UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 6

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

# Título: Identificador de cámaras IP en una red de datos para el sistema de video vigilancia Suria en su versión Qt.

**Autores:** Katery De Paz Bermúdez

Rainer Alberto Sánchez Ruiz

**Tutores:** Merlín Milián Díaz

Fernando Echemendía Tour

La Habana, Junio de 2014

"Año 56 de la Revolución"

# *Declaración de Autoría*

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

Para que así conste firmamos la presente a los ____ días del mes de _____del año_____.

**Katery De Paz Bermúdez**

_____
Firma del Autor

**Rainer Alberto Sánchez Ruiz**

_____
Firma del Autor

**Ing. Merlín Milián Díaz**

_____
Firma del Tutor

**Ing. Fernando Echemendía Tour**

_____
Firma del Tutor

## Datos de Contacto

**Ing. Merlín Milián Díaz**

Graduada de la Universidad de las Ciencias Informáticas (UCI) en el año 2011. Pertenece al departamento Señales Digitales del Centro de Geoinformática y Señales Digitales (GEYSED) de la Facultad 6. Labora como Asesora de Planificación y Control de dicho Centro.
Correo electrónico: mmilian@correo.uci.cu

**Ing. Fernando Echemendía Tour**

Graduado de la Universidad de las Ciencias Informáticas (UCI) en el año 2008. Pertenece al departamento Señales Digitales del Centro de Geoinformática y Señales Digitales (GEYSED) de la Facultad 6. Labora como Programador de Video Sensores para el sistema de Video Vigilancia Xilema Suria.
Correo electrónico: fechemendia@correo.uci.cu

## Dedicatoria

*A mi Padre Celestial, Mi Gran Señor, El Dios Altísimo, El Creador Todopoderoso, El Dios eterno, El Todo Suficiente, quien ha sido mi todo en todo, mi sustentador, mi amparo y fortaleza; y a mi Madre Elizabeth Bermúdez Pazo mi mayor inspiración. A ambos por su amor incomparable y por su apoyo.*

*Kathery De Paz Bermúdez*

*Quiero dedicar esta tesis a mis familiares más cercanos en especial a mi abuelo Bernardo que desde algún lugar sé que ahora me está mirando, el me enseñó gran parte de lo que sé hacer y contribuyó en gran medida a formar la persona que soy hoy en día, a mi madre Tyamiles por ser todo para mí, por querer siempre lo mejor para mí y confiar en mí en todo momento, a mi abuela Dinorah por permitirme ser su chulo durante los 25 años que ya tengo y a mi padre Alberto por ser un amigo fiel en el que siempre he podido confiar. A todos ustedes va dedicado este trabajo.*

*Rainer Alberto Sánchez Ruíz*

# Agradecimientos

*A mi Señor que me ha guiado por este camino, y ha sostenido cada uno de mis pasos, quien me ha dado fuerzas cuando he sentido que desfallezco, en quien he aprendido a confiar y dependo completamente.*

*A mi mamita por serlo todo para mí, mi mejor amiga, mi hermana, mi confidente. Eres mi ejemplo de perseverancia y de Fe, grande es tu Fe y serás recompensada por ello. Tú me empujaste hasta aquí, a pesar del amor protector dejaste que Dios tomara las riendas de mi vida y te lo agradezco. Gracias por ser mi apoyo en estos años, por sostener mis brazos cuando sentía que desfallecía, por darme aliento para continuar, por cada una de las oraciones, de las noches de desvelo y las lágrimas hoy puedes disfrutar las primicias, yo te amo mama. A mi Kevin, quien fue la lluvia tardía de nuestras vidas pero llegó cuando más lo necesitábamos, el que más ha sufrido mi ausencia y que aun tan pequeño me ama tanto que su amor me da fuerzas cada mañana para esforzarme un poquito más. A Emilio quien ha cuidado de mi mama como a un tesoro y le estaré agradecida siempre, eres un ejemplo de padre, tu lograste cambiar mi concepto de padre*

*A mis padres espirituales Yisel y Jose.*

*A mis abuelos gracias por enseñarnos sobre la Fe, yo estaría perdida si no me hubieran educado en el buen camino, si no me hubieran enseñado sobre la verdad, sobre Jesús pues gracias a eso hoy puedo decir soy Libre, yo sé que por sus oraciones estoy parada hoy aquí, y aun cuando mami no me puede ver cuando nos veamos en el cielo le puedo decir no fue en vano…Gracias*

*A mi familia tan linda y unida, su apoyo significo todo para mí, siempre regresar a casa es bueno sabiendo que sin importar que tengas una posición social o un reconocimiento a nivel mundial siempre te reciben con el mejor abrazo con el mejor beso.*

*A mi papa Nelson por malcriarme tanto en estos años, gracias por tus detalles y por tu cariño*

*A Milo, Zaida y Yasel, Ronny, Pori, Marcos, Mariela y Handy, Roberto, Alí y Alayo*

*A mis favoritos Diana y Jose, Mariam y Alejandro, Darayne y Flabia, Yasmín y Daniel Raúl y Eliecer.*

*A mis hermanos en general, no importa a donde Dios me lleve siempre enseñaré las cosas que he aprendido en este pueblo.*

*A mis amigos del grupo Ariel, Yoilán y Yosvanis.*

*A mi compañero de tesis.*

*A mi tutora Merlín.*

*A mi tribunal en especial al profesor Asnay.*

*Kathery*

# Agradecimientos

*Primero que nada quiero agradecer a mi familia por ser ejemplo y guía para mí, por la educación que recibí de ellos, inculcándome valores y principios que estoy orgulloso de haber aprendido de ellos.*

*A mis amigos de infancia que son mis hermanos y a los que he conocido en la UCI que de una forma u otra forman hoy también parte de mi vida.*

*Quiero agradecer a Dios por permitir que mi compañera de tesis y yo lográramos entendernos y salir adelante, y a la vez agradecerle a ella por confiar en mí cuando las cosas se ponían difícil.*

*Agradecer también a todos los profesores y compañeros de aula que he tenido a lo largo de estos 5 años.*

*A los miembros del tribunal por su apoyo y paciencia.*

*A todos los que de alguna manera hicieron posible la culminación de este trabajo de diploma.*

*Por último agradecer también a todas las personas que conocí en la universidad por ayudarme de manera directa o indirecta durante mi estadía en esta gran casa que fue para mí la UCI.*

*Rainer*

Debido al avance tecnológico, se han ido perfeccionando los métodos y formas que permitan garantizar la seguridad con la mayor eficiencia posible. Entre estos métodos se encuentra la video vigilancia IP, la cual se ha posicionado como una alternativa muy fuerte en el mercado, pues permite adaptarse a medios de transmisión más eficientes y flexibles. En la Universidad de Ciencias Informáticas (UCI), se encuentra en proceso de desarrollo el Sistema de Video Vigilancia Suria. Este sistema no cuenta con la funcionalidad de identificar automáticamente las cámaras IP conectadas en la red de datos.

La presente investigación tiene como objetivo la creación de una aplicación basada en componentes encargada de la identificación automática de cámaras IP, que sea capaz de identificarlas y configurarlas. Para la construcción de esta solución han sido utilizados RUP como metodología de desarrollo, UML como lenguaje de modelado, como lenguaje de Programación C++, así como el marco de trabajo Qt. El resultado de esta investigación es una aplicación que permita la identificación de cámaras IP en una red de Datos. Posibilita, además, configurar las cámaras y modificar los datos de configuración como el usuario, la contraseña, modelo, fabricante y dirección IP.

**Palabras Clave:** Cámaras IP, red de datos, video vigilancia

Due to technological advances, have been refined methods and ways that can ensure security with the greatest possible efficiency. These methods find the IP video surveillance, which has positioned itself as a strong alternative in the market, because it fits in to more efficient and flexible ways of transmission. At the University of Informatics Sciences (UIS) is being developed Suria Video Surveillance System, but this system does not have the functionality to automatically identify the IP cameras connected to the data network.

This research aims to create a component-based application, responsible for automatically identifying IP cameras, which is able to identify and configure them. For the construction of this solution have been used RUP as development methodology, UML as modeling language, such as C++ as programming language and Qt as framework. The result of this research is an application that allows the identification of IP cameras in a data network. Also allows configuring the cameras and changing the configuration of data such as the user name, password, pattern maker and IP address.

**Keywords:** Local area network, IP cameras, video surveillance.

**ÍNDICE**

**ÍNDICE DE TABLAS**

## ÍNDICE DE FIGURAS

# Introducción

La seguridad física representa un aspecto importante para las necesidades sociales. A medida que ha ido evolucionando, el hombre ha utilizado diversos instrumentos y alternativas que permitan salvaguardar su vida y sus posesiones, desde puertas y cerraduras hasta guardias de seguridad. Debido al avance de las tecnologías, se han perfeccionado los métodos y formas de garantizar la seguridad con la mayor eficiencia posible. Entre estos métodos se encuentra la video vigilancia, la cual no es más que el empleo de video en el monitoreo de una o varias localizaciones desde un sitio remoto (Mora Saavedra, 2011).

Los sistemas de video vigilancia en sus inicios fueron totalmente análogos y han ido progresivamente digitalizándose; sus comienzos se vieron marcados por los videograbadores que contaban con el cassette como medio de almacenamiento. Este permitía grabar unas horas, lo cual traía como consecuencia, la necesidad de almacenar cierta cantidad de cinta para dar respaldo al sistema, además de volver compleja la búsqueda de información específica en las mismas (Sandoval N, 2010). La video vigilancia ha tenido una evolución acelerada en la última década, impulsada por el desarrollo tecnológico y una preocupación creciente por la seguridad pública y privada. Gracias al desarrollo de las redes y el video digital, los nuevos sistemas hacen uso de cámaras basadas en tecnología IP[1], los cuales se han posicionado como una alternativa muy fuerte en el mercado, pues permiten adaptarse a medios de transmisión más eficientes y flexibles. Estos sistemas utilizan cámaras IP y servidores para la grabación de video en un sistema completamente digitalizado.

Los sistemas de video vigilancia basados en tecnología IP ofrecen toda una serie de ventajas y funcionalidades avanzadas, que no puede proporcionar un sistema de video vigilancia analógica. Entre las ventajas se incluyen la accesibilidad remota, la alta calidad de la imagen, la gestión de eventos, así como las posibilidades de una integración sencilla con otras funciones (Mora Saavedra, 2011).

Cuba, a pesar de las limitaciones económicas que enfrenta debido al bloqueo impuesto durante décadas, no se ha mantenido al margen de los logros alcanzados con los sistemas de video vigilancia y ha invertido gradualmente en el desarrollo de la informatización de la sociedad. A raíz de este propósito se crea la Universidad de las Ciencias Informáticas (UCI), un centro de

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

altos estudios concebido para preparar y formar profesionales que aporten con sus conocimientos al desarrollo de la sociedad y a la economía del país.

En la UCI se desarrollan diferentes proyectos en convenio con varias empresas en todas las ramas de la sociedad, con el objetivo de generar sistemas informáticos que sean de creación nacional y respondan a las necesidades tanto de clientes nacionales como internacionales. Específicamente en la Facultad 6**,** el departamento de Señales Digitales, cuenta con un proyecto dedicado al desarrollo de un sistema para la video vigilancia empleando cámaras IP, que incorpora todas las prestaciones propias de estos sistemas y pretende adaptarse fácilmente a los entornos corporativos en los que se desee utilizar, lo que lo convierte en un sistema efectivo y escalable. Hoy en día, el sistema agrupa las funcionalidades tradicionales de un sistema de seguridad, además de elementos innovadores para el análisis automático en los flujos de video.

En la actualidad, para que el sistema de video vigilancia Suria en su versión Qt, sea capaz de controlar las cámaras IP conectadas en la red de datos, es necesario que el administrador del sistema inserte manualmente cierta información como la marca, el modelo y dirección IP de las cámaras. Para configurarlas se debe ir hacia donde está instalada cada una y buscar el IP donde está conectada e investigar el modelo y fabricante, pues dependiendo de esto es que se puede trabajar con ella y muchas veces es necesario hasta reiniciarla y comenzar todo desde el principio. En despliegues realizados, se ha identificado que cuando el número de dispositivos conectados es numeroso, se genera gran lentitud y demora en el proceso.

La situación planteada anteriormente lleva a formular el siguiente **problema a resolver:** La identificación de las cámaras IP en el sistema de Video Vigilancia Suria, solo se logra introduciendo los datos de las mismas durante la instalación del sistema; Definiendo como **objeto de estudio** la identificación de cámaras IP en una red de datos; Delimitando como **campo de acción** la identificación de cámaras IP en una red de datos para el sistema de video vigilancia Suria en su versión Qt. Para dar solución al problema detectado se propone como **objetivo general:** Desarrollar una aplicación capaz de identificar automáticamente cámaras IP en una red de datos para el sistema de video vigilancia Suria en su versión Qt.

Para guiar la presente investigación se definen las siguientes **preguntas científicas:**
¿Cuáles son los elementos teóricos que sustentan la investigación?
¿Qué elementos intervienen en el proceso de diseño del identificador de cámaras para el sistema de video vigilancia Suria Qt?

¿Cuáles son las pruebas de software que permiten validar la solución propuesta?

Para lograr el objetivo planteado anteriormente se han establecido las tareas de investigación que se muestran a continuación:

1. Caracterizar los procesos relacionados con la identificación de las cámaras IP en una red de datos.

2. Definir las herramientas, tecnologías y metodología seleccionadas para el desarrollo de la aplicación.

3. Realizar el diseño de la aplicación para el sistema de video vigilancia Suria versión Qt.

4. Implementar un identificador de cámaras IP en una red de datos para el sistema de video vigilancia Suria versión Qt.

5. Realizar pruebas de caja negra para validar que la aplicación cumpla con las especificaciones definidas.

Los **métodos científicos** que se utilizan con el fin de dar cumplimiento a los objetivos de la investigación son:

**Métodos Teóricos**:
Permiten estudiar las características del objeto de investigación que no son observables directamente, facilitan la construcción de modelos e hipótesis de investigación y crean las condiciones para ir más allá de las características fenomenológicas y superficiales de la realidad, contribuyendo al desarrollo de las teorías científicas y para su ejecución se apoyan en el proceso de análisis y síntesis (Hernández León, y otros, 2011).

Según (Hernández León, y otros, 2011) El **Análisis histórico-lógico:** *"analiza la trayectoria completa del fenómeno, su condicionamiento a los diferentes periodos de la historia, revela las etapas principales de su desenvolvimiento y las conexiones históricas fundamentales. Expresa en forma teórica la esencia del objeto, explica la historia de su desarrollo, reproduce el objeto en su forma superior y permite unir el estudio de la estructura del objeto de investigación con su concepción histórica."* Este método se utiliza para realizar un estudio sobre los antecedentes de las cámaras IP, obteniendo un conocimiento histórico de forma cronológica desde su surgimiento hasta la actualidad.

