control durante las 24 horas.

Figura 6 Panasonic despunta con su nueva cámara domo en color WV-NF2

Panasonic se caracteriza por vender soluciones de seguridad completa más que hardware independiente (Axis lidera este mercado). Cada kit de solución es muy dependiente a un hardware específico, **cada serie de cámaras viene con un software específico para ella, con poca o ninguna compatibilidad con otras cámaras, lo que dificulta mucho el trabajo para desarrolladores externos, porque tienen que soportar muchas APIs[3] distintas.** (6)

[3] Una API (del inglés Application Programming Interface - Interfaz de Programación de Aplicaciones) es el conjunto de funciones y procedimientos (o métodos si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Figura 7 Panasonic XPBP04 para series BB_x WV-NS32x.

SCATI LABS, S.A: es una compañía tecnológica radicada en Zaragoza que se dedica a la fabricación de sistemas de video vigilancia, que **permiten la gestión de múltiples cámaras de seguridad para el control y supervisión de instalaciones locales y remotas**. Tiene en el mercado la Suite de Gestión Remota VisionSurfer, que se enmarca en la explotación y mantenimiento eficaz de grandes parques de videograbación digital. El módulo que se encarga de la video vigilancia es el SurferWatcher, pero ningún módulo se comercializa de manera independiente. Esta suite, permite elaborar soluciones específicamente diseñadas para el entorno bancario, es muy costosa, pero tiene probada eficacia en su campo. (7)

Figura 8 SCATI LABS S.A SurferWatcher

Sony: es otra de las grandes empresas que se encuentra emergida en este campo de la seguridad. Las cámaras de video vigilancia de Sony **utilizan la tecnología Intelligent Video Analysis para detectar incidentes y alertar al personal de seguridad**. Por ejemplo, con Intelligent Object Detection puede vigilar zonas determinadas dentro de su tienda y emitir una alerta en caso de que retiren un objeto. Intelligent Motion Detection le permite vigilar las instalaciones, emitiendo una alerta en caso de que entren intrusos por la noche, pero ignorando el ruido ambiental (como el ruido de los árboles con el viento), que puede originar falsas alarmas. Los productos de la Sony también **adolecen de la baja compatibilidad con otros fabricantes y del uso extensivo de mpeg4**. (8)

Figura 9 Cámara SNC-RX550P de la Sony

Figura 10 Sony IMZRS332C

Una vez analizadas las principales compañías de fabricación de sistemas de video vigilancia se aprecia que los sistemas actuales tienen características comunes como son: la capacidad de gestionar las cámaras presentes en el sistema(adicionar, eliminar y configurar las cámaras), visualizar flujos de video de varias cámaras de manera individual o conjunta y exportar secuencias de video. Todo esto sobre una interfaz de usuario amigable e intuitiva.

Se considera que las principales características de sus productos son: Los de la Axis "están diseñados para los sectores de la vigilancia, la seguridad y la monitorización remota, y están basados en la innovación y en lo estándares abiertos", **pero tienen un inconveniente en cuanto a que** "requiere para lograr los rendimientos máximos del uso de mpeg4 para la transmisión y almacenamiento de los flujos, el que es un formato propietario con licencias muy costosas". Por su parte, **Panasonic que es otra de las grandes compañías** "ha hecho desde entonces importantes innovaciones en el mundo del hardware de video vigilancia" **pero su deficiencia viene en que** "cada serie de cámaras viene con un

software específico para ella, con poca o ninguna compatibilidad con otras cámaras, lo que dificulta mucho el trabajo para desarrolladores externos, porque tienen que soportar muchas APIs[4] distintas".

A su vez **SCATI LABs es una empresa que se dedica a la fabricación de sistemas de video vigilancia que** "permiten la gestión de múltiples cámaras de seguridad para el control y supervisión de instalaciones locales y remotas".

Otra de las destacadas empresas en el mercado de la video vigilancia es la Sony, la cual fabrica productos que "utilizan la tecnología Intelligent Video Analysis para detectar incidentes y alertar al personal de seguridad" **aunque** "adolecen de la baja compatibilidad con otros fabricantes y del uso extensivo de mpeg4".

Al valorar críticamente los productos de estas compañías, los autores concluyen que la mayoría son sistemas provistos por fabricantes de hardware, y tienen gran dependencia del hardware del fabricante en cuestión. La mayoría de las funcionalidades del sistema vienen implementadas en el hardware y el software solo se encarga de gestionarlas. Lo que encarece el costo del sistema y dificulta su extensión con cámaras u otro equipo que no sea del fabricante que lo proveyó, ya que existe una muy baja compatibilidad.

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

1.3.1 Tecnologías a usar para el desarrollo del sistema.

[4] Una API (del inglés Application Programming Interface - Interfaz de Programación de Aplicaciones) es el conjunto de funciones y procedimientos (o métodos si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Las tecnologías a usar a la hora de desarrollar cualquier aplicación están en estrecha dependencia con respecto a los lenguajes de programación a utilizar. Para el desarrollo de la aplicación se tienen los siguientes lenguajes candidatos:

- **Java:** Potente lenguaje orientado a objetos y multiplataforma. Consta con una librería de clases base para el desarrollo, pero para aplicaciones de envergadura se requiere de Frameworks [5]externos, los cuales son en su mayoría propietarios o libres que están en desarrollo por una comunidad de programadores y cuentan con escasa documentación. Las aplicaciones resultantes requieren de gran cantidad de memoria en las computadoras clientes.
- **Visual C++ .NET:** Reconocido lenguaje, no totalmente orientado a objetos, pero muy eficaz en cuanto a rapidez y uso de memoria en las aplicaciones que se obtienen. Carece de una librería de clases que realmente acelere el desarrollo de aplicaciones empresariales, y tiene muchas dificultades en cuanto a la interfaz de usuario.
- **C# 2.0:** Lenguaje estrella de la plataforma de desarrollo .NET. Cuenta con una extensa librería base para el desarrollo de todo tipo de aplicaciones. C# **es un lenguaje de programación simple pero eficaz**, este **toma las mejores características de lenguajes preexistentes como Visual Basic, Java o C++ y las combina en un solo lenguaje.**

  C# presenta entre otras características:
  - Sencillez.
  - Modernidad.
  - Orientación a Objetos.
  - Orientación a componentes.
  - Gestión automática de memoria.
  - Seguridad de tipos.
  - Instrucciones seguras.
  - Sistema de tipos unificado.

---

[5] Framework: Librería de código (posiblemente ya compilado) que sirve como base para el desarrollo de un tipo determinado de aplicaciones.

- Extensibilidad de tipos básicos.
- Extensibilidad de operadores.
- Extensibilidad de modificadores.
- Posibilidad de crear versiones.
- Eficiencia
- Compatibilidad.

Por las características que tiene el sistema los autores consideraron que el desarrollo de la aplicación se realice con C# 2.0 como lenguaje de desarrollo ya que el mismo "es simple pero eficaz"**. Además porque este** "combina las mejores características de lenguajes preexistentes como Visual Basic, Java o C++"**.**

1.3.2 Herramientas.

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

- **Ingeniería y documentación:** Más conocidas por herramientas CASE[6].Las herramientas más reconocidas en este campo son Rational Suite, Enterprise Architect y Visual Paradigm.
  - **Rational Suite:** Es reconocida como la líder 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[7]. Es bastante cara y poco intuitiva de trabajar.
  - **Enterprise Architect 6.5:** Es una herramienta progresiva que cubre todos los aspectos del ciclo de desarrollo, proporcionando una trazabilidad completa desde la fase inicial del diseño a través del despliegue y mantenimiento. También provee soporte para pruebas, mantenimiento y control de cambio. Entre las principales características que este brinda se pueden mencionar:

[6] CASE: Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador.

[7] UML: Unified Modeling Language: Lenguaje Unificado de Modelado. Lenguaje utilizado para modelar software.

- Crea elementos del modelo UML para un amplio alcance de objetivos.
- Ubica esos elementos en diagramas y paquetes.
- Documenta los elementos que ha creado.
- Genera código para el software que está construyendo.
- Realiza ingeniería directa e inversa de código en lenguajes como ActionScript, C++, C#, Java, PHP, entre otros.

Soporta todos los diagramas y modelos del UML. Puede modelar procesos de negocio, sitios web, interfaces de usuario, redes, configuraciones de hardware, mensajes y más. Estima el tamaño del proyecto en esfuerzo de trabajo en horas. Captura y traza requisitos, recursos, planes de prueba, solicitudes de cambio y defectos.

- **Visual Paradigm:** Es una Suite totalmente multiplataforma con varios módulos que se complementan entre sí. Requiere bastante RAM[8] debido a estar desarrollado en JAVA[9]. Tiene problemas de integración con otras herramientas de desarrollo.
- **Microsoft Office Visio 2007:** Este software de creación de dibujos y diagramas facilita a los profesionales empresariales y de TI la visualización, el análisis y la comunicación de información compleja. Los textos y tablas complicadas y difíciles de comprender en diagramas de Visio pueden comunicar información de un vistazo. En lugar de imágenes estáticas, se pueden crear diagramas de Visio conectados a datos que muestran información, son fáciles de actualizar y pueden hacer la diferencia en cuanto a productividad del equipo de desarrollo. (9)

- **Sistemas de Gestión de Base de Datos (SGBD)**: Algunas de las herramientas más reconocidas son Microsoft SQL Server, Oracle Database, MySQL, PostgreSQL.
  - **Microsoft SQL Server:** Reconocido gestor de base de datos de Microsoft, tiene varias versiones entre las que se puede elegir de acuerdo a la envergadura de las bases de datos a manejar. Se integra a la perfección con todos los demás productos de Microsoft. Las licencias cambian en

[8] RAM: Random Access Memory. Memoria de Acceso Aleatorio. Tipo de memoria utilizada por las computadoras.

[9] JAVA: Lenguaje de desarrollo multiplataforma.

dependencia de la versión del producto, pero se mantienen en la media alta entre los demás gestores.

- **Oracle Database:** Para muchos el mejor gestor de BD, y ciertamente el más caro. Muy utilizado en grandes sistemas. Cuenta con un gran número de herramientas que cubren todas las necesidades del cliente, y es muy fiable.
- **MySQL:** Gestor muy difundido debido a su alta compatibilidad con PHP y a ser hasta mediados del 2008 libre y Open Source[10]. Es incierto su futuro debido a haber sido comprado por Sun Microsystems.
- **PostgreSQL 8.3:** Es el primer SGBD de código abierto en implementar Recorrido Sincronizado, que reduce el uso de E/S en aplicaciones de minería de datos. El grupo de Windows ha implementado un sistema de compilación con Visual C++, lo cual mejora la estabilidad y rendimiento en Windows, así como la accesibilidad para otros contribuyentes Windows. Nuevas opciones de registro (logging) han sido agregadas y el sobrecosto del recolector de estadísticas ha sido disminuido para hacer más fácil el monitoreo de sus servidores. Incluye una cantidad récord de características nuevas y mejoradas, que van a aumentar los beneficios en cuanto al diseño de aplicaciones, administración de la base de datos y usuarios. Otra de las características están en la mayor consistencia en el rendimiento de versiones anteriores, asegurando que cada usuario pueda obtener el mismo alto nivel de rendimiento demostrado en pruebas recientes, para todas las transacciones, tanto en horas pico como fuera de ellas. (10)

- **Control de versiones:** Los principales exponentes son Microsoft Visual Source Safe y Tortoise SubVersion.
  - **Microsoft Visual Source Safe 2005:** Visual SourceSafe 2005 es una aplicación fácil de utilizar destinada a desarrolladores que desean conseguir una manera fácil de administrar cambios en el código fuente. A medida que el equipo se amplía y necesita cambios, se pueden migrar sin problemas los proyectos de Visual SourceSafe 2005 a Team Foundation Server, el componente

[10] Open Source: Código abierto, indica que el código fuente de una aplicación es de libre difusión.

central del nuevo Visual Studio Team System, que combina la seguridad de SQL Server 2005, Active Directory e IIS para formar una plataforma de colaboración completa.

- Visual SourceSafe 2005 y Team Foundation Server proporcionan una solución de administración para el ciclo de vida de la aplicación para el amplio abanico de clientes de Microsoft. (11)

  Las características nuevas incluyen:

  - Compatibilidad con Unicode y XML.
  - Acceso remoto a través de HTTP.
  - Compatibilidad con versiones anteriores.
  - Capacidad aumentada a 4 GB.
  - Ruta de migración a Team Foundation Server.
- **Tortoise SubVersion:** Herramienta totalmente libre y Open Source. Está respaldada por toda una comunidad de desarrollo especializada en el tema. Tiene un funcionamiento muy independiente de los IDEs[11] de desarrollo, y muchos problemas de integración con estos.

- **IDE de desarrollo**: Los IDEs se deben seleccionar en dependencia de los lenguajes en que se piensa desarrollar, puesto que no todos los IDEs soportan estos lenguajes candidatos.

  Como se ha seleccionado el C# 2.0 se tienen como candidatos los siguientes IDEs:

  - **Microsoft Visual Studio 2008**: Microsoft Visual Studio 2008 es el nuevo IDE que Microsoft ha desarrollado enfocado a las nuevas necesidades que involucra un nuevo mundo dentro del Desarrollo de Software y que viene con muchas mejoras y funcionalidades. Ha sido desarrollado especificamente para la plataforma .NET y en especial para el lenguaje C#. Es el IDE lider a nivel mundial en cuanto a funcionalidades y rendimiento. (12)
  - **SharpDevelop**: IDE de libre distribución que soporta completamente el lenguaje C# 2.0, pero que carece de integración con muchas tecnologías asociadas a la plataforma .NET. No está concebido para aplicaciones de gran envergadura.

[11] IDE: Integrated Develop Environment, Entorno Integrado de Desarrollo, software para desarrollar aplicaciones en distintos lenguajes.

Se seleccionó como IDE de desarrollo el Microsoft Visual Studio 2008 por sus funcionalidades y por el soporte del lenguaje C# 2.0 como herramienta. Para la realización de la Ingeniería y Documentación, el Enterprise Architect 6.5 por su integración con el Visual Studio 2008 y compatibilidad con el C# 2.0. Además también se utilizó el Microsoft Office Visio 2007 para realizar diagramas y esquemas ligeros. También se usó para el control de versiones, el Microsoft Visual Source Safe 2005 debido a la facilidad con que se integra con las demás herramientas de Microsoft previamente seleccionadas. Para la gestión de datos, el PostgreSQL 8.3 ya que las necesidades de la aplicación no justifican un sistema mayor y por demás propietario. Este cumple con todas las funcionalidades requeridas por el sistema.

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

Conjuntamente con los lenguajes de desarrollo se utilizan tecnologías que apoyadas en las herramientas de desarrollo permiten de manera integral desarrollar el sistema. Para el sistema se necesitan fundamentalmente tecnología de comunicaciones y de interfaz de usuario.

- **Comunicación:** Tecnologías que permiten la transmisión de datos sobre la red, están condicionadas por el lenguaje de desarrollo a utilizar.
  - **Microsoft .NET Remoting 2.0:** NET Remoting es una interfaz de programación de aplicaciones (API, *application programming interface*) para la comunicación. Permite crear fácilmente aplicaciones ampliamente distribuidas, tanto si los componentes de las aplicaciones están todos en un equipo como si están repartidos por el mundo. Se pueden crear aplicaciones de cliente que utilicen objetos en otros procesos del mismo equipo o en cualquier otro equipo disponible en la red, .NET Remoting permite a las aplicaciones cliente utilizar objetos en otros procesos del mismo equipo o en cualquier otro equipo disponible en la red.

    También puede utilizar .NET Remoting para comunicarse con otros dominios de aplicación en el mismo proceso. .NET Remoting proporciona un enfoque abstracto a la comunicación entre procesos que separa el objeto remoto de un servidor concreto y el proceso de cliente y desde un mecanismo concreto de comunicación. Como resultado, es flexible y se puede personalizar con facilidad. Puede reemplazar un protocolo de comunicaciones con otro protocolo de

comunicaciones o un formato de seríalización con otro sin volver a compilar el cliente o el servidor. Además, el sistema remoto no supone ningún modelo de aplicación determinado. Puede comunicarse desde una aplicación Web, una aplicación de consola, un servicio de Windows hacia cualquier parte. Los servidores remotos también pueden ser cualquier tipo de aplicación ejecutable. Cualquier aplicación puede hospedar objetos remotos, y así proporcionar sus servicios a cualquier cliente en su equipo o red. (13)

- **Sockets:** Tecnología nativa de Windows para la transmisión de todo tipo de datos sobre la red. Utiliza todos los protocolos de comunicación de datos, protocolos como TCP/IP, UDP etc. Es muy eficiente en cuanto a la rapidez con que se logra la transmisión, pero muy complicada de utilizar y mantener en una aplicación empresarial.

- **Interfaz de Usuario:** Los sistemas de video vigilancia necesitan tener interfaces atractivas y una experiencia de usuario enriquecida, para lograrlo se utilizan librerías que enriquezcan visualmente la aplicación. Entre las librerías mas populares estan:
  - **Krypton Toolkit 2.6:** una librería de controles gráficos que permiten crear interfaz de usuarios profesionales al estilo de MS Office, la cual además puede ser personalizada en su totalidad. Parte de la librería es gratis.

    Krypton Toolkit incluye:
    - Código fuente completo en C#.
    - Ayuda integrada con Visual Studio.
    - Ejemplos con el código fuente completo.
  - **PlusSuite .NET 2.0:** Librería de libre distribucion con muchos controles mejorados para una mejor experiencia visual.

Se seleccionó Microsoft .NET Remoting 2.0 para la comunicación entre los módulos del sistema debido a que es una tecnología de alto nivel, inherente a la plataforma .NET que ya se había seleccionado anteriormente y por su probada eficacia en aplicaciones desktop distribuidas (13)**. Para la interfaz de usuario se seleccionó la Krypton Toolkit debido a que el código fuente es accesible, está muy bien documentada y la parte gratis de la librería es la que se necesita para el sistema.**

1.3.4 Metodologías.

Todo el desarrollo de software es guiado por metodologías de desarrollo que guían todos los procesos a realizar. Las metodologías de desarrollo de software se dividen en dos grandes grupos de acuerdo al énfasis realizado en la documentación del desarrollo, estos son Metodologías Ágiles de Desarrollo y Metodologías Tradicionales o Robustas. Para el desarrollo del sistema se considera entre los principales exponentes de los dos grupos: XP (Extreme Programming, Programación Extrema) y RUP (Rational Unified Process, Proceso Unificado de Desarrollo).

- **XP:** (14) Es un enfoque de la **ingeniería de software** formulado por Kent Beck, autor del primer libro sobre la materia, *Extreme Programming Explained: Embrace Change* (1999). Es el más destacado de los procesos ágiles de desarrollo de software. Al igual que estos, la programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad. Los defensores de XP consideran que los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del desarrollo de proyectos. Creen que ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos.

  Se puede considerar la programación extrema como la adopción de las mejores metodologías de desarrollo de acuerdo a lo que se pretende llevar a cabo con el proyecto, y aplicarlo de manera dinámica durante el ciclo de vida del software.

  Los principios básicos de la programación extrema son: Simplicidad, Comunicación, Retroalimentación y Coraje.

  - **Simplicidad**: La simplicidad es la base de la programación extrema. Se simplifica el diseño para agilizar el desarrollo y facilitar el mantenimiento. Un diseño complejo del código junto a sucesivas modificaciones por parte de diferentes desarrolladores hacen que la complejidad aumente exponencialmente. Para mantener la simplicidad es necesaria la refactorización del código, ésta es la manera de mantener el código simple a medida que crece. También se aplica la simplicidad

en la documentación, de esta manera el código debe comentarse en su justa medida, intentando eso sí que el código esté autodocumentado. Para ello se deben elegir adecuadamente los nombres de las variables, métodos y clases. Los nombres largos no disminuyen la eficiencia del código ni el tiempo de desarrollo gracias a las herramientas de autocompletado y refactorización que existen actualmente. Aplicando la simplicidad junto con la autoría colectiva del código y la programación por parejas se asegura que mientras más grande se haga el proyecto, todo el equipo conocerá más y mejor el sistema completo.

- **Comunicación**: La comunicación se realiza de diferentes formas. Para los programadores el código comunica mejor mientras más simple sea. Si el código es complejo hay que esforzarse para hacerlo inteligible. El código autodocumentado es más fiable que los comentarios ya que estos últimos pronto quedan desfasados con el código a medida que es modificado. Debe comentarse solo aquello que no va a variar, por ejemplo el objetivo de una clase o la funcionalidad de un método. Las pruebas unitarias son otra forma de comunicación ya que describen el diseño de las clases y los métodos al mostrar ejemplos concretos de cómo utilizar su funcionalidad. Los programadores se comunican constantemente gracias a la programación por parejas. La comunicación con el cliente es fluida ya que el cliente forma parte del equipo de desarrollo. El cliente decide que características tienen prioridad y siempre debe estar disponible para solucionar dudas.
- **Retroalimentación**: Al estar el cliente integrado en el proyecto, su opinión sobre el estado del proyecto se conoce en tiempo real. Al realizarse ciclos muy cortos tras los cuales se muestran resultados, se minimiza el tener que rehacer partes que no cumplen con los requisitos y ayuda a los programadores a centrarse en lo que es más importante. Considérense los problemas que derivan de tener ciclos muy largos. Meses de trabajo pueden tirarse por la borda debido a cambios en los criterios del cliente o malentendidos por parte del equipo de desarrollo. El código también es una fuente de retroalimentación gracias a las herramientas de desarrollo. Por ejemplo, las pruebas unitarias informan sobre el estado de salud del código. Ejecutar las pruebas unitarias frecuentemente permite descubrir fallos debidos a cambios recientes en el código.
- **Coraje o Valentía**: Los puntos anteriores parecen tener sentido común, entonces, ¿por qué coraje? Para los gerentes la programación en parejas puede ser difícil de aceptar, parece como si la productividad se fuese a reducir a la mitad ya que solo la mitad de los programadores está

escribiendo código. Hay que ser valiente para confiar en que la programación por parejas beneficia la calidad del código sin repercutir negativamente en la productividad. La simplicidad es uno de los principios más difíciles de adoptar. Se requiere coraje para implementar las características que el cliente quiere ahora sin caer en la tentación de optar por un enfoque más flexible que permita futuras modificaciones. No se debe emprender el desarrollo de grandes marcos de trabajo (Frameworks) mientras el cliente espera. En ese tiempo el cliente no recibe noticias sobre los avances del proyecto y el equipo de desarrollo no recibe retroalimentación para saber si va en la dirección correcta. La forma de construir marcos de trabajo es mediante la refactorización del código en sucesivas aproximaciones.

Las características fundamentales del método son:

- **Desarrollo iterativo e incremental**: pequeñas mejoras, unas tras otras.
- **Pruebas unitarias[12] continuas**, frecuentemente repetidas y automatizadas, incluyendo **pruebas de regresión**[13].
- **Programación en parejas**.
- Frecuente integración del equipo de programación con el cliente o usuario.
- **Corrección de todos los errores** antes de añadir nueva funcionalidad. Hacer entregas frecuentes.
- **Refactorización del código**, es decir, reescribir ciertas partes del código para aumentar su legibilidad y mantenibilidad pero sin modificar su comportamiento. Las pruebas han de garantizar que en la refactorización no se ha introducido ningún fallo.
- **Propiedad del código compartida**: en vez de dividir la responsabilidad en el desarrollo de cada módulo en grupos de trabajo distintos, este método promueve que todo el personal

[12] Prueba Unitaria: Es una forma de probar el correcto funcionamiento de un módulo de código, mediante casos de prueba extremos.

[13] Prueba de Regresión: Tipo de pruebas de software que intentan descubrir las causas de nuevos errores, carencias de funcionalidad, o divergencias funcionales con respecto al comportamiento esperado del software

pueda corregir y extender cualquier parte del proyecto. Las frecuentes pruebas de regresión garantizan que los posibles errores serán detectados.

- **Simplicidad** en el código: es la mejor manera de que las cosas funcionen. Cuando todo funcione se podrá añadir funcionalidad si es necesario. La programación extrema apuesta que es más sencillo hacer algo simple y tener un poco de trabajo extra para cambiarlo si se requiere, que realizar algo complicado y quizás nunca utilizarlo.
- La simplicidad y la comunicación son extraordinariamente complementarias. Con más comunicación resulta más fácil identificar qué se debe y qué no se debe hacer. Mientras más simple es el sistema, menos tendrá que comunicar sobre este, lo que lleva a una comunicación más completa, especialmente si se puede reducir el equipo de programadores.

- **RUP:** (15) Es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado (**UML**), constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. El RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización.

  Se caracteriza por ser iterativo e incremental, estar centrado en la arquitectura y guiado por los casos de uso. Incluye artefactos (que son los productos tangibles del proceso como por ejemplo, el modelo de casos de uso, el código fuente, etc.) y roles (papel que desempeña una persona en un determinado momento, una persona puede desempeñar distintos roles a lo largo del proceso).

  El RUP está basado en 5 principios claves que son:

  - **Adaptar el proceso:** El proceso deberá adaptarse a las características propias del proyecto u organización. El tamaño del mismo, así como su tipo o las regulaciones que lo condicionen, influirán en su diseño específico. También se deberá tener en cuenta el alcance del proyecto.
  - **Balancear prioridades:** Los requerimientos de los diversos participantes pueden ser diferentes, contradictorios o disputarse recursos limitados. *Debe encontrarse un balance que satisfaga los deseos de todos*. Debido a este balanceo se podrán corregir desacuerdos que surjan en el futuro.

- **Demostrar valor iterativamente:** Los proyectos se entregan, aunque sea de un modo interno, en etapas iteradas. En cada iteración se analiza la opinión de los inversores, la estabilidad y calidad del producto, y se refina la dirección del proyecto así como también los riesgos involucrados.
- **Elevar el nivel de abstracción:** Este principio dominante motiva el uso de conceptos reutilizables tales como patrón del software, lenguajes 4GL o marcos de referencia (Frameworks) por nombrar algunos. Esto evita que los ingenieros de software vayan directamente de los requisitos a la codificación de software a la medida del cliente, sin saber con certeza qué codificar para satisfacer de la mejor manera los requerimientos y sin comenzar desde un principio pensando en la reutilización del código. Un alto nivel de abstracción también permite discusiones sobre diversos niveles y soluciones arquitectónicas. Éstas se pueden acompañar por las representaciones visuales de la arquitectura, por ejemplo con el lenguaje UML.
- **Enfocarse en la calidad:** El control de calidad no debe realizarse al final de cada iteración, sino en todos los aspectos de la producción. El aseguramiento de la calidad forma parte del proceso de desarrollo y no de un grupo independiente.

Principales características:

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

Se seleccionó RUP como metodología de desarrollo a utilizar debido a que es la más utilizada en el mundo para el desarrollo de sistemas orientados a objeto ya que utiliza UML para el modelado, y la herramienta CASE a utilizar, Enterprise Architect, lo soporta completamente.

Conclusiones

En este capítulo se ha tratado sobre la evolución de los sistemas de video vigilancia abordando desde los sistemas analógicos que fueron los primeros hasta los modernos que son los sistemas digitales. Se mencionaron además las principales empresas del mundo en la fabricación de sistemas de este tipo, haciendo un estudio en cuanto a las desventajas que tienen estos comparados con la solución que se propone en este documento. Por último se realizó un análisis sobre las nuevas tecnologías y metodologías existentes en la actualidad, explicando las que van a ser utilizadas en el proyecto.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

Debido a la poca estructuración de los procesos de negocio y para poder comprender el contexto en el cual se desarrolla el sistema se determinó desarrollar un Modelo de Dominio, donde se expone un marco conceptual y las relaciones entre estas definiciones. Por otra parte, se enumeran los requerimientos funcionales y no funcionales, agrupándose los primeros en Casos de Uso, con el fin de estructurar el Diagrama de Casos de Uso del Sistema.

2.1 Modelo de Dominio.

El Modelo de Dominio o Modelo Conceptual, permite de manera visual mostrar al usuario los principales conceptos que se manejan en el dominio del sistema en desarrollo. Un Modelo del Dominio es una representación de las clases conceptuales del mundo real, no de componentes software. El modelo desarrollado no se trata de un conjunto de diagramas que describen clases de software u objetos de software con responsabilidades, sino que puede considerarse como un diccionario visual de las abstracciones relevantes, vocabulario e información del dominio. Aprovechando las bondades de los diagramas UML para representar conceptos, el Modelo de Dominio se presenta en forma de diagrama de clases donde figuran los principales conceptos y roles del sistema en cuestión.

2.1.1 Conceptos Fundamentales.

Para una mejor comprensión del Diagrama del Modelo de Dominio estructurado en el siguiente epígrafe, a continuación se proporciona un marco conceptual con las definiciones identificadas, estas son:

- **Entidad:** Una institución donde se controla la seguridad interna mediante personal humano.
- **Zona:** Un área física determinada de la entidad. La entidad está compuesta por zonas.
- **Guardia de Seguridad:** Persona responsable de vigilar una o más zonas.
- **Jefe de Seguridad:** Persona que tiene el deber de controlar el trabajo de uno o más guardias de seguridad.
- **Reporte:** Documento donde un guardia de seguridad archiva las ocurrencias de su turno de trabajo.

2.1.2 Diagrama del Modelo de Dominio.

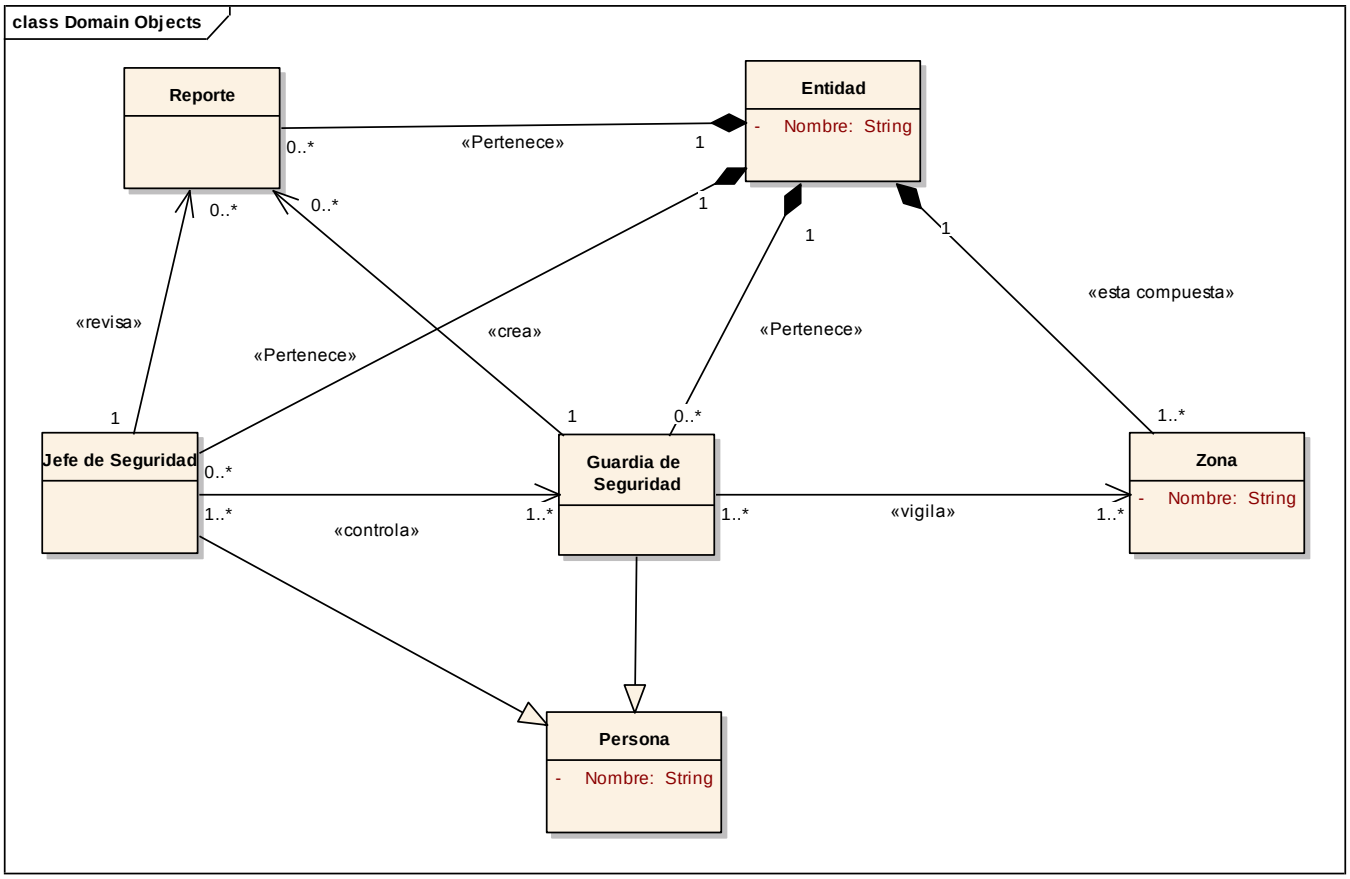

Figura 11 Modelo de Dominio

2.2 Propuesta de Sistema.

La solución que se plantea es la de desarrollar un sistema modular, que permita una futura extensión de manera flexible. El Sistema proveerá las funcionalidades de video vigilancia y manejo de cámaras IP, no sustituirá completamente al personal de seguridad, pero serían necesarios muchos menos guardias de seguridad, los cuales se apoyarían en el Sistema para su trabajo.

El Sistema propuesto es un Sistema Abierto que tendría como entrada los flujos de video obtenidos de las cámaras, y se desenvolvería en el entorno de despliegue con gran influencia de las características de la red informática local, por lo cual en su diseño se ha tenido en cuenta la adaptabilidad a los cambios en este ambiente que permitan al sistema funcionar respondiendo a esos cambios.

El Sistema propuesto consta de 2 partes fundamentales: un Módulo de Visualización (Visor para abreviar) y un Gestor Central de Información (para abreviar solo Gestor). El Gestor es el corazón del Sistema y es el que provee la información a cada instancia del Módulo de Visualización. En cada punto de interés se situaría una Cámara IP, también se podría aprovechar la capacidad de las cámaras PTZ de poder barrer ángulos de visión variables, para con una sola cámara vigilar un área más amplia, que puede incluir varios puntos de interés. El Sistema obtendría los flujos de video capturados por esas cámaras para su visualización, sin importar la localización física de esas cámaras siempre que sean accesibles por red.

El Módulo de Visualización se encontrará en cada una de las estaciones de visualización, y permitirá a los operadores manipular las cámaras, visualizar los flujos de video obtenidos de cada una de ellas de manera individual o en grupo. En caso de ser un administrador del sistema este también podrá adicionar, eliminar y modificar las cámaras en el sistema. De esta forma se podría centrar la vigilancia de un determinado número de puntos de interés en una sola estación de visualización.

Entre las ventajas que se obtendrían del funcionamiento modular está la posibilidad de existencia de varias estaciones de visualización, cada una con una instancia activa del Módulo de Visualización; desde cada una de esas instancias se pueden operar las cámaras existentes, añadir una nueva cámara o modificar una existente. Esos cambios tienen que ser visibles a todas las estaciones de visualización, el encargado de gestionar todo el flujo de información de manera que esté disponible a cada una de ellas y alerte de cambios ocurridos en los datos es el Gestor Central de Información.

Este módulo es el único con acceso a la base de datos del sistema de modo tal que cualquier intento de cambio debe de pasar por él, siendo posible obtener logs de cada operación realizada en el Sistema. También se encarga de gestionar las autenticaciones contra los usuarios registrados en la base de datos del sistema.

El Gestor está compuesto por una capa de negocio y una de acceso a datos, no es necesaria una interfaz de usuario, ya que las operaciones se realiza desde el Módulo de Visualización. El acceso a la Base de Datos se realiza mediante la librería NpgSql, que es la librería más usada para acceder a la base de datos PostgreSQL desde la plataforma .NET.

El Gestor debe de estar siempre activo en el sistema, y desde que se inicia mantiene una lista de todas las cámaras en el sistema, de su configuración y todo tipo de información necesaria para el funcionamiento de cada una de ellas, información que obtiene de la Base de Datos. También tiene una lista con los IPs autorizados a obtener información de él. La comunicación con el Gestor debe de ser a

través de .NET Remoting, el Gestor para cada cliente registra información de su localización, y del usuario que es dueño de esa conexión, y asigna un identificador temporal a ese cliente, que lo distinguirá de las demás estaciones de visualización y le permitirá realizar operaciones futuras con el Gestor.

Cuando se active un módulo de Visualización este descargará del Gestor toda la información relacionada a las cámaras y otros elementos de configuración necesarios para su funcionamiento. El operador podrá entonces manipular las cámaras o hacer cambios sobre estas si es un administrador. Cualquier modificación a los datos existentes se le solicita al Gestor que la procesa y avisa a las demás instancias del Módulo de Visualización que se encuentren activas, para que reflejen dicho cambio.

El Sistema será capaz de manipular diversos tipos de cámaras que a la vez pueden ser de distintos fabricantes. Cada cámara en dependencia del modelo y del fabricante funciona de manera distinta, por lo cual se debe de buscar un mecanismo para dar solución a este problema. Para esto el sistema soportará cada hardware a través de plugins, que serán desarrollados de manera específica para cada cámara, pero que funcionarán de manera semejante, de forma tal que el sistema puede utilizar cualquiera de estos plugins de la misma forma, no importa la cámara que sea, manteniéndose así la sencillez del diseño y la flexibilidad del sistema.

El Módulo de Visualización debe de tener una interfaz amigable, de manera que sea atractivo al usuario y que brinde además un fácil acceso a todas las operaciones del Sistema. Debe de ser un diseño basado en ventanas plegables, para también dejar disponible la mayor cantidad de área expositiva para la visualización de las cámaras.

Los Sistemas de Vigilancia se caracterizan por tener en cada estación de visualización varios monitores y televisores conectados a la PC, el Sistema debe de ser capaz de enviar la señal de una cámara o de un grupo de cámaras determinado hacia cualquier monitor o televisor. Para ello se propone además el desarrollo de controles gráficos específicos para la aplicación utilizando la plataforma .NET, que brinden la posibilidad de visualizar en cada uno de ellos cámaras individuales o grupos de ellas y que esos controles entonces puedan ser enviados a cada monitor o televisor como ventanas apartes.

Es de frecuente interés para los operadores que se encuentran en las estaciones de visualización la necesidad de agrupar un diverso número de cámaras porque están relacionadas de alguna manera (entre todas visualizan un área determinada, o un edificio en particular). En el sistema se propone el concepto de vista como la implementación de esta necesidad, en cada vista se almacenarían las cámaras que

pertenecen a ella, y que posición ocupan dentro de la vista, esta información se guardaría en la Base de Datos y estaría disponible a cada instancia del Módulo de Visualización.

Otra característica del Sistema es la capacidad de tener varios estilos de visualización para las cámaras, algunos de estos son los de Tira de Imágenes, que mostrará al operador las cámaras activas de una en una con intervalos definidos por él (este modo es muy útil cuando el número de cámaras es grande), Modo 2x2 (4 cámaras en un rectángulo de 2x2), Modo Enfocado (una Cámara enfocada y otras visualizándose más pequeñas).

El proceso de obtención de video de las cámaras y la decodificación de este es muy complejo, por lo cual el módulo de Visualización debe de hacer un uso extensivo de las capacidades multi-hilos de la plataforma .NET. La obtención y decodificación de los flujos de video obtenidos se realizan en hilos separados del encargado de realizar la visualización, además con un nivel de prioridad alto para garantizar mayor tiempo de CPU[14].

Esto garantiza estabilidad en los cuadros por segundo de los flujos de video; elemento que es muy importante para una correcta visualización, y es incluso clave en la reproducción de formatos de video tales como MJPEG[15], donde se reciben las imágenes a intervalos prefijados, y cualquier violación de este parámetro deriva en pérdida de información. En adición a esto los cuadros por segundo al que se obtendrían los flujos de video de cada cámara serán configurables para que estén de acuerdo con los cambios en la carga de la red o las posibilidades físicas de estas.

2.3 Requerimientos Funcionales del Sistema.

2.3.1 Requerimientos Funcionales del Visor

RF1 Gestionar Vistas.

RF1.1 Crear Vistas

[14] CPU: Central Processor Unit. Unidad de Procesamiento Central. Es la encargada de todas las operaciones en la computadora.

[15] MJPEG: Motion Joint Photographic Experts Group, formato de codificación de video digital.

Este requerimiento va a permitir crear una vista nueva, definiendo cuantas y cuales cámaras se van a mostrar, y en qué lugar.

RF1.2 Eliminar Vistas

Este requerimiento va a permitir eliminar una o varias vistas del visor.

RF1.3 Modificar Vistas

Este requerimiento va a permitir modificar la dirección de las cámaras.