Según (Hernández León, y otros, 2011) **Analítico-Sintético:** "*Permite la división mental del fenómeno en sus múltiples relaciones y componentes para facilitar su estudio. Establece mentalmente la unión entre las partes previamente analizadas, posibilita descubrir sus características generales y las relaciones esenciales entre ellas.*" La utilización de este método permitió analizar la bibliografía consultada, para efectuar una correcta investigación sobre los elementos que se relacionan con el objeto de estudio.

**Técnica de recopilación de información:**
Se utiliza la técnica de **Entrevista:** se emplea con el objetivo de conocer el funcionamiento del sistema de video vigilancia Suria en su versión Qt e identificar los requisitos funcionales y no funcionales de la aplicación a desarrollar. Para ello en la investigación se toma como muestra un desarrollador del sistema y el líder de proyecto. (Ver Anexo 1).

La investigación está estructurada en los siguientes capítulos:
**Capítulo 1: Fundamentación teórica:** En este capítulo se realiza una explicación de todos los aspectos teóricos y técnicos de importancia para la investigación; se realiza una síntesis del estado actual de estos sistemas en el mundo, así como la experiencia que se tiene sobre el tema en Cuba. También se explican las herramientas y tecnologías propuestas a utilizar.

**Capítulo 2: Características del sistema:** En este capítulo se define el modelo del dominio de la aplicación que se va a desarrollar y se describen las clases y flujos del diagrama modelo del dominio. También se identifican los requisitos funcionales y no funcionales y se definen los casos de usos y actores de la aplicación.

**Capítulo 3: Análisis y Diseño del Sistema:** En este capítulo se define la arquitectura base del Identificador de cámaras IP, mostrándose los artefactos generados en esta fase, así como los diagramas que fueron necesarios para obtener una mayor claridad a la hora de elaborar la solución que se propone, tales como el diagrama de clases del diseño.

**Capítulo 4: Implementación y prueba:** En este capítulo se aborda el tema referente a la construcción del Identificador de cámaras IP, para el sistema Video Vigilancia Suria en su versión Qt, que permitirá configurar automáticamente los datos de las cámaras. Se realiza además una descripción de las pruebas realizadas a la aplicación y los resultados alcanzados.

# Capítulo I: Fundamentación teórica

## Introducción

En este capítulo se tiene como objetivo realizar una introducción al tema de la investigación, haciendo referencia a un conjunto de conceptos asociados al dominio del problema facilitando así el rápido entendimiento del tema y logrando una mayor familiarización con los términos más comunes. Se realiza un estudio acerca de los sistemas identificadores de cámaras IP en una red de datos existentes en Cuba y el mundo y se caracterizan además las principales herramientas y tecnologías que serán utilizadas en el desarrollo de la aplicación.

## 1.1 Conceptos asociados al dominio del problema

A continuación se exponen una serie de conceptos que serán utilizados a lo largo de la investigación y que van a servir de ayuda para lograr una mejor comprensión de la temática tratada.

**IP (Protocolo de Internet):**

Según (Feibel, 1996.) "…*es el protocolo de capa de red con más amplio apoyo para la Internet. Es uno de los protocolos en la suite TCP/IP. Este protocolo define y enruta datagramas a través de Internet y ofrece un servicio de transporte no orientado a conexión. El protocolo IP utiliza la conmutación de paquetes y hace el mejor esfuerzo para entregar sus paquetes."*

**Cámara IP**

Según (Asesorias y computadores LTDA, 2005-2012) *"…son dispositivos autónomos que cuentan con un servidor web de video incorporado, lo que les permite transmitir su imagen a través de redes IP. Las cámaras IP permiten al usuario tener la cámara en una localización y ver el video en tiempo real desde otro lugar a través de Internet."*

Según (Domocam, 2010) *"…las cámaras IP son video-cámaras de vigilancia capaces de enviar las señales de video y audio, bien por medio de un Router, o bien por medio de la red local, para poder visualizar en directo las imágenes dentro de una red local (LAN), o a través de cualquier equipo conectado a internet (WAN)."*

Teniendo en cuenta las definiciones mencionadas las cámaras IP son totalmente autónomas, provistos en su interior de servidores web, y con procesadores para transformar y emitir remotamente todo lo que suceda a su alrededor.

**Red de Datos:**

*Según (Diccionario de la Lengua Española, 2008) "…es una infraestructura cuyo diseño posibilita la transmisión de información a través del intercambio de datos. Cada una de estas redes ha sido diseñada específicamente para satisfacer sus objetivos, con una arquitectura determinada para facilitar el intercambio de los contenidos. Por lo general, estas redes se basan en la conmutación de paquetes. Pueden clasificarse de distintas maneras de acuerdo a la arquitectura física, el tamaño y la distancia cubierta."*

## 1.2 Descripción de la situación problemática

El proyecto de video vigilancia Suria comprende los componentes tradicionales de un sistema de seguridad basado en flujos de video IP, adaptándose fácilmente a los entornos corporativos en los que se deseen utilizar. Entre sus principales características se destacan: la integración de nuevos modelos de cámaras IP sin importar su tipo, la monitorización de varias áreas de forma simultánea en tiempo real, la realización de grabaciones, y la gestión para almacenamiento de grabaciones de forma automática. Aun teniendo el sistema las características y ventajas mencionadas anteriormente, en despliegues realizados el equipo de desarrollo ha identificado una debilidad que presenta el sistema asociadas al proceso de identificación de las cámaras IP. La misma se evidencia en la entrada manual de todos los datos de las cámaras IP a la hora de realizar la instalación y configuraciones iniciales del sistema, pues para cada cámara se debe especificar su dirección IP, puerto, tipo de dispositivo y otros elementos de configuración, haciéndose engorroso el proceso pues se debe ir hacia donde está instalada cada cámara y buscar el IP donde está conectada e investigar el modelo y fabricante pues dependiendo de esto es que se puede trabajar con ella y muchas veces es necesario hasta reiniciarla y comenzar todo desde el principio. Cuando el número de dispositivos conectados es numeroso se genera lentitud y demora en el proceso y se da lugar a la aparición de errores, pues el administrador o la persona encargada de la tarea pueden confundir una dirección o algún otro dato, imposibilitando así que el sistema sea capaz de controlar las cámaras IP.

## 1.3 Descripción del objeto de estudio

La identificación de cámaras IP es un paso importante en la automatización de la gestión de la red, es la capacidad de detectar automáticamente la mayor cantidad de información posible sobre los elementos de red y sus relaciones. (EMC Corporation, 2005) Por lo general, se realiza en dos niveles:

Nivel 3 o Capa de Red[2]: Este nivel incluye la identificación de los dispositivos relacionados con la capa 3 del modelo OSI[3]. Aquí se incluyen todos los dispositivos que tienen al menos una dirección IP (EMC Corporation, 2005).
Nivel 2 o Capa de Enlace de Datos[4]: Este nivel incluye la identificación de los dispositivos relacionados con la capa 2 del modelo OSI (como es el caso de los conmutadores) (EMC Corporation, 2005).

### *1.3.1 Técnicas de identificación de cámaras IP*

Para llevar a cabo la identificación de cámaras IP automáticamente se utilizan dos técnicas principales:
Técnica de Identificación Activa: El término activo se usa para describir esta actividad como algunos estímulos (es decir, paquetes) que son inyectados en la red para generar una respuesta del destinatario. Son muy adecuadas para encontrar los servicios abiertos actualmente en una red (Whyte, 2008). Técnica de Identificación Pasiva: El proceso de identificación de cámaras IP no introduce ningún tráfico en la red, se dedica estrictamente a escuchar (De Montigny-Leboeuf, y otros, 2004).

**Técnica de Identificación Activa**

Las técnicas activas generan ciertos paquetes conocidos como sondas que estimulan la respuesta de los nodos de la red. Los datos de las respuestas y el comportamiento resultante se utilizan para obtener información de la identificación de cámaras IP. Estos paquetes se generan a través de protocolos de red, entre los que se incluyen: SNMP, ARP, ICMP, TCP, DNS y NetBIOS (Mora Saavedra, 2011). De los protocolos mencionados anteriormente, SNMP[5] tiene la capacidad para proporcionar la mayor cantidad de información para el proceso de identificación. Las sondas generan una vista instantánea de la información, por tanto, para mantener una visión actualizada de la red, estos mensajes deben ser generados de forma periódica (Mora Saavedra, 2011).

---

[2] La capa de red del Modelo OSI es la encargada de encaminar los paquetes además de ocuparse de entregarlos.
[3] El modelo de referencia de Interconexión de Sistemas Abiertos (OSI, Open Systems Interconnection) es el modelo de red descriptivo creado por la Organización Internacional para la Estandarización (ISO) lanzado en 1984. Es un marco de referencia para la definición de arquitecturas de interconexión de sistemas de comunicaciones.
[4] La capa de enlace de datos se encarga de desplazar los datos por el enlace físico de comunicación hasta el nodo receptor
[5] SNMP (Simple Network Management Protocol): El Protocolo Simple de Administración de Red es un protocolo de la capa de aplicación que facilita el intercambio de información de administración entre dispositivo s de red. Es parte de la familia de protocolos TCP/IP. SNMP permite a los administradores supervisar el funcionamiento de la red, buscar y resolver sus problemas, así como planear su crecimiento.

En muchos casos, los paquetes de identificación de red activa están limitados a través de routers y cortafuegos, lo que restringe el efecto de la carga de la exploración contenida en la red que está escaneando. El tiempo de exploración utilizando proxy de identificación de red activa debe ser más rápido, ya que el análisis se realiza en el mismo tiempo por diferentes proxys dirigidos a diferentes subredes IP (Arkin, 2005).

**Técnica de Identificación Pasiva**

Las técnicas pasivas emplean aparatos de escucha llamados "sniffers" para obtener la información de identificación. Los sniffers logran su objetivo mediante el análisis de las Unidades de Datos de Protocolo (PDU), así como de la observación de los comportamientos de ciertos protocolos (por ejemplo, TCP). No generan ningún paquete de datos en la red con el fin la identificación de cámaras IP, de esta manera no son limitados por los dispositivos de filtrado como los cortafuegos. Sin embargo, pueden generar tráfico de datos para comunicar sus hallazgos a una entidad central cuando se han implementado de forma distribuida (Treurniet, 2004).

Estas técnicas de identificación pueden detectar los servicios activos, debido a la forma en que operan, los métodos pasivos en general, acumularán la información en un período de tiempo más largo que el empleado por las técnicas activas (De Montigny-Leboeuf, et al., 2004).

Para la realización del Identificador de cámaras IP en una red de Datos se seleccionó la técnica Activa pues permite identificar los puertos abiertos en la red y los servicios de red que se ejecutan en los hosts individuales.

### *1.3.2 Categorías en que se pueden clasificar las herramientas de identificación de cámaras IP*

**Herramientas Activas Híbridas**

Esta categoría incluye las herramientas que hacen uso de múltiples técnicas activas o de la combinación de alguna técnica pasiva con una activa. Una herramienta puede incluir SNMP como parte de sus protocolos de identificación activos, pero no se basa principalmente en SNMP para su funcionamiento. De esta forma, la herramienta es capaz de proporcionar suficiente información interesante, incluso si SNMP no es compatible con los dispositivos de red (Avallone, 2010)

**Herramientas Pasivas**

*Las técnicas pasivas dependen de sensores para monitorizar los paquetes que fluyen por la red e inspeccionar el contenido de estos (cabeceras o datos encapsulados).* En contraste con un enfoque activo, el proceso de recopilación pasiva de información no tiene ningún impacto en el ancho de banda o en los activos controlados. Por lo tanto la vigilancia pasiva se puede utilizar en todo momento sin riesgo de causar interrupciones en el servicio (De Montigny-Leboeuf, et al., 2004).

**Herramientas basadas en SNMP**

Esta categoría incluye las herramientas que se apoyan fuertemente en SNMP para realizar la identificación y recoger la información. Estas herramientas requieren que la generalidad de los dispositivos de red tenga habilitado el protocolo SNMP. Cuando no sucede así, la mayoría será capaz de identificar poca información de interés, por lo general, se limitará a identificar la dirección IP y el estado del nodo (Mora Saavedra, 2011).

Para la realización del Identificador de cámaras IP en una red de Datos se seleccionó como herramienta Activa Nmap pues es una herramienta para la exploración de redes, se trata de un software desarrollado para escanear redes completas.

## 1.4 Análisis de soluciones existentes

### *1.4.1 Identificadores de cámaras IP existentes en el mundo*

**AutoIP™**

GVI Security, Inc. es un proveedor líder de soluciones de seguridad de video vigilancia a la seguridad nacional, los segmentos del mercado institucional y comercial. (Security, 2000) La compañía ofrece un módulo completo para la video vigilancia, una de las soluciones que brindan es un identificador de cámaras IP para los fabricantes Samsung, Axis y Sony. De las cámaras o servidores de video muestra la siguiente información.

- ✓ IP local fija
- ✓ Modelo
- ✓ Protocolo

**Honeywell IP Utility**

Es un programa gratuito que permite a los usuarios configurar dirección IP, el nombre del dispositivo y cambiar contraseñas de las cámaras encontradas. También permite descubrir dispo-

sitivos en una red y abrir las aplicaciones Web-Client individual para cada dispositivo que se va a identificar en la red. (Honeywell, 2010) De las cámaras o servidores de video muestra la siguiente información.

- ✓ Nombre
- ✓ Contraseña
- ✓ Dirección IP

**Milestone Systems**

Fundada en 1998, es el líder mundial de la industria de software de gestión de video IP de plataforma abierta. Detecta y configura automáticamente las nuevas cámaras que se agregan a un sistema existente, permitiendo a los usuarios ampliar rápidamente su instalación lo que requiere una funcionalidad avanzada, monitorización y gestión centralizada. (Company Profile, 2010)

### *1.4.2 Conclusiones sobre las Soluciones Existentes*.

La revisión y estudio de las soluciones existente a nivel internacional evidenció la no existencia de una solución que solo realizara la identificación de cámaras IP para poder integrarla con el sistema Suria en su versión Qt. Las soluciones de estas compañías analizadas son aplicaciones propietarias, por lo tanto, son altamente dependientes de estas y además no son compatibles con el sistema de video vigilancia Suria en su versión Qt. Se necesita una herramienta que sea extensible con la finalidad de añadir soporte a una gran variedad de modelos y a diferentes fabricantes cuando sea necesario. En el análisis realizado se identificaron un conjunto de funcionalidades comunes que pueden ser incorporadas al identificador de cámaras IP para el sistema de video vigilancia Suria en su versión Qt, entre las que se destacan:

- Identificar todas las cámaras IP en la misma red.
- Acceder a los datos de las cámaras IP.
- Configurar automáticamente cada uno de los datos de las cámaras IP.

## 1.5 Herramientas y tecnologías para el desarrollo del software

### *1.5.1 Metodologías de Desarrollo*

Las metodologías de desarrollo de software constituyen el conjunto de procedimientos, técnicas, herramientas y soporte documental que ayuda a los desarrolladores a realizar un nuevo software. Representan un marco de trabajo que tiene entre sus principales funciones: guiar,

planificar, estructurar, controlar, manipular y dirigir el proceso de desarrollo de sistemas de información (Jacobson, y otros, 2000).

Se pueden clasificar en dos grupos:
Metodologías pesadas o tradicionales: Son las metodologías orientadas al control de los procesos, estableciendo rigurosamente las actividades a desarrollar, herramientas a utilizar y notaciones que se usarán (Carrillo Pérez, y otros, 2008).

Metodologías ligeras o ágiles: Son las metodologías orientadas a interactuar con el cliente y el desarrollo incremental del software, mostrando versiones parcialmente funcionales del software al cliente en intervalos cortos de tiempo, para que pueda evaluar y sugerir cambios en el producto según se va desarrollando (Carrillo Pérez, y otros, 2008).

Para desarrollar el identificador de cámaras IP en su versión Qt se utilizará la metodología pesada RUP, pues esta es muy detallada y posibilita documentar todo el proceso de forma minuciosa. Se ajusta tanto a proyectos grandes como a pequeños, siendo así muy conveniente su elección y utilización. Aunque genera un gran volumen de información, esto permite un mayor entendimiento entre los miembros del equipo de desarrollo. La gran cantidad de artefactos que se generan contribuyen a una mejor comprensión del problema y la solución.

**Proceso Unificado de Desarrollo (Rational Unified Process, RUP)**
Es una metodología de desarrollo de software que intenta integrar todos los aspectos a tener en cuenta durante todo el ciclo de vida del software, con el objetivo de hacer abarcables tanto pequeños como grandes proyectos de software (Martínez, y otros).
Plantea principios clave a tener en cuenta durante su utilización: adaptación del proceso, balance de prioridades, colaboración entre equipos, elaboración del nivel de abstracción y deberá permitir enfocarse en la calidad (Martínez, y otros, 2010).

Esta metodología define varias disciplinas dentro de las que se encuentran: Modelado de Negocios, Requerimientos, Análisis y Diseño, Implementación, Pruebas, Transición, Configuración, Administración del Cambio, Administración de Proyectos y Ambiente.

Además divide el desarrollo en cuatro fases que definen el ciclo de vida del producto. La fase de inicio tiene como objetivo determinar la visión del proyecto y definir lo que se desea realizar. En la fase de elaboración se determina la arquitectura óptima del proyecto. Por su parte la tercera fase, construcción, tiene como objetivo obtener la capacidad operacional inicial y por últi-

mo la fase de transmisión tiene la responsabilidad de obtener el producto acabado y definido (Carrillo Pérez, y otros, 2008)

El Proceso Unificado de Desarrollo es:

**Dirigido por casos de uso:** El proceso de desarrollo avanza a través de una serie de flujos de trabajo que parten de los casos de uso. Los casos de uso se especifican, se diseñan y los casos de uso finales son la fuente a partir de la cual se construyen los casos de prueba (Jacobson, y otros, 2000).

**Centrado en la arquitectura:** La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente (Jacobson, y otros, 2000).

**Iterativo e incremental:** RUP propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros. Es práctico dividir el trabajo en partes más pequeñas o mini proyectos. Cada mini proyecto es una iteración que resulta en un incremento. Las iteraciones hacen referencia a pasos en los flujos de trabajo, y los incrementos, al crecimiento del producto (Jacobson, y otros, 2000).

### *1.5.2 Lenguaje de Modelado*

El Lenguaje Unificado de Modelado (UML) se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Se usa para entender, diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas. Da apoyo a la mayoría de los procesos de desarrollo orientados a objetos (Rumabugh, y otros, 1998).

UML capta la información sobre la estructura estática y el comportamiento dinámico de un sistema. El modelar un sistema desde varios puntos de vista, separados pero relacionados, permite entenderlo para diferentes propósitos (Rumabugh, y otros, 1998).

### *1.5.3 Herramientas CASE*

*Las herramientas CASE*[6] *son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste del mismo en términos de tiempo y dinero* (Jacobson, et al., 2000).

Las herramientas CASE proporcionan al ingeniero la posibilidad de automatizar actividades manuales y de mejorar su visión general de la ingeniería. Al igual que las herramientas de ingeniería y de diseño asistidos por computadora que utilizan los ingenieros de otras disciplinas, las herramientas CASE ayudan a garantizar que la calidad se diseñe antes de llegar a construir el producto (Rumabugh, et al., 1998).

Aspectos con los cuales la misma deberá de cumplir:

1. Facilidad de uso.
2. Modelar el software a través de diagramas UML.
3. Lograr integración con diferentes IDE.

**Visual Paradigm para UML v2.0**

Visual Paradigm para UML[7] es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una rápida construcción de aplicaciones de calidad y a un menor coste.

Dentro de las principales características de esta herramienta y que son de utilidad a nuestra aplicación se encuentran:

**1.** Disponibilidad en múltiples plataformas como: Windows y Linux (Visual Paradigm for UML, 2011).

**3.** Diseño centrado en casos de uso y enfocado al negocio que genera un software con mayor calidad (Uml.Slideshare, 2011).

**4.** Interoperabilidad entre diagramas: permite a partir de un diagrama obtener otro que guarde relación con el mismo. (Urbino, 2010)

[6] Computer Aided Software Engineering (Ingeniería de Software Asistida por Computador)
[7] Lenguaje de modelado para especificar o describir métodos o procesos

Según los requisitos planteados para la selección de la herramienta CASE a utilizar en el desarrollo exitoso del producto se decidió emplear Visual Paradigm para UML pues posibilita documentar todo el proceso de forma minuciosa, es una herramienta profesional que soporta todo el ciclo de vida del software y permite el desarrollo de aplicaciones con rapidez, facilidad y calidad con un menor costo de producción.

### *1.5.4 Lenguaje de Programación*

Un lenguaje de programación es un *"conjunto de instrucciones, órdenes, comandos y reglas que permite la creación de programas"* (Glosario.NET., 2007).

**C++ ANSI 98**

C++ es un lenguaje de programación derivado del C que soporta la programación orientada a objetos y la estructurada. Fue creado a mediados de los años 80 por Bjarne Stroustrup con el fin de agregar funcionalidades y características de las que carecía su antecesor, mantiene ventajas en cuanto a riqueza de operadores, expresiones, flexibilidad, concisión y eficiencia. Además, ha eliminado algunas de las dificultades y limitaciones del C original. Desde el punto de vista de los lenguajes orientados a objetos, C++ es un lenguaje híbrido y existen varios IDE que lo soportan como: Eclipse y Qt Creator ( Lenguajes de programación, 2009) (Louden, 2004) (Programación en castellano, 2011).

### *1.5.5 Entorno de Desarrollo Integrado*

Un entorno integrado de desarrollo es un programa informático compuesto por un conjunto de herramientas de programación, que proveen facilidades a los programadores para escribir códigos y agilizar el proceso de desarrollo de software. Pueden ser aplicaciones por sí solas o ser parte de aplicaciones existentes, además permiten ser utilizadas tanto con un único lenguaje de programación como con varios de estos. Las herramientas que normalmente componen un entorno de desarrollo integrado son las siguientes: editor de texto, compilador, intérprete, herramientas para la automatización, depurador, diseñador para la construcción de interfaces gráficas de usuario y, opcionalmente, un sistema de control de versiones (IDE, 2012).

Para la selección del entorno integrado de desarrollo a utilizar en la construcción de la aplicación se definieron los siguientes parámetros a cumplir:

1. Facilidad de uso y abundante documentación.
2. Compatibilidad con GNU\Linux.

3. Integración con el framework de desarrollo escogido para la implementación de la aplicación.
4. Soporte para el lenguaje C++.

**Qt Creator v 2.4.1**

Qt Creator es un entorno integrado de desarrollo creado por Trolltech y diseñado para hacer que el desarrollo en C++ de la aplicación Qt sea más rápido y fácil. Está basado en la biblioteca Qt, la biblioteca multiplataforma de interfaces gráficas de usuario. Pensado especialmente para el desarrollo en varias plataformas como: Windows, Linux y Mac OSX. Adaptado a las necesidades de los desarrolladores, permite crear aplicaciones de escritorio y plataformas de dispositivos móviles (Qt Project Hosting, 2014).

Algunas de las características que se destacan de este IDE son (Programación en castellano, 2011):

1. Avanzado editor de código C++.
2. Tiene integrado diseñadores de interfaz gráfica de usuario.
3. Herramientas para la administración de proyectos.
4. Ayuda sensible al contexto.
5. Depurador visual.
6. Resaltado y autocompletado de código.

### *1.5.6 Herramienta para el escaneo de redes*

Para la implementación de la aplición se utilizará el Nmap, encargado del escaneo de la red de Datos y de encontrar la mayor cantidad de información posible de las cámaras.

**Nmap v6.46**

Rastrea los puertos de la máquina o máquinas en cuestión y establece si un puerto está abierto, cerrado o protegido por un cortafuego. Funciona enviando paquetes o realizando una llamada de conexión (Debich, 2014). Nmap puede utilizarse para muchas finalidades, pero las principales son:

- Descubrimiento de servidores: Identifica computadoras en una red, por ejemplo listando aquellas que responden a los comandos ping.
- Identifica puertos abiertos en una computadora objetivo.
- Determina qué servicios está ejecutando la misma.

- Determinar qué sistema operativo y versión utiliza dicha computadora, esta técnica es también conocida como fingerprinting.
- Obtiene algunas características del hardware de red de la máquina objeto de la prueba.

## Conclusiones

En este capítulo fueron abordados una serie de conceptos fundamentales que giran alrededor del objetivo general de la investigación y que constituyen el soporte teórico del desarrollo de la aplicación.

- ✓ Se caracterizaron las técnicas y herramientas existentes de identificación de cámaras IP, facilitándose una descripción profunda del objeto de estudio, permitiendo entender en que consiste este proceso.
- ✓ Se analizaron soluciones existentes en el mundo, determinándose que no se encontró un identificador de cámaras IP en Cuba, por lo cual fue necesario centrar el estudio en los sistemas de video vigilancia que incluyen la identificación de las cámaras IP, obteniendo las características de funcionamiento que sirven de guía para el desarrollo de la investigación.
- ✓ Se caracterizaron las herramientas y tecnologías informáticas a emplear en el desarrollo del sistema, enunciándose de cada una de ellas las principales características y las facilidades que ofrecen al equipo de desarrollo.

# Capítulo II: Características del Sistema

## Introducción.

En este capítulo se detallan los contenidos relacionados con el análisis de la aplicación que se pretende desarrollar. Definiéndose el modelo de dominio con el objetivo de lograr un mayor acercamiento del problema a resolver. También se capturan los requisitos de software definiéndose las cualidades que debe cumplir la aplicación y se presenta el diagrama de casos de uso del mismo, así como las descripciones de los casos de uso críticos de la aplicación.

## 2.1 Modelo de dominio

Con motivo de la poca estructuración de los procesos del negocio y para poder comprender el contexto en el cual se desarrolla el sistema se determinó desarrollar un Modelo de Dominio. En el proceso de análisis o investigación orientados a objetos, constituye un paso esencial el poder descomponer el problema en conceptos u objetos individuales. La designación de modelo conceptual ofrece la ventaja de subrayar fuertemente una concentración en los conceptos del dominio, no en las entidades del software (Larman, 2da Edición). El modelo de dominio es el encargado de capturar los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan las "cosas" que existen o los eventos que suceden en el entorno en el que trabaja el sistema (Jacobson, y otros, 2000).

### *2.1.1 Descripción del flujo del diagrama del modelo de dominio*

Con el objetivo de que el sistema Suria sea capaz de controlar las cámaras IP conectadas en la red y obtener los datos de la configuración de las mismas, el Administrador obtiene manualmente los datos de la configuración de las cámaras IP conectadas en la Red de Datos. Luego, puede adicionar los datos de las cámaras al sistema, permitiendo así que el sistema de video vigilancia Suria en su versión Qt pueda hacer el resto de las operaciones con ellas.

### *2.1.2 Diagrama de Clases de dominio*.

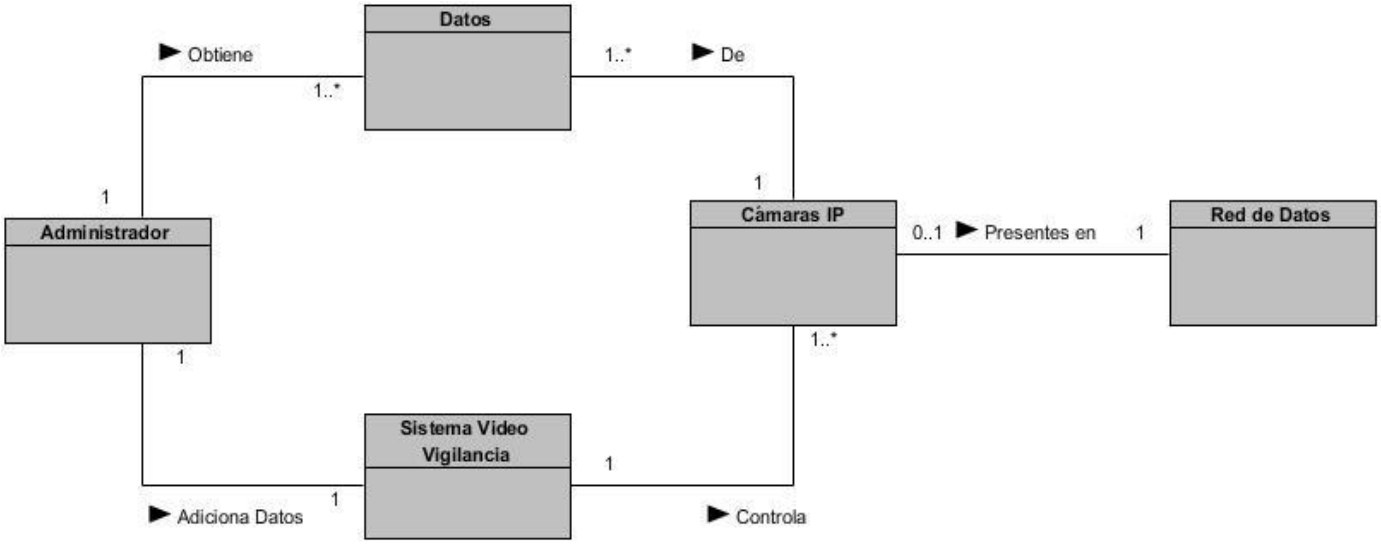



Figura 1: Diagrama de clases de dominio.