RF2 Gestionar Cámara

RF2.1 Adicionar cámara

Este requerimiento va a permitir adicionar una nueva cámara al sistema con todas sus propiedades.

RF2.2 Editar cámara

Este requerimiento va a permitir cambiarle el ip y los datos de acceso a la cámara.

RF2.3 Eliminar Cámara

Este requerimiento va a permitir eliminar una cámara al sistema.

RF3 Manipular cámara

Este requerimiento va a permitir manipular cámaras utilizando funcionalidades individuales de cada unas de las cámaras de acuerdo a las capacidades de estas, además debe mover la cámara de manera remota, ajustar zoom, foco, iris etc.

RF4 Gestionar flujo de video

RF4.1 Obtener flujos de video de las cámaras.

Este requerimiento permite que mediante una petición se capture los flujos de videos y se visualicen.

RF4.2 Transformar flujos de video obtenidos de la cámara.

Este requerimiento va a permitir hacerle transformaciones a los flujos de video tales como: brillo y contraste.

RF5 Visualizar

RF5.1 Cambiar estilos de visualización.

Este requerimiento va a permitir tener diferentes modos de trabajo del visualizador los cuales son: Modo libre, modo tira de imágenes y otros.

RF5.2 Manipular varios visualizadores.

Este requerimiento permite tener abierto varios visualizadores con diferentes modos de trabajo y con diferentes cámaras.

RF6 Configurar visor.

RF6.1 Configurar acceso al Gestor.

Este requerimiento va a permitir configurar el acceso al gestor mediante un ip, un puerto y un nombre.

RF6.2 Configurar localización de los plugins de hardware.

Este requerimiento va a permitir localizar los plugins de hardware, es decir los drivers de la cámara con la cual se va a trabajar, esto permite que cualquier cámara que se conecte pueda trabajar con el sistema.

RF6.3 Actualizar configuración.

Este requerimiento permite recargar la configuración y aplicarla.

RF7 Comunicarse con Sistemas Externos.

RF7.1 Obtener información del Gestor.

Este requerimiento permite conectarse al gestor y pedirle la información necesaria para su funcionamiento.

RF7.2 Capturar eventos sobre cambios en el Sistema.

RF8 Gestionar Zonas Físicas.

RF8.1 Adicionar Zona.

Adicionar una zona que representa un área física dentro del lugar de despliegue del Sistema.

RF8.2 Editar Zona.

Editar una zona que representa un área física dentro del lugar de despliegue del Sistema.

RF8.3 Eliminar Zona.

Eliminar una zona que representa un área física dentro del lugar de despliegue del Sistema.

RF9 Gestionar Zonas Lógicas.

RF9.1 Adicionar Zona Lógica.

Adicionar una zona para agrupar un conjunto de Vistas relacionadas.

RF9.2 Editar Zona Lógica.

RF9.3 Eliminar Zona Lógica.

2.3.2 Requerimientos Funcionales del Gestor.

RF10 Controlar los visores conectados al sistema.

RF10.1 Recibir cuando se modifica una zona, una vista o una cámara y actualizar los datos en el Sistema.

RF10.2 Enviar información a cada visor cuando otro haga una modificación en los datos del Sistema.

RF11 Registrar Logs de Actividad.

RF12 Enviar información del Sistema a cada uno de los visores (cámaras, vistas y zonas).

2.4 Requerimientos No Funcionales del Sistema.

2.4.2 Rendimiento

- **RNF1** El sistema debe permitir la visualización simultánea de al menos 25 cámaras.

2.4.3 Software

- **RNF2** Windows XP SP2 o superior.
- **RNF3** NET Framework 2.0 o superior.
- **RNF4** DirectX 9.0 o superior.

2.4.4 Hardware (Valores recomendados, escalables a las disposiciones reales de explotación).

- **RNF5** 2GB RAM (recomendado).

Descripción:

Debido a que el procesamiento del gran número de flujos de video, es necesario que la computadora cuente con una potente memoria para realizar lo más rápido posible todas las operaciones que el cliente solicite.

- **RNF6** Procesador P 4 a 3.0 GHz o superior (recomendado).

Descripción:

Debido a que el procesamiento del gran número de flujos de video, es necesario que la computadora cuente con una potente memoria para realizar lo más rápido posible todas las operaciones que el cliente solicite.

- **RNF7** Gigabit Ethernet NIC (recomendado).

Descripción:

Provee un ambiente de red muy eficaz, lo cual es muy beneficioso para minimizar el tiempo de las operaciones de visualización de los flujos de video.

2.4.5 Soporte.

- **RNF8** Se necesita los .Net Framework 2.0 o superior instalado en todas las estaciones de Trabajo.
- **RNF9** Se necesita Windows XP SP2 o superior instalado en todas las estaciones de Trabajo.
- **RNF10** Se necesita DirectX 9.0 o superior instalado en todas las estaciones de Trabajo.

2.4.6 Restricciones de diseño.

- **RNF11** Resolución de 1024 x 768.

Descripción:

La aplicación está optimizada para una resolución de 1024x768.

2.4.7 Seguridad.

- **RNF12** Se garantizará la seguridad del Sistema mediante la autenticación de usuarios con varios niveles de permisos.

2.5 Definición de los casos de uso.

2.6.1 Definición de los actores.

Actor	**Descripción**
Administrador	Rol responsable de velar por el funcionamiento apropiado del sistema, configurarlo y realizar todas las demás funcionalidades que permite el sistema.

Operador	Rol que se beneficia de diversas funcionalidades del sistema pero no tiene acceso a las configuraciones del mismo.
Usuario	Rol que representa una generalización especialización de los actores Administrador y Operador.
Sistema	Rol ficticio para representar al Sistema como iniciador de algún caso de uso.
Visor	Rol ficticio para representar al Visor como iniciador de algún caso de uso.
Gestor	Rol ficticio para representar al Gestor como iniciador de algún caso de uso.

2.6.2 Listado de los casos de uso

CU-1	Gestionar Usuarios
Actor	Administrador
Descripción	Adicionar y eliminar usuarios.
Referencia	RNF12

CU-2	Gestionar Cámaras
Actor	Administrador
Descripción	El Administrador añade, edita o elimina una cámara.
Referencia	RF2.1, RF2.2, RF2.3

CU-3	Gestionar Vistas
Actor	Administrador
Descripción	El Administrador añade, edita o elimina una vista.
Referencia	RF1.1, RF1.2, RF1.3

CU-4	Gestionar Flujos de Video
Actor	Usuario
Descripción	El usuario ordena la visualización del flujo de video obtenido de una cámara o lo modifica.
Referencia	RF4.1, RF4.2

CU-5	Visualizar
Actor	Usuario
Descripción	El usuario abre 1 o varios visualizadores y puede cambiar el estilo de visualización de un visualizador determinado.
Referencia	RF5.1, RF5.2

CU-6	Manipular Cámara
Actor	Usuario
Descripción	En dependencia de la cámara que este seleccionada el usuario opera sobre capacidades individuales de esta, tales como mover la cámara, hacer zoom etc.
Referencia	RF3

CU-7	Configurar Visor
Actor	Administrador
Descripción	El Administrador configura los parámetros de accesos al gestor, la localización de los plugins de hardware, y actualiza el Visor para los cambios realizados.
Referencia	RF6.1, RF6.2, RF6.3

CU-8	Comunicarse con Sistemas Externos

Actor	Visor
Descripción	El Sistema cuando es necesario se comunica con el Gestor de Información para obtener alguna información en el Sistema.
Referencia	RF7.1

CU-9	Gestionar Zonas Físicas
Actor	Administrador
Descripción	El Administrador adiciona, edita o elimina zonas que representan una ubicación física en la instalación donde se encuentra desplegado el Sistema.
Referencia	RF8.1, RF8.2, RF8.3

CU-10	Gestionar Zonas Lógicas
Actor	Administrador
Descripción	El Administrador adiciona, edita o elimina zonas lógicas para agrupar vistas en el Sistema.
Referencia	RF9.1, RF9.2, RF9.3

CU-11	Controlar los módulos conectados al Sistema.
Actor	Visor
Descripción	El Gestor controla el flujo de eventos entre los visores.
Referencia	RF7.2,RF10.1, RF10.2

CU-12	Registrar Logs de Actividad.
Actor	Gestor
Descripción	Se registran logs de la actividad de cada instancia de los módulos con respecto a la información almacenada en el Sistema.

Referencia	RF11

CU-13	Servir Información del Sistema
Actor	Visor
Descripción	El Gestor le envía al Visor información almacenada en el Sistema acerca de las Cámaras, las Zonas Físicas y Lógicas y las Vistas.
Referencia	RF12

CU-14	Autenticar Usuarios
Actor	Usuario
Descripción	Autenticar un Usuario mediante su nombre y contraseña.
Referencia	RNF12

2.6 Diagramas de Casos de Uso

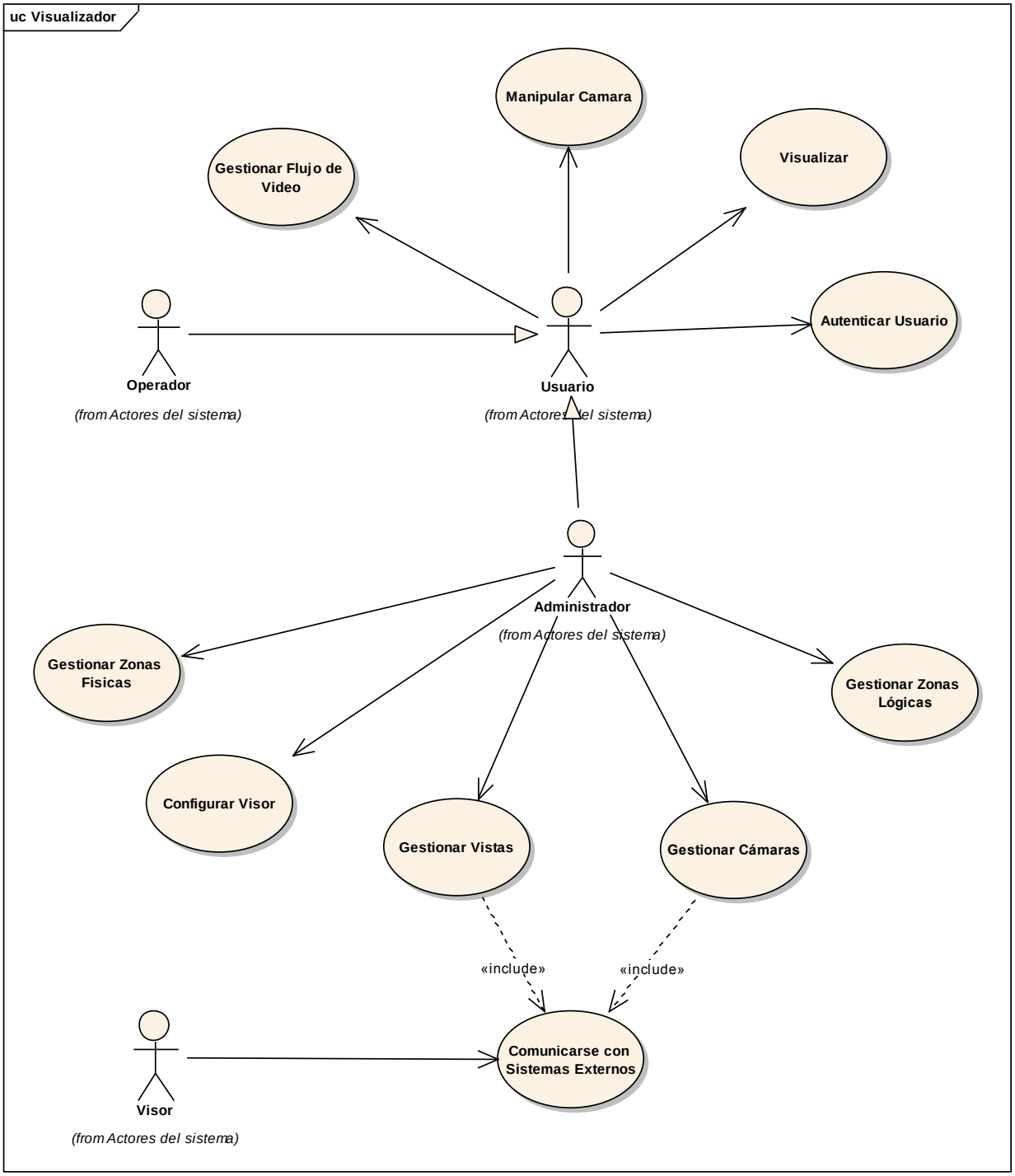

Figura 12 Diagrama de Casos de Uso del Visor

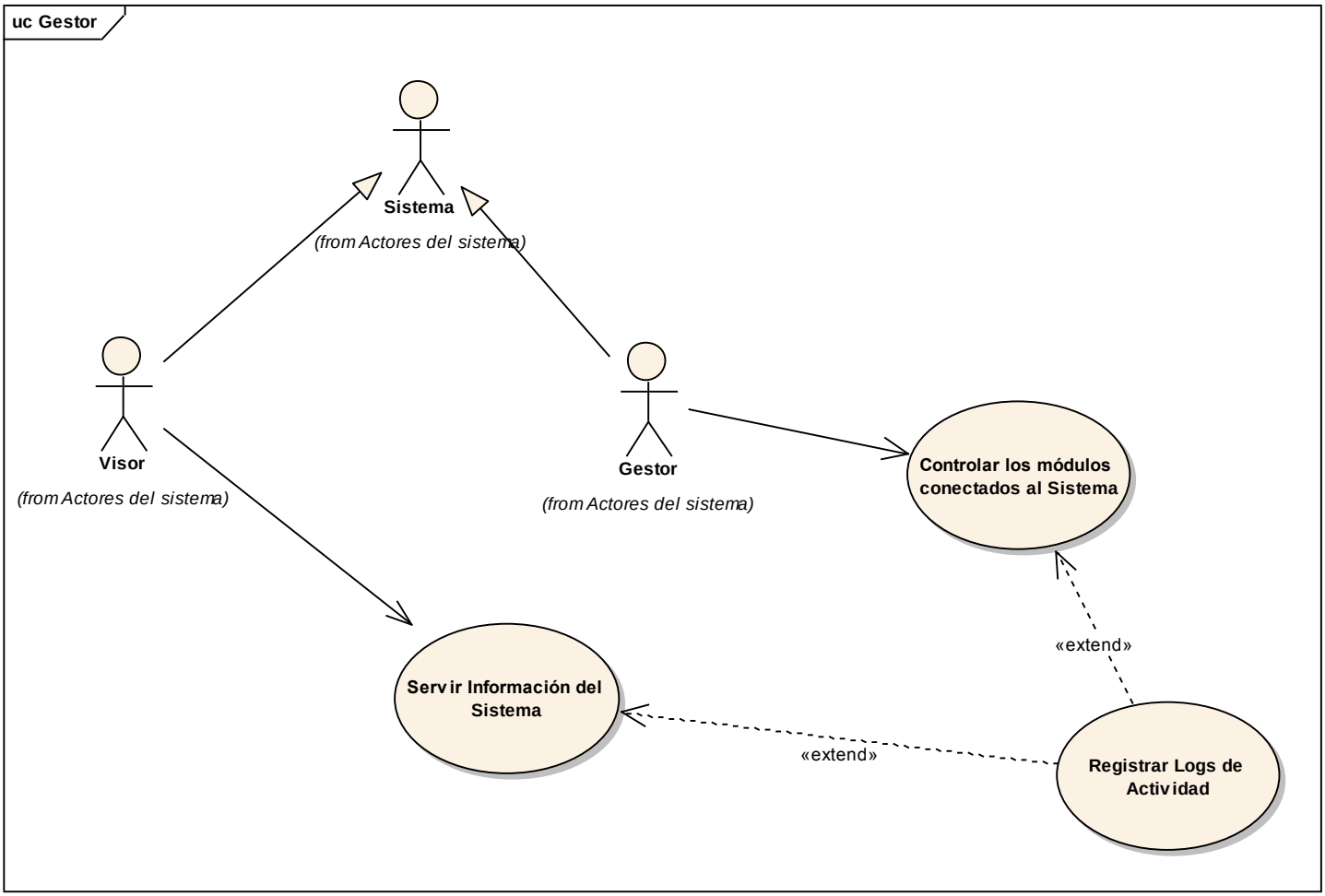

Figura 13 Diagrama de Casos de Uso del Gestor

2.7 Casos de Uso por Ciclo

2.7.1 Primer Ciclo de Desarrollo

Código	Nombre de caso de uso	Paquete	Justificación de la selección.

CU-13	Servir Información del Sistema	Gestor	Es la fuente de la información con la que trabajan los visores.
CU-2	Gestionar Cámaras	Visor	Es una de las funcionalidades bases del Sistema
CU-3	Gestionar Vistas	Visor	Es una de las funcionalidades bases del Sistema
CU-4	Gestionar Flujos de Video	Visor	Es una de las funcionalidades bases del Sistema
CU-5	Visualizar	Visor	Es una de las funcionalidades bases del Sistema
CU-6	Manipular Cámara	Visor	Es una de las funcionalidades bases del Sistema
CU-7	Configurar visor	Visor	Es necesario para poder usar el Visor, incluso en ambientes de prueba.
CU-8	Comunicarse con Sistemas Externos	Visor	Es una de las funcionalidades bases del Sistema
CU-9	Gestionar Zonas Físicas	Visor	Es una de las funcionalidades bases del Sistema
CU-10	Gestionar Zonas Lógicas	Visor	Es una de las funcionalidades bases del Sistema

2.7.2 Segundo Ciclo de Desarrollo

Código	**Nombre de caso de uso**	**Paquete**	**Justificación de la selección.**
CU-1	Gestionar Usuarios	Visor	Es secundario durante el desarrollo
CU-11	Controlar los módulos conectados al Sistema.	Gestor	Requiere un nivel de madurez avanzado en el Sistema.
CU-12	Registrar Logs de Actividad.	Gestor	No es una de las funcionalidades prioritarias durante el desarrollo

CU-14	Autenticar Usuarios	Gestor	Es secundario durante el desarrollo

2.8 Casos de Uso Expandidos

Caso de uso		
CU-1	Gestionar Usuarios	
Propósito	Garantizar la seguridad en el Sistema mediante el uso de usuarios con privilegios distintos, que están protegidos por contraseña.	
Actores: Administrador		
Resumen: El caso de uso se inicia cuando el Administrador selecciona la opción de adicionar o eliminar o editar un usuario determinado.		
Referencias	RNF12	
Acción del actor		**Respuesta del sistema**
Escenario Adicionar		
1.- El Administrador selecciona la opción de adicionar un usuario.		2.- El Sistema le muestra una forma para introducir los datos necesarios.
3.- El Administrador entra los datos necesarios y acepta la entrada.		4.- El Sistema almacena los nuevos datos.
Escenario Eliminar		
1.- El Administrador selecciona la opción de eliminar un usuario.		2.- El Sistema muestra los usuarios que existen.
3.-El Administrador selecciona un usuario y acepta.		4.- El Sistema muestra una advertencia.
5.-El Administrador elige continuar con la operación.		6.- El Sistema elimina el usuario.
Escenario Editar		
1.- El Administrador selecciona la opción de editar un usuario.		2.- El Sistema muestra los usuarios que existen.
3.-El Administrador selecciona un usuario.		4.- El Sistema muestra los datos del usuario.

5. El Administrador modifica los datos y acepta.	6.-El Sistema almacena los datos modificados.
Casos de Uso Asociados	CU-8 Comunicarse con Sistemas Externos(caso de uso incluido)
Puntos de extensión	Se usa el caso de uso Comunicarse con Sistemas Externos en: Escenario Adicionar: Acción 4 Escenario Eliminar: Acción 2,4 Escenario Editar: Acción 2,4

Caso de uso	
CU-2	Gestionar Cámaras
Propósito	Modificar las cámaras a manejar por el Sistema.
Actores: Administrador	
Resumen: El caso de uso se inicia cuando el Administrador selecciona la opción de adicionar o eliminar o editar una cámara.	
Referencias	RF2.1, RF2.2, RF2.3

Acción del actor	**Respuesta del sistema**
Escenario Adicionar	
1.- El Administrador selecciona la opción de adicionar una cámara.	2.- El Sistema le muestra una forma para introducir los datos necesarios.
3.- El Administrador entra los datos necesarios y acepta la entrada.	4.- El Sistema almacena los nuevos datos.
Escenario Eliminar	
1.- El Administrador selecciona la opción de eliminar una cámara.	2.- El Sistema muestra una advertencia.
3.-El Administrador elige continuar con la operación.	4.- El Sistema elimina la cámara.
Escenario Editar	

1.- El Administrador selecciona la opción de editar una cámara.	2.- El Sistema muestra los datos de la cámara.
5. El Administrador modifica los datos y acepta.	6.-El Sistema almacena los datos modificados.
Casos de Uso Asociados	CU-8 Comunicarse con Sistemas Externos(caso de uso incluido)
Puntos de extensión	Se usa el caso de uso Comunicarse con Sistemas Externos en: Escenario Adicionar: Acción 4 Escenario Eliminar: Acción 4 Escenario Editar: Acción 2,6

Caso de uso	
CU-3	Gestionar Vistas
Propósito	Modificar las vistas a manejar por el Sistema.
Actores: Administrador	
Resumen: El caso de uso se inicia cuando el Administrador selecciona la opción de adicionar o eliminar o editar una vista.	
Referencias	RF3.1, RF3.2, RF3.3

Acción del actor	**Respuesta del sistema**
Escenario Adicionar	
1.- El Administrador selecciona la opción de adicionar una vista.	2.- El Sistema le muestra una forma para introducir los datos necesarios.
3.- El Administrador entra los datos necesarios y acepta la entrada.	4.- El Sistema almacena los nuevos datos.
Escenario Eliminar	
1.- El Administrador selecciona la opción de eliminar una vista.	2.- El Sistema muestra una advertencia.
3.-El Administrador elige continuar con la operación.	4.- El Sistema elimina la vista.

Escenario Editar	
1.- El Administrador selecciona la opción de editar una vista.	2.- El Sistema muestra los datos de la vista.
5. El Administrador modifica los datos y acepta.	6.-El Sistema almacena los datos modificados.
Casos de Uso Asociados	CU-8 Comunicarse con Sistemas Externos(caso de uso incluido)
Puntos de extensión	Se usa el caso de uso Comunicarse con Sistemas Externos en: Escenario Adicionar: Acción 4 Escenario Eliminar: Acción 4 Escenario Editar: Acción 2,6

Caso de uso		
CU-4	Gestionar Flujos de Video	
Propósito	Obtención y modificación de los flujos de video obtenidos de las cámaras.	
Actores: Usuario		
Resumen: El Usuario ordena visualizar una cámara o un grupo de cámaras, o selecciona una que se esté visualizando, y elige modificar algún parámetro en el flujo de video.		
Referencias	RF4.1, RF4.2	
Acción del actor		**Respuesta del sistema**
Escenario Obtener Flujo de Video		
1.- El Usuario selecciona la opción de visualizar una cámara o vista.		2.- El Sistema obtiene el flujo de video de la cámara.
		4.- El Sistema visualiza el flujo de video obtenido.
Escenario Modificar Flujo de Video		
1.- El Usuario selecciona una cámara		2.- El Sistema muestra los parámetros posibles a modificar.
3.-El Usuario modifica los parámetros que		4.- El Sistema aplica los cambios.

requiera.	

Caso de uso	
CU-5	Visualizar
Propósito	Manejar la interfaz de visualización del Sistema.
Actores: Usuario	
Resumen: Permite al usuario crear más de un área de visualización, y cambiar los modos en que trabaja esa área de acuerdo a estilos predefinidos.	
Referencias	RF5.1, RF5.2
Acción del actor	**Respuesta del sistema**
Escenario Crear Visualizador	
1.- El Usuario selecciona la opción de crear un nuevo visualizador.	2.- El Sistema crea un visualizador vacio.
Escenario Cambiar Modo de Trabajo	
1.- El Usuario selecciona la opción de cambiar modo de trabajo en el visualizador activo.	2.- El Sistema aplica el cambio.

Caso de uso	
CU-6	Manipular Cámara
Propósito	Utilizar las capacidades individuales de cada cámara
Actores: Usuario	
Resumen: Permite al usuario acceder a las funcionalidades individuales de cada cámara que el Sistema soporta.	
Referencias	RF3
Acción del actor	**Respuesta del sistema**
1.- El Usuario selecciona una cámara determinada	2.- El Sistema muestra las opciones de manipulación.
3.- El Usuario modifica los controles.	4.-El Sistema aplica los cambios

Caso de uso	
CU-7	Configurar Visor
Propósito	Configurar el visor para su correcto funcionamiento.
Actores: Administrador	
Resumen: El caso de uso se inicia cuando el Administrador selecciona la opción de configurar el Visor.	
Referencias	RF6.1, RF6.2, RF6.3

Acción del actor	**Respuesta del sistema**
1.- El Administrador selecciona la opción configurar Visor.	2.- El Sistema muestra las opciones de configuración.
3.- El Administrador modifica las opciones y acepta.	4.-El Sistema aplica los cambios

Caso de uso	
CU-8	Comunicarse con Sistemas Externos
Propósito	Proporcionar una vía de entrada de información al Visor.
Actores: Visor	
Resumen: El caso de uso se inicia cuando el Visor necesita información del Gestor.	
Referencias	RF7.1

Acción del actor	**Respuesta del sistema**
1.- El Visor solicita las cámaras del Sistema al Gestor.	2.- El Gestor procesa la solicitud y envía la información solicitada.
3.- El Visor solicita las zonas físicas del Sistema al Gestor.	4.- El Gestor procesa la solicitud y envía la información solicitada.
5.- El Visor solicita las vistas del Sistema al Gestor.	6.- El Gestor procesa la solicitud y envía la información solicitada.

7.- El Visor solicita las zonas lógicas del Sistema al Gestor.	8.- El Gestor procesa la solicitud y envía la información solicitada.

Caso de uso		
CU-9	Gestionar Zonas Físicas	
Propósito	Modificar las zonas físicas a manejar por el Sistema.	
Actores: Administrador		
Resumen: El caso de uso se inicia cuando el Administrador selecciona la opción de adicionar o eliminar o editar una zona física.		
Referencias	RF8.1, RF8.2, RF8.3	
Acción del actor		**Respuesta del sistema**
Escenario Adicionar		
1.- El Administrador selecciona la opción de adicionar una zona.		2.- El Sistema le muestra una forma para introducir los datos necesarios.
3.- El Administrador entra los datos necesarios y acepta la entrada.		4.- El Sistema almacena los nuevos datos.
Escenario Eliminar		
1.- El Administrador selecciona la opción de eliminar una zona.		2.- El Sistema muestra una advertencia.
3.-El Administrador elige continuar con la operación.		4.- El Sistema elimina la zona.
Escenario Editar		
1.- El Administrador selecciona la opción de editar una zona.		2.- El Sistema muestra los datos de la zona.
5. El Administrador modifica los datos y acepta.		6.-El Sistema almacena los datos modificados.
Casos de Uso Asociados		CU-8 Comunicarse con Sistemas Externos(caso de uso incluido)
Puntos de extensión		Se usa el caso de uso Comunicarse con Sistemas

	Externos en: Escenario Adicionar: Acción 4 Escenario Eliminar: Acción 4 Escenario Editar: Acción 2,6

Caso de uso		
CU-10	Gestionar Zonas Lógicas	
Propósito	Modificar las zonas lógicas a manejar por el Sistema.	
Actores: Administrador		
Resumen: El caso de uso se inicia cuando el Administrador selecciona la opción de adicionar o eliminar o editar una zona lógica.		
Referencias	RF9.1, RF9.2, RF9.3	
Acción del actor		**Respuesta del sistema**
Escenario Adicionar		
1.- El Administrador selecciona la opción de adicionar una zona.		2.- El Sistema le muestra una forma para introducir los datos necesarios.
3.- El Administrador entra los datos necesarios y acepta la entrada.		4.- El Sistema almacena los nuevos datos.
Escenario Eliminar		
1.- El Administrador selecciona la opción de eliminar una zona.		2.- El Sistema muestra una advertencia.
3.-El Administrador elige continuar con la operación.		4.- El Sistema elimina la zona.
Escenario Editar		
1.- El Administrador selecciona la opción de editar una zona.		2.- El Sistema muestra los datos de la zona.
5. El Administrador modifica los datos y acepta.		6.-El Sistema almacena los datos modificados.
Casos de Uso Asociados		CU-8 Comunicarse con Sistemas Externos(caso de

	uso incluido)
Puntos de extensión	Se usa el caso de uso Comunicarse con Sistemas Externos en: Escenario Adicionar: Acción 4 Escenario Eliminar: Acción 4 Escenario Editar: Acción 2,6

Caso de uso		
CU-11	Controlar los módulos conectados al Sistema.	
Propósito	El Gestor controla el flujo de eventos entre los visores.	
Actores: Visor		
Resumen: El caso de uso se inicia cuando una instancia del Visor hace alguna modificación en la información gestionada por el Sistema, el gestor procesa el evento y alerta a las instancias del Visor.		
Referencias	RF7.2,RF10.1, RF10.2	
Acción del actor		**Respuesta del sistema**
Escenario Cámaras Modificadas		
1.- El Visor hace una modificación en las cámaras		2.- El Gestor captura el evento y procesa la información.
		3.- El Gestor informa a las instancias del Módulo de Visualización de la ocurrencia del evento, y proporciona información acerca de este.
4.-El Visor Captura la información y la procesa.		5.- El Gestor archiva el log de la actividad
Escenario Zonas Físicas Modificadas		
1.- El Visor hace una modificación en las zonas físicas.		2.- El Gestor captura el evento y procesa la información.

	3.- El Gestor informa a las demás instancias del Módulo de Visualización de la ocurrencia del evento, y proporciona información acerca de este.
4.-El Visor Captura la información y la procesa.	5.- El Gestor archiva el log de la actividad
Escenario Vistas Modificadas	
1.- El Visor hace una modificación en las vistas.	2.- El Gestor captura el evento y procesa la información.
	3.- El Gestor informa a las demás instancias del Módulo de Visualización de la ocurrencia del evento, y proporciona información acerca de este.
4.-El Visor Captura la información y la procesa.	5.- El Gestor archiva el log de la actividad
Escenario Zonas Lógicas Modificadas	
1.- El Visor hace una modificación en las zonas lógicas.	2.- El Gestor captura el evento y procesa la información.
	3.- El Gestor informa a las demás instancias del Módulo de Visualización de la ocurrencia del evento, y proporciona información acerca de este.
4.-El Visor Captura la información y la procesa.	5.- El Gestor archiva el log de la actividad
Casos de Uso Asociados	CU-12 Registrar Logs de Actividad
Puntos de Extensión	CU-12 en Actividad 5 de Todos los Escenarios.

Caso de uso	
CU-12	Registrar Logs de Actividad
Propósito	El Gestor archiva logs de cada una de las operaciones.

Actores: Gestor		
Resumen: El caso de uso se inicia cuando se realiza alguna operación en el Gestor que incurre en el acceso o la modificación a alguna información del Sistema.		
Referencias	RF11	
Acción del actor		**Respuesta del sistema**
		1.-El Gestor formatea la información de la actividad.
		2.-El Gestor archiva el log de la actividad.

Caso de uso		
CU-13	Servir Información del Sistema	
Propósito	El Gestor envía sobre demanda información acerca de las cámaras y otros elementos del sistema a alguna de las instancias del Módulo de Visualización.	
Actores: Visor		
Resumen: El caso de uso se inicia cuando se realiza alguna operación en el Gestor que incurre en el acceso o la modificación a alguna información del Sistema.		
Referencias	RF12	
Acción del actor		**Respuesta del sistema**
1.-El Visor solicita información sobre alguno de los elementos del Sistema al Gestor.		2.-El Gestor envía la información al visor.
		3.-El Gestor archiva el log de la actividad.
Casos de Uso Asociados		CU-12 Registrar Logs de Actividad
Puntos de Extensión		CU-12 en Actividad 3

Caso de uso	
CU-14	Autenticar Usuario
Propósito	Garantizar que la persona que esté operando el Sistema tenga los derechos requeridos.
Actores: Usuario	

Resumen: El caso de uso se inicia cuando el Usuario solicita autenticarse.		
Referencias	RNF12	
Acción del actor		**Respuesta del sistema**
1.-El Usuario selecciona la opción de autenticarse.		2.-El Sistema muestra una forma para entrar los datos del usuario.
3.-El Usuario entra los datos y selecciona Aceptar.		4.-El Sistema valida los datos del Usuario.
Casos de Uso Asociados		CU-8 Comunicarse con Sistemas Externos(caso de uso incluido)
Puntos de extensión		Se usa el caso de uso Comunicarse con Sistemas Externos en: Acción 4

Conclusiones

En este capítulo se establecieron las características principales del sistema así como el proceso para su funcionamiento. Se muestra el modelo del dominio, conjuntamente con la especificación de los requisitos funcionales y no funcionales. Además se elaboró el modelo de caso de usos del sistema.

CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA

3.1 Arquitectura.

El Sistema está diseñado para seguir una arquitectura primaria, en forma de pizarra en su variante de tablero de control. Esta desacopla el sistema en componentes denominados agentes autónomos, los cuales son independientes en la realización atómica de su funcionalidad. Pero dependen de una entrada de información externa, que es provista por otros agentes, y a su vez producen un resultado que puede a su vez ser entrada de otros agentes.

Esta forma de trabajar brinda grandes beneficios en la modularización del Sistema, que gana en flexibilidad y posibilidades de evolución. 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 Visor uno de los agentes autónomos que funcionarían con el Sistema, en las Recomendaciones se proponen otros agentes autónomos que podrían ser incluidos.

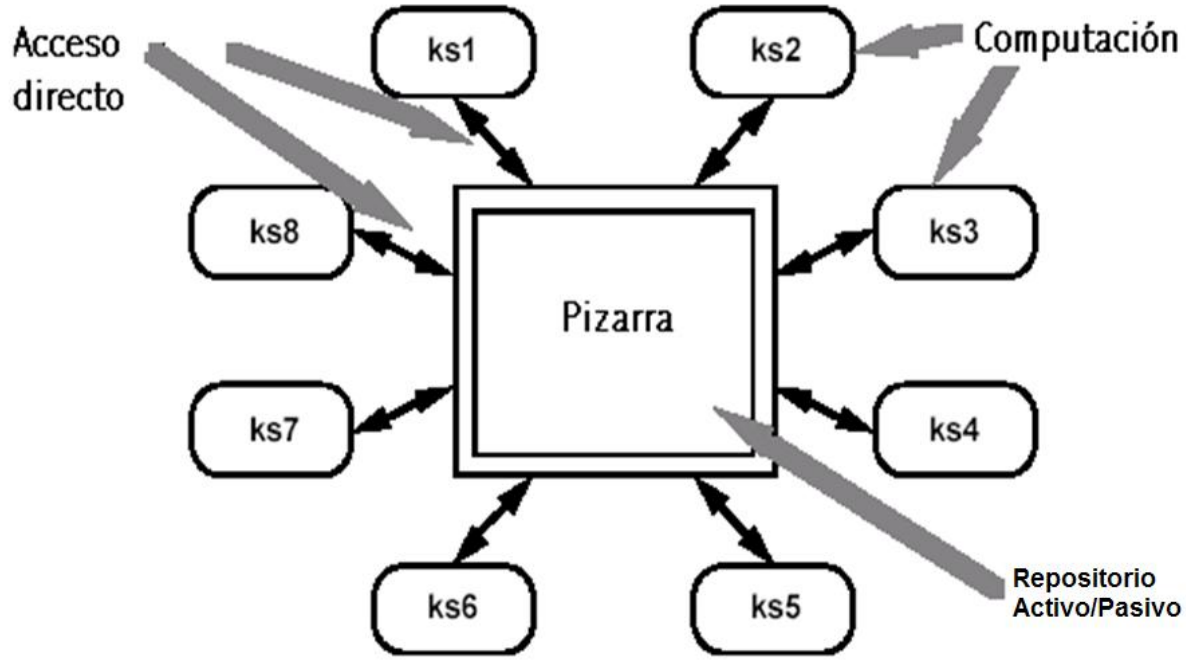

Figura 14 Arquitectura en Pizarra

La elección de este estilo de arquitectura se basa en que el mundo de la video vigilancia está en constante movimiento y evolución. Con frecuencia surgen nuevas tecnologías que son adoptadas automáticamente en la realización de estos sistemas, o se desarrollan algunas específicamente para ellos. Estos cambios van desde hardwares con funcionalidades novedosas hasta nuevas características o prestaciones del software que todos se apresuran a incorporar para no quedar en desventaja competitiva. La Arquitectura en Pizarra brinda una sólida estructura, que es flexible al cambio, por lo cual es una respuesta adecuada a las necesidades del Sistema.

Para afrontar los cambios frecuentes en el hardware, y a la vez para soportar diferentes tipos de este se usa un estilo basado en Plugins o Extensiones. Para cada modelo de hardware se crea un plugin, compuesto por elementos de software que implementan interfaces definidas en el Sistema. Estos los fuerzan a funcionar de una forma estándar y permite al sistema soportar cualquier hardware siempre que cuente con el plugin específico para este. Su complejidad estructural es baja, el mayor esfuerzo está en manejar las capacidades individuales del hardware. Esto permite desarrollar los plugins en períodos de tiempo eficiente para los clientes, lo que puede llegar incluso a horas en dependencia de la complejidad del hardware.

Aunque el Sistema mantiene una Arquitectura de Pizarra a manera global, en cada uno de los módulos se usa el estilo Modelo-Vista-Controlador (MVC) y el de 3 Capas. El primero afronta una de las consecuencias de los cambios frecuentes en los soportes de hardware y de añadir nuevas funcionalidades, que es la de constantes modificaciones en la interfaz de usuario. Se separa los módulos en 3 componentes fundamentales, el Modelo, la Vista y el Controlador. El Modelo es inmutable a los cambios en la interfaz de usuario, se encarga de toda la gestión interna del negocio, la Vista se encarga de presentar los datos y de interactuar con el usuario, mientras que el controlador maneja todos los cambios de estado de la Vista. Mediante el uso de

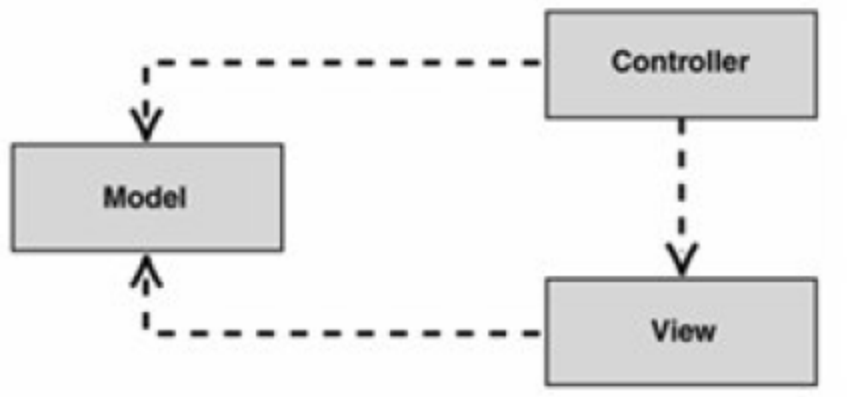

Figura 15 Patrón Modelo-Vista-Controlador

este patrón el Visor es capaz de utilizar diferentes interfaces de usuario manteniendo la misma lógica de negocio. El segundo distribuye todo el diseño del módulo en 3 capas bien definidas, cada una de ellas agrupando clases con tareas muy relacionadas en pos de una meta general. En el caso del Sistema se usa para delimitar las capas de Presentación, Negocio y Acceso a Datos.

La combinación de estos estilos arquitectónicos enfrenta los posibles escenarios a los que el Sistema debe enfrentarse en su funcionamiento, por lo cual fueron seleccionados para plantear el diseño completo del Sistema.

3.2 Modelo de Análisis.

El análisis consiste en obtener una visión del sistema que se preocupa de ver que es lo que hace el mismo. El modelo de análisis va a ser la entrada fundamental para el comienzo del diseño.

Diagrama de clases de análisis.