### *2.1.3 Descripción de las clases*

- Administrador: Es el usuario encargado de obtener los datos de las cámaras y luego adicionarlas al sistema.
- Datos: representa los datos de las cámaras IP que el administrador configurará.
- Sistema Video Vigilancia: Representa al sistema de Video Vigilancia Suria.
- Cámara IP: Son cámaras IP diseñadas para enviar los flujos de video a través de la red.
- Red de Datos: Infraestructura a la que están conectadas las cámaras IP.

## 2.2 Especificación de los requisitos de software

Los requisitos, de manera general, son una descripción de lo que se necesita o desea de un producto. La meta primaria de la fase de requisitos es identificar y documentar lo que en realidad se necesita, en una forma clara y entendible tanto para los clientes como para los miembros del equipo de desarrollo (Mora Saavedra, 2011).

### *2.2.1. Requisitos Funcionales*

Los requisitos funcionales de un sistema son declaraciones de los servicios que debe proporcionar el sistema, de la manera que este debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares (Sommerville, 2005).

**R.F1** Cargar plugins de identificación de cámaras.

La aplicación debe permitir al usuario cargar los plugins necesarios para identificar determinado modelo de cámara que desee encontrar, en caso de adicionarse a la red un dispositivo cuyo modelo no figure entre los soportados por el Identificador.

- ✓ Rangos de direcciones IP (Formato: Numérico).
- ✓ Rangos de Puertos (Formato: Alfabético).

**R.F2** Identificar cámaras IP.

La aplicación debe permitir encontrar las cámaras IP en un rango de direcciones definido por el usuario.

- ✓ Rangos de direcciones IP (Formato: Numérico.; valor entre 0 y 255).
- ✓ Rangos de Puertos (Formato: Numérico).

**R.F3** Seleccionar Rango de direcciones IP.

La aplicación debe permitir que el usuario seleccione los rangos de IP en los que se va a realizar la identificación de cámaras IP.

**R.F4** Adicionar Rango de direcciones IP.

La aplicación debe permitir que el usuario adicione los rangos de IP en los que se va a realizar la identificación de cámaras IP.

- ✓ Rangos de direcciones IP (Formato: Numérico.; valor entre 0 y 255).

**R.F5** Modificar Rango de direcciones IP.

La aplicación debe permitir que el usuario modifique los rangos de IP en los que se va a realizar la identificación de cámaras IP.

- ✓ Rangos de direcciones IP (Formato: Numérico.; valor entre 0 y 255).

**R.F6** Eliminar Rango de direcciones IP.

La aplicación debe permitir que el usuario elimine los rangos de IP en los que se va a realizar la identificación de cámaras IP.

**R.F7** Seleccionar Rango de Puertos.

La aplicación debe permitir que el usuario seleccione los rangos de puertos que se probarán inicialmente para cada dirección IP.

**R.F8** Adicionar Rango de Puertos.

La aplicación debe permitir que el usuario adicione los rangos de puertos que se probarán inicialmente para cada dirección IP.

- ✓ Rangos de Puertos (Formato: Numérico).

**R.F9** Modificar Rango de Puertos.

La aplicación debe permitir que el usuario modifique los rangos de puertos que se probarán inicialmente para cada dirección IP.

- ✓ Rangos de Puertos (Formato: Numérico).

**R.F10** Eliminar Rango de Puertos.

La aplicación debe permitir que el usuario elimine los rangos de puertos que se probarán inicialmente para cada dirección IP.

**R.F11** Modificar datos de la cámara.

La aplicación debe permitir que el usuario se conecte a la página web del dispositivo, y de esta forma modificar los datos de configuración del mismo.

- ✓ Direcciones IP (Formato: Numérico.; valor entre 0 y 255).
- ✓ Usuario (Formato: Alfabético).
- ✓ Contraseña (Formato: Alfanumérico).

### *2.2.2. Requisitos No Funcionales*

Los requisitos no funcionales del sistema son restricciones de los servicios o funciones ofrecidas por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. (Sommerville, 2005)

**Requisitos No Funcionales de Diseño e Implementación.**

**RnF1.** Restricciones de Diseño e Implementación: Se debe desarrollar la aplicación en el lenguaje C++, utilizando como marco de trabajo el framework Qt en su versión 5.2.

**Requisitos No Funcionales de Usabilidad.**

**RnF2.** Tipo de usuario Final: La aplicación debe ser usado por un administrador del sistema Suria que deberá tener conocimientos medios en informática.

**RnF3.** Tipo de Aplicación Informática: El Identificador de cámaras IP en una red de datos será una aplicación de escritorio.

**RnF4.** Finalidad: La aplicación que se pretende desarrollar para el sistema de video vigilancia Suria en su versión Qt, tiene como objetivo agilizar los procesos de identificación de cámaras IP que garantice el escaneo de la red.

**Requisitos No Funcionales de Ambiente.**

**RnF5.** Software: Sistema Operativo: Windows o Linux. Debe tener instalado el Nmap v6.46.

**RnF6.** Hardware: El hardware de la PC donde se ejecute la aplicación debe contar con una memoria 2GB de RAM, Core i3 a 2.4GHz y estar conectada a una red Ethernet en funcionamiento.

**Requisitos de la interfaz externa.**

**RnF7.** Interfaz de usuario: La aplicación debe tener un botón para cada funcionalidad evitando así ambigüedades. De cada botón saldrá una etiqueta especificando su funcionalidad. Los datos de las cámaras se mostrarán en columna de manera que el usuario pueda entender el resultado obtenido.

**Requisitos de confiabilidad.**

**RnF8.** El sistema debe garantizar un tratamiento adecuado cuando exista un error en los datos introducidos, mostrando un mensaje al administrador informando las causas del error.

## 2.4 Descripción del sistema

La solución será una aplicación de escritorio cuya principal funcionalidad será identificar las cámaras IP en una Red de Datos. Además, brindará la posibilidad de modificar los datos de las cámaras como la dirección IP, usuario y contraseña. Los administradores del sistema de Video Vigilancia serán los únicos con acceso a la aplicación, la cual será basada en componentes, permitiendo, de esta manera, añadir soporte para diferentes marcas y modelos de las cámaras IP en la medida que se necesite.

Luego de identificados los requisitos funcionales del identificador de cámaras IP en una red de Datos para el sistema Suria en su versión Qt se agruparon en casos de usos (CU), los cuales se representan el diagrama de casos de uso del sistema. Cada uno de los casos de usos identificados engloba un conjunto de acciones, las cuales son inicializadas por los actores del sistema.

### 2.4.1. Definición de los actores

Definir los actores que interactuarán con la aplicación es uno de los primeros pasos en la definición del uso del mismo. Cada tipo de fenómeno externo con el que debe interactuar el sistema está representado por un actor. Se debe definir cada actor especificando una breve descripción que incluya el área de responsabilidad del actor. Como los actores representan elementos externos al sistema, no es necesario describirlos en detalle.

Los actores identificados para el identificador del sistema de video vigilancia Suria en su versión Qt son:

Tabla 1: Descripción de los actores.

| Actor | Descripción |
|---|---|
| Administrador | Persona encargada de velar por el funcionamiento apropiado de la aplicación, es el único responsable de configurar y gestionar las funcionalidades del sistema. |

### 2.4.2 Diagrama de Casos de Uso del Sistema

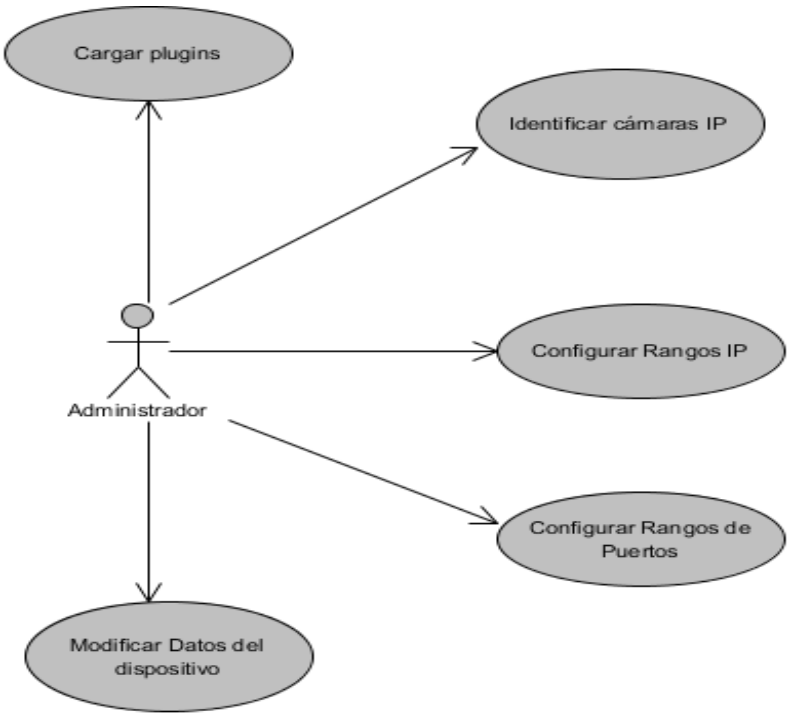



Figura 2: Diagrama de casos de uso del Sistema.

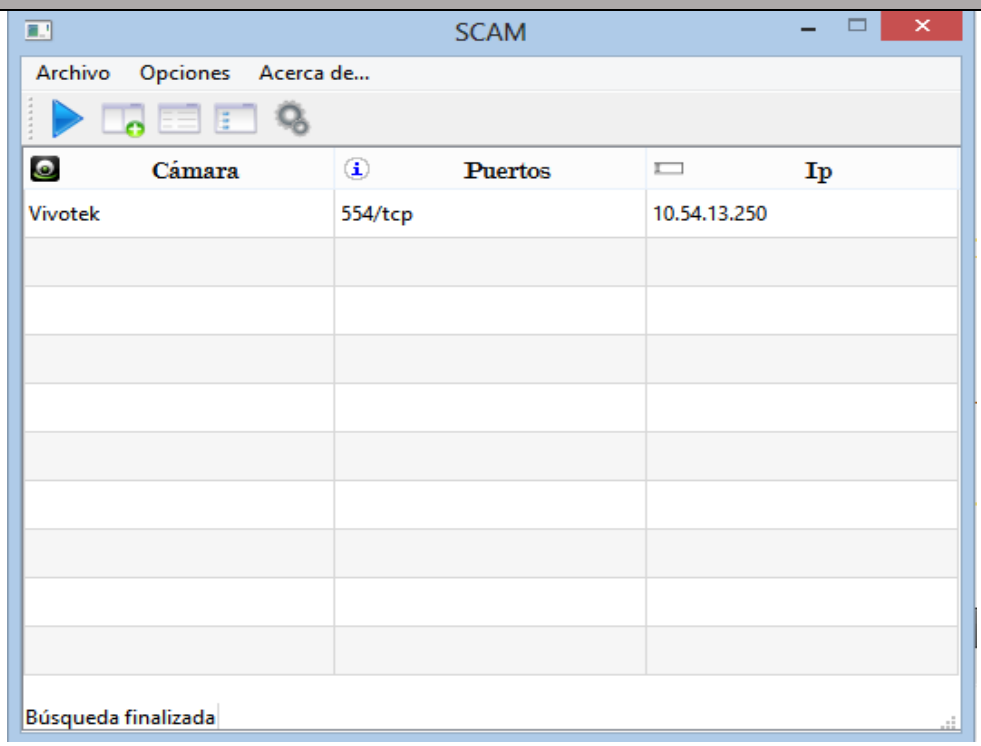
### *2.4.3. Especificación de los Casos de Uso del Sistema*

En el presente epígrafe se presenta la descripción textual de los casos de uso Identificar cámaras IP por ser considerado arquitectónicamente significativo. La especificación de los otros casos de uso se encuentra en el Anexo 2.

Tabla 2: Descripción textual del caso de uso Identificar cámaras IP.

| | | |
|---|---|---|
| **CU** | Identificar Cámaras IP | |
| **Actores** | Administrador | |
| **Resumen** | El caso de uso se inicia cuando el Administrador selecciona la opción “Identificar cámaras”. | |
| **Referencias** | R.F2 | |
| **Prioridad** | Crítico | |
| **Precondiciones** | El actor debe haber seleccionado un rango de direcciones IP y un rango de puertos. | |
| **Postcondiciones** | Se muestra un listado con las cámaras encontradas, sobre las cuales se pueden realizar otras operaciones. | |
| **Flujo de eventos** | | |
| **Flujo básico:** | | |
| | **Actor** | **Aplicación** |
| | | |
| | 1. El actor selecciona la opción “Identificar cámaras”. | 2. La aplicación escanea por el rango de direcciones IP y por el rango de puertos en caso de ser seleccionado.<br>2.1. La aplicación compara cada dirección IP y rango de puertos seleccionados.<br>2.2. Si en una dirección IP hay una cámara, obtiene los datos de esta, y los añade al listado de cámaras.<br>2.3. La aplicación actualiza la interfaz mostrando el listado de cámaras modificado. |
| **Flujos alternos** | | |
| | **Actor** | **Aplicación** |

| | | |
|---|---|---|
| | | 2.3. La aplicación muestra en la barra de estado que el escaneo ha finalizado. |
| **Prototipo de Interfaz:** | | |




## Conclusiones

En este capítulo se especificaron los temas relacionados con el análisis de la aplicación que se desea desarrollar, a partir de la información recopilada con anterioridad.

- ✓ Se realizó una modelación de la situación problémica obteniéndose el diagrama del modelo de dominio y la descripción del flujo de dominio lográndose una mejor comprensión sobre el funcionamiento de la aplicación así como de las clases que interactúan en el mismo.
- ✓ Se identificaron los requisitos funcionales y no funcionales de la aplicación que se pretende desarrollar, obteniéndose un total de 6 requisitos funcionales que fueron agrupados en 5 casos de uso de ellos 1 críticos y el resto secundarios, obteniéndose una mejor comprensión de las funcionalidades de la aplicación, permitiéndole a los implementadores un mejor entendimiento de la aplicación que se desea desarrollar.
- ✓ Se identificaron los actores que interactúan con la aplicación, obteniéndose finalmente el diagrama de casos de uso del sistema, lo cual facilitó la realización de las especificaciones que permitirán a los desarrolladores conocer de forma detallada el flujo de cada una de las funcionalidades.

# Capítulo III: Análisis y Diseño

## Introducción

En este capítulo se abordan temas fundamentales para la construcción de la aplicación propuesta, definiéndose los patrones de diseño empleados para garantizar la calidad y viabilidad de la aplicación a desarrollar. Se presenta además el modelo de diseño, conformado por el diagrama de clases y el diagrama de secuencia.

## 3.1 Descripción de la Arquitectura

La arquitectura de software de un programa o un sistema de cómputo es la estructura que incluyen los componentes de software, las propiedades visibles externamente de esos componentes y las relaciones entre ellos (Pressman, 2005).