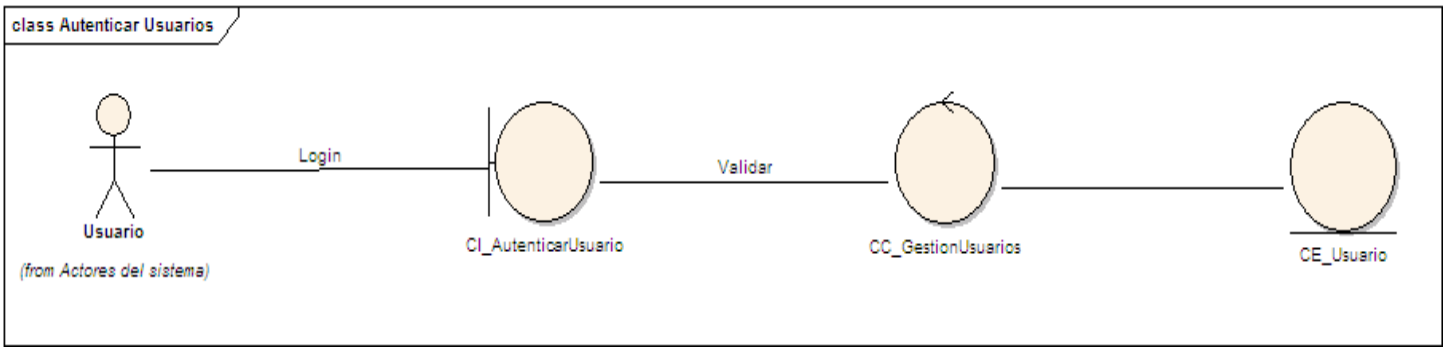

Figura 16 CU Autenticar usuario.

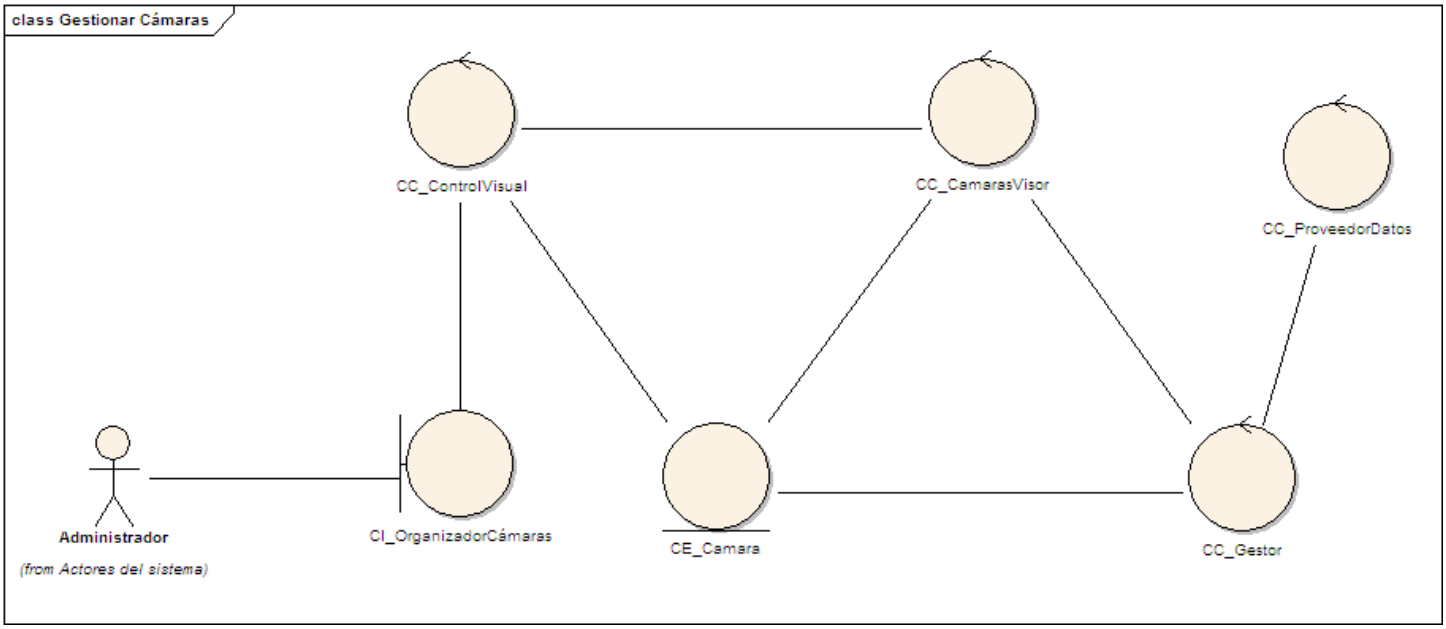

Figura 17 CU Gestionar Cámaras.

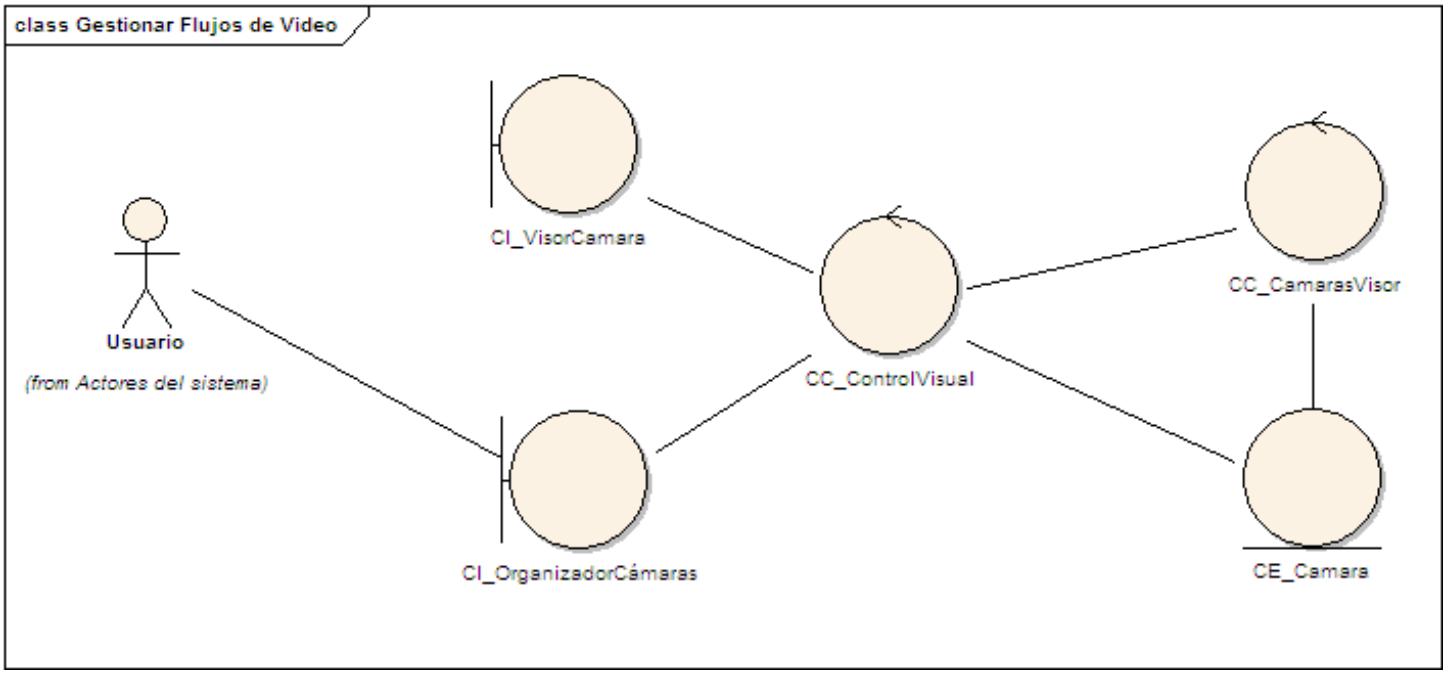

Figura 18 CU Gestionar Flujos de Video.

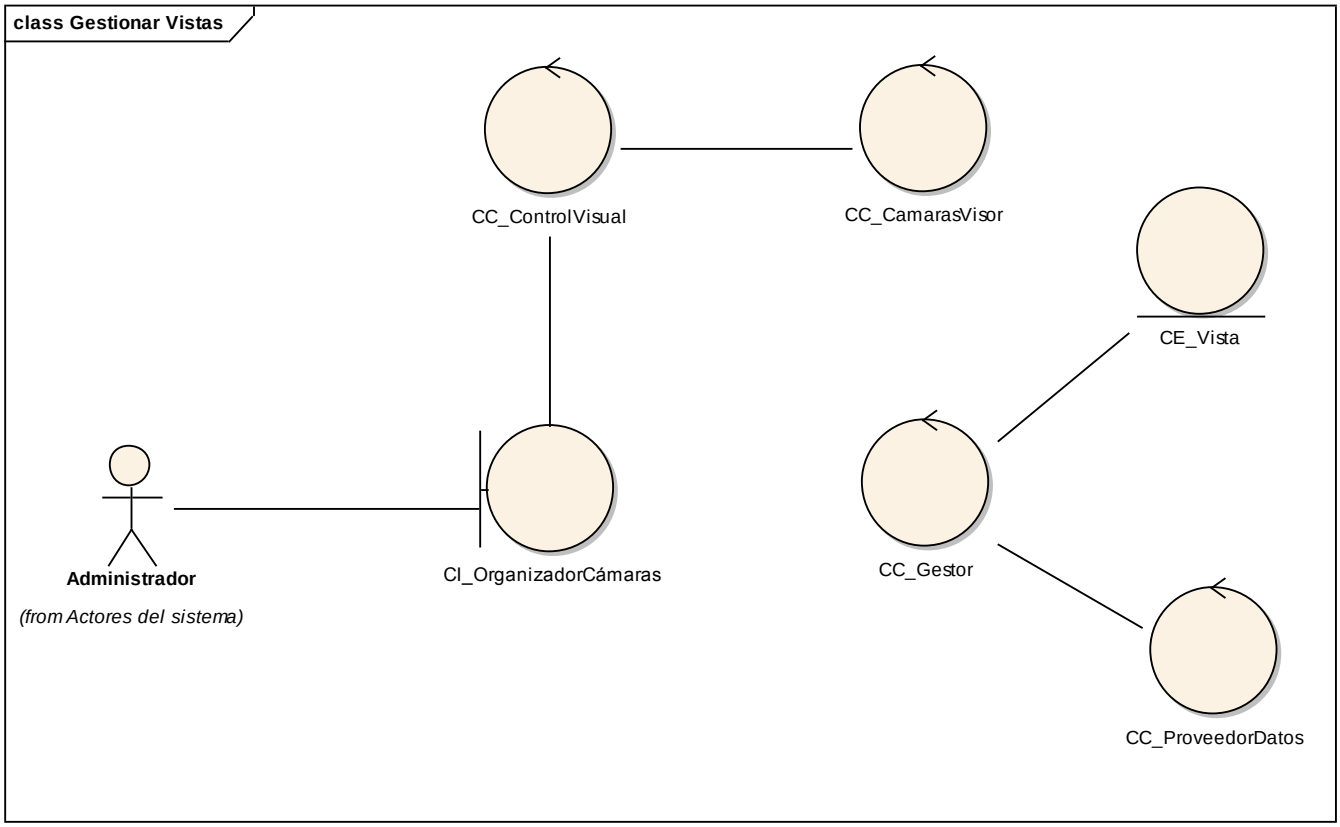

Figura 19 CU Gestionar Vistas

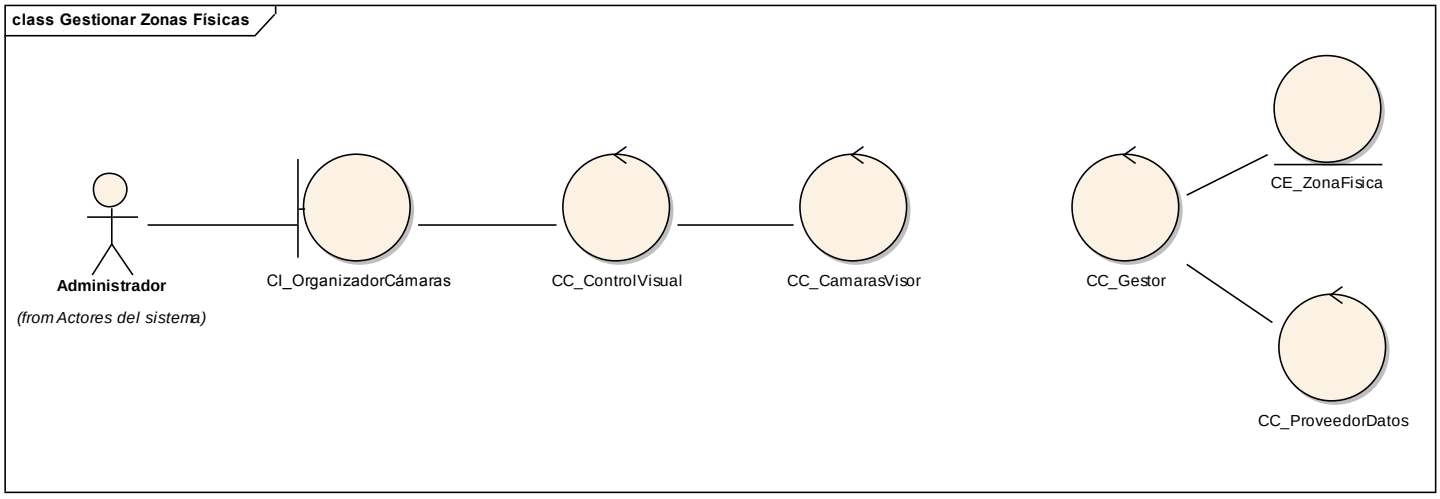

Figura 20 CU Gestionar Zonas Físicas.

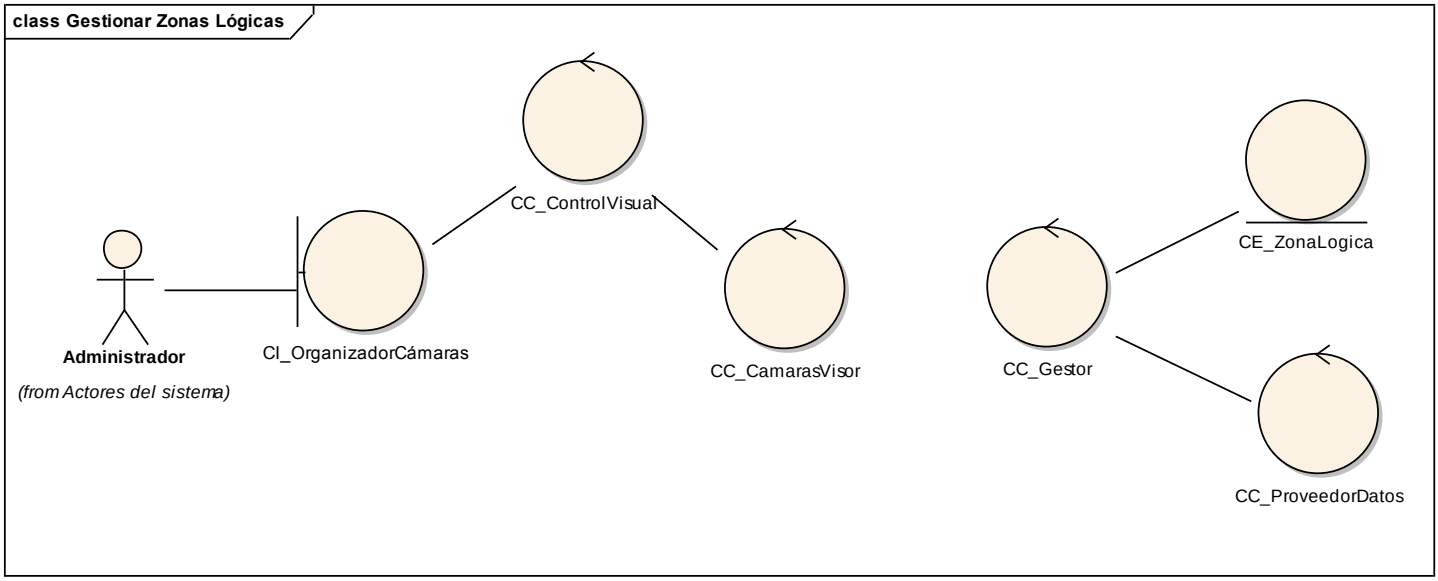

Figura 21 CU Gestionar Zonas Lógicas.

3.3 Modelo de Diseño.

Un Modelo de Diseño es un modelo de objetos que describe la realización física de los casos de uso centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema. Este artefacto constituye la entrada fundamental utilizada para el correcto desarrollo de las entradas de implementación.

A continuación se presentan los diagramas de clases de algunos de los casos de uso fundamentales y un diagrama de secuencia o colaboración para cada uno de ellos y las descripciones de las principales clases involucradas en ellos. Los restantes diagramas por su extensión se encuentran en el Anexo I para los diagramas de Clases, el Anexo II para los diagramas de interacción y el Anexo III para la descripción de las clases.

En algunos de los diagramas no se verán los atributos y métodos de las clases, por el gran volumen de información por lo que se recomienda ver el fichero adjunto al documento SVV.eap, el cual contiene la

Ingeniería realizada en el Enterprise Architect 6.5, en este archivo se encuentra toda esta información detallada.

3.3.1 CU Autenticar usuario.

Diagrama de Clases:

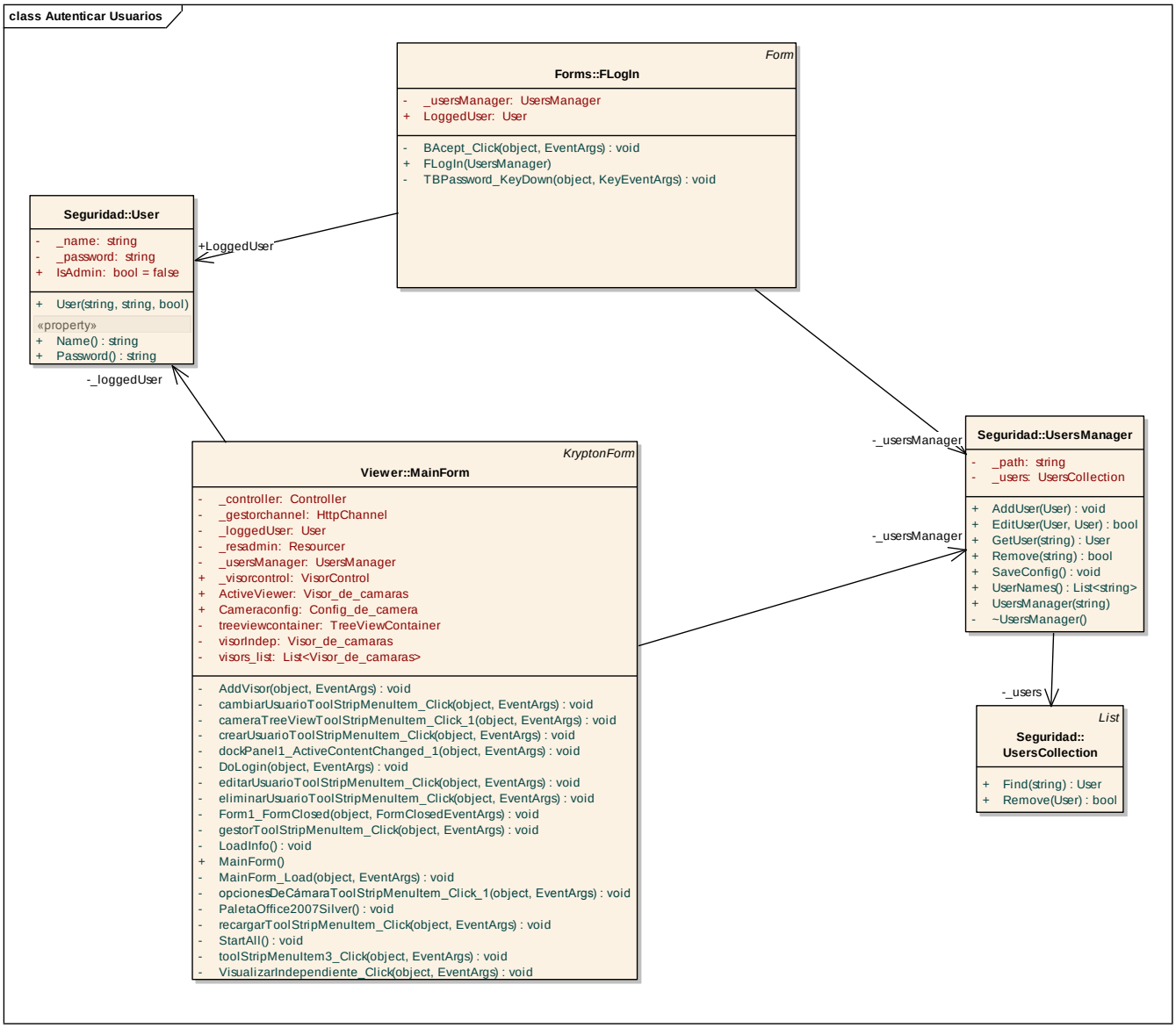

Figura 22 Diagrama de Clases del CU Autenticar usuario

Diagrama de Secuencia

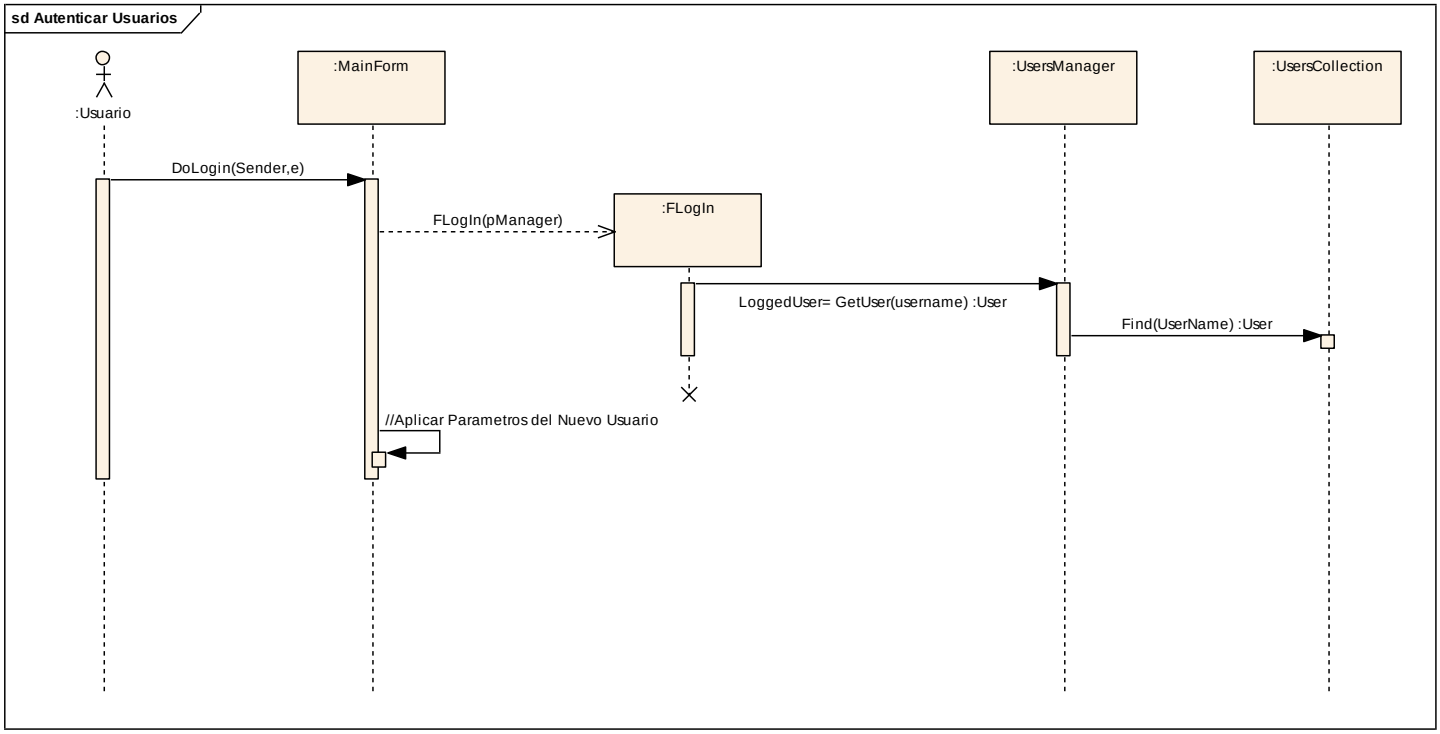

Figura 23 Diagrama de Secuencia del CU Autenticar Usuario

3.3.2 CU Gestionar Cámaras.

Diagrama de Clases:

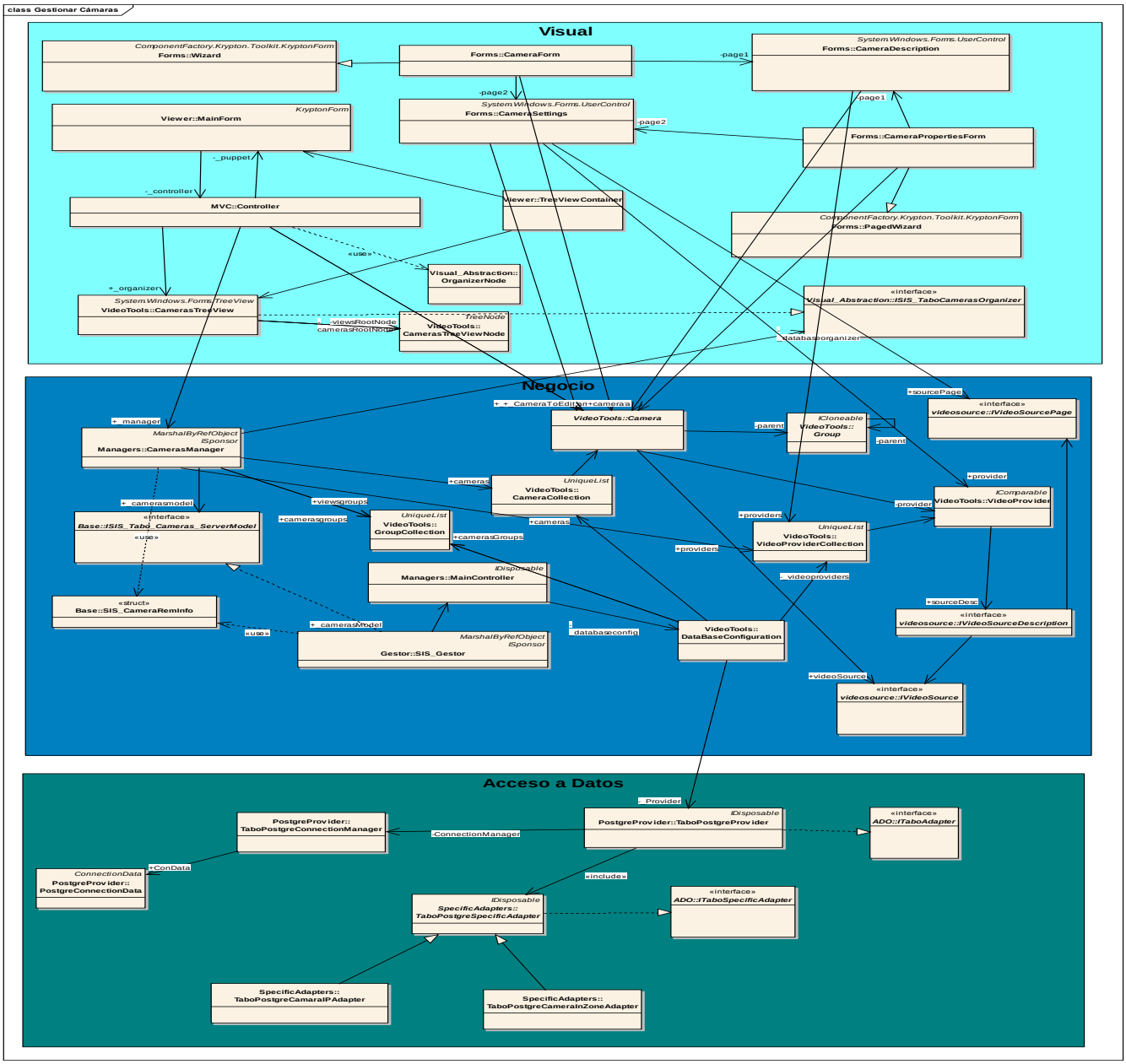

Figura 24 Diagrama de Clases del CU Gestionar Cámaras

Diagrama de Colaboración Escenario Adicionar Cámara:

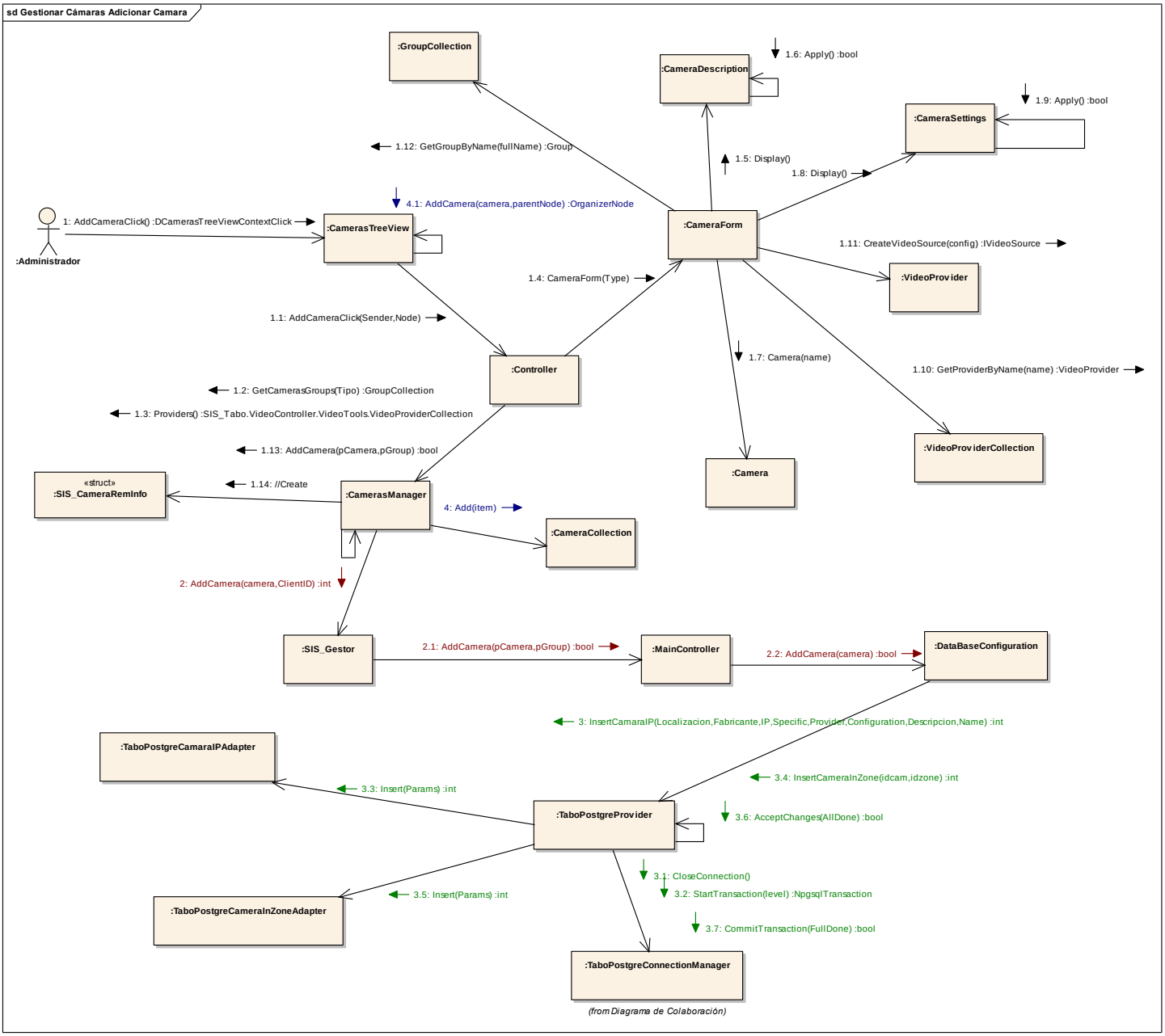

Figura 25 Diagrama de Colaboración Escenario Adicionar Cámara.

3.3.3 CU Gestionar Flujos de Video.

Diagrama de Clases:

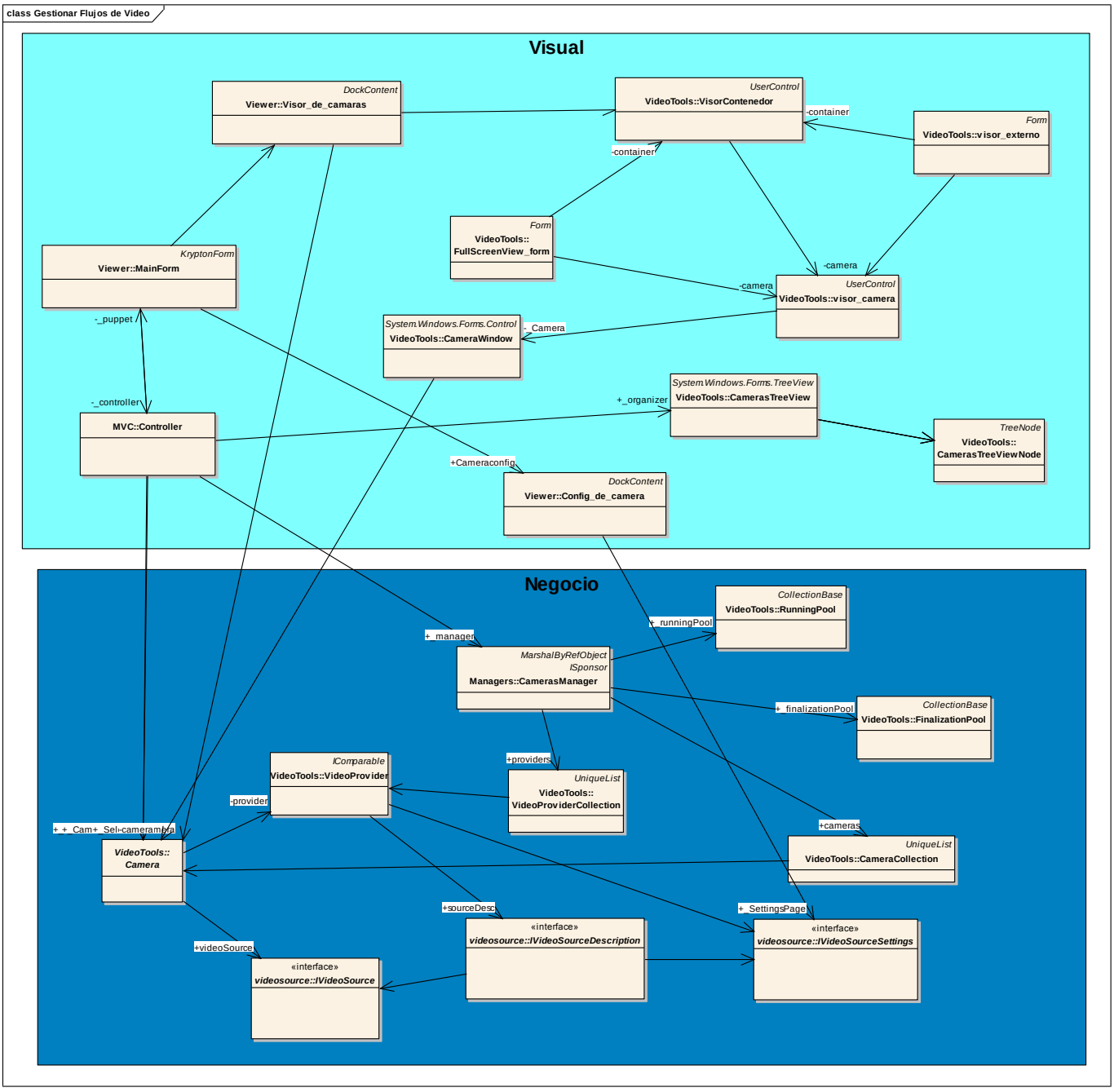

Figura 26 Diagrama de Clases CU Gestionar Flujos de Video

Diagrama de Colaboración del Escenario Obtener Flujo de Video.

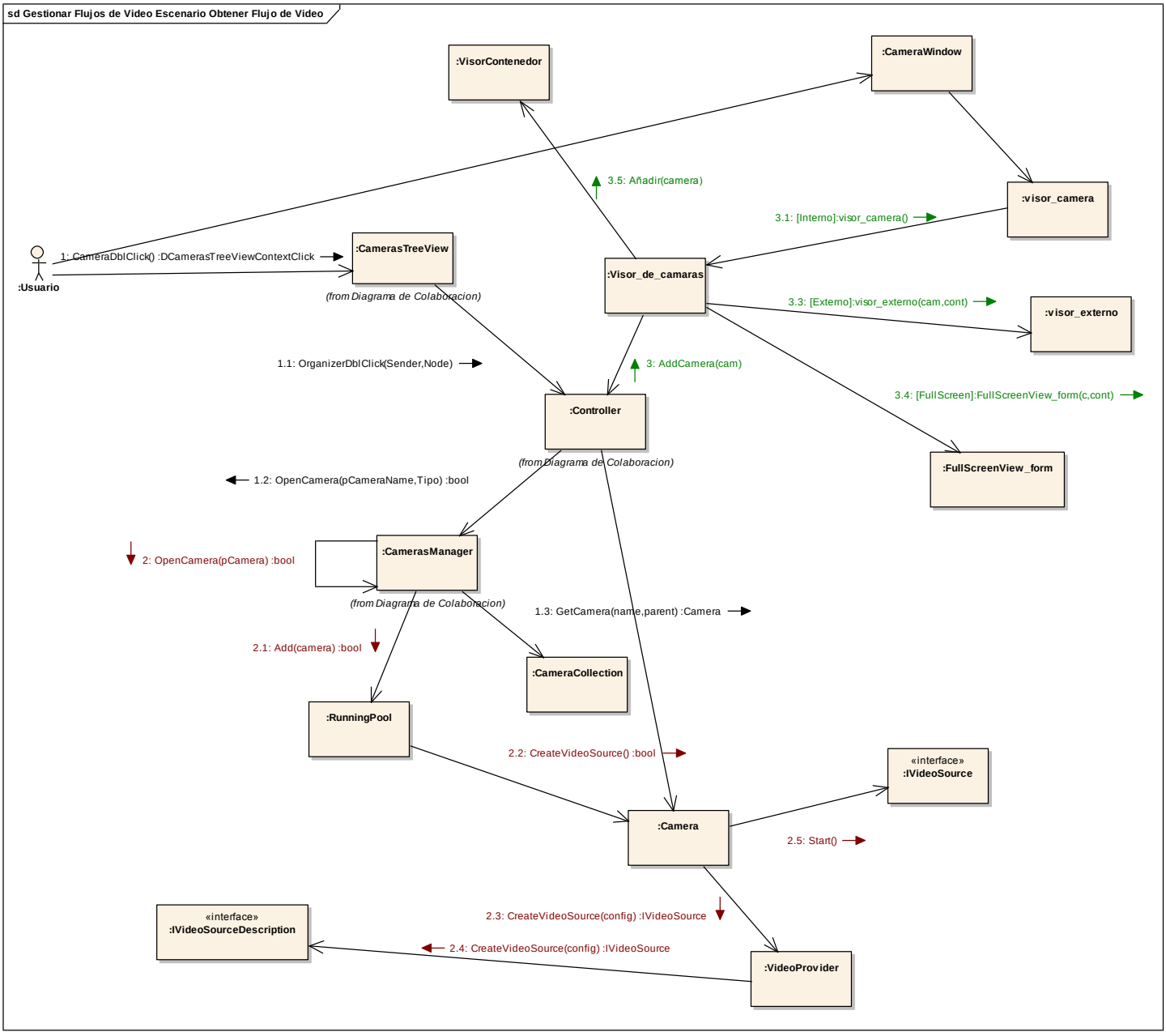

Figura 27 Diagrama de Colaboración del Escenario Obtener Flujo de Video.