### *3.1.1 Arquitectura en Capas*

Este patrón pertenece al estilo llamada y retorno, descompone una aplicación en un conjunto de capas independientes y ordenadas jerárquicamente. Cada nivel o capa usa los servicios de la capa inmediatamente inferior y ofrece servicios a la inmediatamente superior. Permite estructurar aplicaciones que se pueden descomponer en grupos de sub-tareas, donde cada grupo está en un determinado nivel de abstracción (Claramunt, 2012).

Tiene como ventaja la reutilización de capas, facilita la estandarización y la contención de cambios a una o pocas capas (Garlan, et al., 1994). Para la realización del identificador de cámaras IP en una red de Datos para el sistema de video vigilancia Suria en su versión Qt, se define la arquitectura en dos capas.

**La capa de la Presentación:**

Esta capa reúne todos los aspectos del software que tienen que ver con las interfaces y la interacción con los diferentes tipos de usuarios. Estos aspectos típicamente incluyen el manejo y aspecto de las ventanas.

**La capa Lógica del Negocio:**

Esta capa reúne todos los aspectos del software que automatizan o apoyan los procesos que llevan a cabo los usuarios. Estos aspectos típicamente incluyen las tareas que forman parte de los procesos, las reglas y restricciones que aplican.

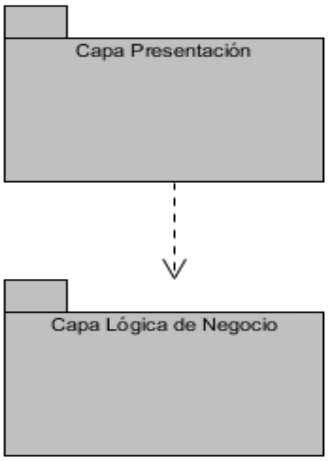



Figura 3: Arquitectura en Capas

### *3.1.2 Arquitectura basada en componentes*

Debido a que el sistema debe soportar una gran variedad de cámaras, es necesario un mecanismo que permita dar soporte a nuevos modelos, sin tener que recompilar el código de la aplicación cada vez que esto sea necesario. Es por eso que la arquitectura basada en componentes es ideal para dar solución a este problema. Para cumplir con este patrón se ha diseñado un mecanismo de plugins, agregados externos a la aplicación. Las funcionalidades de agregar y visualizar cámaras de un tipo en específico se implementan en forma de plugins, lo que brinda ventajas como:

- Extensibilidad del sistema: Se pueden agregar plugins al sistema para dar soporte a nuevos modelos de cámaras, sin tener que recompilar todo el sistema.
- Ambiente controlado: Cada plugin implementa un ambiente controlado de visualización. En caso de que ocurra un error, este puede ser descargado sin afectar al resto de la aplicación.

## 3.2 Patrones de diseño

Un patrón de diseño es una descripción de un problema bien conocido, expresan esquemas para definir estructuras de diseño (o sus relaciones), con los que se pueden construir sistemas de software (Larman, 1999).

### *3.2.1 Patrones GRASP*

GRASP es el acrónimo para General Responsibility Assignment Software Patterns (Patrones Generales de Software para Asignar Responsabilidades). Describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Constituyen un apoyo para la enseñanza que ayuda a entender el diseño de objeto esencial y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable (Larman, 1999).

A continuación se describen los patrones utilizados en la solución propuesta:

- Experto**:** Es el principio básico de asignación de responsabilidades. Este patrón garantiza que cada clase cumpla con sus responsabilidades, según la información que contenga y las funcionalidades que se deseen implementar (Larman, 1999). En la aplicación cada clase mantiene este principio, por ejemplo la clase **CC_Configuracion** es la experta en conocer todo lo relacionado con los rangos de IP y los rangos de puertos, por tanto es aquí donde se realizan las acciones referentes a la configuración de los mismos.

- Creador: este patrón plantea que se debe guiar la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. Se utiliza este patrón debido a la característica del sistema de ser orientado a objetos (Larman, 2003). Este patrón se evidencia en la clase **CI_Principal**. En esta se crean los objetos de todas las clases.

- Controlador: este patrón plantea que se debe guiar la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. Se utiliza este patrón debido a la característica del sistema de ser orientado a objetos (Larman, 2003). En la aplicación se evidencia mediante la clase **CC_Configuracion** que no forma parte del negocio y cuya única función es re direccionar los datos desde la interfaz hasta las diferentes clases, dependiendo de la función invocada.

- Alta cohesión: Una clase tiene una responsabilidad moderada en un área funcional y colabora con otras clases para llevar a cabo las tareas. (Larman, 2003). Como ejemplo en la aplicación se encuentran las clases **CC_Configuracion y CC_Cargar_plugin** con responsabilidades altamente relacionadas, y que no hacen una gran cantidad de trabajo.

- Bajo Acoplamiento: Asignar una responsabilidad para mantener bajo acoplamiento (una clase con bajo acoplamiento no depende de muchas otras). El acoplamiento es una medida de la fuerza con que una clase está conectada a otras. El patrón propone el diseño de clases más independientes, lo que reduce el impacto del cambio y facilita la reutilización en otros sistemas. Es un patrón evaluativo que el diseñador aplica al juzgar sus decisiones de diseño (Larman, 1999). Con este patrón se logra en la aplicación un grado de dependencia mínimo entre las clases **CC_Configurar_Rango_Ip** y **CC_Configurar_Rango_Puerto** por ejemplo, que aunque parecen estar estrechamente ligadas fueron implementadas de forma tal que la fuerza con que están conectadas entre sí es casi nula.

### *3.2.2 Patrones GoF empleados*

Los patrones *GoF* (Gang of Four, en español Pandilla de los Cuatro) se clasifican en 3 categorías basadas en su propósito: creacionales, estructurales y de comportamiento.

- ✓ Creacionales: los patrones creacionales se encargan de la creación de instancias de los objetos. Abstraen la forma en que se crean los objetos, permitiendo tratar las clases a crear de forma genérica, dejando para después la decisión de qué clase crear o cómo crearla.

- ✓ Estructurales: los patrones estructurales se ocupan de cómo las clases y objetos se combinan para formar grandes estructuras y proporcionar nuevas funcionalidades. Tratan de conseguir que los cambios en los requisitos de la aplicación no ocasionen cambios en las relaciones entre los objetos.

- ✓ Comportamiento: los patrones de comportamiento plantean la interacción y cooperación entre las clases. Son utilizados para organizar, manejar y combinar comportamientos.

Observador: Garantiza que cuando un objeto cambia, todos los objetos que dependen del mismo sean notificados y actualizados. Este patrón se evidencia en el uso del mecanismo de señales y slots de Qt. También se puede ver en la comunicación entre los plugins para descubrir las cámaras y la aplicación, pues esta última se convierte en observadora de los plugins y es notificada en el momento en que uno de ellos culmina el análisis (Mora Saavedra, 2011).

Command: permite solicitar una operación a un objeto sin conocer realmente el contenido de esta operación, ni el receptor real de la misma. Para ello se encapsula la petición como un objeto, con lo que además se facilita la parametrización de los métodos. En la aplicación se utiliza este patrón cuando se llama al proceso externo Nmap.

## 3.3 Modelo de diseño

El modelo de diseño es una abstracción de la implementación del sistema. Se utiliza para concebir y documentar el diseño del sistema de software. Es un producto de trabajo integral y compuesto que contiene los artefactos, clase de diseño, interfaz, paquete de diseño, subsistema de diseño, sucesos, señal, clase con probabilidad, ejecución de guiones de uso, operación, realización de una operación y componente de servicio (De la Torre, 2006).

### *3.3.1 Diagrama de secuencia*

Un diagrama de secuencia del sistema es un artefacto creado de manera rápida y fácil, que muestra los eventos de entrada y salida relacionados con el sistema que se está estudiando. UML incluye la notación de los diagramas de secuencia, para representar los eventos que parten de los actores externos hacia el sistema (Larman, 1999). Los demás diagramas de secuencia relacionados con esta aplicación, se encuentran en el Anexo 3.

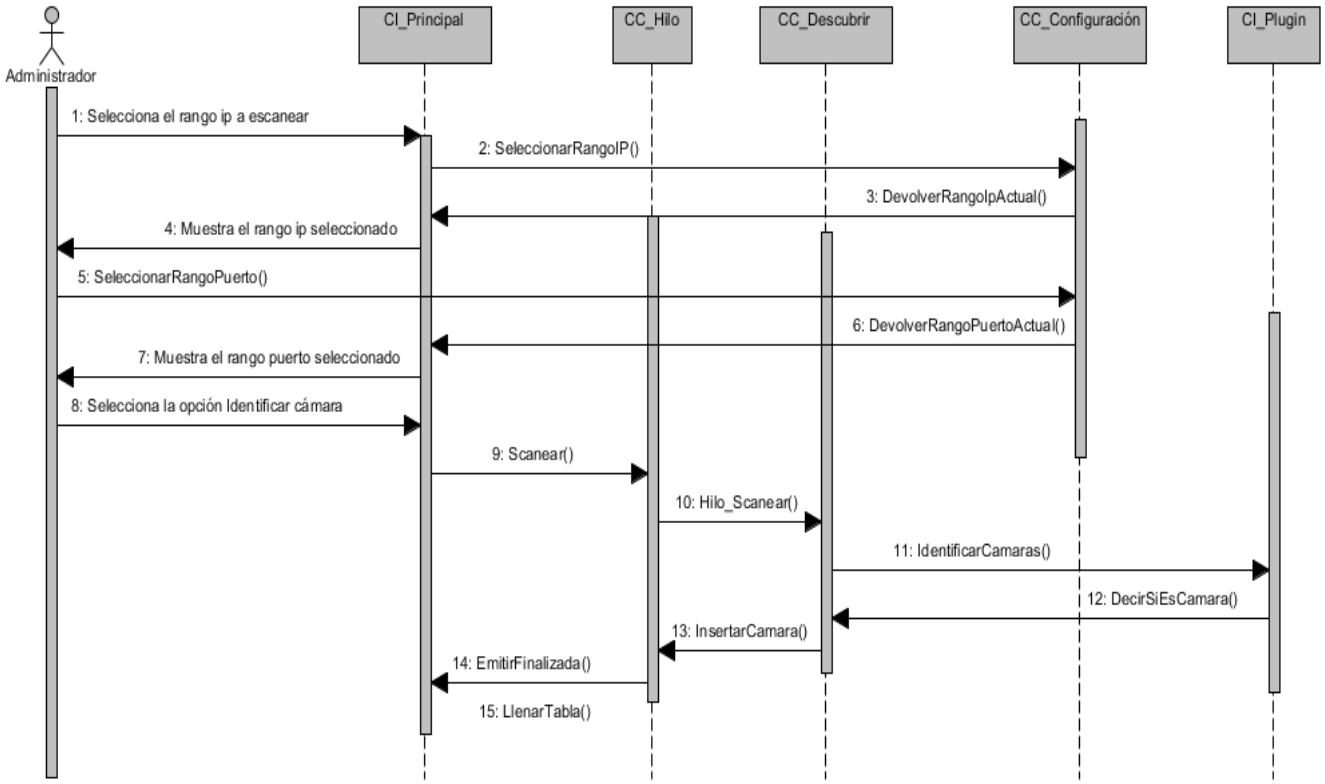



Figura 4: Diagrama de Secuencia del Caso de Uso Identificar Cámaras IP.

### *3.3.2 Diagrama de clases*

Un diagrama de clases proporciona una perspectiva estática que representa el diseño estructural del sistema, mostrando un conjunto de clases, sus atributos y las relaciones entre ellos (Larman, 2003). En este diagrama se muestra las clases definidas en el diseño, así como la relación entre ellas, comenzando por las clases agrupadas en el paquete de Vistas. En este paquete se encuentran las clases **CI_Principal** y **CI_Configuracion,** que serán las encargadas de interactuar con el administrador. A su vez la clase **CI_Configuracion** se relaciona con la clase **CC_Configuracion,** que tiene como objetivo regular el flujo de información entre la vista antes mencionada y las controladoras **CC_Cargar_Plugin**, **CC_Configurar_Rango_Ip** y **CC_Configurar_Rango_Puerto**, logrando de esta forma también una especialización en cada caso. La clase **CC_CargarPlugin** por otra parte, usa la interfaz **CI_Plugin** a la

hora de cargar los plugins necesarios al iniciar la aplicación. Otra clase que utiliza la interfaz **CI_Plugin** es la controladora **CC_Descubrir**, encargada de implementar los métodos relacionados con la identificación de cámaras. La clase **CE_Camara** posee datos acerca del dispositivo y la forma de acceder a ellos y está asociada a las clases **CC_Descubrir** antes mencionada y con la controladora **CC_Modificar_Datos_Camara**. Esta última nos ofrece acceso al software de la cámara, permitiendo de esta forma modificar la información perteneciente a ella.

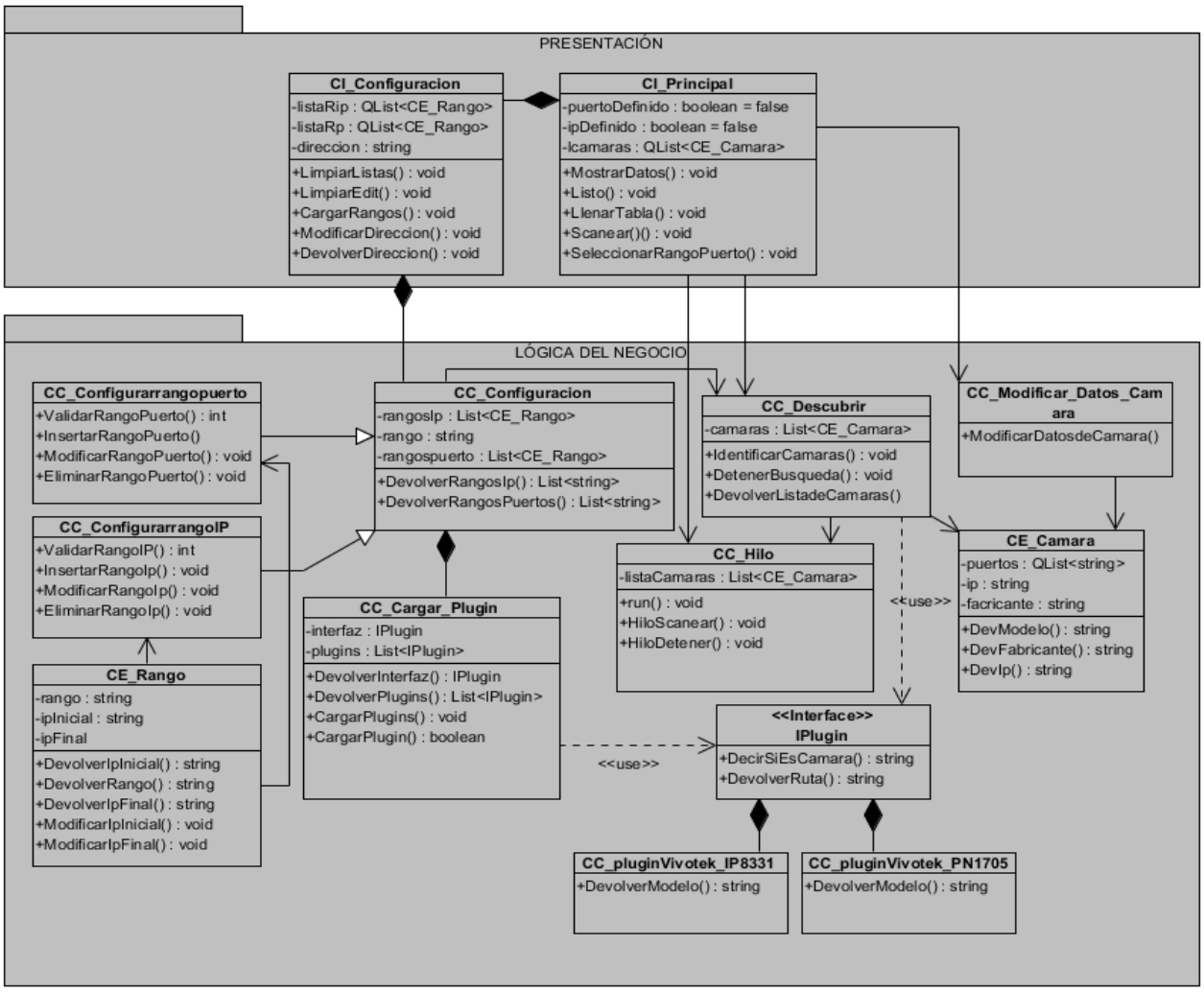



Figura 5: Diagrama de Clase de Diseño.