3.3.4 CU Gestionar Vistas

Diagrama de Clases:

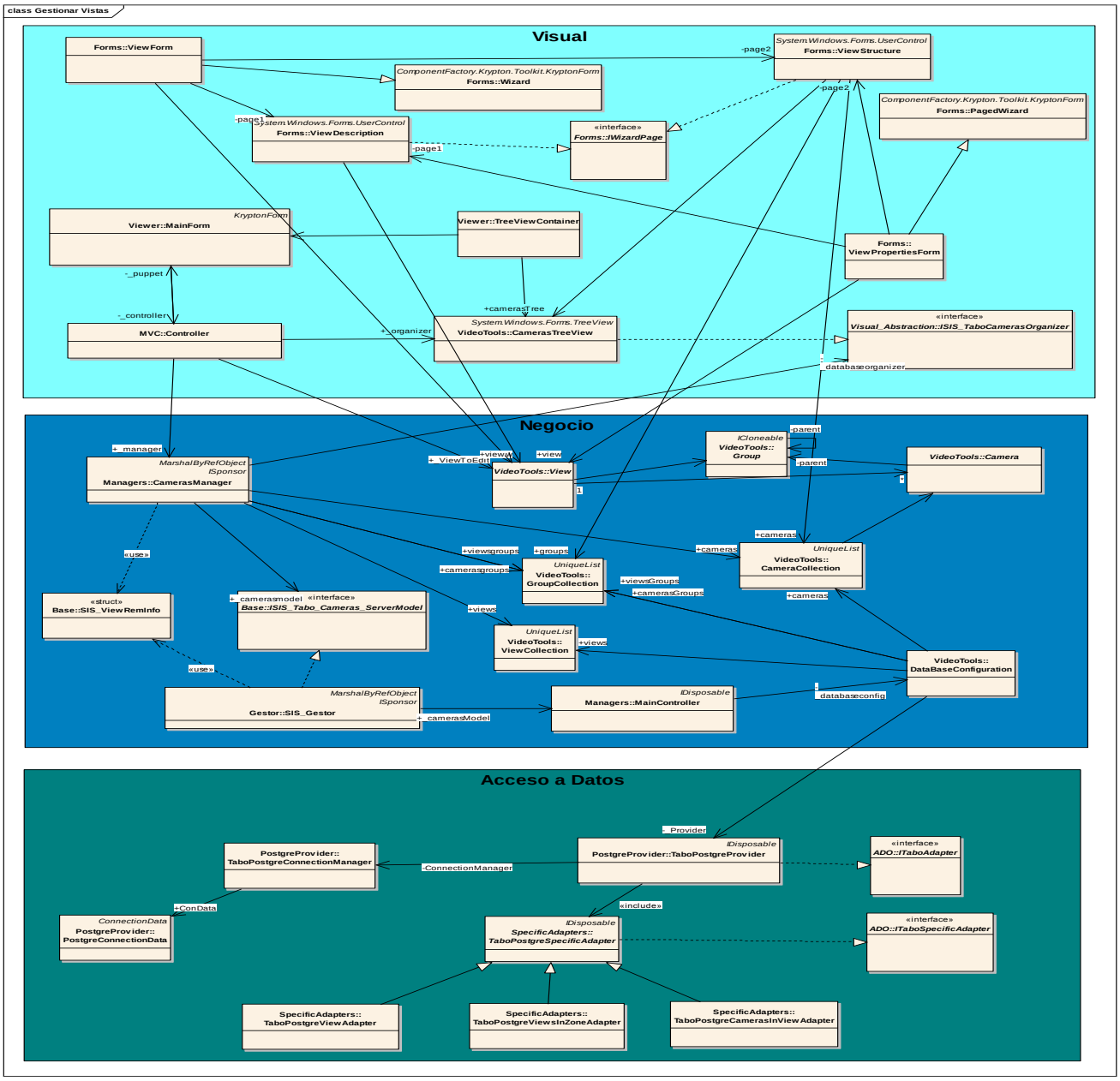

Figura 28 Diagrama de Clases CU Gestionar Vistas.

Diagrama de Colaboración del Escenario Adicionar Vista:

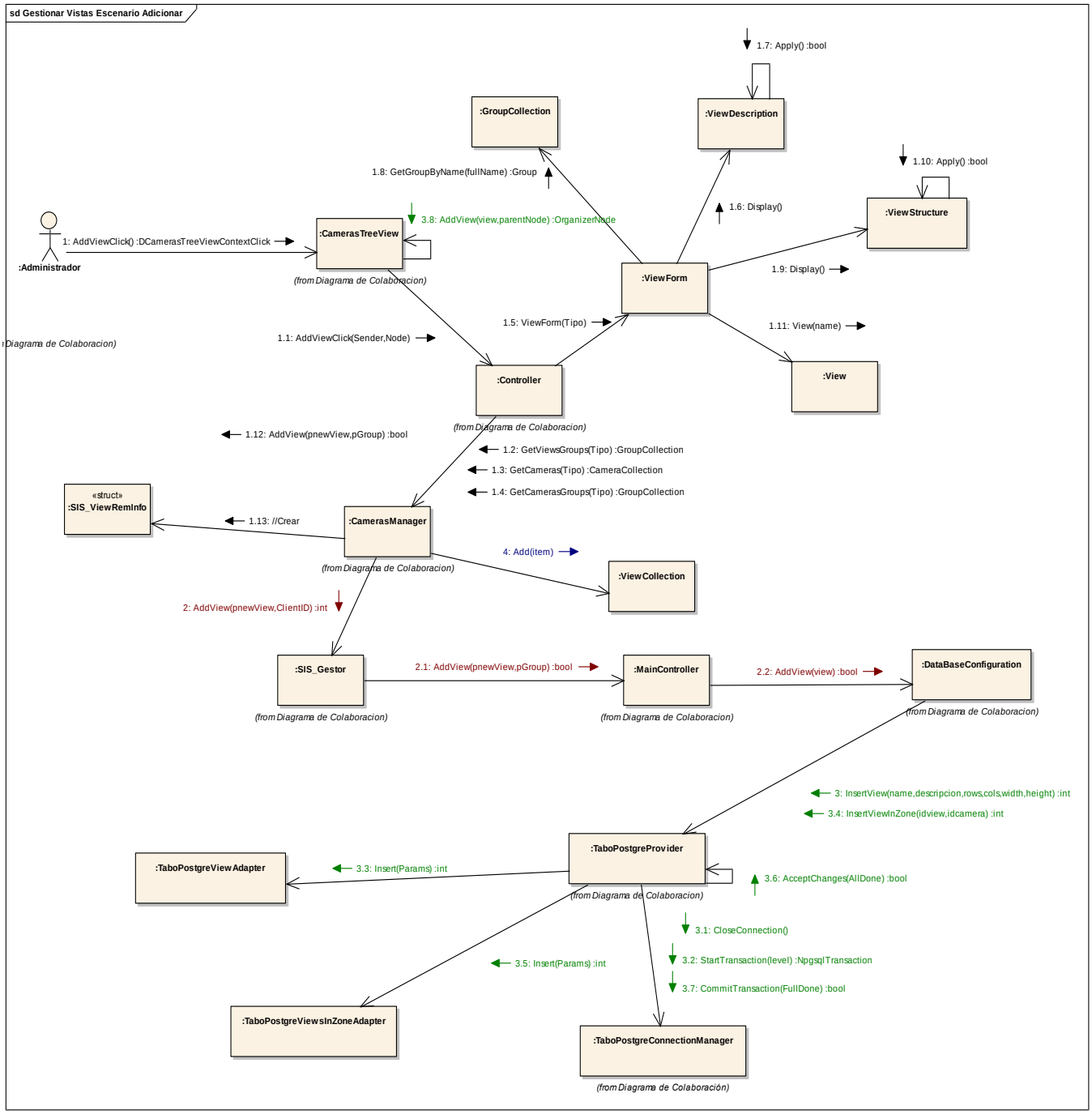

Figura 29 Diagrama de Colaboración del Escenario Adicionar Vista.

3.3.5 CU Gestionar Zonas Físicas.

Diagrama de Clases:

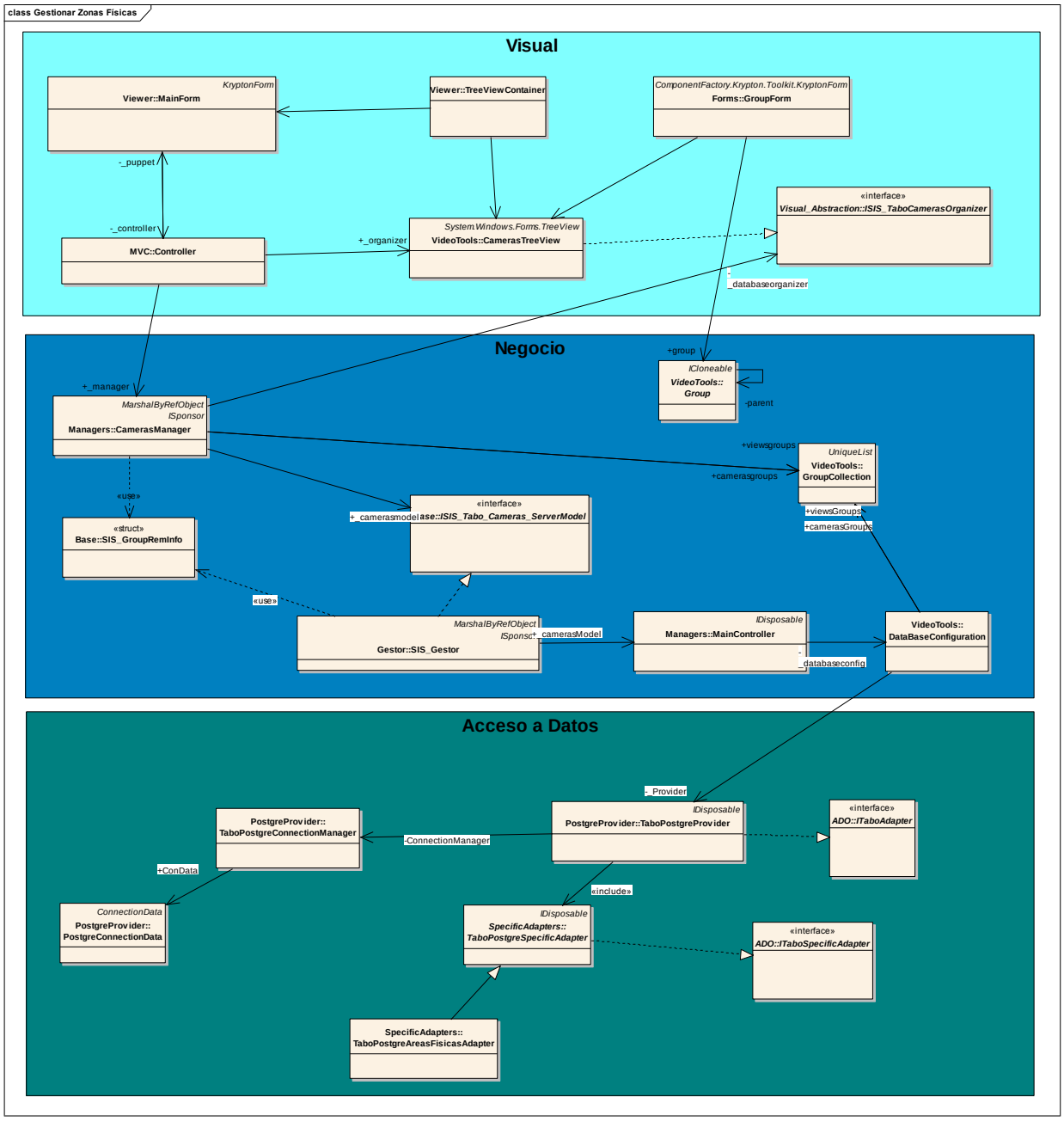

Figura 30 Diagrama de Clases CU Gestionar Zonas Físicas.

Diagrama de Colaboración del Escenario Adicionar Zona Física:

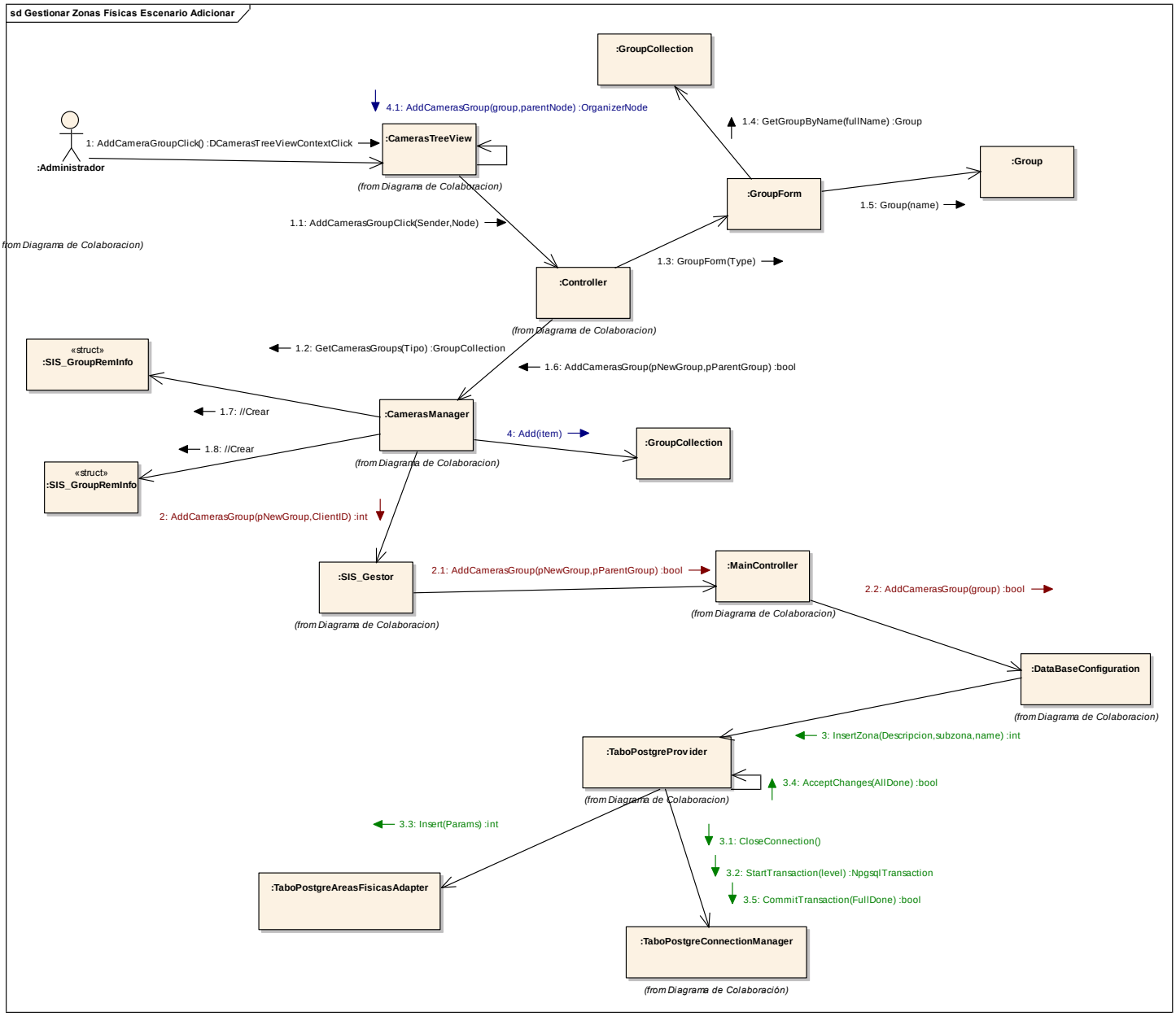

Figura 31 Diagrama de Colaboración del Escenario Adicionar Zona Física.

3.3.6 CU Gestionar Zonas Lógicas.

Diagrama de Clases:

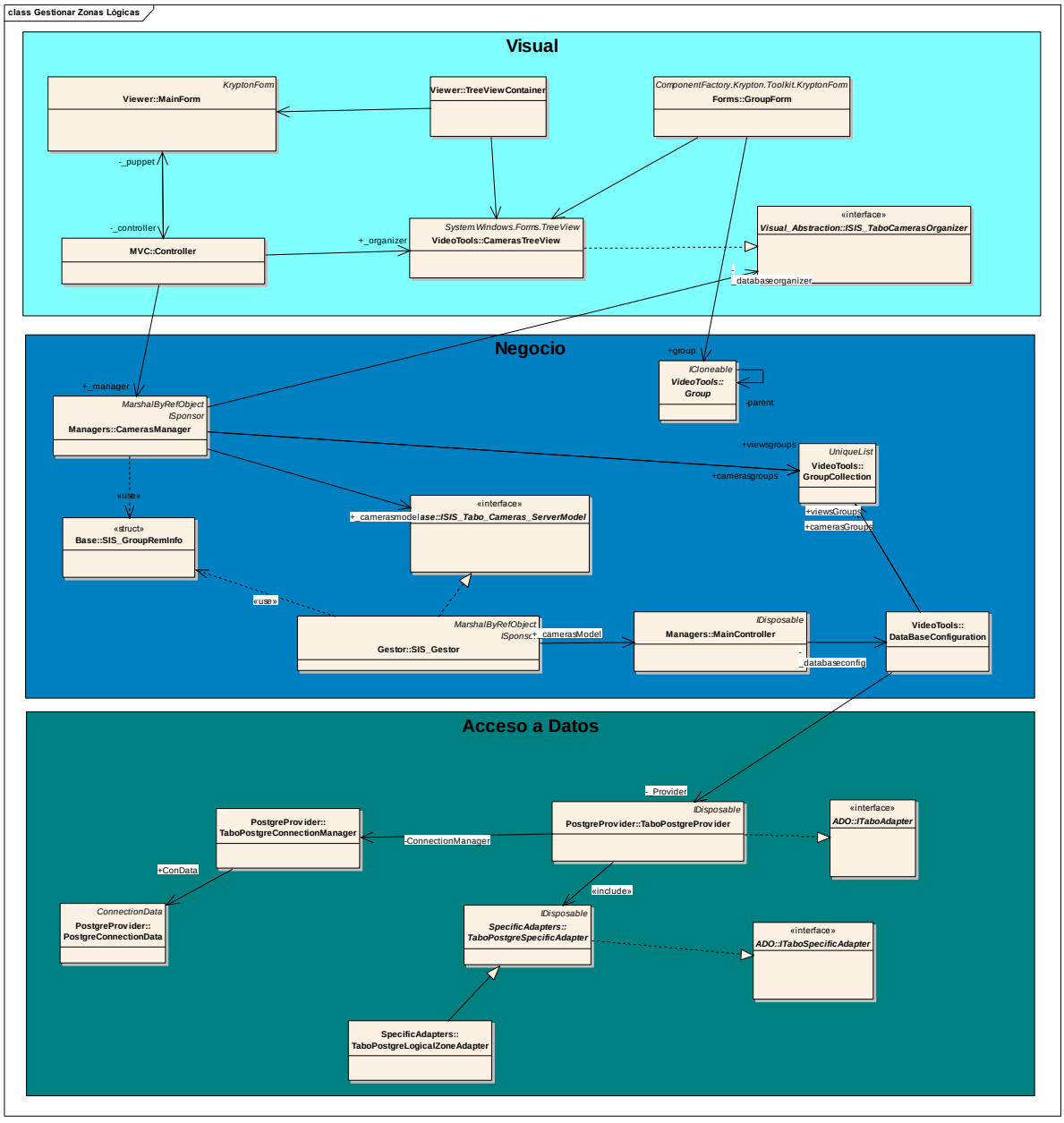

Figura 32 Diagrama de Clases CU Gestionar Zonas Lógicas.

Diagrama de Colaboración del Escenario Adicionar Zona Lógica:

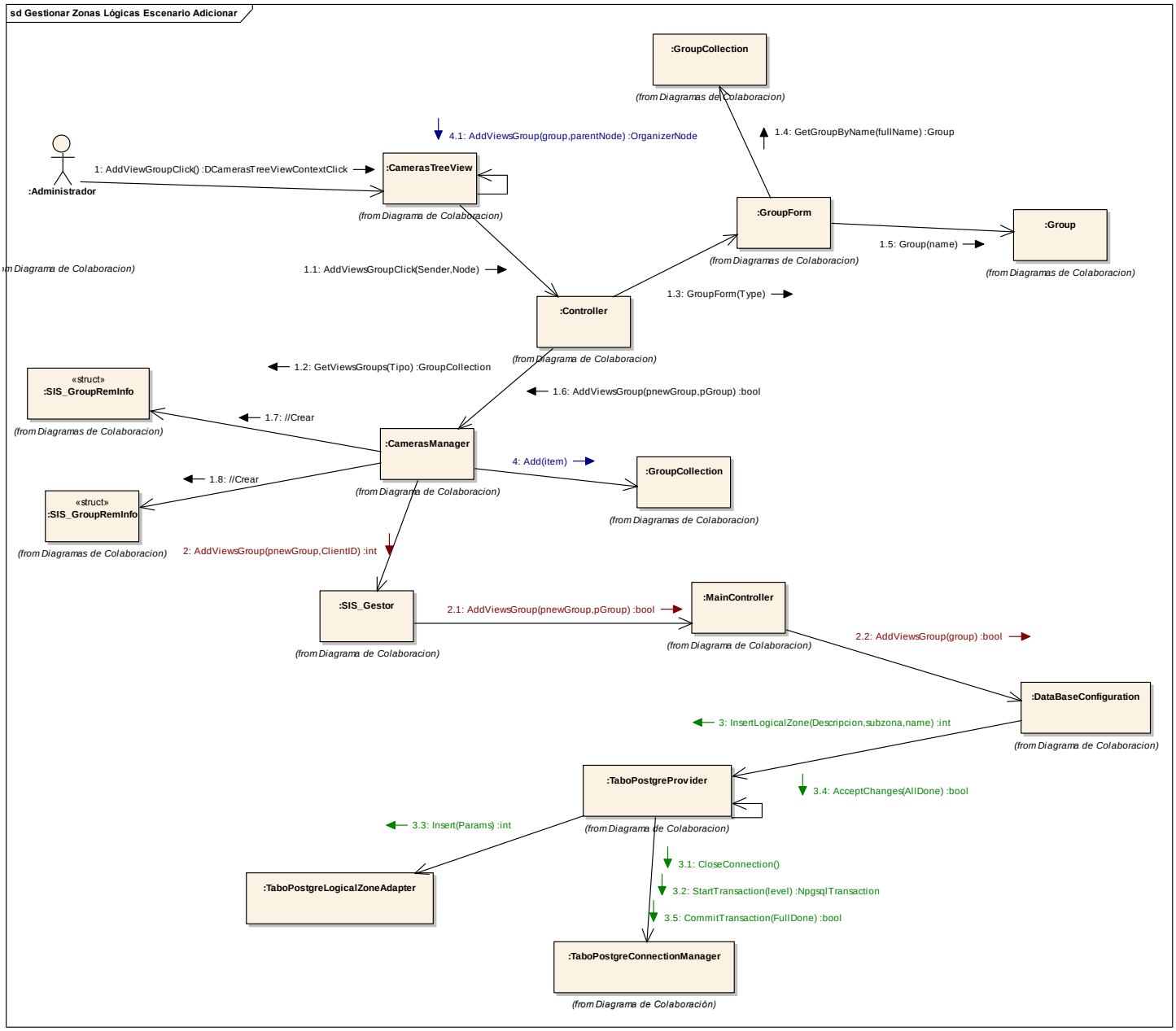

Figura 33 Diagrama de Colaboración del Escenario Adicionar Zona Lógica.

3.3.7 Descripción de las Clases.

Nombre: UsersManager	
Tipo de clase : Controladora	
Atributo	**Tipo**
_path	string
_users	UsersCollection
Para cada responsabilidad:	
Nombre:	UsersManager(string Path)
Descripción:	Constructor que inicializa el manager con los datos requeridos.
Nombre:	AddUser(User usuario)
Descripción:	Adiciona un nuevo usuario a la base de datos.
Nombre:	EditUser(User old, User new)
Descripción:	Edita los datos de un usuario
Nombre:	Remove(string Nombre)
Descripción:	Elimina un usuario de la base de datos.
Nombre:	SaveConfig()
Descripción:	Salva la configuración actual de los usuarios
Nombre:	UserNames()
Descripción:	Devuelve los usuarios que existen en el Sistema

Nombre: Camera	
Tipo de clase : Entidad	
Atributo	**Tipo**
_attendants	int
_Fabricacion	string
_IP	string
_priority	ThreadPriority
_saveBitmap	Bitmap
camerasettings	Object

Configuration	Object
Description	String
FrameRate	int
Height	Int
Width	Int
Id	Int
lastFrame	Bitmap
Name	String
Parent	Group
Provider	VideoProvider
VideoSource	VideoSource

Para cada responsabilidad:	
Nombre:	Camera(string Nombre)
Descripción:	Crea una cámara
Nombre:	CloseVideoSource
Descripción:	Cierra la fuente de video
Nombre:	CopyFrom(Camera)
Descripción:	Copia los datos de una cámara ya creada
Nombre:	CreateVideoSource
Descripción:	Crea una fuente de video para la cámara.
Nombre:	GetVideoSettingPage
Descripción:	Devuelve un control para la configuración de video de la cámara.
Nombre:	Lock
Descripción:	Bloquea la cámara para acceso exclusivo
Nombre	Unlock
Descripción:	Desbloquea el acceso exclusivo a la cámara.
Nombre:	Video_NewFrame(object Sender,CameraEventArgs e)
Descripción:	Avisa a los interesados de que ha llegado un nuevo frame a la cámara.
Nombre:	WaitForStop

Descripción:	Bloquea el hilo de la llamada hasta que la cámara se haya detenido.

Nombre: Controller	
Tipo de clase : Controladora	
Atributo	**Tipo**
_CameraToEdit	Camera
_EditNode	TreeNode
_manager	CamerasManager
_organizer	CamerasTreeView
_puppet	MainForm
_SelectedCamera	Camera
_ViewToEdit	View
Para cada responsabilidad:	
Nombre:	_manager_StatisticsChange(string fps, string bps)
Descripción:	Capturador de los cambios en las estadísticas de la red.
Nombre:	AddCameraClick(CamerasTreeView, OrganizerNode)
Descripción:	Handler para los eventos visuales de adicionar una cámara
Nombre:	AddCamerasGroupClick(CamerasTreeView, OrganizerNode)
Descripción:	Handler para los eventos visuales de adicionar una zona física.
Nombre:	AddViewClick(CamerasTreeView, OrganizerNode)
Descripción:	Handler para los eventos visuales de adicionar una vista.
Nombre:	AddViewsGroupClick(CamerasTreeView, OrganizerNode)
Descripción:	Handler para los eventos visuales de adicionar una zona lógica.
Nombre:	AuthorizeEffect(DragDropEffects, TreeNode, TreeNode, TreeNode, ISIS_TaboCamerasOrganizer)
Descripción:	Handler para capturar los cambios en las cámaras.
Nombre	CameraGroupChanged(Group, bool)
Descripción:	Handler para capturar los cambios en las zonas físicas.
Nombre:	Close

Descripción:	Cierra el controlador liberando todas las conexiones necesarias.
Nombre:	CloseCamera(Camera)
Descripción:	Manda a cerrar una cámara
Nombre:	CloseCameraOnViewer(Visor_de_cámaras, Camera)
Descripción:	Cierra una cámara en un visor manteniéndola abierta en otros visores.
Nombre:	CloseCameras(List<Camera>)
Descripción:	Cierra todas las cámara de la lista
Nombre:	Controller(string, CamerasTreeView)
Descripción:	Crea el controlador para funcionar desconectado del Sistema.
Nombre:	Controller(VisorConfigInfo, CamerasTreeView)
Descripción:	Crea el controlador para funcionar conectado al Sistema
Nombre:	EditCameraClick(CamerasTreeView,OrganizerNode)
Descripción:	Handler para los eventos visuales de edición de una cámara.
Nombre:	EditCamerasGroupClick(CamerasTreeView,OrganizerNode)
Descripción:	Handler para los eventos visuales de edición de una zona fisica.
Nombre	EditViewClick(CamerasTreeView, OrganizerNode)
Descripción:	Handler para los eventos visuales de edición de una vista.
Nombre:	EditViewsGroupClick(CamerasTreeView, OrganizerNode)
Descripción:	Handler para los eventos visuales de edición de una zona lógica.
Nombre:	FillOrganizer(ISIS_TaboCamerasOrganizer)
Descripción:	Llena de datos el organizador de la cámara
Nombre:	OrganizerDblClick(CamerasTreeView, OrganizerNode)
Descripción:	Handler para capturar los doble clicks en el Organizador.
Nombre:	RemoveCameraClick(CamerasTreeView, OrganizerNode)
Descripción:	Handler para capturar los eventos de eliminar una cámara.
Nombre:	RemoveCamerasGroupClick(CamerasTreeView, OrganizerNode)
Descripción:	Handler para capturar los eventos de eliminar una zona física.
Nombre:	RemoveViewClick(CamerasTreeView, OrganizerNode)
Descripción:	Handler para capturar los eventos de eliminar una vista.

Nombre:	RemoveViewsGroupClick(CamerasTreeView, OrganizerNode)
Descripción:	Handler para capturar los eventos de eliminar una zona lógica.
Nombre:	SelectedCameraChange(Visor_de_cámaras, Camera)
Descripción:	Handler para capturar cuando cambia la cámara activa.
Nombre	ViewChanged(View, bool)
Descripción:	Handler para capturar cuando ha cambiado una vista en el Sistema.
Nombre:	ViewGroupChanged(Group, bool)
Descripción:	Handler para capturar cuando ha cambiado una zona lógica en el Sistema.

Nombre: Gestor	
Tipo de clase : Controladora	
Atributo	**Tipo**
_camerasModel	MainController
_clients	Hashtable
_dbAdapter	TaboPostgreProvider
_OnCameraChanged	DCameraGroupChanged
_OnViewChanged	DViewChanged
_OnViewGroupChanged	DViewGroupChanged
_ViewToEdit	View
Para cada responsabilidad:	
Nombre:	AddCamera(SIS_CameraRemInfo, int)
Descripción:	Adicionar una cámara al Sistema
Nombre:	AddCamerasGroup(SIS_GroupRemInfo, int)
Descripción:	Adicionar una zona física al Sistema.
Nombre:	AddView(SIS_ViewRemInfo, int)
Descripción:	Adicionar una vista al Sistema.
Nombre:	AddViewsGroup(SIS_GroupRemInfo, int)
Descripción:	Adicionar una zona lógica al Sistema.
Nombre:	CameraChangedAsyncCallBackHandler(IAsyncResult)

Descripción:	Handler para capturar los cambios en las cámaras.
Nombre	CameraGroupChangedAsyncCallBackHandler(IAsyncResult)
Descripción:	Handler para capturar los cambios en las zonas físicas.
Nombre:	CheckCamera(SIS_CameraRemInfo, int)
Descripción:	Chequear si la cámara existe en el Sistema.
Nombre:	CheckCamerasGroup(SIS_GroupRemInfo, int)
Descripción:	Chequear si la zona física existe en el Sistema.
Nombre:	CheckClientID(int)
Descripción:	Chequear si es válido el ID presentado por un módulo cliente.
Nombre:	CheckView(SIS_ViewRemInfo, int)
Descripción:	Chequear si la vista existe en el Sistema.
Nombre:	CheckViewsGroup(SIS_GroupRemInfo, int)
Descripción:	Chequear si la zona lógica existe en el Sistema.
Nombre:	DeleteCamera(SIS_CameraRemInfo, int)
Descripción:	Eliminar una cámara del Sistema
Nombre:	DeleteCamerasGroup(SIS_GroupRemInfo, int)
Descripción:	Eliminar una zona física del Sistema.
Nombre:	DeleteView(SIS_ViewRemInfo, int)
Descripción:	Eliminar una vista del Sistema.
Nombre	DeleteViewsGroup(SIS_GroupRemInfo, int)
Descripción:	Eliminar una zona lógica del Sistema.
Nombre:	GetCameraFromID(int, int)
Descripción:	Encontrar una cámara a partir de su ID.
Nombre:	GetCameraFromName(string, int)
Descripción:	Encontrar una cámara a partir de su nombre.
Nombre:	GetCameraGroupFromID(int, int)
Descripción:	Encontrar una zona física a partir de su ID
Nombre:	GetCameraGroupFromName(string, int)
Descripción:	Encontrar una zona física a partir de su nombre

Nombre:	GetCameras(int)
Descripción:	Devuelve todas las cámaras del Sistema.
Nombre:	GetCamerasGroups(int)
Descripción:	Devuelve todas las zonas físicas en el Sistema.
Nombre:	GetClientID(ISIS_ServerClient)
Descripción:	Genera un ID valido para el cliente.
Nombre:	GetViewFromID(int, int)
Descripción:	Encontrar una vista a partir de su ID.
Nombre	GetViewFromName(string, int)
Descripción:	Encontrar una vista a partir de su nombre.
Nombre:	GetViewGroupFromID(int, int)
Descripción:	Encontrar una zona lógica a partir de su ID.
Nombre:	GetViewGroupFromName(string, int)
Descripción:	Encontrar una zona lógica a partir de su nombre.
Nombre:	GetViews(int)
Descripción:	Devuelve todas las vistas del sistema.
Nombre:	GetViewsGroups(int)
Descripción:	Devuelve todas las zonas lógicas del Sistema.
Nombre:	SendCameraChangedMessage(DCameraChanged, int, SIS_CameraRemInfo, bool, string)
Descripción:	Enviar a cada módulo conectado el aviso de que una cámara ha cambiado.
Nombre:	SendCameraGroupChangedMessage(DCameraGroupChanged, int, SIS_GroupRemInfo, bool, string)
Descripción:	Enviar a cada módulo conectado el aviso de que una zona física ha cambiado.
Nombre:	SendViewChangedMessage(DViewChanged, int, SIS_ViewRemInfo, bool, string)
Descripción:	Enviar a cada módulo conectado el aviso de que una vista ha cambiado.
Nombre	SendViewGroupChangedMessage(DViewGroupChanged, int, SIS_GroupRemInfo, bool, string)
Descripción:	Enviar a cada módulo conectado el aviso de que una zona lógica ha cambiado.

Nombre:	SIS_Gestor
Descripción:	Crea una nueva instancia del Gestor
Nombre:	UpdateCamera(SIS_CameraRemInfo, SIS_CameraRemInfo, int)
Descripción:	Actualizar una cámara
Nombre:	UpdateCamerasGroup(SIS_GroupRemInfo, SIS_GroupRemInfo, int)
Descripción:	Actualizar una zona física.
Nombre:	UpdateView(SIS_ViewRemInfo, SIS_ViewRemInfo, int)
Descripción:	Actualizar una vista.
Nombre:	UpdateViewsGroup(SIS_GroupRemInfo, SIS_GroupRemInfo, int)
Descripción:	Actualizar una zona lógica.
Nombre:	ViewChangedAsyncCallBackHandler(IAsyncResult)
Descripción:	Handler para capturar los cambios en las vistas del Sistema.
Nombre:	ViewGroupChangedAsyncCallBackHandler(IAsyncResult)
Descripción:	Handler para capturar los cambios en las zonas lógicas del Sistema.

Nombre: CamerasManager	
Tipo de clase : Controladora	
Atributo	**Tipo**
_camerasModel	ISIS_Tabo_Cameras_ServerModel
_finalizationPool	FinalizationPool
_GestorEventsReceiver	_GestorEventsReceiver
_id	int
_modelchannel	IChannel
_port	int
_runningPool	RunningPool
_statCount	int
_statIndex	int
_statLength	int
_statReady	int

_statReceived	long
cameras	CameraCollection
camerasgroups	GroupCollection
providers	VideoProviderCollection
ServerMode	bool
views	ViewCollection
viewsgroups	GroupCollection

Para cada responsabilidad:	
Nombre:	AddCamera(Camera, Group)
Descripción:	Adicionar una cámara al Sistema
Nombre:	AddCamerasGroup(Group, Group)
Descripción:	Adicionar una zona física al Sistema.
Nombre:	AddView(View, Group)
Descripción:	Adicionar una vista al Sistema.
Nombre:	AddViewsGroup(Group, Group)
Descripción:	Adicionar una zona lógica al Sistema.
Nombre:	CameraChanged (SIS_CameraRemInfo, bool, int)
Descripción:	Handler para capturar los cambios en las cámaras.
Nombre	CameraGroupChanged (SIS_GroupRemInfo, bool, int)
Descripción:	Handler para capturar los cambios en las zonas físicas.
Nombre	CamerasManager(string, bool)
Descripción:	Crea una nueva instancia del manejador.
Nombre:	CheckCamera(Camera)
Descripción:	Chequear si la cámara existe en el Sistema.
Nombre:	CheckCamerasGroup(Group)
Descripción:	Chequear si la zona física existe en el Sistema.
Nombre:	CheckView(View)
Descripción:	Chequear si la vista existe en el Sistema.
Nombre:	CheckViewsGroup(Group)

Descripción:	Chequear si la zona lógica existe en el Sistema.
Nombre:	CloseAllOpenedCameras
Descripción:	Cierra todas las cámaras abiertas.
Nombre:	CloseCamera(Camera)
Descripción:	Cerrar una cámara.
Nombre:	CloseConnection
Descripción:	Cierra la conexión con el Gestor.
Nombre:	DeleteCamera(Camera)
Descripción:	Eliminar una cámara del Sistema
Nombre:	DeleteCamerasGroup(DeleteCamerasGroup)
Descripción:	Eliminar una zona física del Sistema.
Nombre:	DeleteView(View)
Descripción:	Eliminar una vista del Sistema.
Nombre	DeleteViewsGroup(Group)
Descripción:	Eliminar una zona lógica del Sistema.
Nombre:	GetCameraFromID(int, CameraType)
Descripción:	Encontrar una cámara a partir de su ID.
Nombre:	GetCameraFromName(string, CameraType)
Descripción:	Encontrar una cámara a partir de su nombre.
Nombre:	GetCameraGroupFromID(int, CameraType)
Descripción:	Encontrar una zona física a partir de su ID
Nombre:	GetCameraGroupFromName(string, CameraType)
Descripción:	Encontrar una zona física a partir de su nombre
Nombre:	GetCameras(CameraType)
Descripción:	Devuelve todas las cámaras del Sistema.
Nombre:	GetCamerasGroups(CameraType)
Descripción:	Devuelve todas las zonas físicas en el Sistema.
Nombre:	GetViewFromID(int, CameraType)
Descripción:	Encontrar una vista a partir de su ID.

Nombre	GetViewFromName(string, CameraType)
Descripción:	Encontrar una vista a partir de su nombre.
Nombre:	GetViewGroupFromID(int, CameraType)
Descripción:	Encontrar una zona lógica a partir de su ID.
Nombre:	GetViewGroupFromName(string, CameraType)
Descripción:	Encontrar una zona lógica a partir de su nombre.
Nombre:	GetViews(CameraType)
Descripción:	Devuelve todas las vistas del sistema.
Nombre:	GetViewsGroups(CameraType)
Descripción:	Devuelve todas las zonas lógicas del Sistema.
Nombre:	LoadAllData(string, string, int, RemotingUtils.ChannelProtocol)
Descripción:	Cargar todos los datos desde el Gestor.
Nombre:	LoadCameras
Descripción:	Cargar todas las cámaras del Sistema
Nombre:	LoadCamerasGroups
Descripción:	Cargar todas las zonas físicas del Sistema.
Nombre	LoadSettings(string)
Descripción:	Cargar la configuración
Nombre:	LoadViews
Descripción:	Carga todas las vistas en el Sistema
Nombre:	LoadViewsGroups
Descripción:	Cargar todas las zonas físicas en el Sistema
Nombre:	OpenCamera(Camera)
Descripción:	Abrir una cámara y adicionarla a la piscina de cámaras activas.
Nombre:	OpenCamera(string, CameraType)
Descripción:	Abrir una cámara según su nombre
Nombre:	OpenPort(int, bool)
Descripción:	Abrir un puerto para la comunicación con el Gestor.
Nombre:	OpenPort(bool)

Descripción:	Abrir un puerto para la comunicación con el Gestor.
Nombre:	OpenView(View)
Descripción:	Abrir una vista
Nombre	OpenView(string, CameraType)
Descripción:	Abrir una vista
Nombre:	RegisterWithServer
Descripción:	Registrarse con el Gestor como Cliente.
Nombre:	SaveCameras(CameraType)
Descripción:	Salva todas las cámaras del sistema.
Nombre:	SaveViews(CameraType)
Descripción:	Salva todas las vistas.
Nombre:	UpdateCamera(Camera, Camera)
Descripción:	Actualizar una cámara
Nombre:	UpdateCamerasGroup(Group, Group)
Descripción:	Actualizar una zona física.
Nombre:	UpdateView(View, View)
Descripción:	Actualizar una vista.
Nombre:	UpdateViewsGroup(Group, Group)
Descripción:	Actualizar una zona lógica.
Nombre:	ViewChanged (SIS_ViewRemInfo, bool, int)
Descripción:	Handler para capturar los cambios en las vistas del Sistema.
Nombre:	ViewGroupChanged (SIS_GroupRemInfo, bool, int)
Descripción:	Handler para capturar los cambios en las zonas lógicas del Sistema.

Nombre: Group	
Tipo de clase : Entidad	
Atributo	**Tipo**
description	string
Id	int

name	string
parent	Group
Para cada responsabilidad:	
Nombre:	Clone()
Descripción:	Devuelve un clon del grupo en una nueva instancia.
Nombre:	CopyFrom(Group)
Descripción:	Copia los datos desde un grupo ya creado.
Nombre:	Description()
Descripción:	Devuelve una descripción del grupo.
Nombre:	FullName()
Descripción:	Devuelve el nombre completo del grupo.
Nombre:	Group(string)
Descripción:	Crea una nueva instancia.
Nombre:	ID
Descripción:	Devuelve el ID del grupo
Nombre:	Name
Descripción:	Devuelve el nombre del grupo
Nombre:	Parent
Descripción:	Devuelve el padre del grupo.