## Conclusiones

En este capítulo se definieron temas relacionados con el análisis y el diseño del Identificador de Cámaras IP, para el sistema Video Vigilancia Suria en su versión Qt:

- ✓ Se estableció la línea base de la arquitectura que tendrá el identificador que se quiere desarrollar, empleándose el patrón arquitectónico dos capas, permitiendo así la reutilización de capas y facilitando la estandarización y la contención de cambios a una o pocas capas.
- ✓ Se identificaron los patrones de diseño utilizados en el desarrollo de la aplicación, facilitando la asignación de responsabilidades y logrando un diseño de software que sirva de apoyo a la implementación del sistema.
- ✓ Se realizó el diagrama de secuencia del caso de uso más crítico, mostrándose las relaciones entre las clases por medio de funciones y garantizando así una mejor comprensión del flujo de vida del caso de uso. Además, se realizó el diagrama de clases de diseño, obteniendo con esto una visión estática del sistema y facilitando la comunicación entre los programadores y la identificación de cámaras IP de fallas de la aplicación en el diseño.

# Capítulo IV: Implementación y Prueba

## Introducción

En este capítulo se presentan todos los elementos relacionados con el flujo de trabajo de implementación, describiéndose detalladamente las técnicas utilizadas en el desarrollo de la aplicación. También se detallan las pruebas realizadas a la aplicación, con el objetivo de validar su correcto funcionamiento.

## 4.1 Diagrama de Despliegue

Un modelo de despliegue es un modelo de objeto que describe la distribución física del sistema, en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo. El modelo de despliegue se utiliza como entrada fundamental en las actividades de diseño e implementación, debido a que la distribución del sistema tiene una influencia principal en su diseño (Jacobson, y otros, 2000).

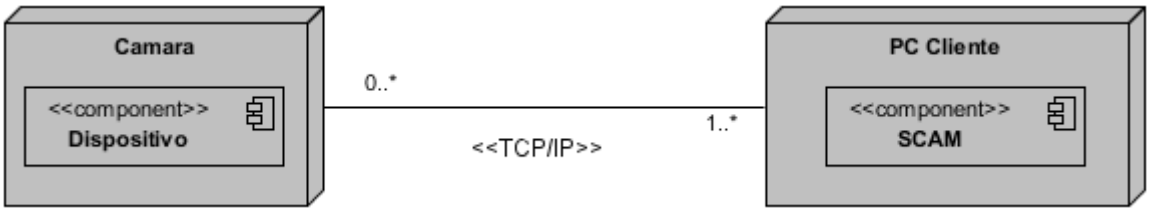



Figura 6: Diagrama de Despliegue.

## 4.2 Modelo de Componente

Los modelos de componente son usados para estructurar el modelo de implementación en términos de subsistemas de implementación y mostrar las relaciones entre los elementos de implementación. Además de mostrar las dependencias entre elementos de implementación y los correspondientes elementos de diseños que son implementados (Jacobson, y otros, 2000).

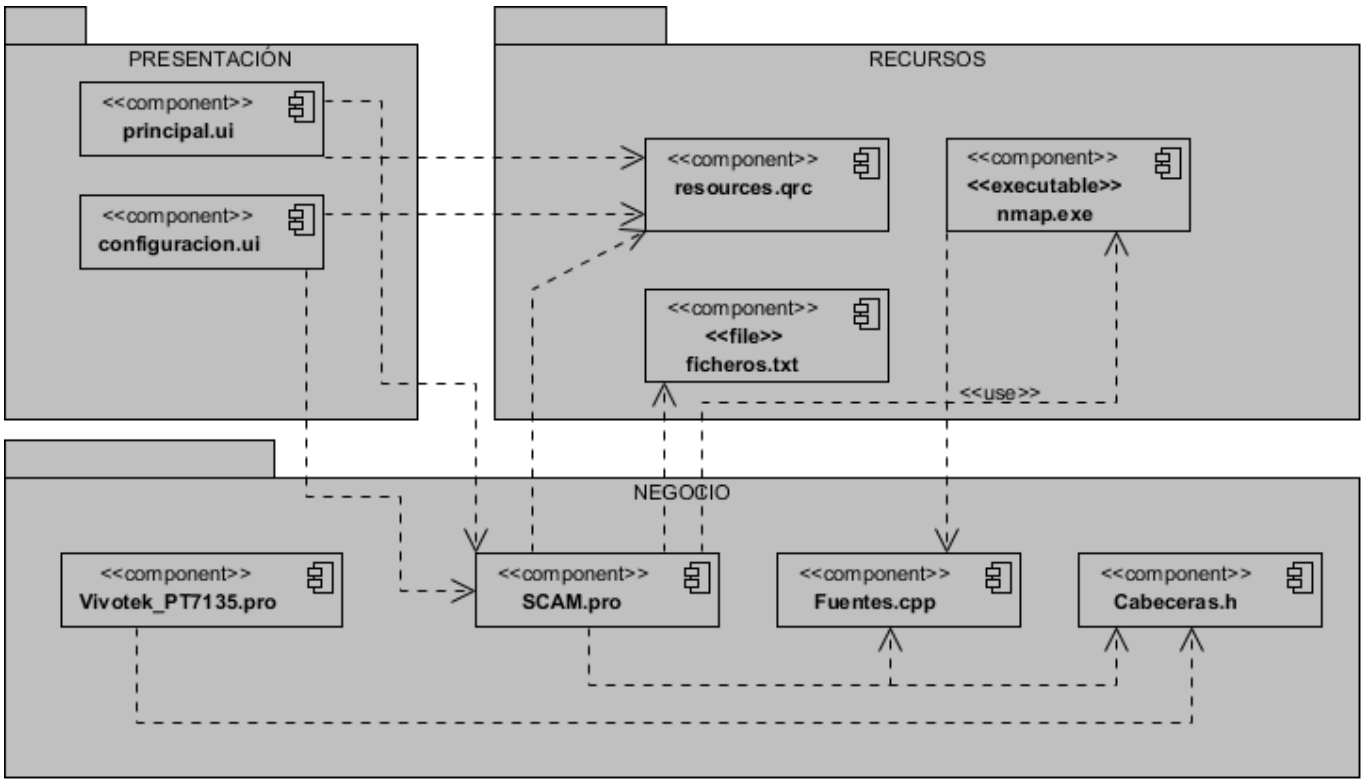



Figura 7: Diagrama de Componente.

## 4.3 Estándares de codificación

Es importante que durante la codificación se consideren permanentemente los siguientes criterios de calidad:

- ✓ Facilidad de Comunicación: Proporcionar al usuario entradas y salidas fácilmente asimilables.
- ✓ Auto-descripción: Proporcionar en el código, explicaciones sobre la implantación realizada.
- ✓ Simplicidad: La implantación realizada debe hacerse de la forma más comprensible posible.

Los estándares utilizados para el desarrollo del sistema fueron los siguientes:

**Notación Camello**

Consiste en escribir los identificadores de variables o funciones con la primera letra en mayúscula y el resto en minúscula o viceversa, en dependencia de la variante que escojas. El nombre se debe porque los identificadores recuerdan las jorobas de un camello.

```
QString InicRango,FinRango;
bool banderita;
for(int i=0; i<=4;++i)
{
     InicRango += rang->DevolverInic().split(".").at(i);
     FinRango += rang->DevolverFin().split(".").at(i);
    qDebug()<<FinRango;

if((InicRango.toInt()) > (FinRango.toInt()))
return true;

else
    banderita = false;
}
if(InicRango!=FinRango)
return banderita;
```

Figura 8: Ejemplo de notación camello.

## 4.4 Prueba del Software

Las pruebas son una actividad en la cual el sistema o los componentes son ejecutados bajo ciertas condiciones o requerimientos especificados donde los resultados son observados y registrados para realizar una evaluación de algún aspecto del sistema o del componente. Entre las buenas prácticas para realizar pruebas al sistema es que dichas pruebas se realicen por un actor externo que no sea el desarrollador de la aplicación (Quesada López, 2009).

**Objetivos de las pruebas**

Las pruebas buscan encontrar los posibles fallos en la implementación, en la usabilidad y en la calidad de un programa determinado, para validar su correcto funcionamiento. Entre los objetivos de la realización de las pruebas se tiene:

- ✓ Detectar defectos en la aplicación.
- ✓ Verificar que todos los requisitos se han implementado satisfactoriamente.
- ✓ Identificar y asegurar que los defectos encontrados se han corregido antes de entregar el producto al usuario o cliente final.
- ✓ Diseñar los casos de prueba que permitan identificar diferentes clases de errores utilizando la menor cantidad de tiempo y esfuerzo.

### *4.4.1 Métodos de Pruebas.*

La prueba de **caja negra** se refiere a las pruebas que se llevan a cabo sobre la interfaz del software; o sea, los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integri-

dad de la información externa se mantiene. La prueba de la **caja blanca** del software comprueba los caminos lógicos del software, proponiendo casos de prueba que ejerciten conjuntos específicos de condiciones y/o bucles; se puede examinar el estado del programa en varios puntos para determinar si el estado real coinciden con el esperado o mencionado. Para validar el correcto desempeño del sistema implementado, se escogió la prueba de caja negra, para validar que el sistema responde adecuadamente a los diferentes requisitos del diseño.

### *4.4.2 Pruebas de caja negra.*

Esta prueba examina algunos aspectos del modelo fundamentalmente del sistema sin tener mucho en cuenta la estructura interna del software. Se centran principalmente en los requisitos funcionales del software. Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos.

La prueba de Caja Negra no es una alternativa a las técnicas de prueba de la Caja Blanca, sino un enfoque complementario que intenta descubrir diferentes tipos de errores a los encontrados en los métodos de la Caja Blanca (Jacobson, et al., 2000).

Según Pressman, 1997, la prueba de caja negra intenta encontrar errores de las siguientes categorías:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a bases de datos externas.
- Errores de rendimiento.
- Errores de inicialización y de terminación.

### *4.4.2 Técnicas de prueba de Caja Negra.*

Para desarrollar la prueba de caja negra existen varias técnicas, entre ellas están: [Pressman, 2000]

- Técnica de la Partición de Equivalencia: esta técnica divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del *software*.
- Técnica del Análisis de Valores Límites: esta Técnica prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
- Técnica de Grafos de Causa-Efecto: es una técnica que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

**Técnica Partición de Equivalencia**

Dentro del método de Caja Negra, la técnica de la Partición de Equivalencia es una de las más efectivas, pues permite examinar los valores válidos e inválidos de las entradas existentes en el software, descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente, se dirige a la definición de casos de pruebas que descubran clases de errores, reduciendo así en número de clases de prueba que hay que desarrollar.

Una partición equivalente es una técnica de prueba de Caja Negra, que divide el dominio de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. El diseño de estos casos de prueba para la partición equivalente se basa en la evaluación de las clases de equivalencia.

El diseño de casos de prueba para la partición equivalente, se basa en una evaluación de las clases de equivalencia para una condición de entrada. Una clase de equivalencia representa un conjunto de estados válidos o inválidos para condiciones de entrada. Regularmente, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición lógica (Pressman, 1997).

**Casos de Prueba**

Los casos de prueba son el diseño de un conjunto de variables o condición de entrada, para demostrar que los requisitos de una aplicación se cumplen parcial o completamente. Estos casos de prueba normalmente se describen por cada caso de uso del sistema y contiene un identificador, la descripción del caso de uso y variables asociadas ha dicho caso de uso.

**Caso de Prueba # 1 “CU Configurar Rango de IP”.**

**Descripción general:** El CU inicia cuando el administrador desea añadir los rangos de IP. El CU termina cuando se han añadido los rangos de IP.

**Condiciones de Ejecución:** -

**Sección 1: "Configurar Rango de IP"**

Tabla 3: Diseño de caso de prueba Sección "Configurar Rango de IP".

| Escenario | Descripción | Rango de IP | Respuesta del sistema | Flujo central |
|---|---|---|---|---|
| EC 1.1 Configurar Rango de IP de forma correcta. | El sistema muestra un formulario con los campos necesarios para adicionar el rango de IP. El administrador llena los campos correctamente y luego selecciona la opción: Guardar. | V<br>10.54.13.1-10.54.13.254 | El sistema muestra un listado con el rango de IP. | Clic izquierdo en el botón Configurar Rangos IP. |
| EC 1.2 Configurar Rango de IP de forma incorrecta. | El sistema muestra un formulario con los campos necesarios para adicionar el rango de IP. El administrador llena los campos incorrecta. | I<br>10.00.42.77-10.34.00.63 | El sistema muestra un mensaje de error. | Clic izquierdo en el botón Configurar Rangos IP. |

**Descripción de las variables.**

Tabla 4: Descripción de las variables del caso de prueba # 1: CU Configurar Rango de IP.

| No | Nombre de campo | Clasificación | Valor Nulo | Descripción |
|---|---|---|---|---|
| 1 | Rango de IP | Campo de texto | No | Este campo permite entrar una cadena de caracteres. |

**Caso de Prueba # 2 "Configurar Rango de Puerto".**

**Descripción general:** El CU inicia cuando el administrador desea añadir los rangos de Puertos. El CU termina cuando se han añadido los rangos de Puertos.

**Condiciones de Ejecución:** -

**Sección 2: "Configurar Rango de Puerto"**

Tabla 5: Diseño de caso de prueba Sección "Configurar Rango de Puerto".

| Escenario | Descripción | Rango de Puerto | Respuesta del sistema | Flujo central |
|---|---|---|---|---|
| EC 1.1 Configurar Rango de Puerto de forma correcta. | El sistema muestra un formulario con los campos necesarios para adicionar el rango de Puerto. El administrador llena los campos correctamente y luego selecciona la opción: Aceptar. | *V*<br>*60-80* | El sistema muestra un listado con el rango de Puerto. | Clic izquierdo en el botón de configuración. |
| EC 1.2 Configurar Rango de Puerto de forma incorrecta. | El sistema muestra un formulario con los campos necesarios para adicionar el rango de Puerto. El administrador llena los campos incorrecta. | *I*<br>*00-00* | El sistema muestra un mensaje de error. | Clic izquierdo en el botón de configuración. |

**Descripción de las variables.**

Tabla 6: Descripción de las variables del caso de prueba # 1: CU Configurar Rango de Puerto.

| No | Nombre de campo | Clasificación | Valor Nulo | Descripción |
|---|---|---|---|---|
| 1 | Rango de Puerto | Campo de texto | No | Este campo permite entrar una cadena de caracteres. |

**Caso de Prueba # 3 "Cargar Plugin".**

**Descripción general:** El CU inicia cuando el administrador carga el plugin.

**Condiciones de Ejecución:** -

**Sección 3: "Cargar Plugin"**

Tabla 7: Diseño de caso de prueba Sección "Cargar Plugin".

| Escenario | Descripción | Cargar Plugin | Respuesta del sistema | Flujo central |
|---|---|---|---|---|
| EC 1.1 Identificar cámaras IP de forma correcta. | El sistema muestra un seleccionador de archivos. El administrador carga el archivo y luego selecciona la opción: Aceptar. | V<br>pluginVivotek_IP8331 | El sistema muestra el plugin cargado. | Clic izquierdo en el botón de configuración. |
| EC 1.2 Identificar cámaras IP de forma incorrecta. | El sistema muestra un seleccionador de archivos. El administrador carga el archivo y luego selecciona la opción: Aceptar. | I<br>(Campo vacío) | El sistema muestra un mensaje de error. | Clic izquierdo en el botón de configuración. |

**Caso de Prueba # 4 "CU Identificar cámaras IP".**

**Descripción general:** El CU inicia cuando el administrador desea escanear la red y encontrar las cámaras IP conectadas a la red de Datos. El CU termina cuando se han encontrado las cámaras IP, mostrándose un listado de las mismas.

**Condiciones de Ejecución:** Debe ser añadido los rangos de puertos, rangos de IP y cargado el plugin para poder escanear la red.

**Sección 4: "Identificar cámaras IP"**

Tabla 8: Diseño de caso de prueba Sección "Identificar cámara IP".

| Escenario | Descripción | Respuesta del sistema | Flujo central |
|---|---|---|---|
| EC 1.1 Identificar cámaras IP de forma correcta. | El usuario selecciona la opción de escanear después de haber configurado el rango de direcciones IP y de puerto. | El sistema escanea la red de datos y muestra un listado de las cámaras y sus datos de configuración. | Clic izquierdo en el botón escanear. |
| EC 1.2 Identificar cámaras IP de forma incorrecta. | El usuario desea escanear la red sin haber configurado antes el rango de direcciones IP y de puerto. | El sistema tiene el botón desactivado. | Clic izquierdo en el botón escanear. |

### 4.3.3 Resultado de las pruebas.

Luego de realizar el diseño de caso de prueba para el CU Identificar cámaras IP, se obtuvo un total de 4 casos de prueba, se obtuvieron 4 no conformidades en la primera iteración, de ellas 3 altas y 1 baja. Las mismas se muestran en la siguiente tabla:

Tabla 9: Resultado de la primera iteración de pruebas.

| No. | No conformidad | CP | Clasificación |
|---|---|---|---|
| 1. | Se detectó que los valores de entrada de los campos Rango de IP y Rango de Puertos no validaban correctamente los datos. | 1 y 2 | Alta |
| 2. | En la lista de Rango de IP y Rango de Puertos permitía añadir un rango que ya existía. | 1 y 2 | Alta |
| 3. | Cuando la aplicación estaba escaneando no se podía dar click sobre otro botón para acceder a otras funcionalidades. | 4 | Alta |
| 4. | Cada vez que iniciaba la aplicación era necesario insertar nuevamente los rangos de IP y de puertos a escanear. | 1 y 2 | Baja |

Como resultado de la segunda iteración se obtuvieron 2 no conformidades, las 2 bajas. Estas no conformidades se muestran en la siguiente tabla:

Tabla 10: Resultado de la segunda iteración de pruebas.

| No. | No conformidad | CP | Clasificación |
|---|---|---|---|
| 1. | No existía una barra de estado que informara al usuario el estado en que se encontraba la aplicación. | 4 | Baja |
| 2. | No existía un botón que permitiera detener la búsqueda. | 4 | Baja |

En la tercera iteración no se encontraron no conformidades, lo que significa que la aplicación desarrollada cumple con la calidad requerida para el sistema de video vigilancia Suria en su versión Qt.

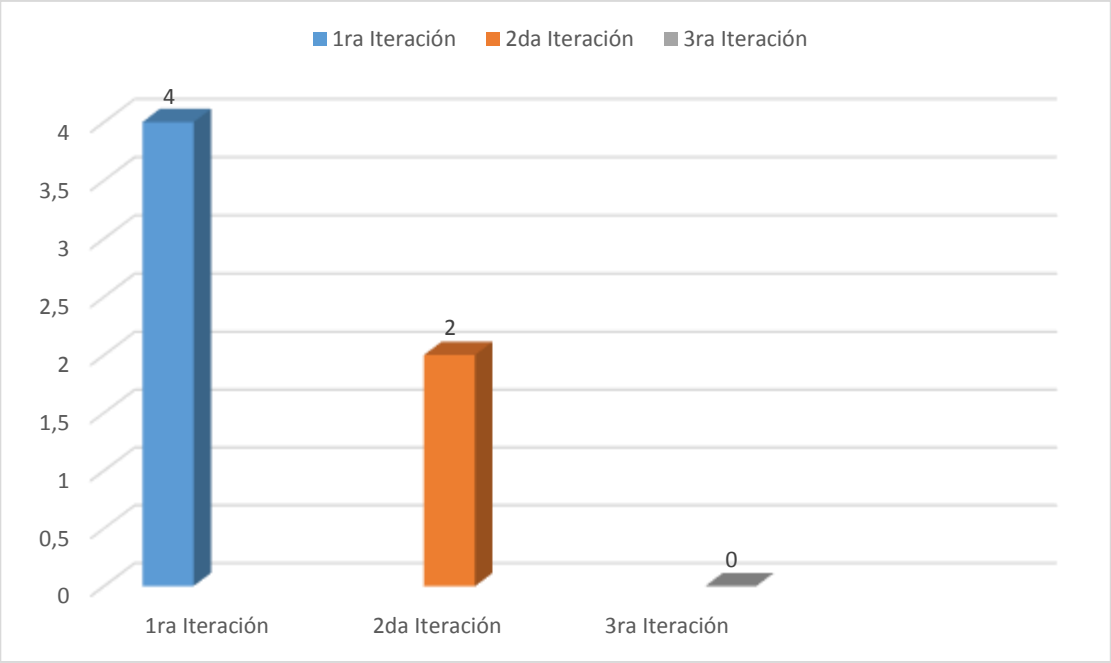



Figura 9: Resultado de las pruebas de caja negra.

## Conclusiones

En este capítulo se definieron temas relacionados con el proceso de implementación y prueba, del módulo de administración web para el sistema de video vigilancia Suria versión Qt:

- Se realizaron los diagramas de despliegue y de componentes de implementación, proporcionando una vista de la implementación del sistema y garantizando una descripción detallada de su estructura.
- Se realizaron las pruebas de caja negra, permitiendo comprobar el correcto funcionamiento de la aplicación en cuanto a las funcionalidades definidas, garantizando que la aplicación desarrollada cumpla con la calidad requerida para ser utilizada por el administrador del sistema de video vigilancia Suria versión Qt.

## Conclusiones Generales

El resultado principal de este trabajo de investigación, es la obtención de una aplicación capaz de identificar las cámaras IP presentes en una red de Datos. Se cuenta con una documentación completa y bien explicada, que se fue fundamentando durante la fase de diseño y de implementación, siendo guiada por la metodología RUP. Con el propósito de darle cumplimiento al objetivo general y basado en la problemática expuesta, se llevaron a cabo satisfactoriamente cada una de las tareas que fueron perfiladas al comienzo de la investigación:

- ✓ El análisis de otros sistemas identificadores, demostró que no existe a nivel nacional e internacional una aplicación que se pueda integrar al sistema de video vigilancia Suria en su versión Qt; sin embargo, el estudio de estos permitió identificar funcionalidades que posteriormente fueron incluidas en el identificador, lográndose obtener un producto que cumpliese con los requisitos del cliente.
- ✓ Los artefactos generados en la fase de diseño e implementación, permitieron establecer un único flujo de trabajo, así como la obtención de una arquitectura sólida para el sistema, garantizándose un mejor soporte en la aplicación y una mejor organización durante el desarrollo del *software*.
- ✓ Las pruebas realizadas al módulo desarrollado, permitieron detectar errores y mejorar la calidad del *software*, garantizando con esto el cumplimiento de las funcionalidades definidas y la entrega de un producto que cumpla con las condiciones requeridas.

## Recomendaciones

Al concluir el desarrollo de este trabajo de diploma se recomienda:

- ✓ Continuar con la implementación de plugins con el fin de que el sistema sea capaz de soportar más modelos de cámara.
- ✓ Automatizar la entrada de datos obtenidos de las cámaras al Sistema de Video Vigilancia Suria en su versión Qt.

# Bibliografía

**Asesorias y computadores LTDA. 2005-2012.** Asesoría Informática. *Asesoría Informática.* [En línea] 2005-2012. [Citado el: 11 de diciembre de 2012.] http://www.aseinformatica.com/camarasip.php.

**Diccionario de la Lengua Española. 2008.** Definiciones.es. [En línea] 2008. [Citado el: 26 de Noviembre de 2013.] http://definicion.de/.

**Domocam. 2010.** Domocam Soluciones Tecnológicas. [En línea] 2010. [Citado el: 10 de Diciembre de 2013.] http://www.domocam.es/camarasipdefinicion.html.

**Feibel, Werner. 1996. .** The Encyclopedia of Networking. Sybex, : Alameda,CA, 1996.

## Referencias Bibliográficas

**De Montigny-Leboeuf, Annie y Massicotte, Frédéric. 2004.** Passive Network Discovery for Real Time. Ottawa, Canadá : Communication Research Centre Canada, 2004.

**Lenguajes de programación. 2009.** Lenguajes de programación. [En línea] 2009. [Citado el: 30 de Noviembre de 2013.] http://www.lenguajes-de-programacion.com/lenguajes-de-programacion.shtml.

**López Monge, Alejandro. 2005.** *Aprendiendo a programar con Libpcap.* 2005.

**—. 2005.** *Aprendiendo a programar con Libpcap.* 2005.

**Urbino, Rafael Jacobo Hidalgo. 2010.** Análisis, diseño e implementación de una herramienta que permita la centralización de la información gestionada por el módulo Resultados de la Colección Multisaber. Cuidad de la Habana: s.n., 2010 : s.n., 2010.

**Arkin, Ofir. 2005.** Deficiencies of Active Network Discovery Systems. s.l. : Insightix Ltd, 2005.

**Avallone, S. 2010.** A Topology Discovery Module based on a Hybrid Methodology. s.l. : University of Napoli —Federico, 2010.

**Axis Communications. 2010.** Axis Camera Station. [En línea] 2010. [Citado el: 29 de Noviembre de 2013.] http://www.axis.com/files/datasheet/ds_acs_44712_en_1110_lo.pdf.

**AxxonSoft. 2003.** Axxon. [En línea] 2003. [Citado el: 04 de 04 de 2014.] http://www.axxonsoft.com/.

**Beasley, Jeffrey S. 2009.** Networking, Second Edition. s.l. : Prentice Hall, 2009.

**Burbeck., Steve.** Application programming in Smalltalk-80: How to use Model-View-Controller (MVC). [En línea] University of Illinois in Urbana-Champaign. http://st-www.cs.uiuc.edu/users/smarch/st-docs/mvc.html..

**Carrillo Pérez, Isaías, Pérez González, Rodrigo y Rodríguez Martín, Aureliano David. 2008.** Metodología de desarrollo del software. 2008.

**Claramunt, Javier Lambert. 2012.** *Implementación de un prototipo funcional para automatizar el proceso de pruebas de Caja Blanca del departamento Señales Digitales del Centro GEySED.* La Habana : s.n., 2012.

**Company Profile. 2010.** Milestone Systems. [En línea] 2010. [Citado el: 30 de Noviembre de 2013.] http://www.milestonesys.com/.

**De la Torre, Anibal. 2006.** [En línea] 2006. [Citado el: 27 de marzo de 2014.] http://www.adelat.org/media/docum/nuke_publico/lenguajes_del_lado_servidor_o_cliente.html..

**Debich. 2014.** Out Tribute to Debian GNU/Linux. [En línea] 2014. [Citado el: 1 de mayo de 2014.] http://www.debianhackers.net/nmap-escaner-de-puertos.

**Definición.De. 2014.** Definicion.De. [En línea] 2014. [Citado el: 30 de abril de 2014.] http://definicion.de/seguridad/.

**EMC Corporation. 2005.** EMC Smarts IP Availability Manager. 2005.

**Free Download Manager.ORG. 2007.** Free Download Manager. [En línea] 2007. [Citado el: 30 de Noviembre de 2013.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%2.

**Fyodor. December 2010.** *Remote OS detection via TCP/IP stack fingerprinting.* s.l. : Phrack, December 2010. 54.

**Glosario.NET. 2007.** HispaNetwork Publicidad y Servicios, S.L. [En línea] 16 de marzo de 2007. [Citado el: 30 de Noviembre de 2013.] http://tecnologia.glosario.net/terminos-viricos/lenguaje-de-rogramaci%F3n-9768.html.

**Hernández León, Rolando Alfredo y Coello González, Sayda. 2011.** *El proceso de investigación científica.* Ciudad de La Habana: Universitaria del Ministerio de Educación Superior : s.n., 2011. ISBN 978-959-16-1307-3.