Nombre: View	
Tipo de clase: Entidad	
Atributo	**Tipo**
cameraIDs	int
cellHeight	short
cellWidth	short
cols	short
description	short
id	int

MaxCols	int
MaxRows	int
name	string
parent	UniqueList<Group>
rows	short

Para cada responsabilidad:	
Nombre:	AddParent(Group)
Descripción:	Adiciona la vista a una zona.
Nombre:	CopyFrom()
Descripción:	Copia los datos de una vista ya creada.
Nombre:	DataChange()
Descripción:	Avisa de que han cambiado los datos en la vista.
Nombre:	GetCamera(int, int)
Descripción:	Devuelve una cámara determinada de la vista.
Nombre:	IsInGroup(Group)
Descripción:	Devuelve si la vista esta en una zona determinada.
Nombre:	RemoveParent(Group)
Descripción:	Saca la vista de una zona.
Nombre:	SetCamera(int, int, int)
Descripción:	Cambia la cámara que se encuentra en una posición determinada.
Nombre:	View(string)
Descripción:	Crea una nueva instancia de la vista.

Conclusiones

Durante este capítulo, se mostró la arquitectura escogida para la realización de la aplicación, se desarrolló el modelo de análisis y diseño del sistema. Primeramente se elaboraron los diagramas de clases del análisis en donde se exponen los conceptos básicos del sistema, sus partes y relaciones. Luego los diagramas de clases del diseño y los diagramas de interacción por cada uno de los casos de usos, y por último las descripciones de las clases del diseño.

CAPÍTULO 4: IMPLEMENTACIÓN DEL SISTEMA.

En la implementación, se parte de los resultados obtenidos en el diseño y se implementa el sistema en términos de componentes, como son los ficheros de código fuente, ficheros de código binario, ejecutables, entre otros. Primeramente, se detalla el modelo de datos del sistema, que es el modelo donde se ve la estructura en la cual se almacenan toda la información requerida en el sistema. Luego se muestra el modelo de implementación que está compuesto por el diagrama de despliegue y el diagrama de componentes. Estos diagramas, describen los componentes a construir, su organización y dependencias entre los nodos físicos en la que funcionará la aplicación.

4.1 Modelo de datos.

El modelo de datos es el modelo utilizado para el diseño de la base de datos.

4.1.1 Descripción de las tablas.

Nombre: tbl_camara		
Descripción: Representa una cámara en el Sistema.		
Atributo	**Tipo**	**Descripción**
id_camara	integer	ID de la cámara.
tipo_camara	varchar(20)	Tipo de la cámara.
fabricacion	varchar(50)	Fabricante
name_cam	varchar(80)	Nombre de la cámara.
ip	varchar(255)	IP
provider	varchar(100)	Plugin asociado
configprovider	varchar(1000)	Configuración del plugin.
descripcion	varchar(255)	Descripción de la cámara.

Nombre: tbl_view		
Descripción:		
Atributo	**Tipo**	**Descripción**
id_view	integer	ID de la vista

name	varchar(50)	Nombre
descripcion	varchar(200)	Descripción
rows	integer	Cantidad de filas que componen la vista
cols	integer	Cantidad de columnas de la vista
width	integer	Ancho de las celdas
height	integer	Alto de las celdas

Nombre: tbl_ zona		
Descripción: Representa una zona física de la institución en la que se despliega el Sistema.		
Atributo	**Tipo**	**Descripción**
id_zona	integer	ID de la zona física
descripcion	varchar(255)	Descripción
sub_zona	integer	Zona a la que pertenece
shortname	varchar(50)	Nombre de la zona

Nombre: tbl_logic_group		
Descripción: Representa una zona lógica utilizada para agrupar vistas.		
Atributo	**Tipo**	**Descripción**
id_zona	integer	ID de la zona lógica
descripcion	varchar(255)	Descripción
sub_zona	integer	Zona lógica a la que pertenece.
shortname	varchar(50)	Nombre

4.2 Modelo de implementación.

4.2.1 Diagrama de despliegue.

El diagrama de despliegue que se muestra se utiliza para capturar los elementos de configuración del procesamiento y las conexiones entre esos elementos. Se utiliza además para visualizar la distribución de los componentes de software en los nodos físicos.

La aplicación desarrollada aquí tiene 4 elementos fundamentales: PC Visor, Cámara, PC Gestor y el Servidor de Base de Datos.

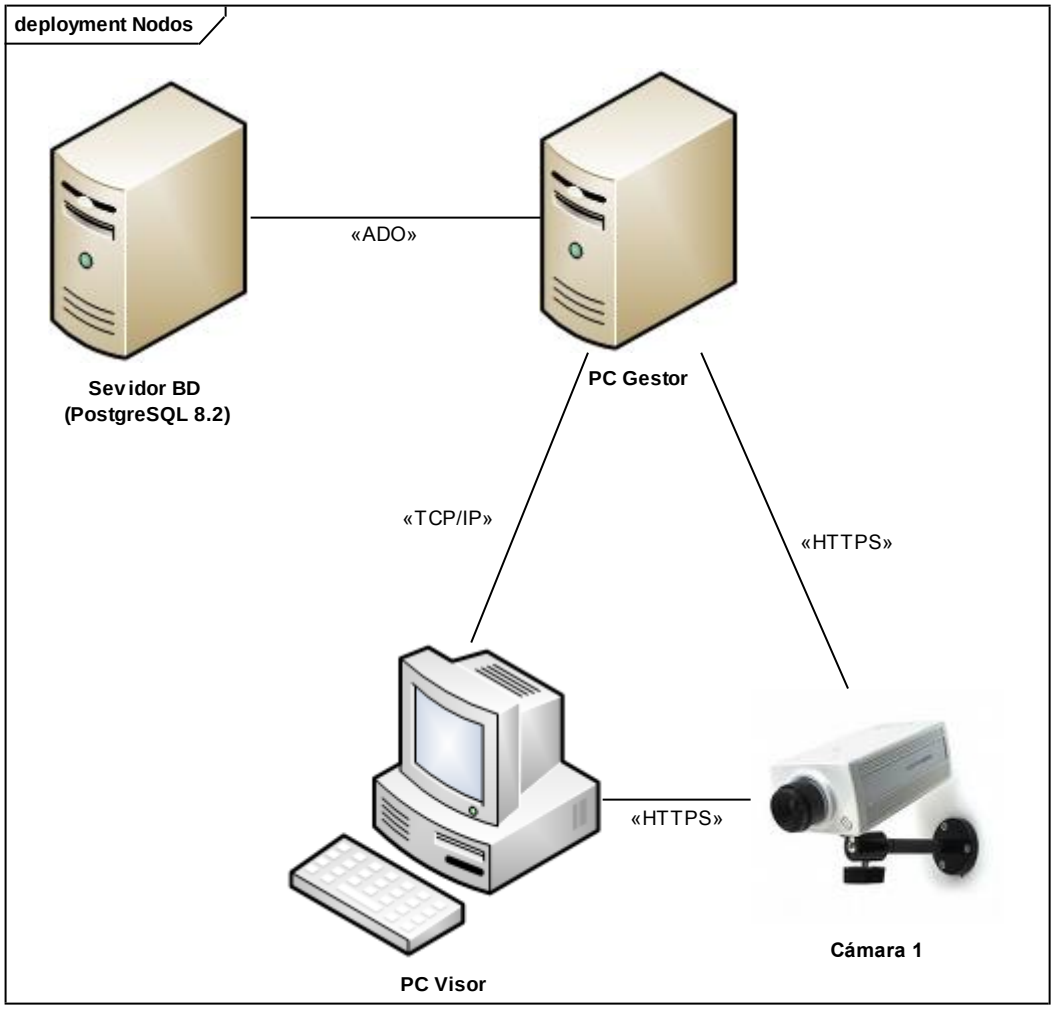

Figura 34 Diagrama de Despliegue del Sistema.

4.2.2 Diagrama de Componentes.

Un diagrama de componentes muestra las organizaciones y dependencias lógicas entre componentes de software, sean estos componentes de código fuente, binarios o ejecutables. Desde el punto de vista del diagrama de componentes se tienen en consideración los requisitos relacionados con la facilidad de desarrollo, la gestión de software, la reutilización, y las restricciones impuestas por los lenguajes de programación y las herramientas utilizadas en el desarrollo.

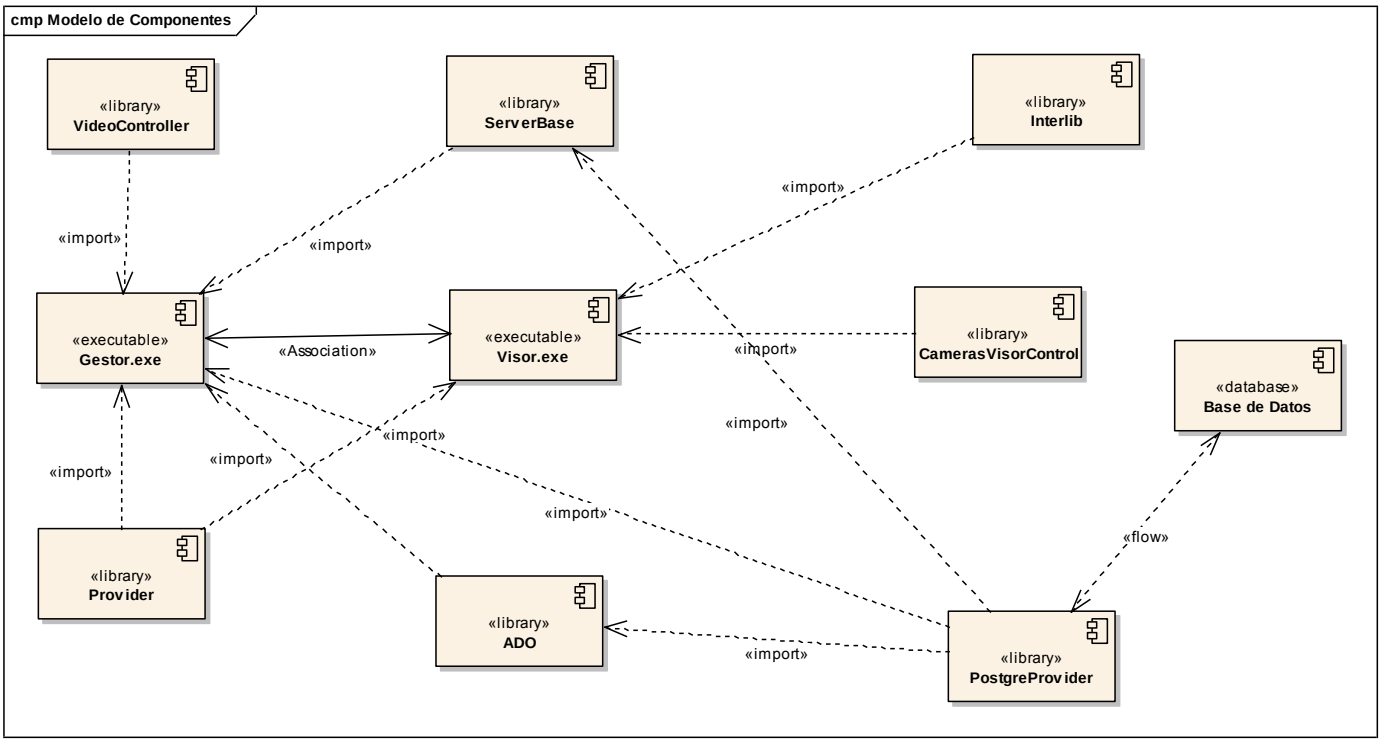

Figura 35 Diagrama de Componentes del Sistema.

Conclusiones.

En este capítulo se da una visión general de como quedó finalmente implementada la aplicación. Se mostró la misma en términos de componentes, su ubicación y utilización a la hora de la implementación final. Se obtuvo como resultado que todos los componentes se implementaron cumpliendo con esta etapa de desarrollo.

CONCLUSIONES

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

- ✓ Se realizó un análisis sobre los principales sistemas de video vigilancia existente en la actualidad. Las principales limitaciones estuvieron en que estos software tienen en su funcionamiento gran dependencia del fabricante.
- ✓ Se valoraron las principales tecnologías y herramientas actuales lo que permitió seleccionar las adecuadas para el desarrollo del sistema permitiendo que el producto final tenga la calidad requerida.
- ✓ Se implementó un sistema que brinda los servicios de visualización de todas las áreas en la que estén conectadas las cámaras, así como la obtención de flujos de videos en tiempo real. Además de gestionar toda la configuración referentes a las cámaras ip, cumpliendo con las funcionalidades trazadas.
- ✓ Se logró que el sistema esté preparado para evolucionar hacia nuevas funcionalidades y actualizar las ya existentes.

Es importante destacar la capacidad del sistema de soportar variados hardware de diversos fabricantes, que es una de las delimitaciones actuales de la mayoría de los sistemas en el mercado. Además todos los datos que usa o genera se acogen a los estándares internacionales que rigen estos sistemas, tanto para la compresión de imágenes, como para la trasmisión de las mismas.

RECOMENDACIONES

Durante el desarrollo del trabajo han surgido ideas que podrían implementarse en un futuro, de forma que se logren desarrollar más funcionalidades para la aplicación, para lo cual se recomienda:

- La implementación de un módulo de grabación.
- La implementación de un módulo de detección de movimientos.
- El despliegue piloto del sistema en alguna institución para hacer un seguimiento del rendimiento y enriquecer el Sistema con las peticiones de los clientes.

REFERENCIA BIBLIOGRÁFICA

1. **Axis Communications.** Axis Communications. *Axis Communications.* [En línea] 21 de 11 de 2008. [Citado el: 21 de 11 de 2008.] http://www.axis.com.

2. Videovigilancia. *TELENORMA Solucines en comunicación.* [En línea] [Citado el: 30 de November de 2008.] http://www.telenorma.com.co/index.php?option=com_content&view=section&layout=blog&id=3&Itemid=7.

3. **Herranz, Arantxa.** Video Vigilancia IP. *PCWORLD PROFESIONAL.* [En línea] 24 de July de 2008. [Citado el: 23 de November de 2008.] http://www.idg.es/pcworldtech/Video_vigilancia_IP:_un_Gran_Hermano_en_nuestra_re/art191729-comunicaciones.htm.

4. **Axis Communications.** *Coste total de propiedad(TCO) Comparativa entre los sistemas de vigilancia con cámaras IP y cámaras analógicas.* s.l. : Axis Communications, 2008.

5. **MPEG LA®.** MPEG-4 Visual. *MPEG LA.* [En línea] MPEG LA, 2004. [Citado el: 20 de Noviembre de 2008.] http://www.mpegla.com/m4v/m4v-licensees.cfm.

6. **Panasonic.** Construnario.com - Notiweb: Sistemas de Seguridad Panasonic: 50 años haciendo la vida más fácil. *Construnario >> Notiweb.* [En línea] 25 de April de 2007. [Citado el: 25 de November de 2008.] http://www.construnario.es/notiweb/titulares_resultado.asp?regi=15902.

7. Scati Labs - Wikipedia, la enciclopedia libre. *Wikipedia, la enciclopedia libre.* [En línea] [Citado el: 25 de November de 2008.] http://es.wikipedia.org/wiki/Scati_Labs.

8. Sony: Vigilancia inteligente en Verbania: España. *Sony: Soluciones profesionales, corporativas y de broadcast: España.* [En línea] [Citado el: 25 de November de 2008.] http://www.sony.es/biz/view/ShowContent.action?site=biz_es_ES&contentId=1189437949251&sectiontype=NVM+CaseStudies.

9. Las 10 ventajas principales de Microsoft Office Visio 2007. *Microsoft Office Online.* [En línea] [Citado el: 25 de November de 2008.] http://office.microsoft.com/es-es/visio/HA101650313082.aspx.

10. PostgreSQL8.3 ya disponible. *Portal de Astra NTi.* [En línea] [Citado el: 25 de November de 2008.] http://www.astra.es/noticias/postgresql-8-3-ya-disponible.

11. Visual SourceSafe 2005. *Visual Studio.* [En línea] [Citado el: 25 de November de 2008.] http://msdn.microsoft.com/es-es/vcsharp/aa718670.aspx.

12. Microsoft Visual Studio - Wikipedia, la enciclopedia libre. *Wikipedia, la enciclopedia libre.* [En línea] [Citado el: 25 de November de 2008.] http://es.wikipedia.org/wiki/Microsoft_Visual_Studio#Visual_Studio_2008.

13. Información general de .NET Framework Remoting. *Visual Studio.* [En línea] [Citado el: 25 de November de 2008.] http://msdn.microsoft.com/es-es/library/kwdt6w2k(VS.80).aspx .

14. **Wikipedia.** Programación Extrema. *Wikipedia La Enciclopedia Libre.* [En línea] 22 de Noviembre de 2008. [Citado el: 29 de Noviembre de 2008.] http://es.wikipedia.org/wiki/Programaci%C3%B3n_Extrema.

15. —. Proceso Unificado de Rational. *Wikipedia.* [En línea] 19 de Noviembre de 2008. [Citado el: 29 de Noviembre de 2008.] http://es.wikipedia.org/wiki/RUP.

BIBLIOGRAFÍA

1. Antonio Albiol, Valery Naranjo, and Josep Prades. *Tratamiento digital de la señal. Teoría y aplicaciones.* SPUPV-99.4162. Servicio de publicaciones de la Universidad politécnica de Valencia, 1999.
2. Antonio Albiol, Cristina Sandoval, and Alberto Albiol. Robust motion detector for video surveillande applications. Technical report, Departamento de comunicaciones, Universidad Politécnica de Valencia, 2003.
3. B.G. Batchelor and P.F. Whelan. Machine visión systems: proverbs, principles, prejudices and priorities. In *Machine Vision Applications, Architectures, and Systemtems Integration III*,. Number 2347 in Proceeding of the SIPE, pages 374-383, Boston, 1994. The International Society for optical Engineering.
4. Robert Collins and Alan Lipton. A system for video surveillance and monitoring. Technical Report CMU-RI-TR-00-12, Robotics Institute, Carnegie mellon University, Pittsburgh, PA, May 2000.
5. E. Duracan and T.Ebrahimi. Change detection and background extraction by linear algebra. *Proceedings of the IEEE*, 89(10): 1382-1402, October 2001.
6. G.L. Foresti, Lucio Marcenaro, and C.S. Regazzoni. Automatic detection and indexing of video-event shots for surveillanceapplications. In I*EEE transactions on multimedia, vol 4*, number 4, 2002.
7. Pedro García, Encarna Segarra, Tomás Pérez, José M. Sempere, José Ruiz, and M. Vázquez de Parga. *Apuntes sobre la teoría de autómatas y lenguajes formales.* SPUPV-96.846. Servicio de publicaciones de la Universidad Politécnica de Valencia 1996.
8. John E. Hopcroft and Jeffrey D. ullman. *Introduction to automata theory languages, and computation*. Compañía editorial continental, 1993.

9. Dieter Koller. Moving object recognition and classification based on recursive shape parameter estimation. In *12th Israeli conference on artificial intelligence, computer vision and neuralnetworks*, pages 359-368, Tel-Aviv, 1996.

10. Takashi komuro, Idaku Ishii, and Masatoshi ishikawa. General-purpose vision chip architecture for real-time machine vision. In *Advanced robotics 12.,* pages 619'627, 1999.

11. Akio Namiki, Yoshihiro Nakabo, idaku Ishii, and Masatoshi ishikawa. High speed grasping using visual and force feedback. *In Int Conf. on Robotics and Automation*., proceedings of the IEEE, pages 3195-3200, Detroit, 1999.

12. J. H. Piater and J.L. Crowley. Multi-modal tracking of interacting trgets using Gaussian approximations. In *Workshop on PETS*, Proceedings 2nd IEEE, Hawaii, 2001.

13. Julio Pons, Josep prades-nebot, Antonio Albiol, and Jesus Molina. Fast motion detection in the compressed domain for video surveillance. *IEEE Electronics letters*, 38(9):409-411, April 2002.

14. C.S. Regazzoni, V.Ramesh, and G.L. Foresti. Special issue on video communications, processing and understanding for third generation surveillance systems. In *Proceedings IEEE*, vol 89, number 10, 2001.

15. C. Ridder, O. Munkelt, and H. Kirchner. Adaptive background estimation and foreground detection using kalman filtering, 1995.

16. Kentaro Toyama, John Krumm, Barry Brummit, and Brian Meyers. Wallflowers: Principles and practice of background maintenance. Technical report, Microsoft Research. Redmond., 1999.

17. Videovigilancia. *TELENORMA Solucines en comunicación.* [En línea] [Citado el: 30 de November de 2008.]

18. Axis Communications. Axis Communications. *Axis Communications.* [En línea] 21 de 11 de 2008. [Citado el: 21 de 11 de 2008.] http://www.axis.com.

19. 3. Herranz, Arantxa. Video Vigilancia IP. *PCWORLD PROFESIONAL.* [En línea] 24 de July de 2008. [Citado el: 23 de November de 2008.] http://www.idg.es/pcworldtech/Video_vigilancia_IP:_un_Gran_Hermano_en_nuestra_re/art191729-comunicaciones.htm.
20. 4. MPEG LA®. MPEG-4 Visual. *MPEG LA.* [En línea] MPEG LA, 2004. [Citado el: 20 de Noviembre de 2008.] http://www.mpegla.com/m4v/m4v-licensees.cfm.
21. 5. Panasonic. Construnario.com - Notiweb: Sistemas de Seguridad Panasonic: 50 años haciendo la vida más fácil. *Construnario >> Notiweb.* [En línea] 25 de April de 2007. [Citado el: 25 de November de 2008.] http://www.construnario.es/notiweb/titulares_resultado.asp?regi=15902.
22. 6. Scati Labs - Wikipedia, la enciclopedia libre. *Wikipedia, la enciclopedia libre.* [En línea] [Citado el: 25 de November de 2008.] http://es.wikipedia.org/wiki/Scati_Labs.
23. 7. Sony: Vigilancia inteligente en Verbania: España. *Sony: Soluciones profesionales, corporativas y de broadcast: España.* [En línea] [Citado el: 25 de November de 2008.] http://www.sony.es/biz/view/ShowContent.action?site=biz_es_ES&contentId=1189437949251&sectiontype=NVM+CaseStudies.
24. 8. Las 10 ventajas principales de Microsoft Office Visio 2007. *Microsoft Office Online.* [En línea] [Citado el: 25 de November de 2008.] http://office.microsoft.com/es-es/visio/HA101650313082.aspx.
25. 9. PostgreSQL8.3 ya disponible. *Portal de Astra NTi.* [En línea] [Citado el: 25 de November de 2008.] http://www.astra.es/noticias/postgresql-8-3-ya-disponible.
26. 10. Visual SourceSafe 2005. *Visual Studio.* [En línea] [Citado el: 25 de November de 2008.] http://msdn.microsoft.com/es-es/vcsharp/aa718670.aspx.
27. Videovigilancia. *TELENORMA Solucines en comunicación.* [En línea] [Citado el: 30 de November de 2008.] http://www.telenorma.com.co/index.php?option=com_content&view=section&layout=blog&id=3&Itemid=7.
28. Axis Communications. Axis Communications. *Axis Communications.* [En línea] 21 de 11 de 2008. [Citado el: 21 de 11 de 2008.] http://www.axis.com.
29. Herranz, Arantxa. Video Vigilancia IP. *PCWORLD PROFESIONAL.* [En línea] 24 de July de 2008. [Citado el: 23 de November de 2008.]

http://www.idg.es/pcworldtech/Video_vigilancia_IP:_un_Gran_Hermano_en_nuestra_re/art191729-comunicaciones.htm.

30. MPEG LA®. MPEG-4 Visual. *MPEG LA.* [En línea] MPEG LA, 2004. [Citado el: 20 de Noviembre de 2008.] http://www.mpegla.com/m4v/m4v-licensees.cfm.
31. Panasonic. Construnario.com - Notiweb: Sistemas de Seguridad Panasonic: 50 años haciendo la vida más fácil. *Construnario >> Notiweb.* [En línea] 25 de April de 2007. [Citado el: 25 de November de 2008.] http://www.construnario.es/notiweb/titulares_resultado.asp?regi=15902.
32. Scati Labs - Wikipedia, la enciclopedia libre. *Wikipedia, la enciclopedia libre.* [En línea] [Citado el: 25 de November de 2008.] http://es.wikipedia.org/wiki/Scati_Labs.

    Sony: Vigilancia inteligente en Verbania: España. *Sony: Soluciones profesionales, corporativas y de broadcast: España.* [En línea] [Citado el: 25 de November de 2008.] http://www.sony.es/biz/view/ShowContent.action?site=biz_es_ES&contentId=1189437949251&sectiontype=NVM+CaseStudies.
33. Las 10 ventajas principales de Microsoft Office Visio 2007. *Microsoft Office Online.* [En línea] [Citado el: 25 de November de 2008.] http://office.microsoft.com/es-es/visio/HA101650313082.aspx.
34. PostgreSQL8.3 ya disponible. *Portal de Astra NTi.* [En línea] [Citado el: 25 de November de 2008.] http://www.astra.es/noticias/postgresql-8-3-ya-disponible.
35. Visual SourceSafe 2005. *Visual Studio.* [En línea] [Citado el: 25 de November de 2008.] http://msdn.microsoft.com/es-es/vcsharp/aa718670.aspx.
36. Microsoft Visual Studio - Wikipedia, la enciclopedia libre. *Wikipedia, la enciclopedia libre.* [En línea] [Citado el: 25 de November de 2008.] http://es.wikipedia.org/wiki/Microsoft_Visual_Studio#Visual_Studio_2008.
37. Información general de .NET Framework Remoting. *Visual Studio.* [En línea] [Citado el: 25 de November de 2008.] http://msdn.microsoft.com/es-es/library/kwdt6w2k(VS.80).aspx .
38. Wikipedia. Programación Extrema. *Wikipedia La Enciclopedia Libre.* [En línea] 22 de Noviembre de 2008. [Citado el: 29 de Noviembre de 2008.] http://es.wikipedia.org/wiki/Programaci%C3%B3n_Extrema.
39. Axis Communications. *Coste total de propiedad(TCO) Comparativa entre los sistemas de vigilancia con cámaras IP y cámaras analógicas.* s.l. : Axis Communications, 2008.

40. Wikipedia. Proceso Unificado de Rational. *Wikipedia.* [En línea] 19 de Noviembre de 2008. [Citado el: 29 de Noviembre de 2008.] http://es.wikipedia.org/wiki/RUP.

ANEXOS

Anexo I Diagramas de Clases

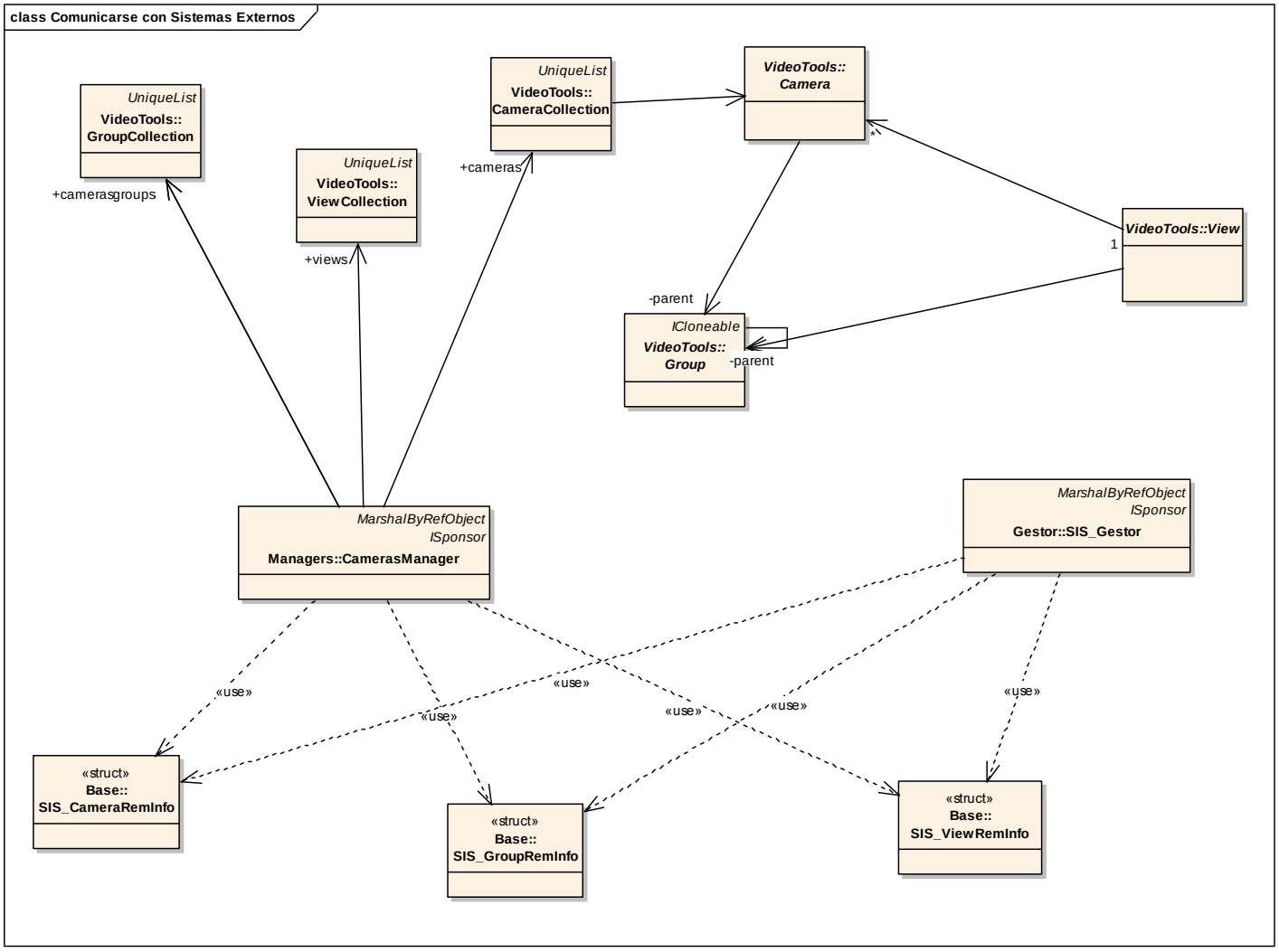

Figura 36 Diagrama de Clases del CU Comunicarse con Sistemas Externos

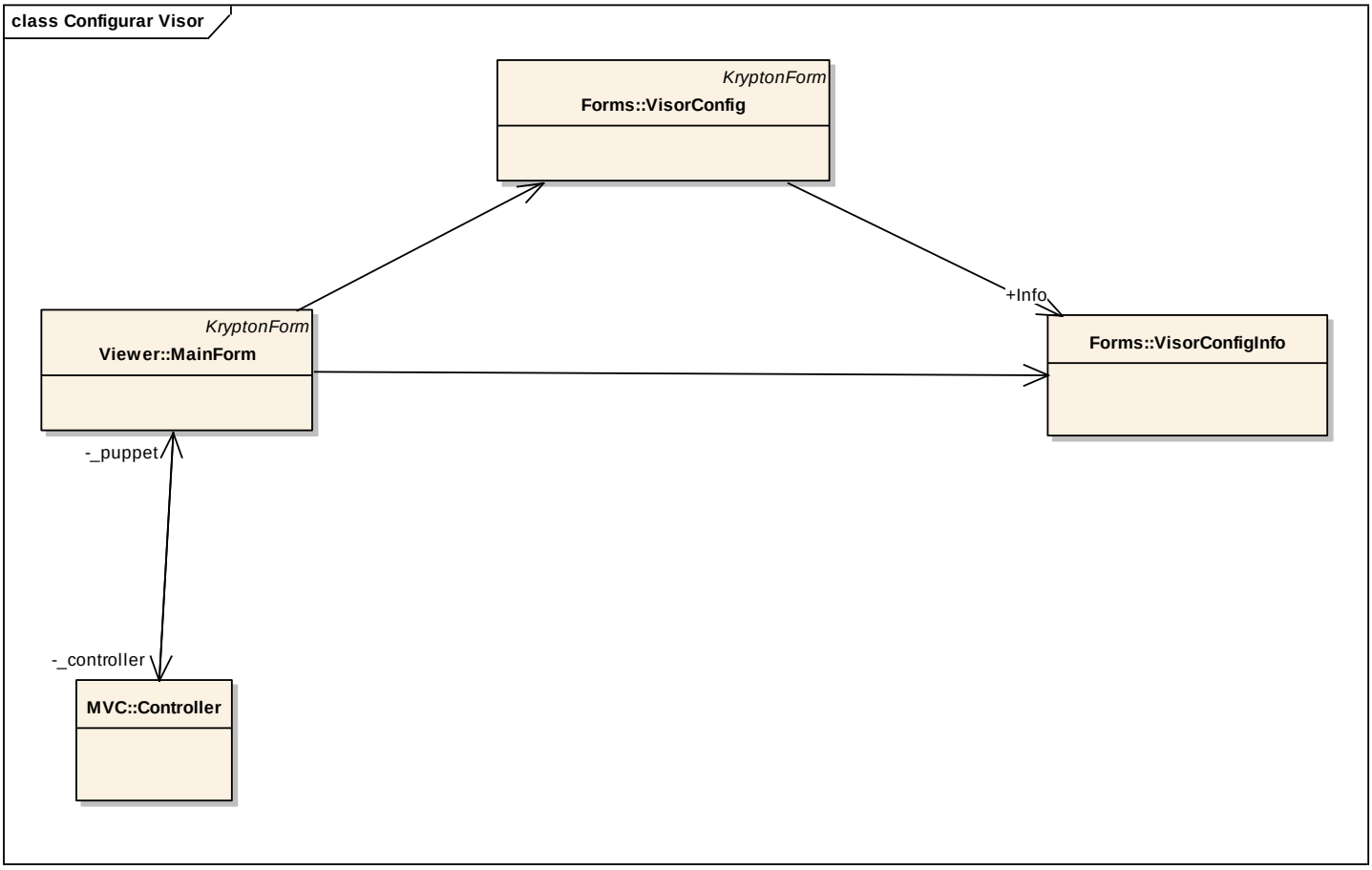

Figura 37 Diagrama de Clases del CU Configurar Visor

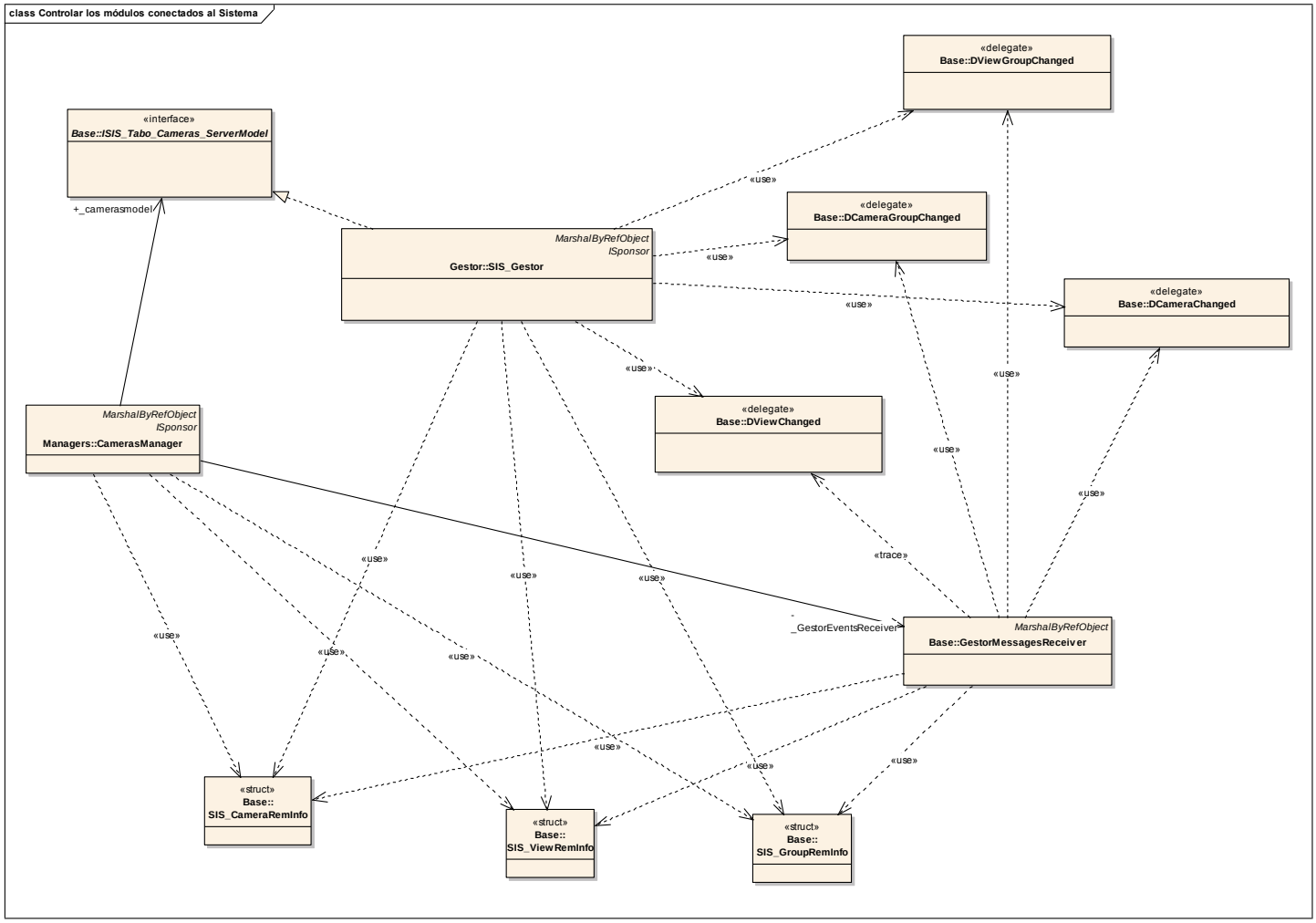

Figura 38 Diagrama de Clases del CU Controlar los Módulos Conectados al Sistema

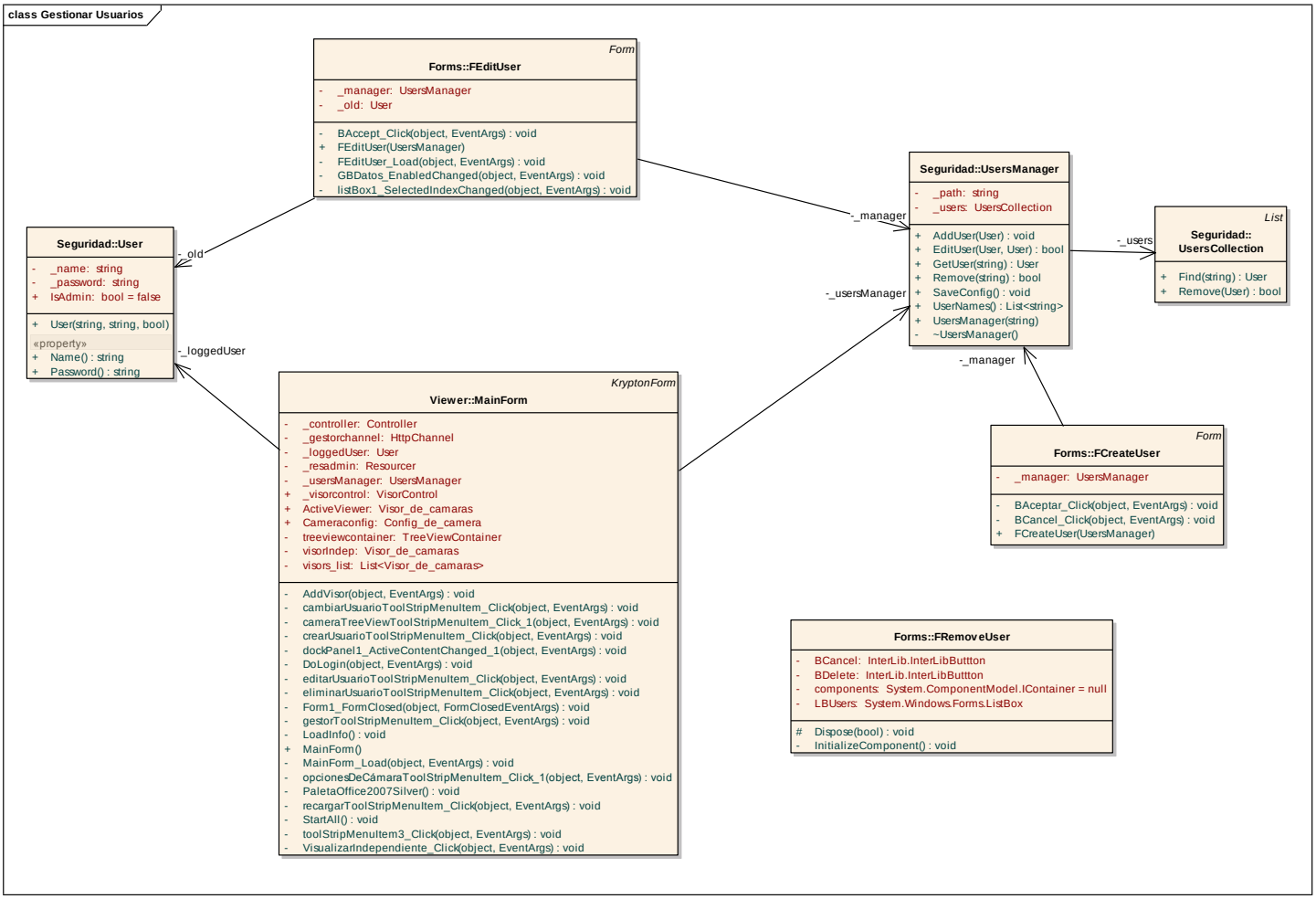

Figura 39 Diagrama de Clases del CU Gestionar Usuarios

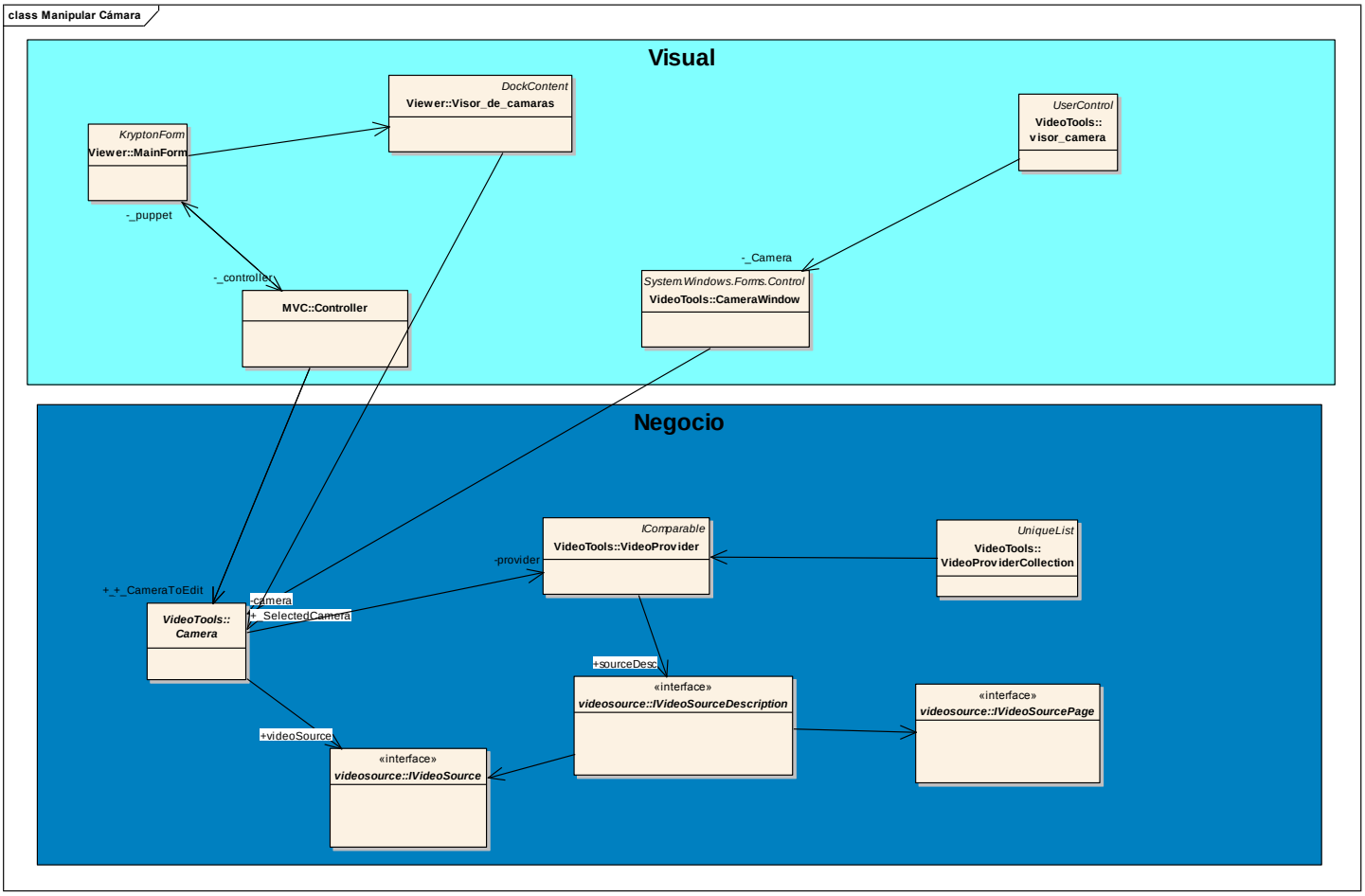

Figura 40 Diagrama de Clases del CU Manipular Cámara

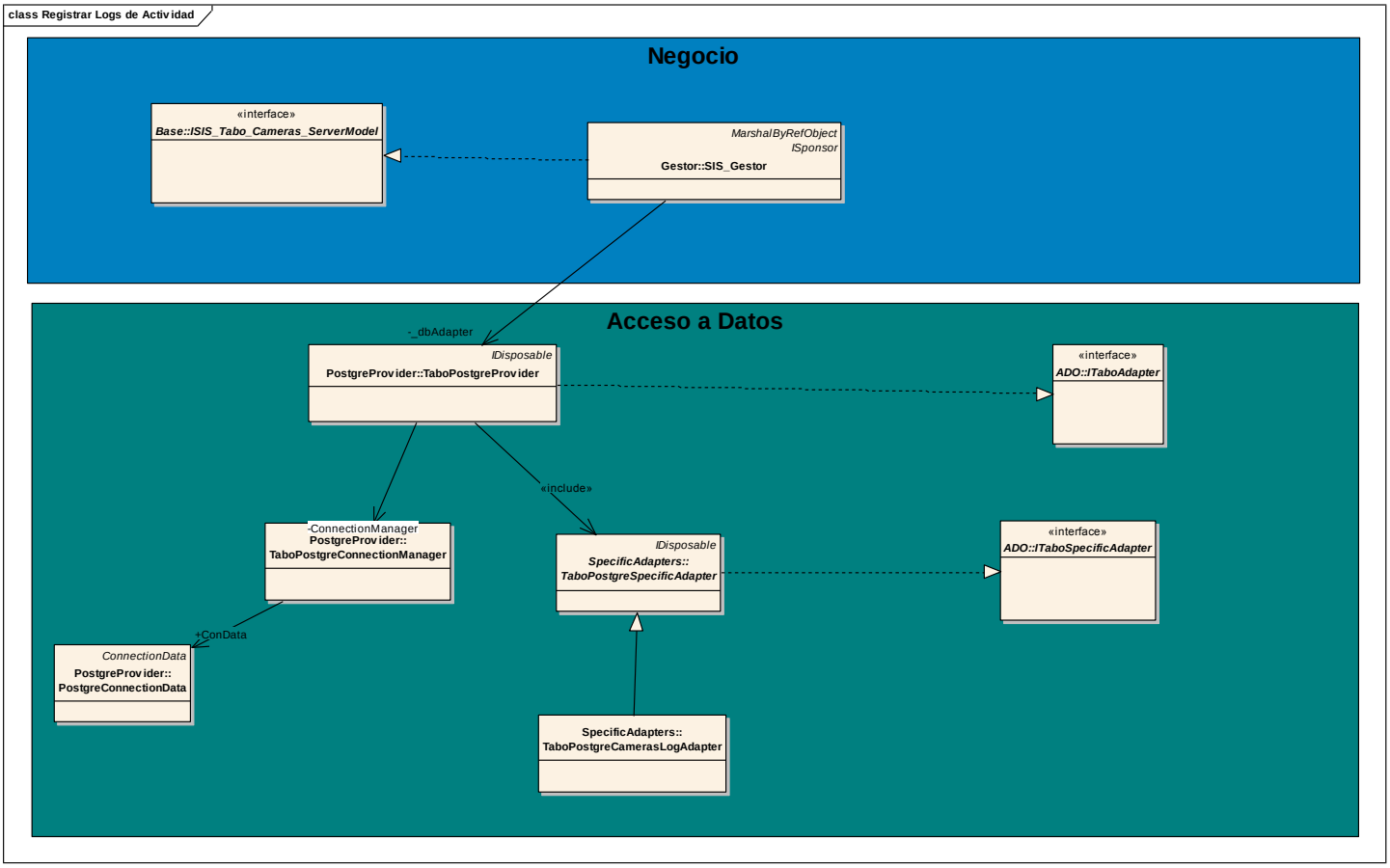

Figura 41 Diagrama de Clases del CU Registrar Logs de Actividad

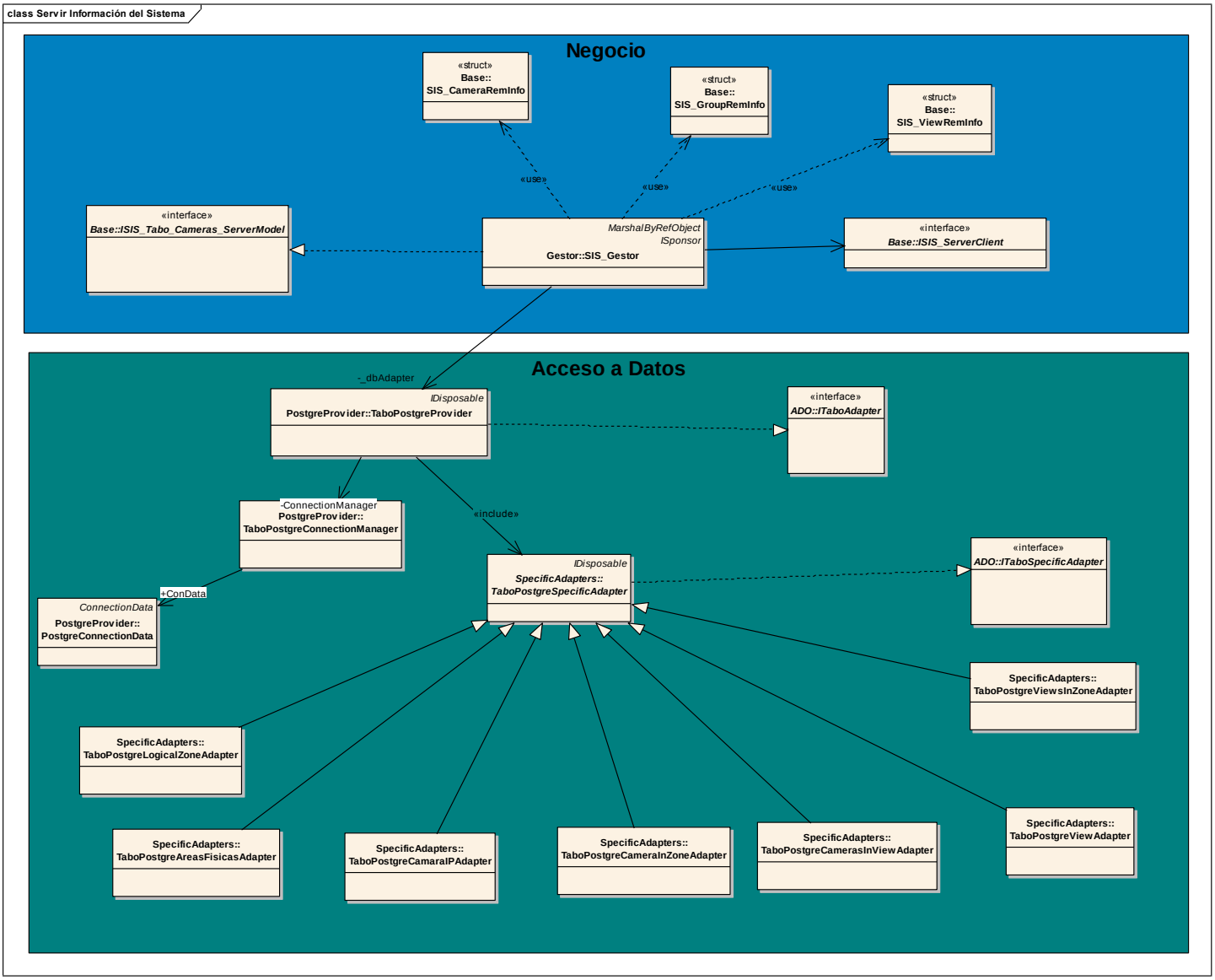

Figura 42 Diagrama de Clases del CU Servir Información del Sistema

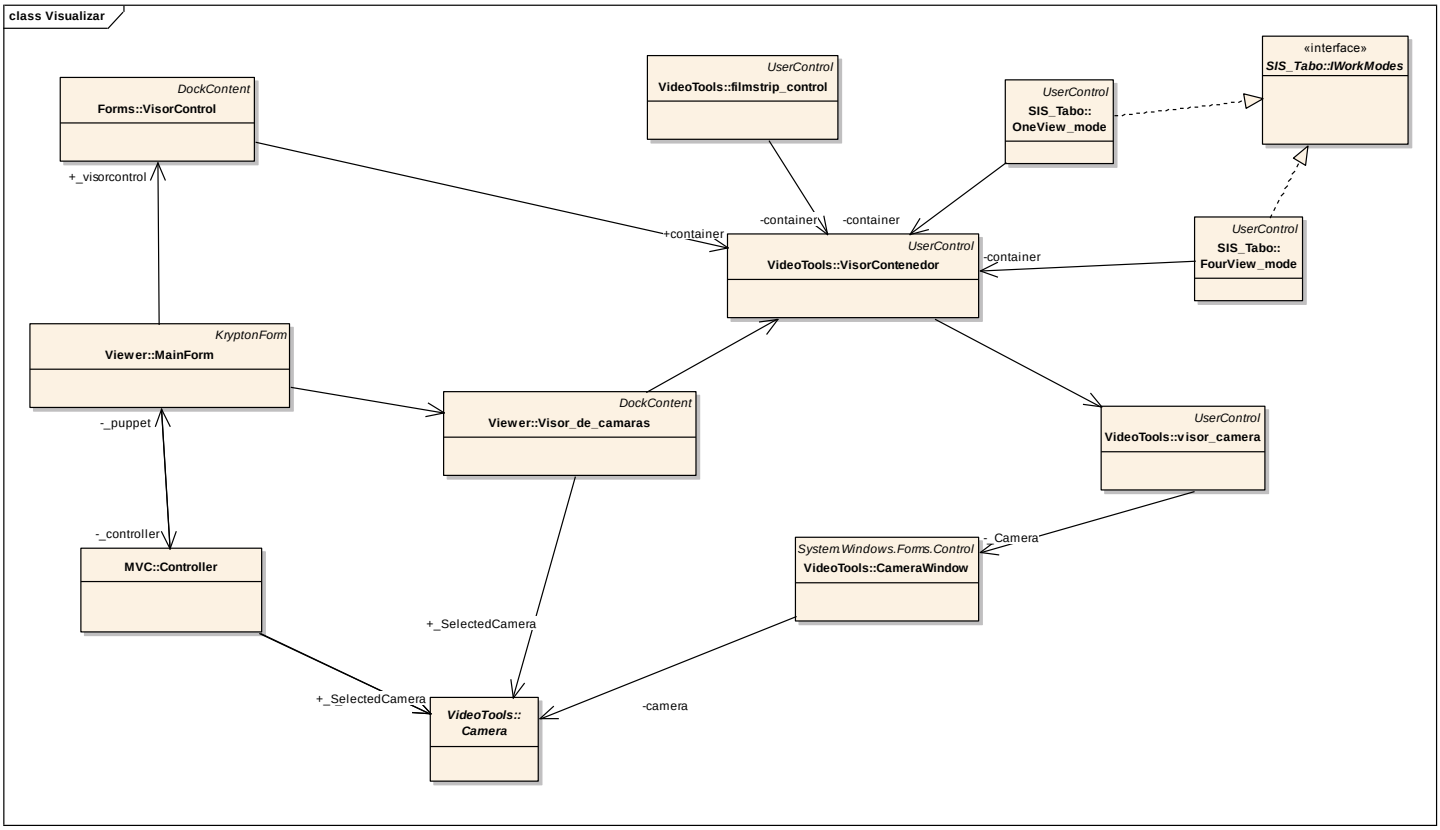

Figura 43 Diagrama de Clases del CU Visualizar

Anexo II Diagramas de interacción del diseño.

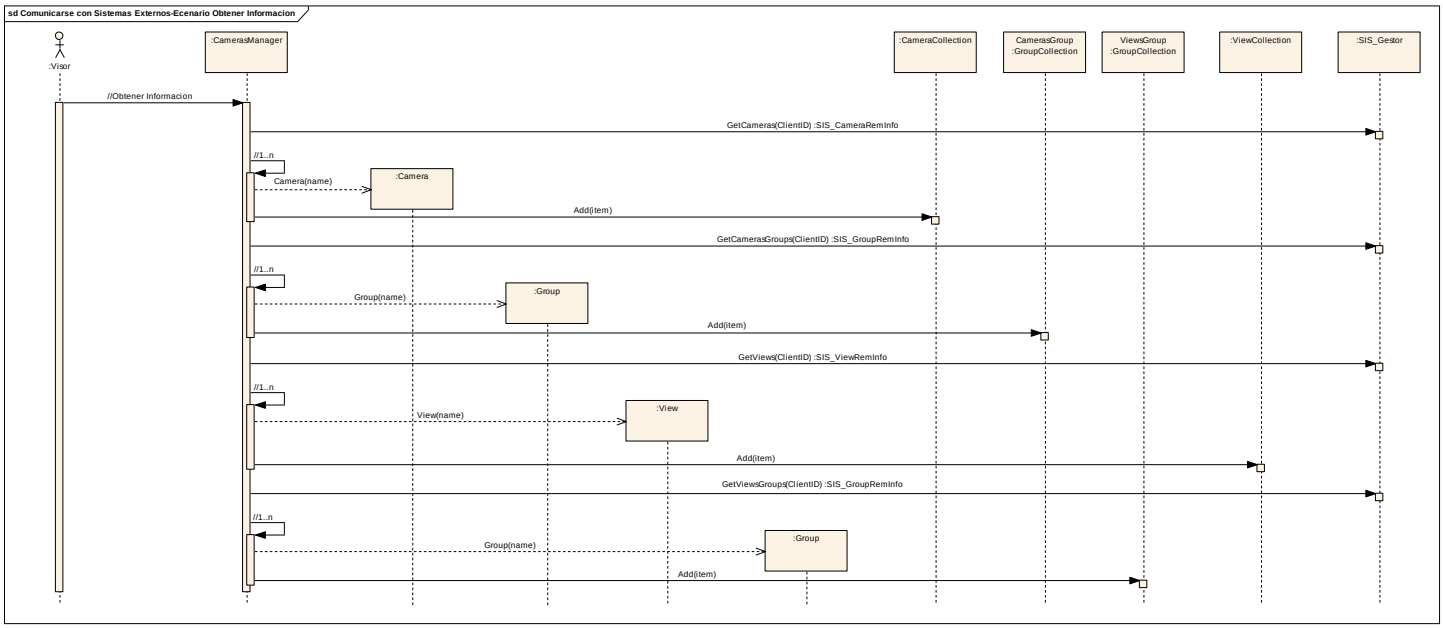

Figura 44 Diagrama de secuencia del CU Comunicarse con Sistemas Externos.

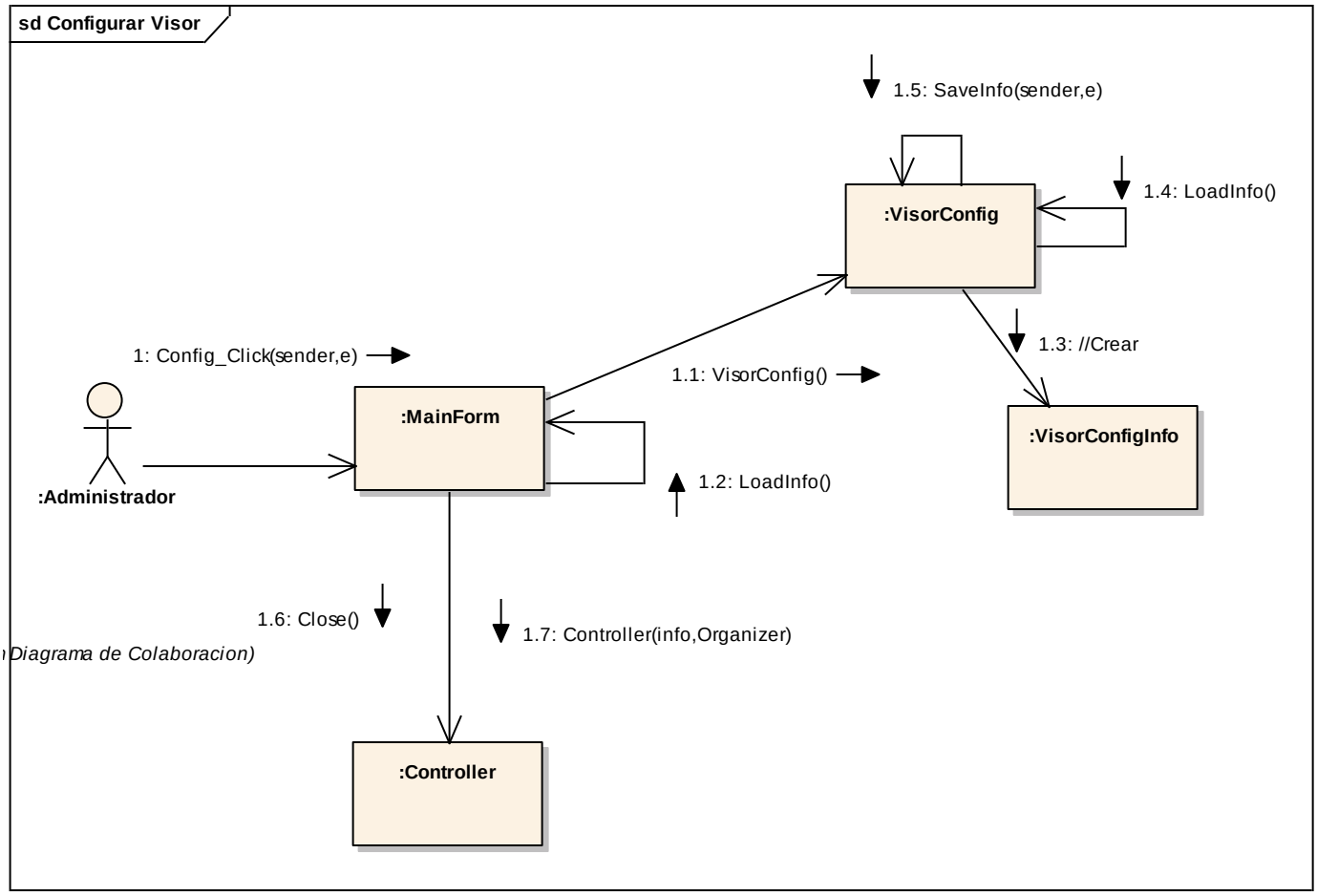

Figura 45 Diagrama de colaboración del CU Configurar Visor.

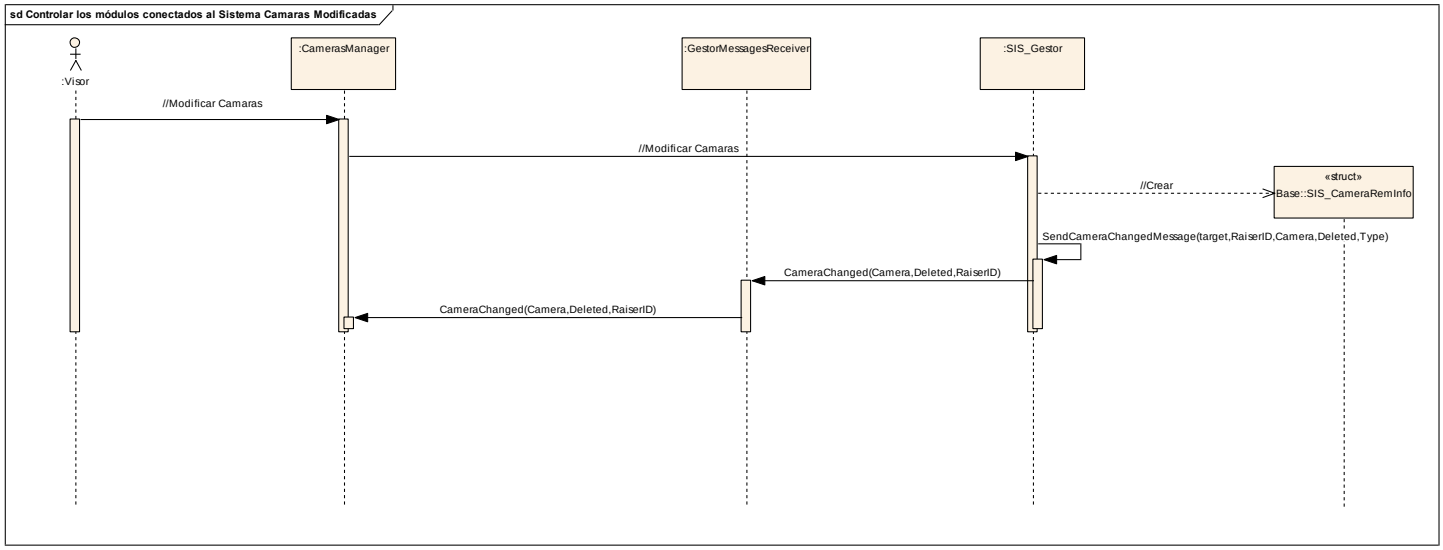

Figura 46 Diagrama de secuencia del CU Controlar los módulos conectados al Sistema (Escenario Cámaras Modificadas).

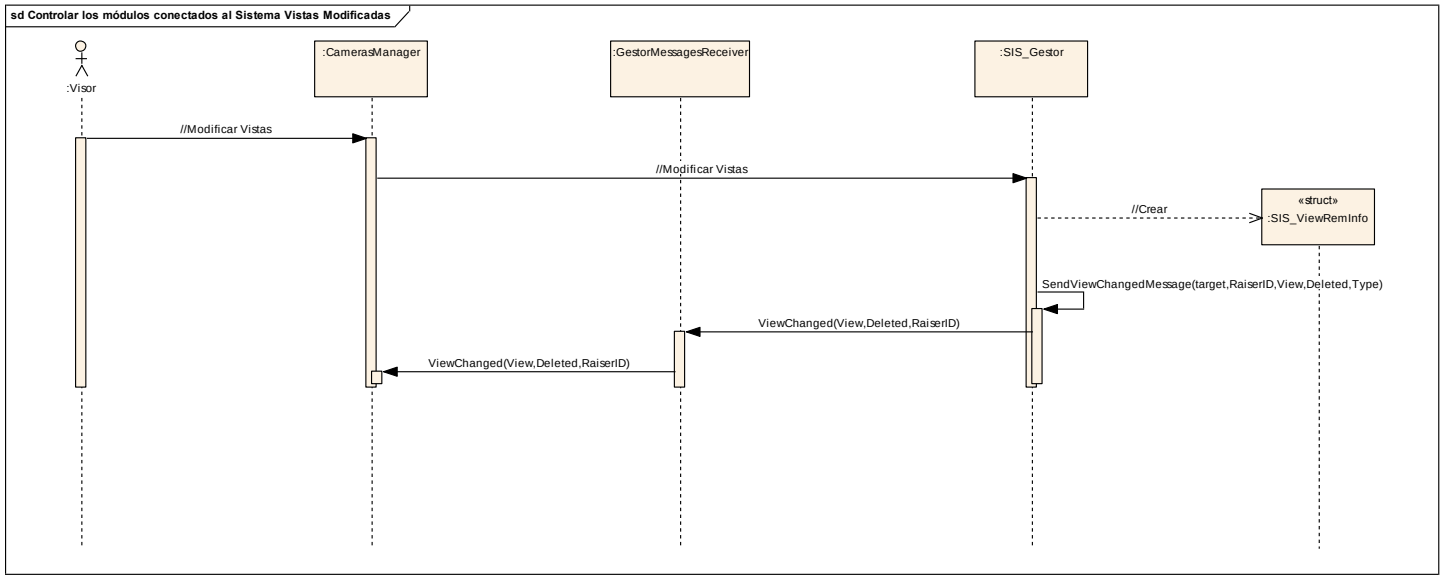

Figura 47 Diagrama de secuencia del CU Controlar los módulos conectados al Sistema (Escenario Vistas Modificadas).

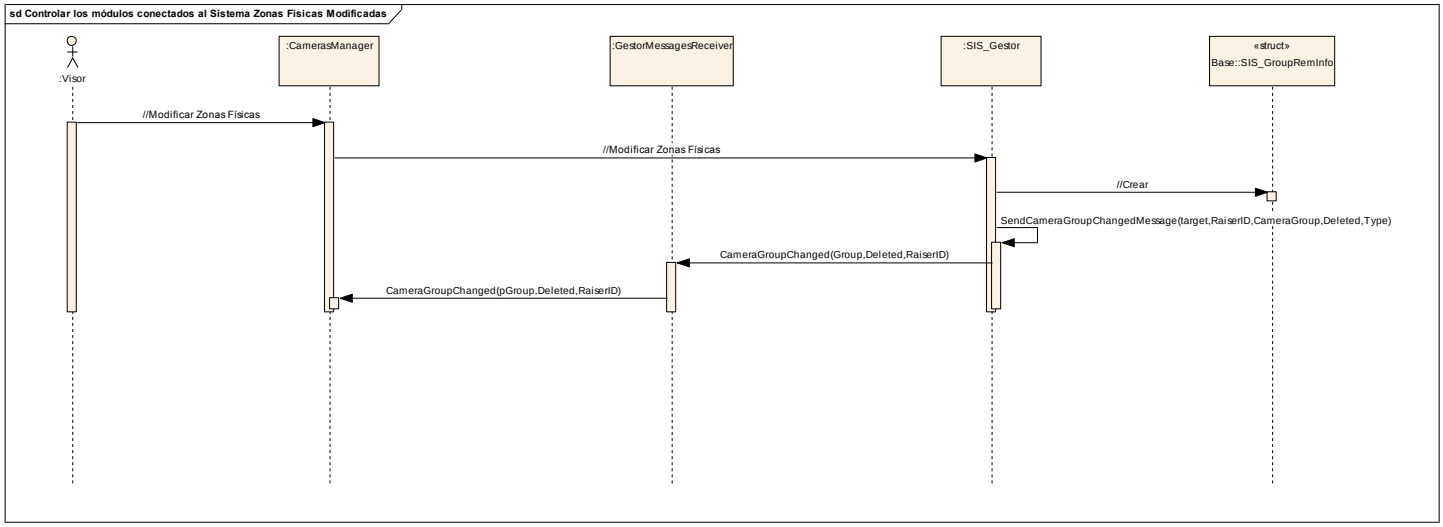

Figura 48 Diagrama de secuencia del CU Controlar los módulos conectados al Sistema (Escenario Zonas Físicas Modificadas).

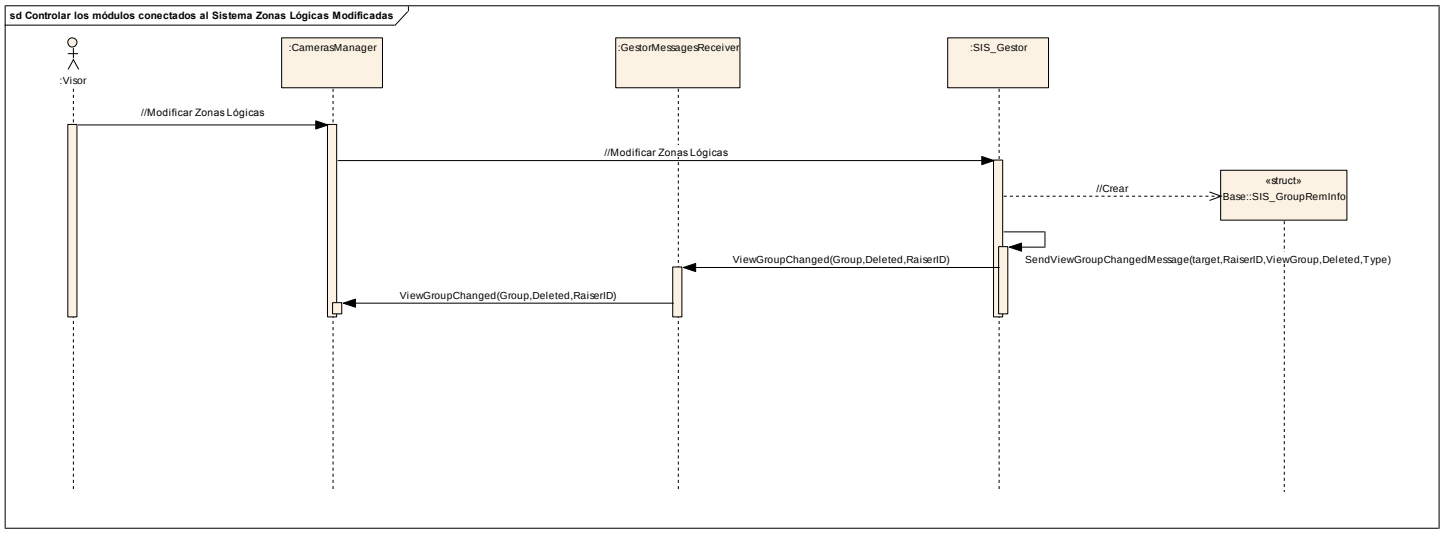

Figura 49 Diagrama de secuencia del CU Controlar los módulos conectados al Sistema (Escenario Zonas Lógicas Modificadas).

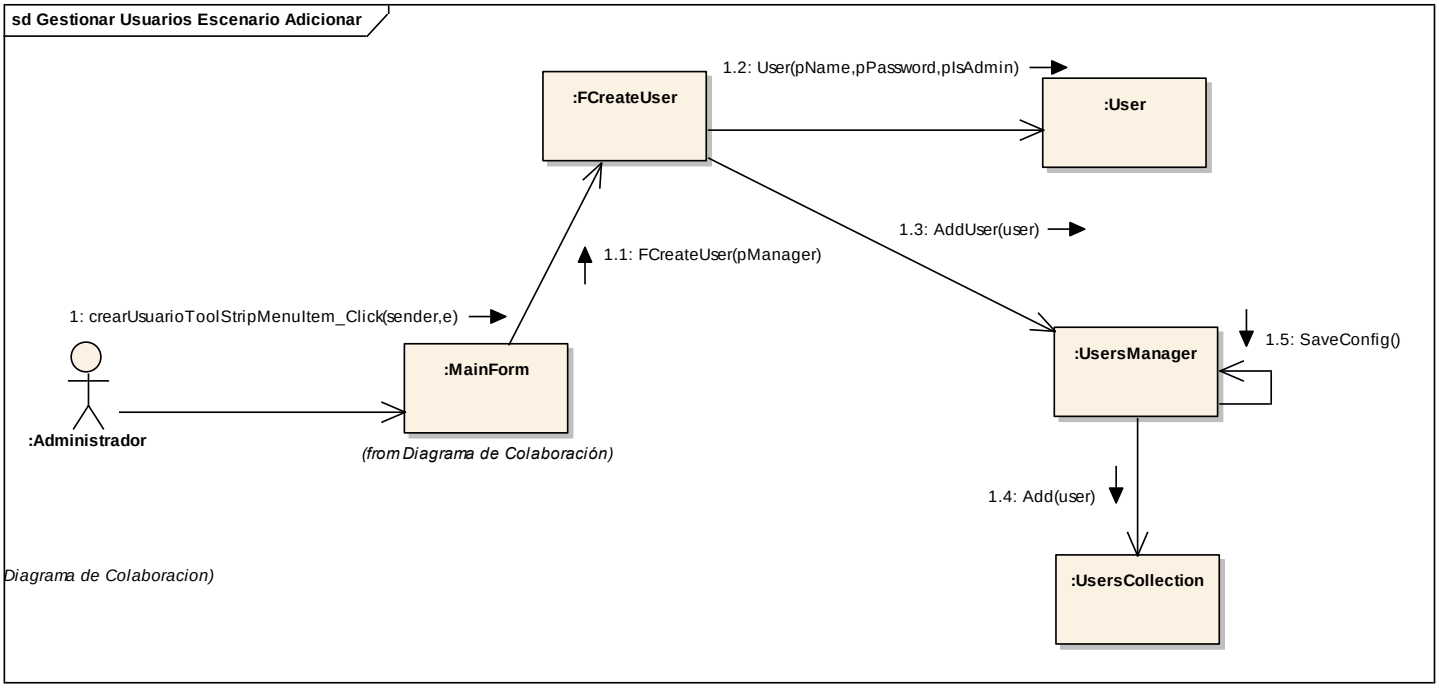

Figura 50 Diagrama de colaboración del CU Gestionar Usuarios (Escenario Adicionar).

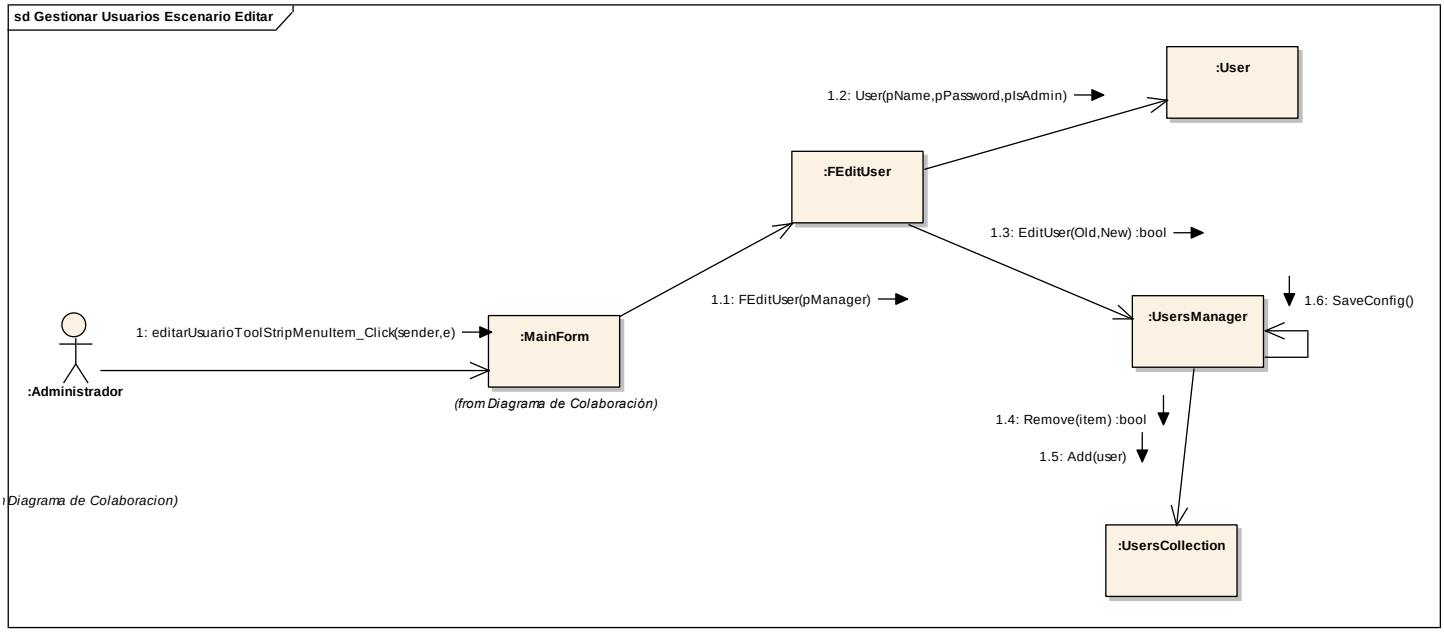

Figura 51 Diagrama de Colaboración del CU Gestionar Usuarios (Escenario Editar).

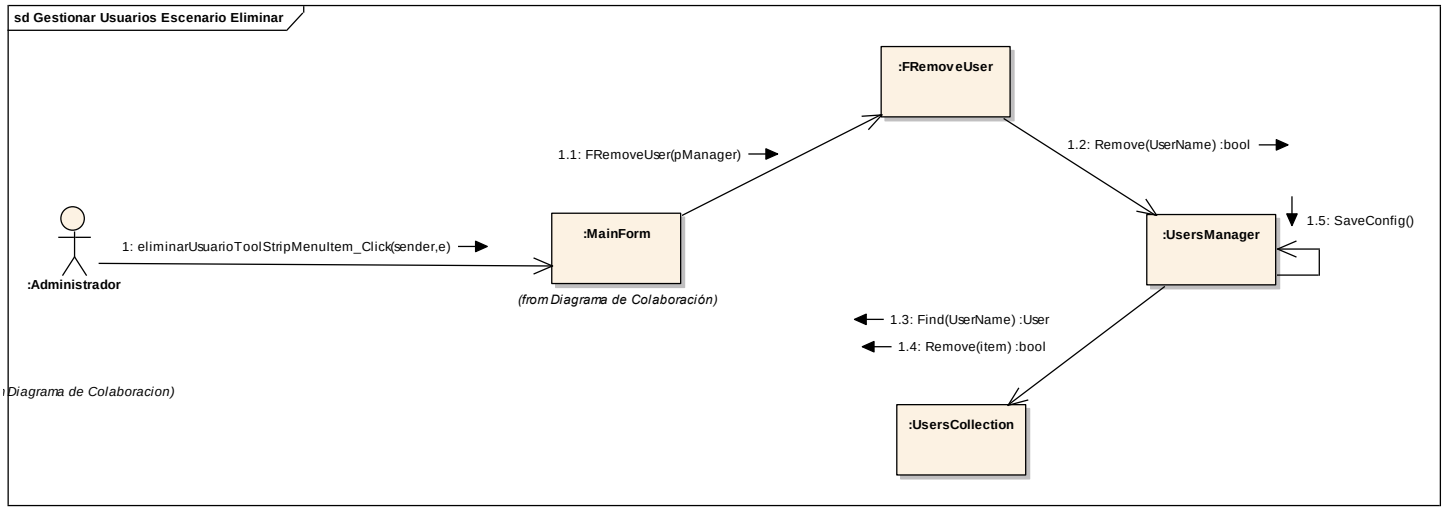

Figura 52 Diagrama de Colaboración del CU Gestionar Usuarios (Escenario Eliminar).

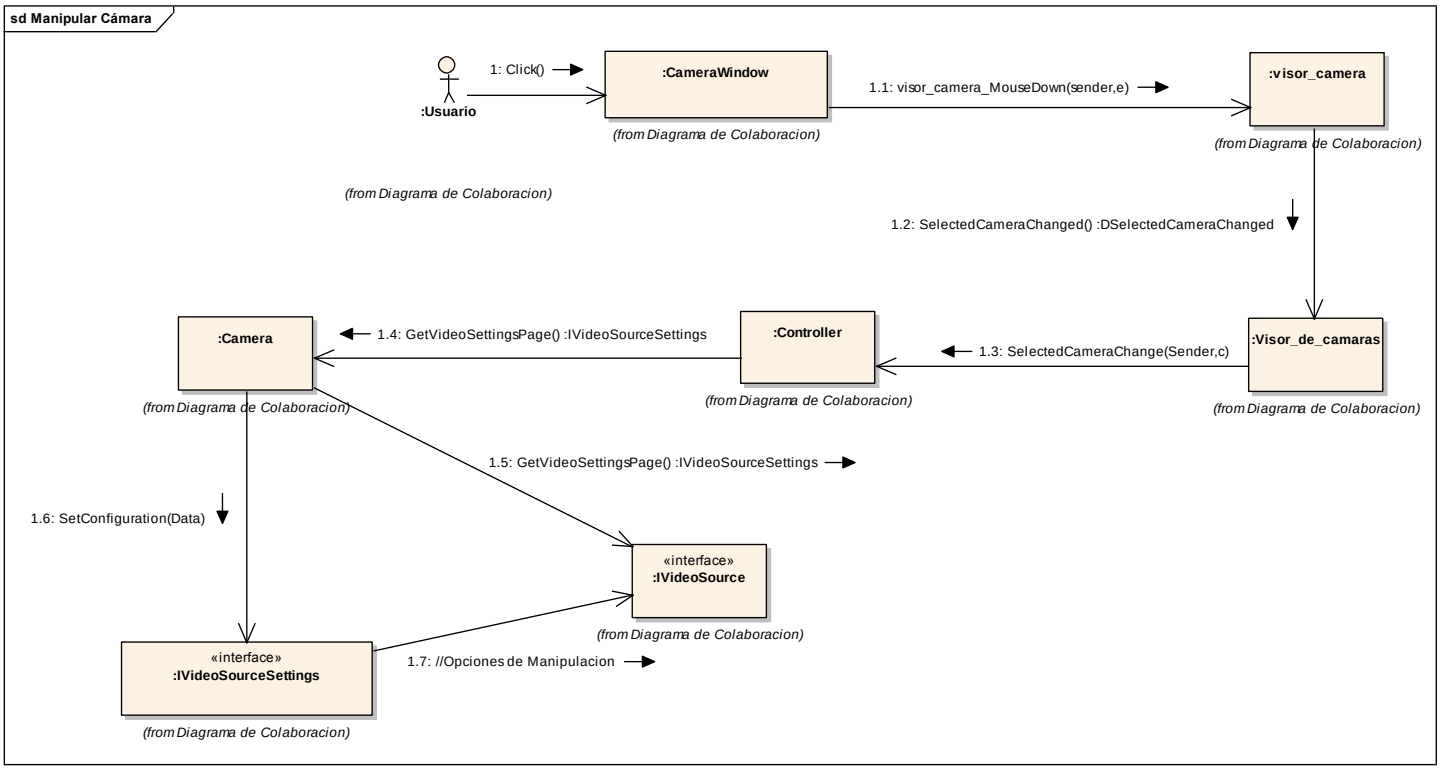

Figura 53 Diagrama de colaboración del CU Manipular Cámara.

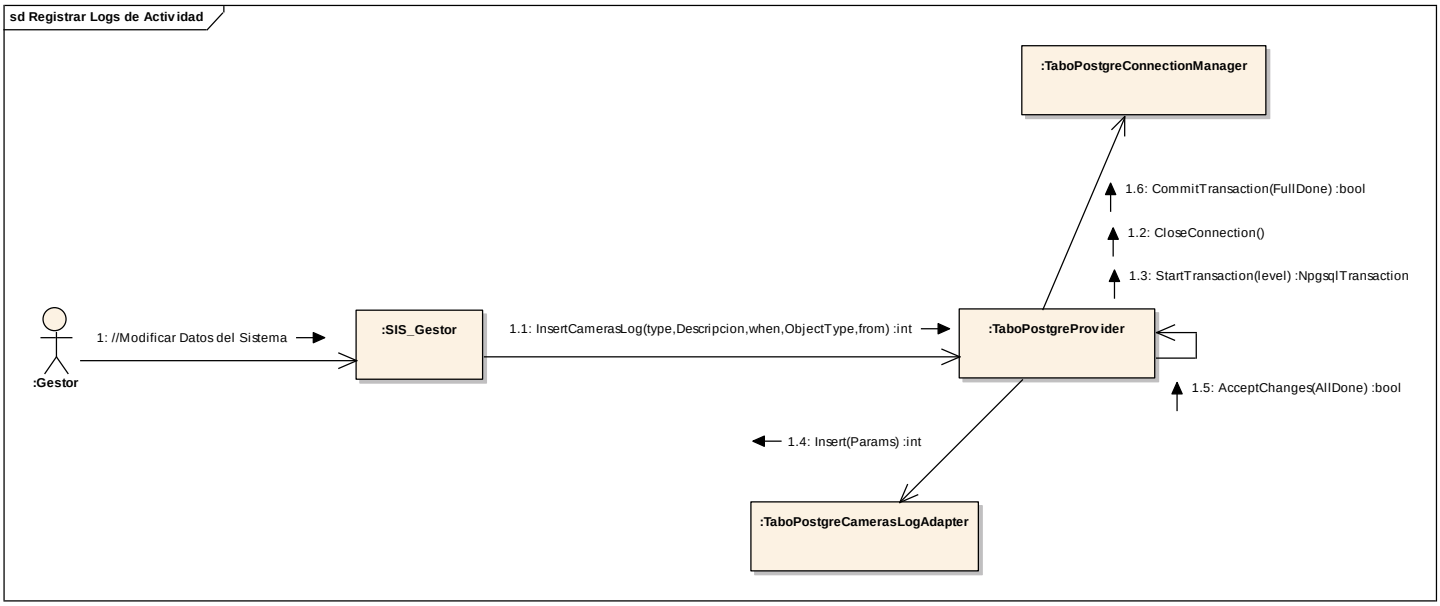

Figura 54 Diagrama de colaboración del CU Registrar Logs de Actividad.

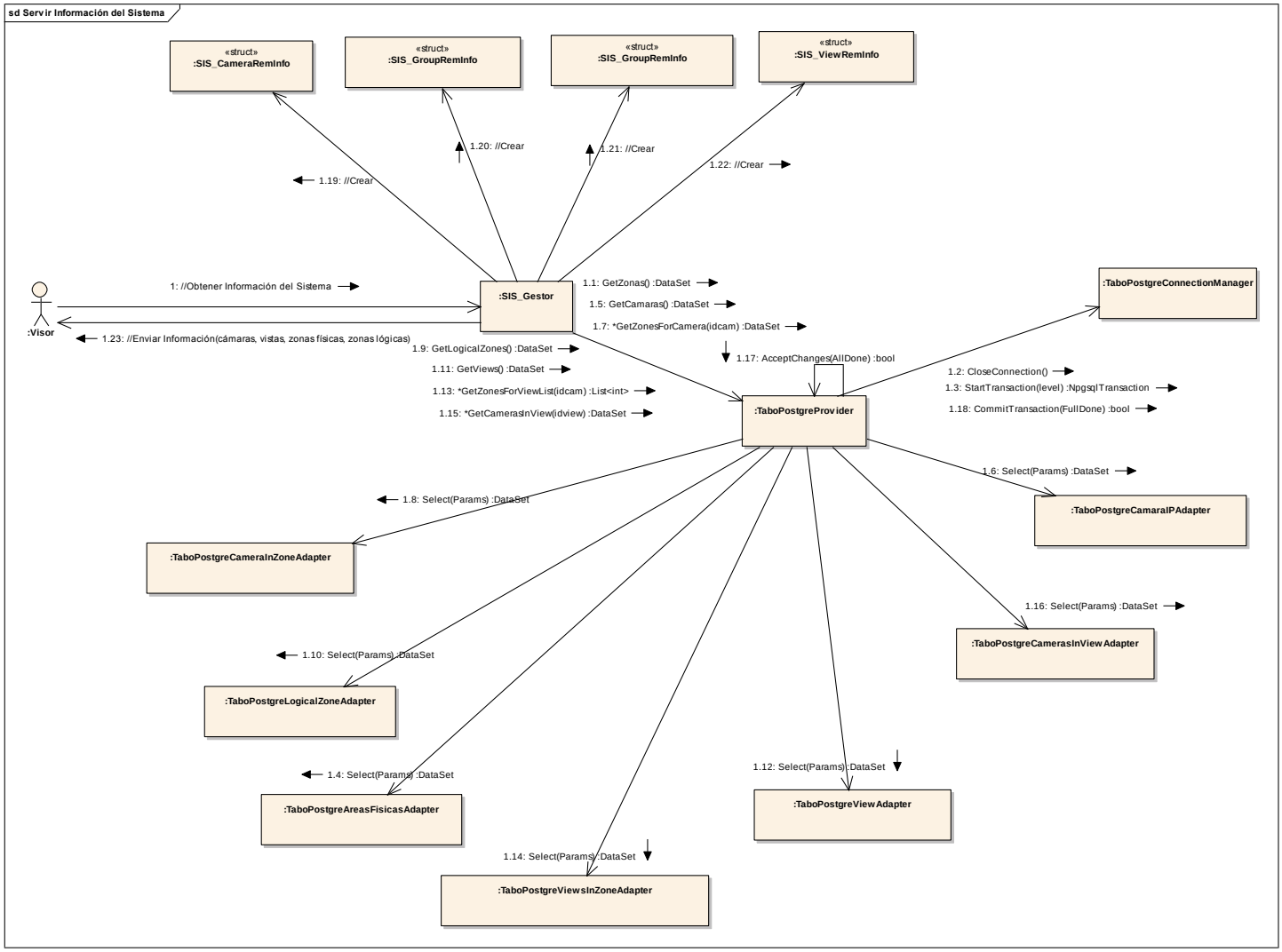

Figura 55 Diagrama de colaboración del CU Servir Información del Sistema.

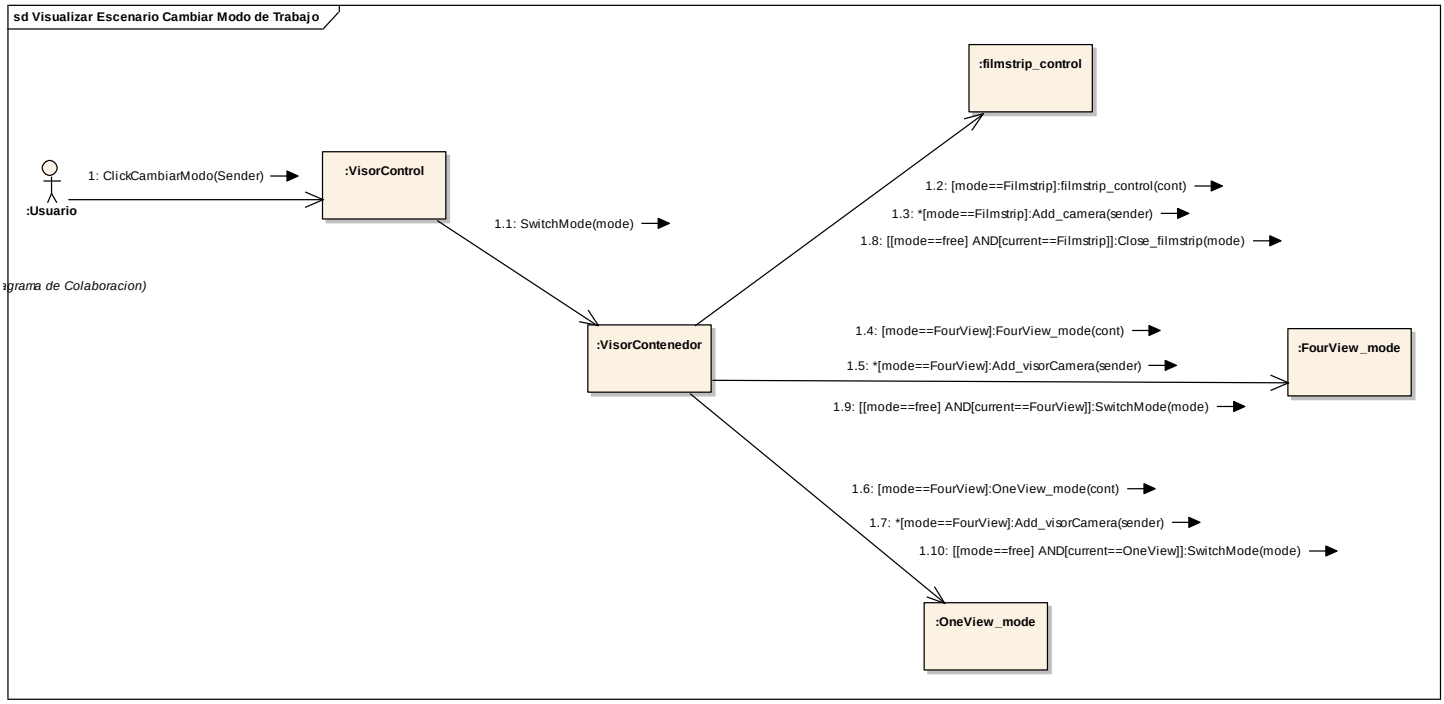

Figura 56 Diagrama de colaboración del CU Visualizar (Escenario Cambiar Modo de Trabajo).

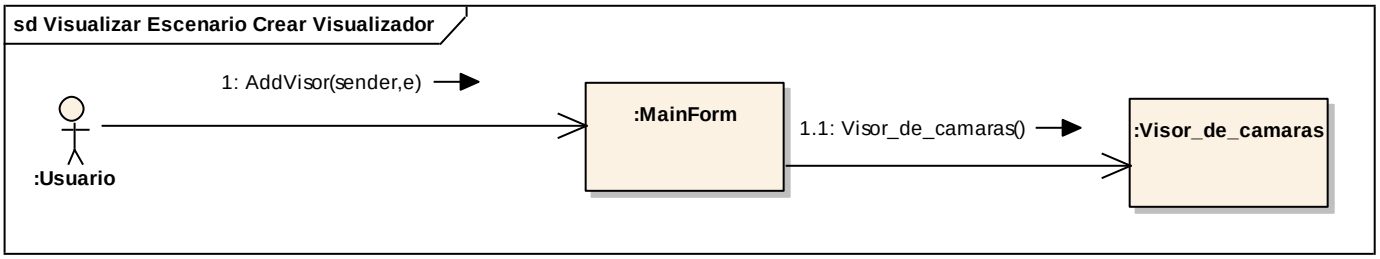

Figura 57 Diagrama de colaboración del CU Visualizar (Escenario Crear Visualizador).

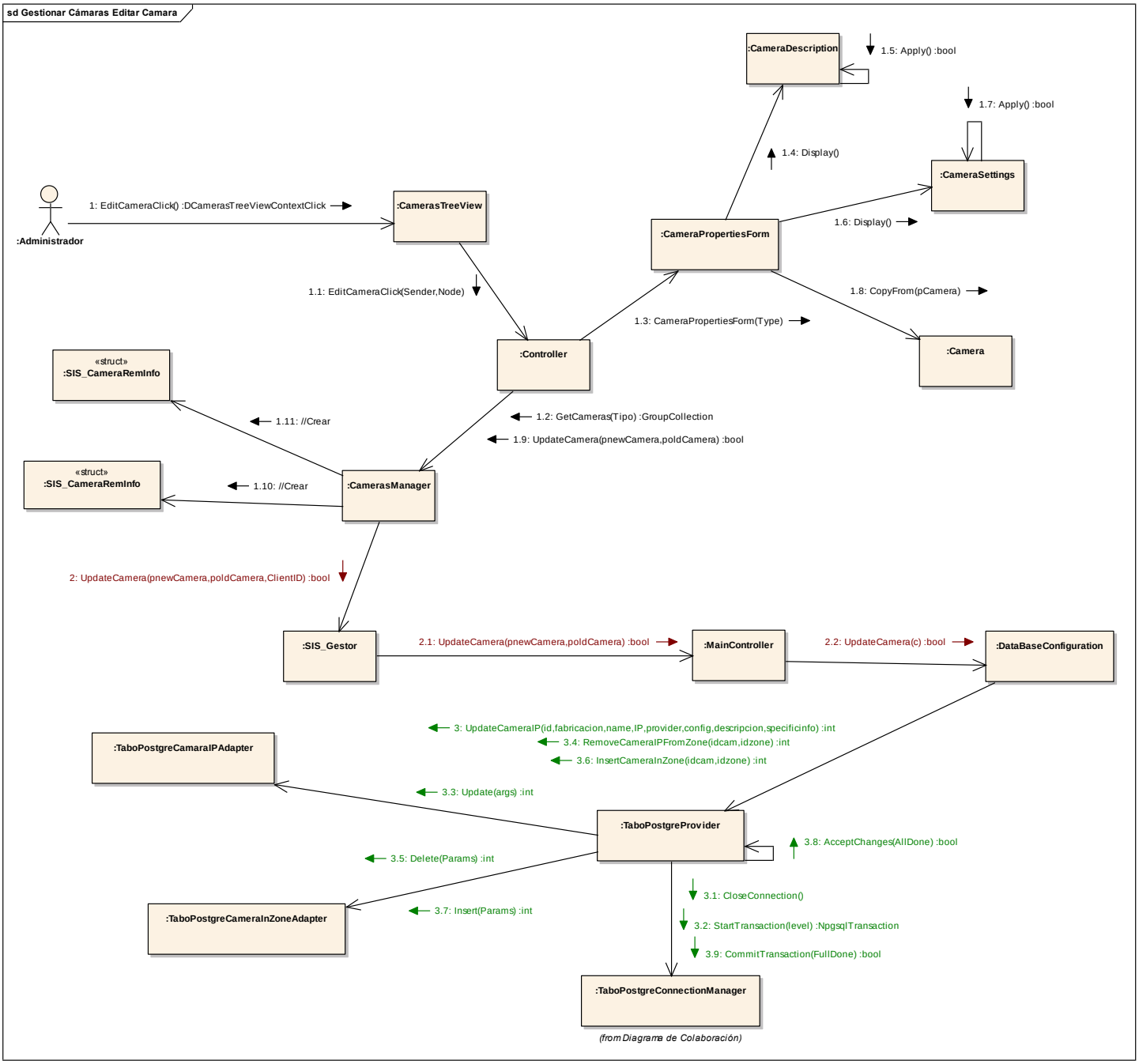

Figura 58 Diagrama de colaboración del CU Gestionar Cámaras (Escenario Editar Cámara).

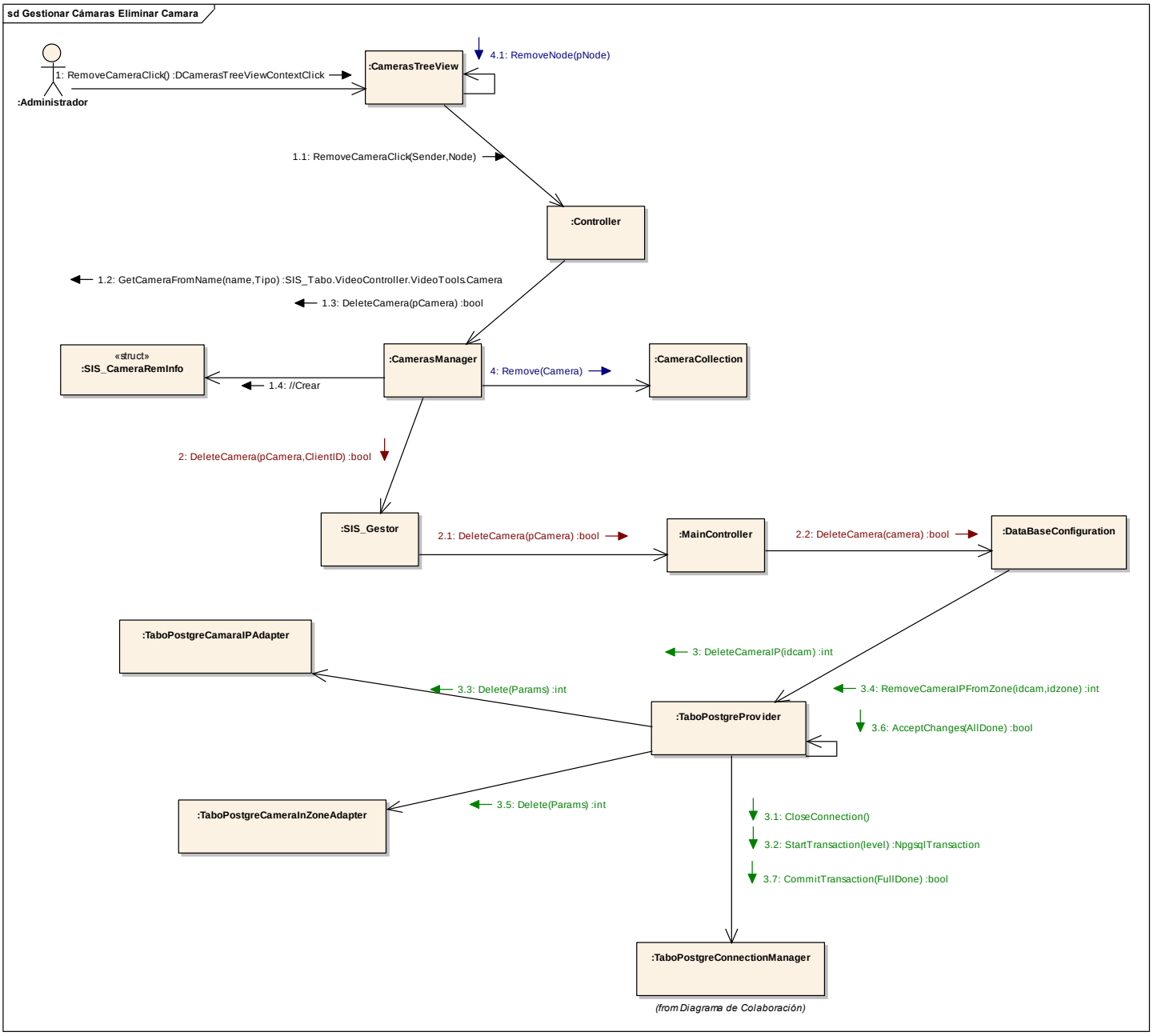

Figura 59 Diagrama de colaboración del CU Gestionar Cámaras (Escenario Eliminar Cámara).

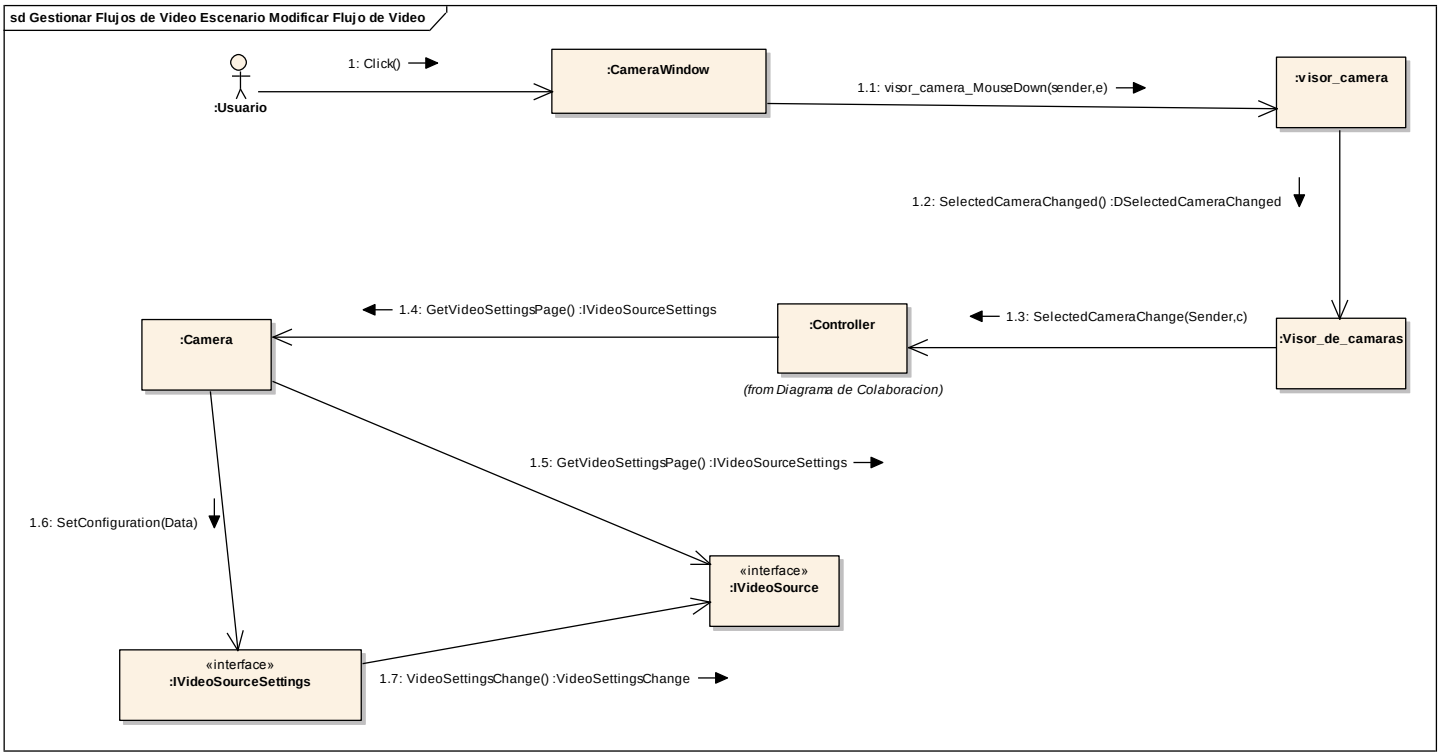

Figura 60 Diagrama de Secuencia del CU Gestionar Flujos de Videos (Escenario Modificar flujo de video).

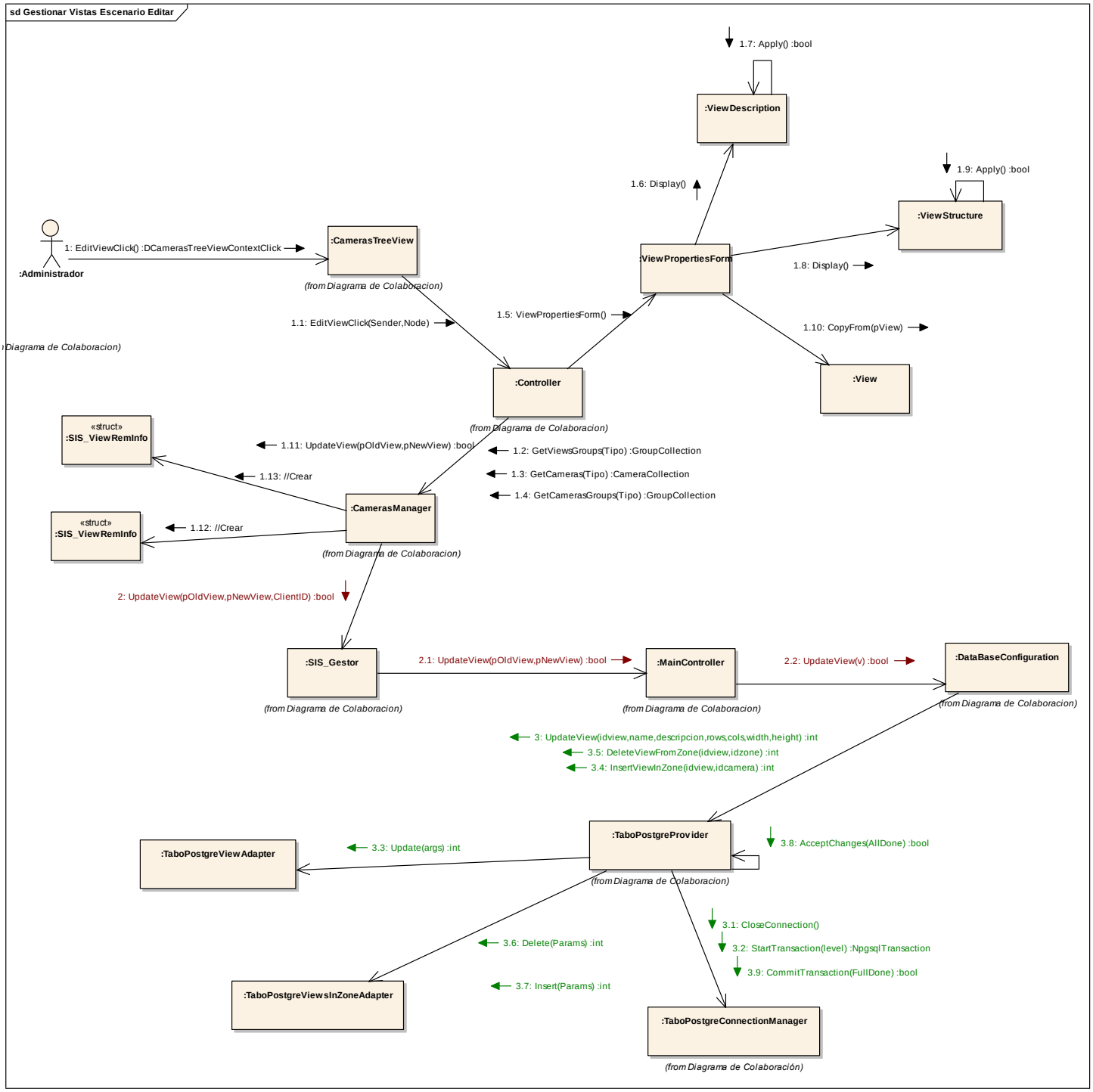

Figura 61 Diagrama de colaboración del CU Gestionar Vistas (Escenario Editar).

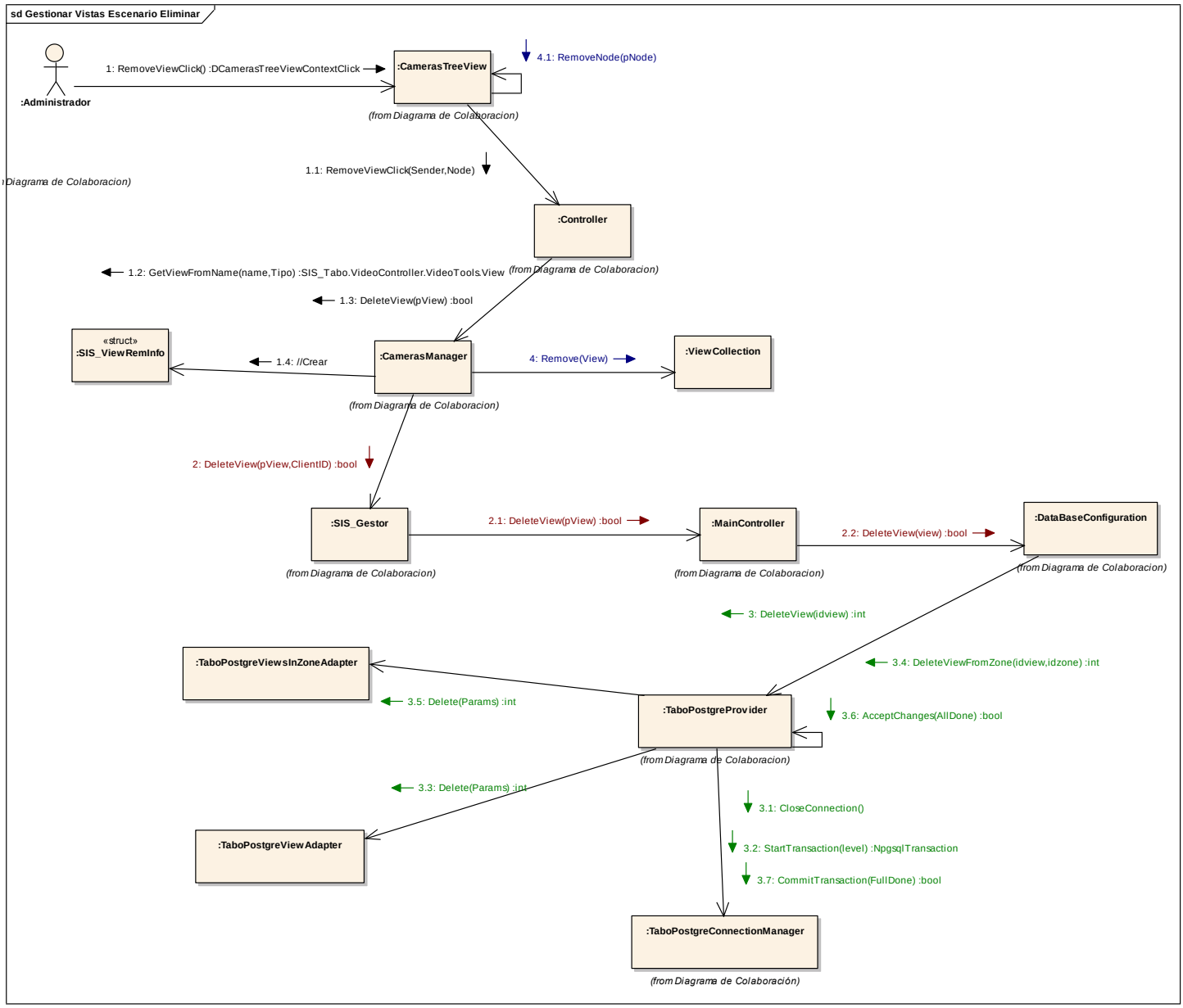

Figura 62 Diagrama de colaboración del CU Gestionar Vistas (Escenario Eliminar).

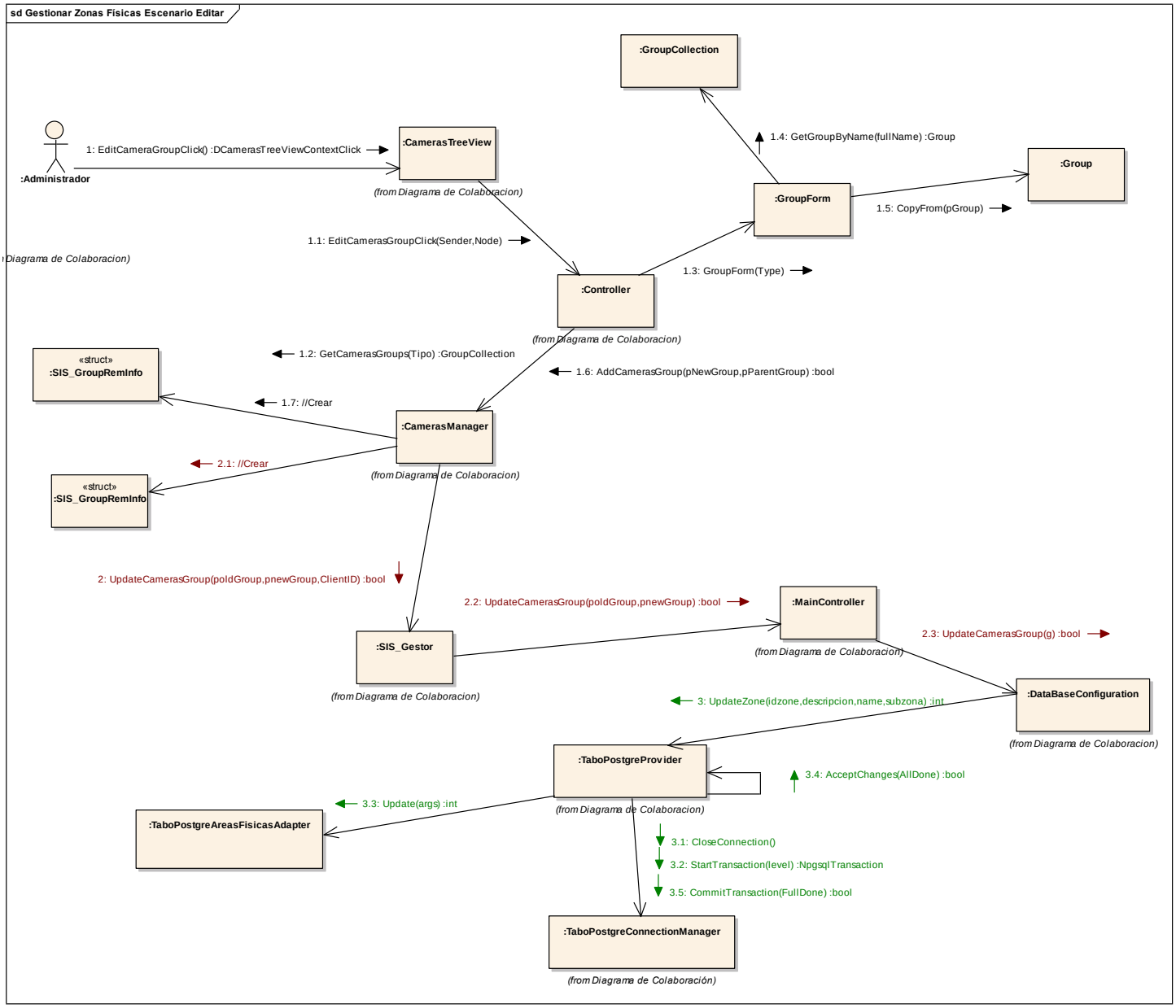

Figura 63 Diagrama de colaboración del CU Gestionar Zonas Físicas (Escenario Editar).

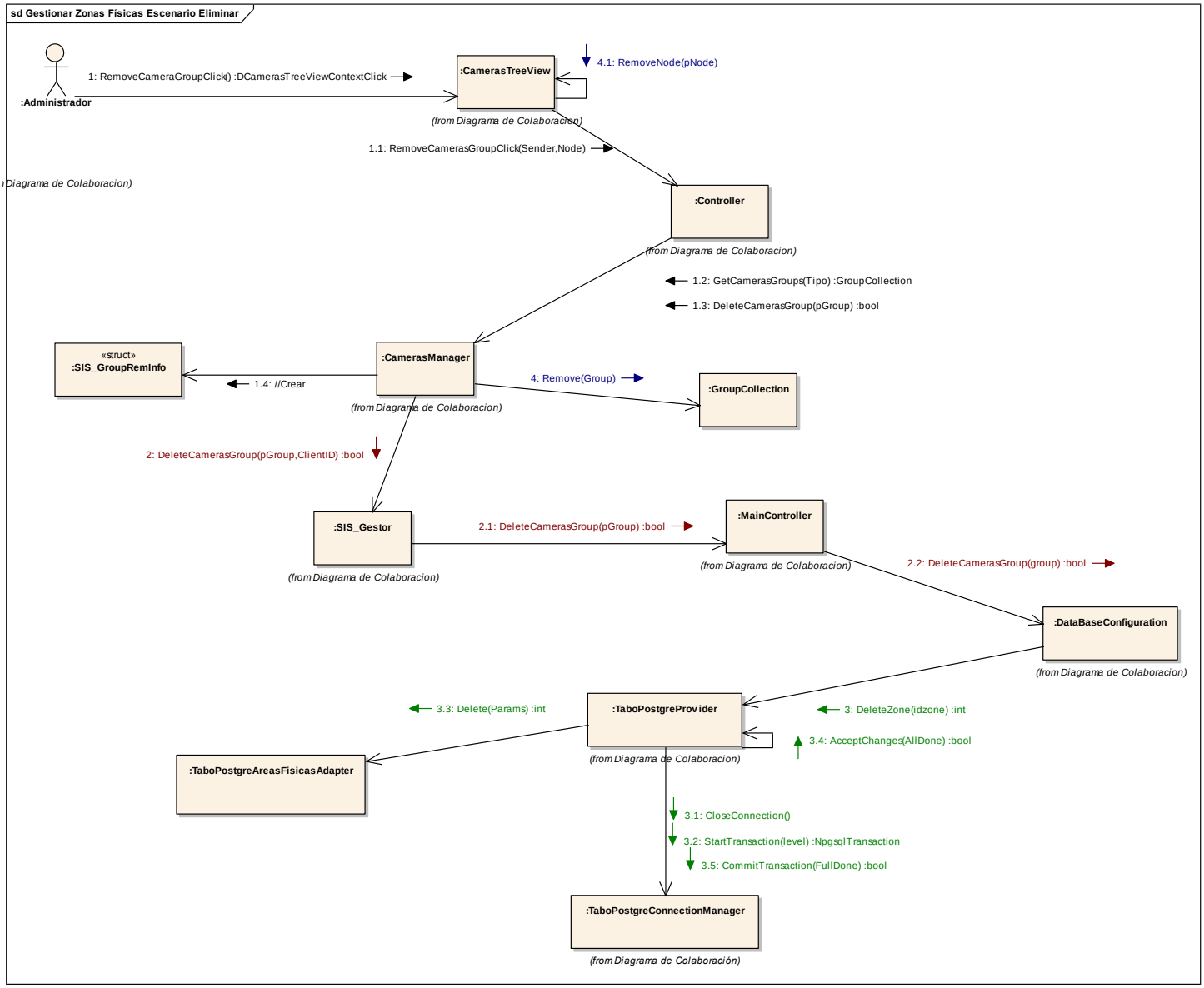

Figura 64 Diagrama de colaboración del CU Gestionar Zonas Físicas (Escenario Eliminar).

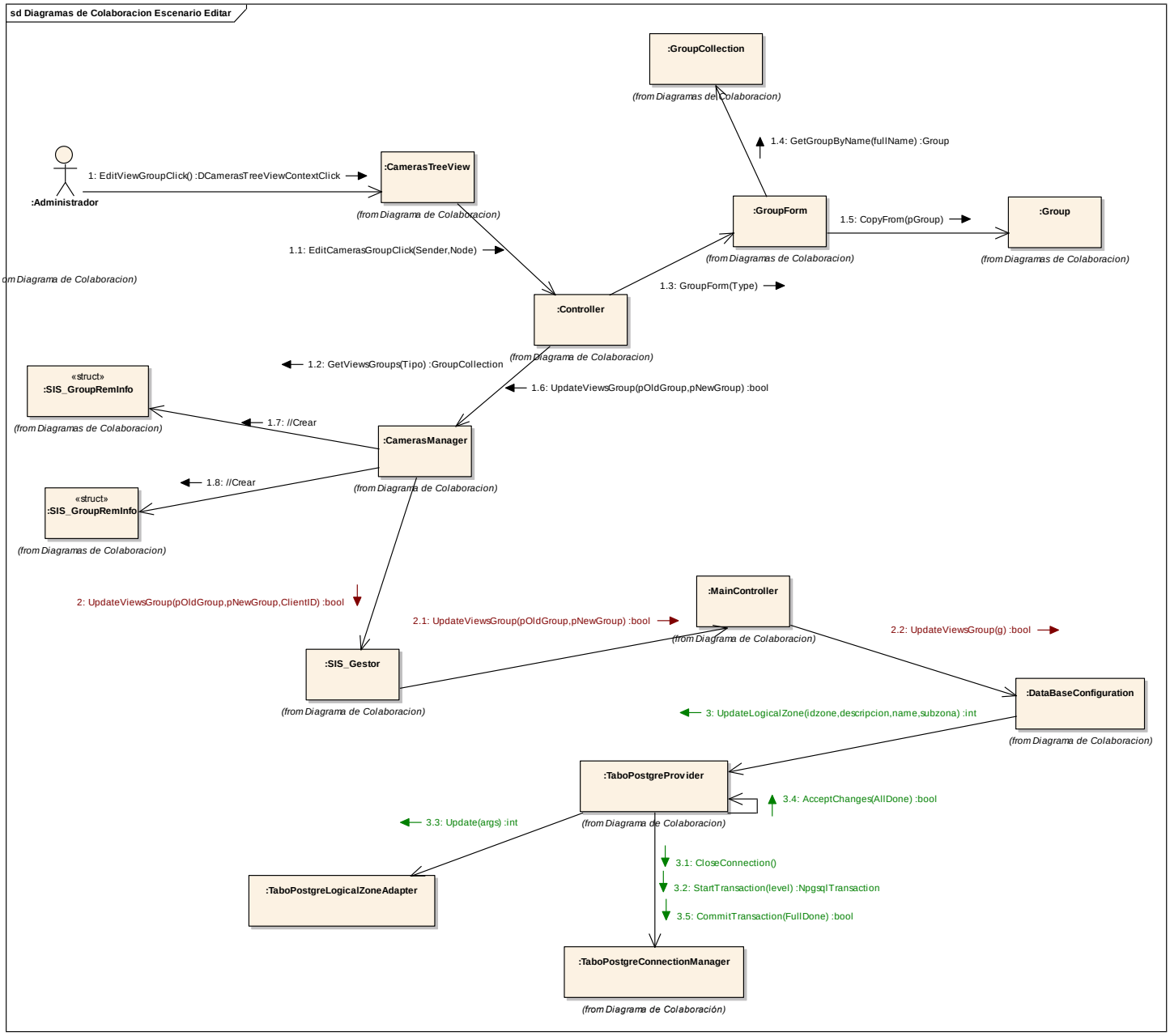

Figura 65 Diagrama de colaboración del CU Gestionar Zonas Lógicas (Escenario Editar).

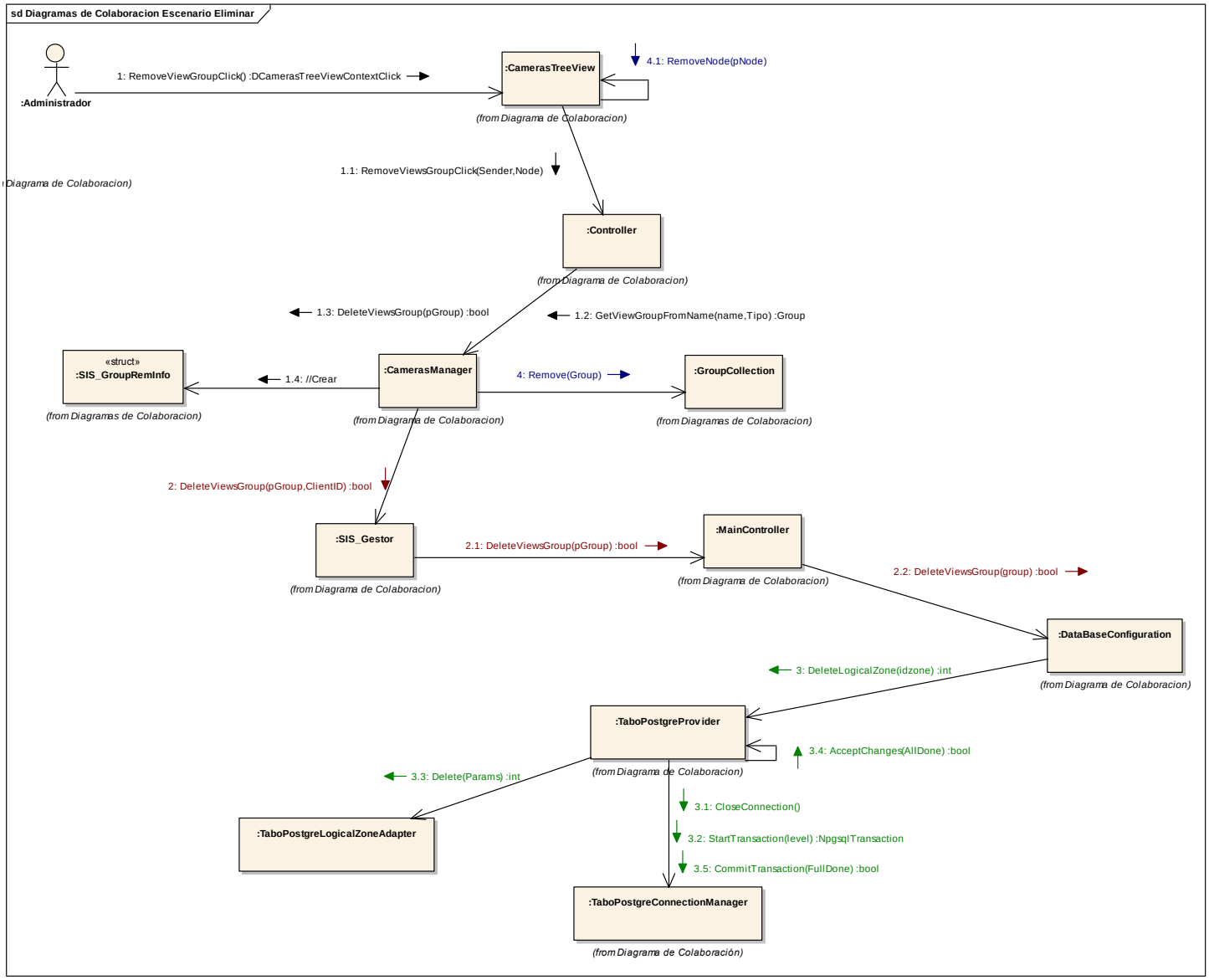

Figura 66 Diagrama de colaboración del CU Gestionar Zonas Lógicas (Escenario Eliminar).

Anexo III Descripción de las clases de diseño.

Nombre: User	
Tipo de clase : Entidad	
Atributo	**Tipo**
_name	string
_password	string
IsAdmin	bool
Para cada responsabilidad:	
Nombre:	User(string Nombre, string Password ,bool IsAdmin)
Descripción:	Constructor que crea el Usuario con los datos requeridos.

Nombre: UsersCollection : List<User>	
Tipo de clase : Entidad	
Para cada responsabilidad:	
Nombre:	Find(string Nombre)
Descripción:	Busca un usuario por su nombre en la colección y lo devuelve
Nombre:	Remove(User usuario)
Descripción:	Elimina un usuario de la colección.

Nombre: GestorMessagesReceiver	
Tipo de clase : Controladora	
Atributo	**Tipo**
_OwnerID	int
Para cada responsabilidad:	
Nombre:	CameraChanged (SIS_CameraRemInfo, bool, int)
Descripción:	Handler para capturar los cambios en las cámaras.
Nombre:	CameraGroupChanged (SIS_GroupRemInfo, bool, int)

Descripción:	Handler para capturar los cambios en las zonas físicas.
Nombre:	ViewChanged (SIS_ViewRemInfo, bool, int)
Descripción:	Handler para capturar los cambios en las vistas.
Nombre:	ViewGroupChanged (SIS_GroupRemInfo, bool, int)
Descripción:	Handler para capturar los cambios en las zonas lógicas.

Nombre: TaboPostgreProvider	
Tipo de clase : Controladora	
Atributo	**Tipo**
_AutomaticAccept	bool
ConnectionManager	TaboPostgreConnectionManager
Para cada responsabilidad:	
Nombre:	AcceptChanges (bool)
Descripción:	Acepta todos los cambios realizados en la base de datos
Nombre:	DeleteCameraFromView (int, int, int)
Descripción:	Elimina una cámara de una vista
Nombre:	DeleteCameraIP (int)
Descripción:	Elimina una cámara del Sistema.
Nombre:	DeleteCameraIPFromZone (int, int)
Descripción:	Elimina una cámara IP de una zona
Nombre:	DeleteLogicalZone (int)
Descripción:	Elimina una zona lógica.
Nombre	DeleteView (int)
Descripción:	Elimina una vista del Sistema
Nombre	DeleteViewFromZone (int, int)
Descripción:	Saca a una vista de una zona.
Nombre:	DeleteZone (int)
Descripción:	Elimina una zona física.
Nombre:	GetCámaras

Descripción:	Devuelve las cámaras en el Sistema
Nombre:	GetCamerasInView(int)
Descripción:	Devuelve las cámaras que están en una vista.
Nombre:	GetCamerasLogs
Descripción:	Devuelve los logs de las cámaras.
Nombre:	GetLogicalZones
Descripción:	Devuelve todas las zonas lógicas en el Sistema.
Nombre:	GetViews
Descripción:	Devolver las vistas en el Sistema
Nombre:	GetZonas
Descripción:	Devolver las zonas físicas en el Sistema.
Nombre:	GetZonesForCamera(int)
Descripción:	Devolver las zonas en las que esta una cámara.
Nombre:	GetZonesForView (int)
Descripción:	Devolver las zonas en las que se encuentra una vista
Nombre:	InsertCamaraIP (int, string, string, string, string, string, string, string)
Descripción:	Adicionar una cámara IP al Sistema.
Nombre	InsertCameraInView (int, int, int)
Descripción:	Adicionar una cámara a una vista.
Nombre:	InsertCameraInZone (int, int)
Descripción:	Adicionar una cámara a una zona física.
Nombre:	InsertCamerasLog (string, string, DateTime, string, string)
Descripción:	Adicionar un log
Nombre:	InsertLogicalZone (string, int, string)
Descripción:	Adicionar una zona lógica.
Nombre:	InsertView (string, string, int, int, int, int)
Descripción:	Adicionar una vista al Sistema.
Nombre:	InsertViewInZone (int, int)
Descripción:	Insertar una vista en una zona

Nombre:	InsertZona (string, int, string)
Descripción:	Adicionar una zona física al Sistema.
Nombre:	OpenConnection
Descripción:	Abrir una nueva conexión con la Base de Datos.
Nombre	TaboPostgreProvider (object, ConnectionDataKind, int)
Descripción:	Crea una nueva instancia del proveedor de datos.
Nombre:	UpdateCameraIP (int, string, string, string, string, string, string, string)
Descripción:	Actualizar los datos de una cámara.
Nombre:	UpdateLogicalZone (int, string, string, int)
Descripción:	Actualizar los datos de una zona lógica.
Nombre:	UpdateView (int, string, string, int, int, int, int)
Descripción:	Actualizar los datos de una vista.
Nombre:	UpdateZone (int, string, string, int)
Descripción:	Actualizar los datos de una zona física.

Nombre: MainController	
Tipo de clase : Controladora	
Atributo	**Tipo**
_databaseconfig	DataBaseConfiguration
_workmode	WorkMode
_camerasModel	ISIS_Tabo_Cameras_ServerModel
_finalizationPool	FinalizationPool
_GestorEventsReceiver	_GestorEventsReceiver
_id	int
_modelchannel	IChannel
_port	int
_runningPool	RunningPool
_statCount	int
_statIndex	int

_statLength	int
_statReady	int
_statReceived	long
ServerMode	bool
Para cada responsabilidad:	
Nombre:	AddCamera(Camera, Group)
Descripción:	Adicionar una cámara al Sistema
Nombre:	AddCamerasGroup(Group, Group)
Descripción:	Adicionar una zona física al Sistema.
Nombre:	AddView(View, Group)
Descripción:	Adicionar una vista al Sistema.
Nombre:	AddViewsGroup(Group, Group)
Descripción:	Adicionar una zona lógica al Sistema.
Nombre:	CameraChanged (SIS_CameraRemInfo, bool, int)
Descripción:	Handler para capturar los cambios en las cámaras.
Nombre	CameraGroupChanged (SIS_GroupRemInfo, bool, int)
Descripción:	Handler para capturar los cambios en las zonas físicas.
Nombre	CamerasManager(string, bool)
Descripción:	Crea una nueva instancia del manejador.
Nombre:	CheckCamera(Camera)
Descripción:	Chequear si la cámara existe en el Sistema.
Nombre:	CheckCamerasGroup(Group)
Descripción:	Chequear si la zona física existe en el Sistema.
Nombre:	CheckView(View)
Descripción:	Chequear si la vista existe en el Sistema.
Nombre:	CheckViewsGroup(Group)
Descripción:	Chequear si la zona lógica existe en el Sistema.
Nombre:	CloseAllOpenedCameras
Descripción:	Cierra todas las cámaras abiertas.

Nombre:	CloseCamera(Camera)
Descripción:	Cerrar una cámara.
Nombre:	CloseConnection
Descripción:	Cierra la conexión con el Gestor.
Nombre:	DeleteCamera(Camera)
Descripción:	Eliminar una cámara del Sistema
Nombre:	DeleteCamerasGroup(DeleteCamerasGroup)
Descripción:	Eliminar una zona física del Sistema.
Nombre:	DeleteView(View)
Descripción:	Eliminar una vista del Sistema.
Nombre	DeleteViewsGroup(Group)
Descripción:	Eliminar una zona lógica del Sistema.
Nombre:	GetCameraFromID(int, CameraType)
Descripción:	Encontrar una cámara a partir de su ID.
Nombre:	GetCameraFromName(string, CameraType)
Descripción:	Encontrar una cámara a partir de su nombre.
Nombre:	GetCameraGroupFromID(int, CameraType)
Descripción:	Encontrar una zona física a partir de su ID
Nombre:	GetCameraGroupFromName(string, CameraType)
Descripción:	Encontrar una zona física a partir de su nombre
Nombre:	GetCameras(CameraType)
Descripción:	Devuelve todas las cámaras del Sistema.
Nombre:	GetCamerasGroups(CameraType)
Descripción:	Devuelve todas las zonas físicas en el Sistema.
Nombre:	GetViewFromID(int, CameraType)
Descripción:	Encontrar una vista a partir de su ID.
Nombre	GetViewFromName(string, CameraType)
Descripción:	Encontrar una vista a partir de su nombre.
Nombre:	GetViewGroupFromID(int, CameraType)

Descripción:	Encontrar una zona lógica a partir de su ID.
Nombre:	GetViewGroupFromName(string, CameraType)
Descripción:	Encontrar una zona lógica a partir de su nombre.
Nombre:	GetViews(CameraType)
Descripción:	Devuelve todas las vistas del sistema.
Nombre:	GetViewsGroups(CameraType)
Descripción:	Devuelve todas las zonas lógicas del Sistema.
Nombre:	LoadAllData(string, string, int, RemotingUtils.ChannelProtocol)
Descripción:	Cargar todos los datos desde el Gestor.
Nombre:	LoadCameras
Descripción:	Cargar todas las cámaras del Sistema
Nombre:	LoadCamerasGroups
Descripción:	Cargar todas las zonas físicas del Sistema.
Nombre	LoadSettings(string)
Descripción:	Cargar la configuración
Nombre:	LoadViews
Descripción:	Carga todas las vistas en el Sistema
Nombre:	LoadViewsGroups
Descripción:	Cargar todas las zonas físicas en el Sistema
Nombre:	OpenCamera(Camera)
Descripción:	Abrir una cámara y adicionarla a la piscina de cámaras activas.
Nombre:	OpenCamera(string, CameraType)
Descripción:	Abrir una cámara según su nombre
Nombre:	OpenPort(int, bool)
Descripción:	Abrir un puerto para la comunicación con el Gestor.
Nombre:	OpenPort(bool)
Descripción:	Abrir un puerto para la comunicación con el Gestor.
Nombre:	OpenView(View)
Descripción:	Abrir una vista

Nombre	OpenView(string, CameraType)
Descripción:	Abrir una vista
Nombre:	RegisterWithServer
Descripción:	Registrarse con el Gestor como Cliente.
Nombre:	SaveCameras(CameraType)
Descripción:	Salva todas las cámaras del sistema.
Nombre:	SaveViews(CameraType)
Descripción:	Salva todas las vistas.
Nombre:	UpdateCamera(Camera, Camera)
Descripción:	Actualizar una cámara
Nombre:	UpdateCamerasGroup(Group, Group)
Descripción:	Actualizar una zona física.
Nombre:	UpdateView(View, View)
Descripción:	Actualizar una vista.
Nombre:	UpdateViewsGroup(Group, Group)
Descripción:	Actualizar una zona lógica.
Nombre:	ViewChanged (SIS_ViewRemInfo, bool, int)
Descripción:	Handler para capturar los cambios en las vistas del Sistema.
Nombre:	ViewGroupChanged (SIS_GroupRemInfo, bool, int)
Descripción:	Handler para capturar los cambios en las zonas lógicas del Sistema.

Nombre: GroupCollection	
Tipo de clase : Entidad	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	GetGroup(string, Group)
Descripción:	Devuelve un grupo de la colección
Nombre:	GetGroupByName(string)

Descripción:	Encuentra un grupo según su nombre
Nombre:	GetGroupFromID(int)
Descripción:	Encuentra un grupo segun su id.
Nombre:	GroupCollection()
Descripción:	Crea una nueva instancia de la colección.

Nombre: ViewCollection	
Tipo de clase : Entidad	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	GetView(string, Group)
Descripción:	Devuelve una vista según su nombre
Nombre:	GetView(int)
Descripción:	Devuelve una vista según su id.
Nombre:	ViewCollection()
Descripción:	Crea una nueva instancia de la colección.

Nombre: CameraCollection	
Tipo de clase : Entidad	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	CameraCollection()
Descripción:	Crea una nueva instancia de la colección.
Nombre:	GetCamera(string, Group)
Descripción:	Devuelve una cámara según su nombre.
Nombre:	GetCamera(int)
Descripción:	Devuelve una cámara según su id.

Nombre: VideoProvider	
Tipo de clase: Entidad	
Atributo	**Tipo**
sourceDesc	IVideoSourceDescription
Para cada responsabilidad:	
Nombre:	CompareTo(object)
Descripción:	Compara el provider con un objeto.
Nombre:	CreateVideoSource(object)
Descripción:	Crea una nueva fuente de video.
Nombre:	GetSettingsPage()
Descripción:	Devuelve la pagina de configuración del provider
Nombre:	LoadConfiguration(XmlTextReader)
Descripción:	Carga la configuración del provider desde un XML.
Nombre:	LoadVideoSettings (XmlTextReader)
Descripción:	Carga la configuración del provider desde un XML.
Nombre:	SaveConfiguration(XmlTextWriter, object)
Descripción:	Salva la configuración del provider a un XML.
Nombre:	SaveVideoSettings(XmlTextWriter, object)
Descripción:	Salva la configuración del provider a un XML.
Nombre:	VideoProvider(IVideoSourceDescription)
Descripción:	Crea una nueva instancia del provider.

Nombre: VideoProviderCollection	
Tipo de clase: Entidad	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	GetProviderByName(string)

Descripción:	Encuentra un provider según su nombre.
Nombre:	Load(string)
Descripción:	Carga los provider desde un directorio.
Nombre:	LoadAssembly(string)
Descripción:	Carga los providers de un DLL.
Nombre:	VideoProviderCollection()
Descripción:	Crea una nueva instancia de la colección.

Nombre: IVideoSourceDescription	
Tipo de clase: Interfaz	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	CreateVideoSource(object)
Descripción:	Crea una nueva fuente de video.
Nombre:	GetSettingsPage()
Descripción:	Devuelve la pagina de configuración.
Nombre:	LoadConfiguration(XmlTextReader)
Descripción:	Carga la configuración de un XML
Nombre:	LoadVideoSettings(XmlTextReader)
Descripción:	Carga la configuración de un XML
Nombre:	SaveConfiguration(XmlTextWriter, object)
Descripción:	Salva la configuración a un XML
Nombre:	SaveVideoSettings(XmlTextWriter, object)
Descripción:	Slva la configuración a un XML

Nombre: IVideoSource	
Tipo de clase: Interfaz	
Atributo	**Tipo**

Para cada responsabilidad:	
Nombre:	ChangeVideoSettings(object)
Descripción:	Cambia la configuración de video.
Nombre:	GetVideoSettings ()
Descripción:	Devuelve la configuración de video.
Nombre:	GetVideoSettingsPage ()
Descripción:	Devuelve la pagina de configuración de video.
Nombre:	NewFrame()
Descripción:	Avisa cuando ha llegado un nuevo frame,
Nombre:	SignalToStop()
Descripción:	Marca una señal de parada a la fuente de video.
Nombre:	Start()
Descripción:	Comienza a obtener video.
Nombre:	Stop()
Descripción:	Detiene la captura de video.
Nombre:	VideoSettingsChange()
Descripción:	Avisa cuando ha cambiado la configuración de video.
Nombre:	WaitForStop()
Descripción:	Bloquea el hilo actual hasta que se detienen la captura de video.

Nombre: DataBaseConfiguration	
Tipo de clase: Controladora	
Atributo	**Tipo**
_AutomaticAccept	bool
_Provider	TaboPostgreProvider
_videoproviders	VideoProviderCollection
Cameras	CameraCollection
camerasGroups	GroupCollection

views	ViewCollection
viewsGroups	GroupCollection
Para cada responsabilidad:	
Nombre:	AcceptChanges(bool)
Descripción:	Acepta todos los cambios hechos en la Base de Datos..
Nombre:	AddCamera(SIS_Tabo.VideoController.VideoTools.Camera)
Descripción:	Adiciona una cámara al sistema.
Nombre:	AddCamerasGroup(SIS_Tabo.VideoController.VideoTools.Group)
Descripción:	Adiciona una zona física al sistema.
Nombre:	AddView(SIS_Tabo.VideoController.VideoTools.View)
Descripción:	Adicionar una vista al sistema.
Nombre:	AddViewsGroup(SIS_Tabo.VideoController.VideoTools.Group)
Descripción:	Adicionas una zona lógica al sistema.
Nombre:	CameraNeedsUpdate(Camera)
Descripción:	Avisa que una cámara necesita actualización.
Nombre:	CamerasInGroup(SystemGroup)
Descripción:	Devuelve si una cámara esta en un grupo.
Nombre:	CheckCamera(SIS_Tabo.VideoController.VideoTools.Camera)
Descripción:	Chequea que existe una cámara determinada.
Nombre:	CheckCamerasGroup(SIS_Tabo.VideoController.VideoTools.Group)
Descripción:	Chequea que existe una zona física determinada,
Nombre:	CheckView(SIS_Tabo.VideoController.VideoTools.View)
Descripción:	Chequea que existe una vista determinada.
Nombre:	CheckViewsGroup(SIS_Tabo.VideoController.VideoTools.Group)
Descripción:	Chequea que existe una zona lógica determinada.
Nombre:	DataBaseConfiguration(string)
Descripción:	Crea una nueva instancia de la clase.
Nombre:	DeleteCamera(SIS_Tabo.VideoController.VideoTools.Camera)
Descripción:	Elimina una cámara.

Nombre:	DeleteCamerasGroup(SIS_Tabo.VideoController.VideoTools.Group)
Descripción:	Elimina una zona.
Nombre:	DeleteView(SIS_Tabo.VideoController.VideoTools.View)
Descripción:	Elimina una vista.
Nombre:	DeleteViewsGroup(SIS_Tabo.VideoController.VideoTools.Group)
Descripción:	Elimina una zona lógica.
Nombre:	GetCameraByName(string)
Descripción:	Devuelve una cámara según su nombre.
Nombre:	GetCameraConfig(SystemCamera)
Descripción:	Devuelve la configuración de una cámara.
Nombre:	GetCameraFromID(int)
Descripción:	Devuelve una cámara según su id.
Nombre:	GetCamerasGroupByName(string)
Descripción:	Devuelve una zona según su nombre.
Nombre:	GetCameraVideoSettings(SystemCamera)
Descripción:	Devuelve la configuración de video de una cámara.
Nombre:	GetViewByName(string)
Descripción:	Devuelve una vista según su nombre.
Nombre:	GetViewsGroupByName(string)
Descripción:	Devuelve una zona lógica según su nombre.
Nombre:	IsCamerasGroupEmpty(SystemGroup)
Descripción:	Devuelve si una zona esta vacia.
Nombre:	IsViewsGroupEmpty(SystemGroup)
Descripción:	Devuelve si una zona lógica esta vacia.
Nombre:	LoadCameras()
Descripción:	Carga todas las cámaras del sistema.
Nombre:	LoadCameras(System.Xml.XmlTextReader, SIS_Tabo.VideoController.VideoTools.Group)
Descripción:	Carga las cámaras del sistema desde un XML.

Nombre:	LoadCamerasInView(SystemView)
Descripción:	Caarga todas las cámaras de una vista.
Nombre:	LoadSettings()
Descripción:	Carga toda la configuración.
Nombre:	LoadViews()
Descripción:	Carga todas las vistas.
Nombre:	SaveCameras()
Descripción:	Salva las cámaras del sistema.
Nombre:	SaveSettings()
Descripción:	Salva toda la configuración.
Nombre:	SaveViews()
Descripción:	Salva las vistas.
Nombre:	SetCameraConfig(string, SystemCamera)
Descripción:	Cambia la configuración de una camara.
Nombre:	SetCameraVideoSettings(SystemCamera, string)
Descripción:	Cambia la configuración de video de una cámara.
Nombre:	TryAcceptChanges()
Descripción:	Trata de guardar los cambios realizados en la base de datos.
Nombre:	UpdateCamera(Camera)
Descripción:	Actualiza un cámara determinada.
Nombre:	UpdateCamerasGroup(Group)
Descripción:	Actualiza una zona determinada.
Nombre:	UpdateView(View)
Descripción:	Actualiza una vista.
Nombre:	UpdateViewsGroup(Group)
Descripción:	Actualiza una zona lógica.

Nombre: TaboPostgreConnectionManager
Tipo de clase: Controladora

Atributo	**Tipo**
_CurrentState	ManagerState
_Transaction	NpgsqlTransaction
ConData	PostgreConnectionData
Connection	NpgsqlConnection
Para cada responsabilidad:	
Nombre:	CloseConnection()
Descripción:	Cierra la conexión con la base de datos.
Nombre:	CommitTransaction(bool)
Descripción:	Finaliza una transacción.
Nombre:	OpenConnection()
Descripción:	Abre una conexión.
Nombre:	RollBackTransaction()
Descripción:	Deshace una transacción.
Nombre:	StartTransaction(IsolationLevel)
Descripción:	Comienza una transacción.
Nombre:	TaboPostgreConnectionManager(object, ConnectionDataKind, int)
Descripción:	Crea un nuevo manager.
Nombre:	TestConnection(NpgsqlConnection, int)
Descripción:	Cheque que la conexión este viva.
Nombre:	WaitForAConnection(NpgsqlConnection, int)
Descripción:	Se bloquea hasta que se abra la conexión.

Nombre: TaboPostgreSpecificAdapter	
Tipo de clase: Controladora	
Atributo	**Tipo**
_Connection	NpgsqlConnection
_Transaction	NpgsqlTransaction
Para cada responsabilidad:	

Nombre:	Delete(object[])
Descripción:	Ejecuta los scripts de eliminar.
Nombre:	Insert(object[])
Descripción:	Ejecuta los scripts de Insertar.
Nombre:	Select(object[])
Descripción:	Ejecuta los scripts de seleccionar.
Nombre:	SetConnection (NpgsqlTransaction)
Descripción:	Cambia la conexión.
Nombre:	TaboPostgreSpecificAdapter(NpgsqlTransaction)
Descripción:	Crea una nueva instancia del adaptador.
Nombre:	Update(object[])
Descripción:	Ejecuta los scripts de actualizar.

Nombre: postgreConnectionData	
Tipo de clase: Entidad	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	GetFromConnection(NpgsqlConnection)
Descripción:	Obtiene los parámetros de una conexión.
Nombre:	ParseConnectionString(string)
Descripción:	Obtiene los parámetros desde una cadena de conexión.

Nombre: CamerasTreeView	
Tipo de clase: Interfaz	
Atributo	**Tipo**
_ContentType	CameraType
_DropManager	IOrganizerDropManager
cameraImage	int

cameraSelectedImage	int
camerasFolderImage	int
camerasFolderSelectedImage	int
camerasOnly	bool
camerasRootNode	CamerasTreeViewNode
lastClickNode	CamerasTreeViewNode
TreeContextMenu	ContextMenuStrip
viewImage	int
viewSelectedImage	int
viewsFolderImage	int
viewsFolderSelectedImage	int
viewsRootNode	CamerasTreeViewNode

Para cada responsabilidad:	
Nombre:	AddCamera(Camera, TreeNode)
Descripción:	Adiciona una nueva cámara al árbol.
Nombre:	AddCamera(Camera, OrganizerNode)
Descripción:	Adiciona una nueva cámara al árbol.
Nombre:	AddCameraClick()
Descripción:	Captura cuando se ordena adicionar una nueva cámara.
Nombre:	AddCameraGroupClick()
Descripción:	Captura cuando se ordena adicionar una nueva zona física.
Nombre:	AddCamerasGroup(Group, TreeNode)
Descripción:	Adicionar una nueva zona al árbol.
Nombre:	AddCamerasGroup(Group, OrganizerNode)
Descripción:	Adicionar una nueva zona al árbol.
Nombre:	AddView(view, TreeNode)
Descripción:	Adicionar una nueva vista al árbol.
Nombre:	AddView(view, OrganizerNode)
Descripción:	Adicionar una nueva vista al árbol.

Nombre:	AddViewClick()
Descripción:	Captura cuando se ordena adicionar una nueva vista.
Nombre:	AddViewGroupClick()
Descripción:	Captura cuando se ordena adicionar una nueva zona lógica.
Nombre:	AddViewsGroup(Group, TreeNode)
Descripción:	Adicionar una nueva zona lógica al árbol.
Nombre:	AddViewsGroup(Group, OrganizerNode)
Descripción:	Adicionar una nueva zona lógica al árbol.–
Nombre:	BuildCamerasTree(GroupCollection, CameraCollection)
Descripción:	Construir el árbol de las cámaras.
Nombre:	BuildCamerasTree(CameraCollection, Group, TreeNode)
Descripción:	Construir el árbol de las cámaras.
Nombre:	BuildViewsTree(GroupCollection, ViewCollection)
Descripción:	Construir el árbol de las vistas.
Nombre:	BuildViewsTree(GroupCollection, ViewCollection, TreeNode, Group)
Descripción:	Construir el árbol de las vistas.
Nombre:	CamerasTreeView()
Descripción:	Crea una nueva instancia del árbol.
Nombre:	CamerasTreeView_DoubleClick(object, EventArgs)
Descripción:	Capturar los doble click.
Nombre:	CamerasTreeView_DragDrop(object, DragEventArgs)
Descripción:	Capturar los soltar.
Nombre:	CamerasTreeView_DragEnter(object, DragEventArgs)
Descripción:	Capturar la entrada de elementos arrastrados.
Nombre:	CamerasTreeView_DragOver(object, DragEventArgs)
Descripción:	Captura el movimiento de elementos arrastrados.
Nombre:	CamerasTreeView_ItemDrag(object, ItemDragEventArgs)
Descripción:	Captura el movimiento de elementos arrastrados.
Nombre:	ConfigStart()

Descripción:	Configurar el árbol para comenzar a trabajar.
Nombre:	ContainsNode(TreeNode, TreeNode)
Descripción:	Devuelve si el árbol contiene un nodo determinado.
Nombre:	GetCameraFullName(TreeNode)
Descripción:	Devuelve el nombre completo de una cámara.
Nombre:	GetGroupFullName(TreeNode)
Descripción:	Devuelve el nombre completo de una zona física.
Nombre:	GetNodeFor(Group, bool)
Descripción:	Devuelve el nodo de una zona.
Nombre:	GetNodeFor(Camera)
Descripción:	Devuelve el nodo de una cámara.
Nombre:	GetNodeFor(View)
Descripción:	Devuelve el nodo de una vista.
Nombre:	GetNodeType(TreeNode)
Descripción:	Devuelve el tipo de un nodo.
Nombre:	GetViewFullName(TreeNode)
Descripción:	Devuelve el nombre completo de una vista.
Nombre:	Search(string, NodeType, CamerasTreeViewNode)
Descripción:	Busca un elemento por su nombre.
Nombre:	UpdateCameraGroupNode(Group, Group)
Descripción:	Actualiza el nodo de una zona.
Nombre:	UpdateCameraNode(Camera, Camera)
Descripción:	Actualiza el nodo de una cámara.
Nombre:	UpdateViewGroupNode(Group, Group)
Descripción:	Actualiza el nodo de una zona lógica.

Nombre: Visor_de_cámaras	
Tipo de clase : interfaz	
Atributo	**Tipo**

_closesilence	bool
_Content	Dictionary<CameraWindow, visor_camera>
_SelectedCamera	Camera
Para cada responsabilidad:	
Nombre:	AddCamera(Camera)
Descripción:	Adicionar una cámara al visor.
Nombre:	CloseRequested(visor_camera)
Descripción:	Captura las peticiones de cerrar una cámara.
Nombre:	DCloseCamera(Visor_de_cámaras, Camera)
Descripción:	Manda a cerrar una cámara.
Nombre:	DSelectedCameraChanged(Visor_de_cámaras, Camera)
Descripción:	Captura cuando cambia la cámara seleccionada.
Nombre:	DVisorClosing(List<Camera>)
Descripción:	Cierra las cámaras del visor cuando este se esta cerrando.
Nombre:	FullScreenRequested(visor_camera)
Descripción:	Captura las peticiones de pantalla completa.
Nombre:	SizeAdjustRequested(visor_camera)
Descripción:	Captura las peticiones de ajuste de tamaño

Nombre: visor_externo	
Tipo de clase: interfaz	
Atributo	**Tipo**
camera	visor_camera
container	VisorContenedor
Para cada responsabilidad:	
Nombre:	entrar_click(object, EventArgs)
Descripción:	Entra el visor al visualizar activo.
Nombre:	visor_externo(visor_camera, VisorContenedor)
Descripción:	Crea una nueva instancia del control.

Nombre:	Insertar()
Descripción:	Insertar un nuevo visor en el Contenerdor

Anexo IV Descripción de las clases del Modelo de Datos.

Ejemplo

Nombre: tbl_view_in_zone		
Descripción:		
Atributo	**Tipo**	**Descripción**
id_view	integer	ID de la vista
id_zona	integer	ID de la zona en que esta la vista

Nombre: tbl_provider_roles		
Descripción:		
Atributo	**Tipo**	**Descripción**
rolename	varchar(255)	Nombre del rol
applicationname	varchar(255)	Nombre de la aplicación

Nombre: tbl_provider_users		
Descripción:		
Atributo	**Tipo**	**Descripción**
pkid	char(40)	
username	varchar(255)	
applicationname	varchar(255)	
email	varchar(128)	
comment	varchar(255)	
password	varchar(255)	
passwordsalt	varchar(128)	
passwordquestion	varchar(255)	
passwordanswer	varchar(255)	
isapproved	boolean	
lastactivitydate	time	
lastlogindate	time	

lastpasswordchangedate	time	
creationdate	time	
islockedout	boolean	
lastlockedoutdate	time	
failedpasswordattemptcount	integer	
failedpasswordattemptwindowstart	time	
failedpasswordanswerattemptcount	integer	
failedpasswordanswerattemptwindowstart	time	

Nombre: tbl_provider_usersinroles		
Descripción:		
Atributo	**Tipo**	**Descripción**
username	varchar(255)	
rolename	varchar(255)	
applicationname	varchar(255)	

Nombre: tbl_camerainzone		
Descripción:		
Atributo	**Tipo**	**Descripción**
id_camara	integer	
id_zona	integer	

Nombre: tbl_view_in_zone		
Descripción:		
Atributo	**Tipo**	**Descripción**
id_view	integer	
id_zona	integer	

Nombre: tbl_eventlog

Descripción:		
Atributo	**Tipo**	**Descripción**
id_log	integer	
descripcion		
type	varchar(30)	
eventdate	time	
fromwho	varchar	

Nombre: tbl_camaraip		
Descripción:		
Atributo	**Tipo**	**Descripción**
id_camara	integer	
tipo_camara		
fabricacion	varchar(50)	
name_cam	varchar(80)	
ip	varchar(255)	
provider	varchar(100)	
configprovider	varchar(1000)	
descripcion	varchar(255)	
specificinfo	varchar(1000)	

Nombre: tbl_cameraslog		
Descripción:		
Atributo	**Tipo**	**Descripción**
id_log	integer	
descripcion		
type	varchar(30)	
eventdate	time	
fromwho	varchar	

objecttype	varchar	

Nombre: tbl_securitylog		
Descripción:		
Atributo	**Tipo**	**Descripción**
id_log	integer	
descripcion		
type	varchar(30)	
eventdate	time	
fromwho	varchar	