**Honeywell, Ip. 2010.** Honeywell Ip Solution. [En línea] 2010. [Citado el: 29 de Noviembre de 2013.] http://www.honeywellipsolutions.com/us/index.html.

**IDE. 2012. 2012.** Entornos de desarrollo integrado. *Concepto de IDE.* [En línea] 2012. [Citado el: 29 de Noviembre de 2013.] http://petra.euitio.uniovi.es/~i1667065/HD/documentos/Entornos%20de%20Desarrollo%20Integrado.pdf..

**Institución Tecnológico de Colima.** Tutorial de Fundamentos y Base de Datos. [En línea] [Citado el: 27 de marzo de 2014.] http://labredes.itcolima.edu.mx/fundamentosbd/sd_u2_1.htm.

**Instituto Superior de Formación y Recursos en Red para el Profesorado. 2008.** Diseño de Materiales Multimedia. [En línea] 2008. [Citado el: 25 de Noviembre de 2013.] http://www.ite.educacion.es/formacion/materiales/107/cd/video/video0105.html.

**Jacobson, Ivar y Booch, Grady y Rumbaugh, James. 2000.** El Proceso Unificado de Desarrollo de Software. Madrid : s.n., 2000.

**Larman, Craig. 1999.** *UML y Patrones. Introducción al Análisis y Diseño Orientado a Objetos.* s.l. : Prentice Hall, 1999. 2da Edición.

**—. 2003.** *Una introduccion al análisis y diseño orientado a objetos y al proceso unificado.* 2003. 2da Edicion.

**Louden, Kenneth C. . 2004.** Lenguajes de programación: principios y prácticas. Mexico : s.n. : s.n., 2004.

**Martínez, Alejandro y Martínez, Raúl. 2010.** Guía a Rational Unified Process. Universidad de Castilla la Mancha: s.n. : s.n., 2010.

**Mora Saavedra, Elizabeth. 2011.** Aplicación para descubrir Cámaras IP en una Red de Área Local. 2011.

**Pressman. 1997.** *Real Time Systems Design and Analysis .* 1997.

**—. 2005.** *Real Time Systems Design and Analysis .* 2005.

**Programación en castellano. 2011.** Qt Creator, un completo entorno de . [En línea] 2011. [Citado el: 28 de Noviembre de 2013.] http://www.programacion.com/noticia/qt_creator-_un_completo_entorno_de_desarrollo_1723..

**—. 2011.** Qt Creator, un completo entorno de desarrollo. [En línea] 2011. [Citado el: 30 de Noviembre de 2013.] http://www.programacion.com/noticia/qt_creator-_un_completo_entorno_de_desarrollo_1723.

**Qt Project Hosting. 2014.** Qt Proyect. [En línea] 2014. [Citado el: 24 de Marzo de 2014.] http://qt-project.org/doc/qt-4.8/model-view-programming.html.

**Quesada López, Juan Antonio . 2009.** *Pruebas del software.* 2009.

**Rumabugh, James, Jacobson, Ivar y Booch, Grady. 1998.** El Lenguaje Unificado de Modelado.Manual de Referencia. s.l. : s.l. : Adisson-Wesley, 1998.

**Sandoval N, Francisco A. 2010.** [En línea] 01 de Septiembre de 2010. [Citado el: 25 de Noviembre de 2013.] http://fralbe.com/2010/09/01/video-vigilancia-ip/.

**Security, GVI. 2000.** GVI Security Solutions. [En línea] 2000. [Citado el: 29 de Noviembre de 2013.] http://www.linkedin.com/company/gvi-security-solutions.

**Sommerville, Ian. 2005.** *Ingeniería de Software 7ma Edicion.* Madrid : PEARSON EDUCATION, 2005.

**Torres Faría, Daniela A. 2009.** Protocolos de Internet(ARP,RARP,TCP/IP). s.l. : Universidad Simón Bolívar, 2009.

**Treurniet, J. 2004.** An Overview of Passive Information Gathering Techniques for Network Security. Ottawa,Canada : Defence R&D Canada – Ottawa, 2004.

**Uml.Slideshare. 2011.** Visual Paradigm for Uml. [En línea] Slideshare, 2011. [Citado el: 2013 de Noviembre de 2013.] http://www.slideshare.net/vanquishdarkenigma/visual -paradigm-for-uml..

**Visual Paradigm for UML. 2011. .** Herramienta CASE para modelamiento UML. [En línea] 2011. . [Citado el: 29 de Noviembre de 2013.] http://www.todoprogramas.com/programalinux/visualparadigmforuml.

**Whyte, David. 2008.** Network scanning detection etrategies for enterprise networks. Ottawa,Canada : School of , 2008.

**WordPress. 2014.** Mundo Informático. [En línea] 2014. [Citado el: 10 de marzo de 2014.] http://infow.wordpress.com/category/patrones-de-disenogof/.

# Anexos

**Anexo 1. Entrevista realizada.**

Fecha: Noviembre 2013

Entrevistados: Reynier Pupo Gómez

Preguntas:

1. ¿Considera usted importante el desarrollo de una aplicación identificadora de cámaras IP? ¿Qué beneficios traería para el proyecto Video Vigilancia una aplicación de este tipo?
2. ¿Qué requisitos considera que sean importantes a desplegar por la aplicación?

Respuestas:

1- Una aplicación identificadora de cámaras sería de gran importancia ya que reduciría en gran medida el tiempo de despliegue del sistema Suria. Evitaría el desgaste que constituye la enorme labor de visitar las cámaras una por una para recopilar sus datos y añadirlos a la aplicación. A la vez, traería grandes beneficios para el proyecto Video Vigilancia partiendo de la situación de que hay dos estados en los que podemos encontrar el lugar de turno con relación al aplicación: uno es que lleguemos y estén instaladas todas las cámaras con que se va trabajar y el otro que las cámaras se vayan a ir colocando según se avance en el despliegue del software, en ambos casos es de vital importancia conocer las cámaras que están instaladas hasta el momento y como acceder a ellas. Por lo tanto, es un gran aporte para el proyecto y entre los beneficios que podría traer por solo mencionar algunos está:

- El hecho de que se logra una mayor concentración del equipo de trabajo en la instalación del software lo que repercute sin dudas en la calidad de la entrega final del mismo y en su posterior mantenimiento.
- También podría ahorrar recursos al proyecto mirando desde el punto de vista de que no habría que contratar ningún personal para realizar el proceso que de forma manual se hacía antes de esta aplicación.

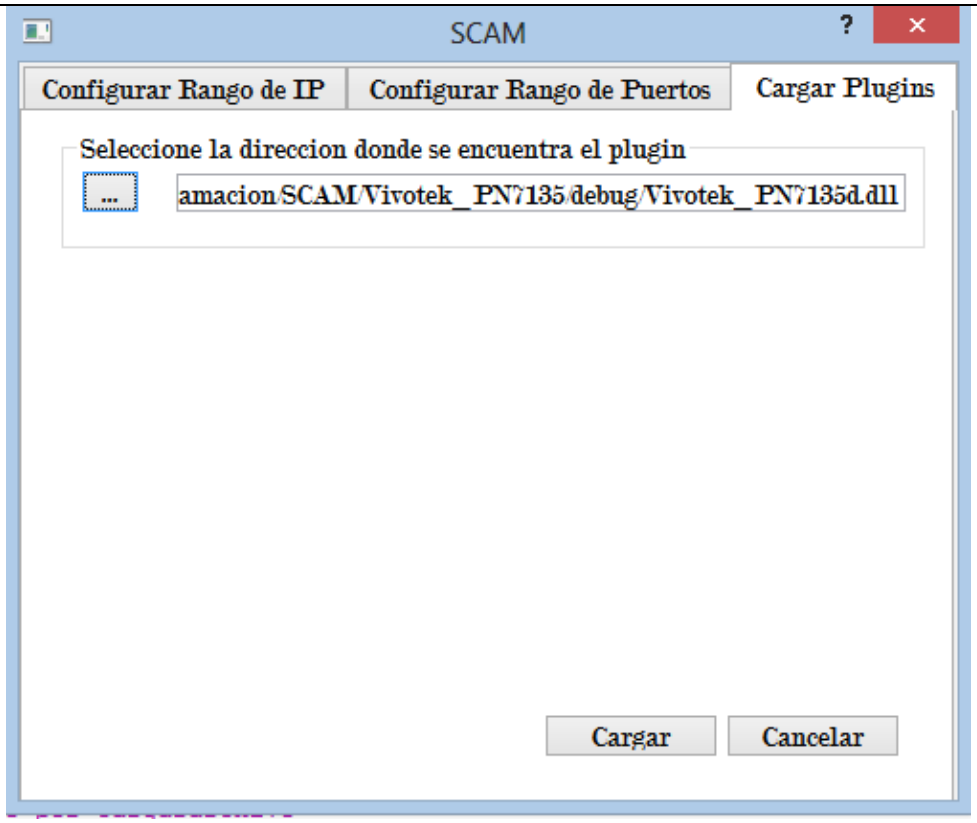
2- Entre las funcionalidades que mencionaron los entrevistados se ven varias opiniones en común, algunas de ellas son:

- Identificar cámaras.
- Gestionar rangos de IP a buscar.
- Administrar cámaras.

**Anexo 2. Descripción de los Casos de Uso**

CU Cargar plugins

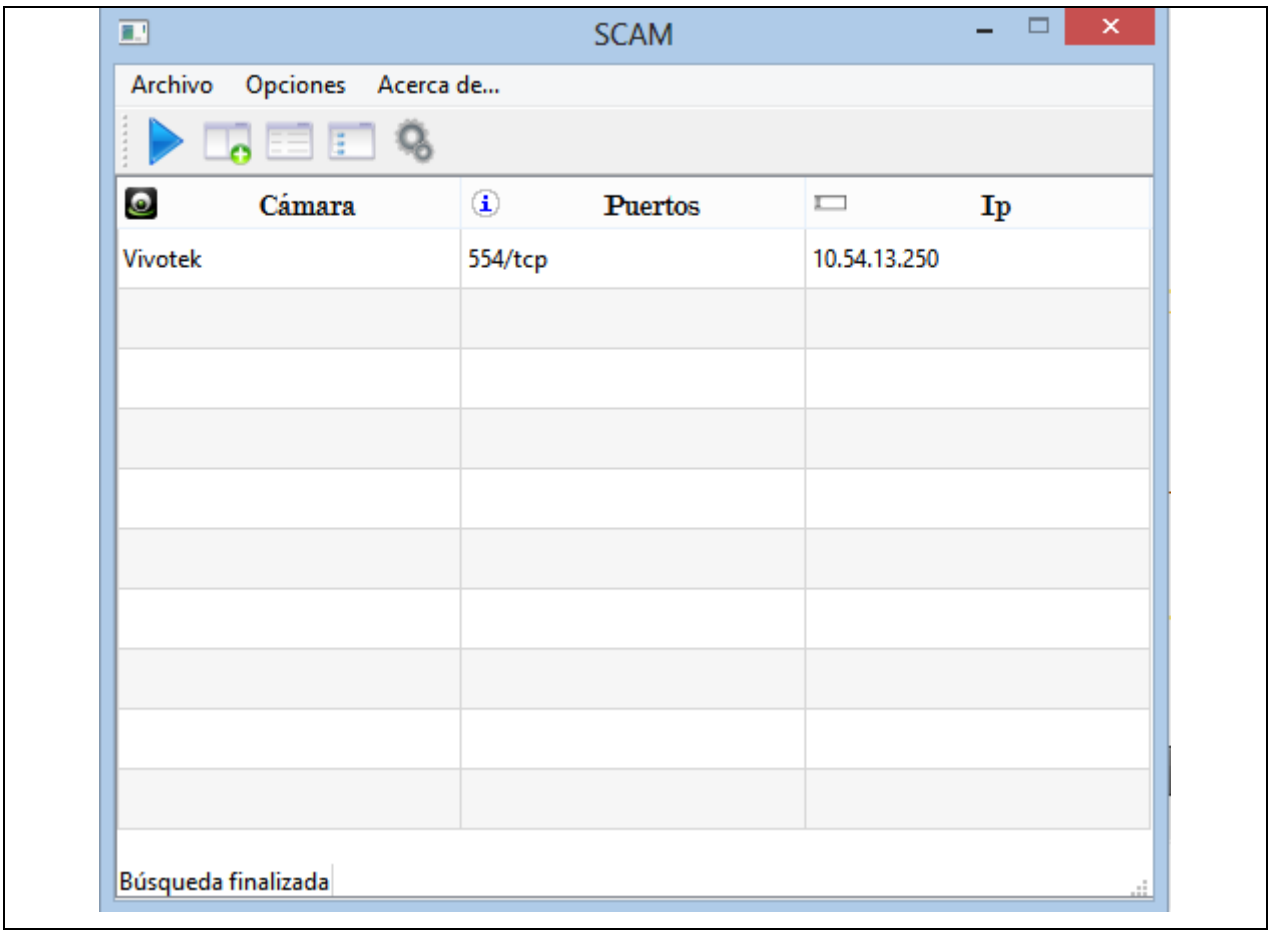
| **CU** | **Cargar plugins** | |
|---|---|---|
| **Actores** | Administrador | |
| **Resumen** | El caso de uso se inicia cuando el Administrador selecciona la opción "Cargar Plugin". | |
| **Referencias** | **R.F1** | |
| **Prioridad** | - | |
| **Precondiciones** | - | |
| **Postcondiciones** | - | |
| **Flujo de eventos** | | |
| **Flujo básico:** | | |
| | **Actor** | **Aplicación** |
| | 1. El actor selecciona la opción "Cargar Plugin".<br>2. El actor selecciona la dirección donde se encuentra el plugin. | 3. Adiciona el plugin a la lista de plugin que se utilizará para el escaneo de la red. |
| **Prototipo de Interfaz:** | | |
| | | |




CU Identificar cámaras IP

| | | |
|---|---|---|
| **CU** | Identificar Cámaras IP | |
| **Actores** | Administrador | |
| **Resumen** | El caso de uso se inicia cuando el Administrador selecciona la opción "Identificar cámaras". | |
| **Referencias** | R.F2 | |
| | | |
| **Prioridad** | Crítico | |
| **Precondiciones** | El actor debe haber seleccionado un rango de direcciones IP y un rango de puertos. | |
| **Postcondiciones** | Se muestra un listado con las cámaras encontradas, sobre las cuales se pueden realizar otras operaciones. | |
| **Flujo de eventos** | | |
| **Flujo básico:** | | |
| | **Actor** | **Aplicación** |
| | | |

| | | |
|---|---|---|
| | 1. El actor selecciona la opción “Identificar cámaras”. | 2. La aplicación escanea por el rango de direcciones IP o por el rango de puertos.<br>2.1. La aplicación compara cada dirección IP y los puertos de los rangos seleccionados.<br>2.2. Si en una dirección IP hay una cámara, obtiene los datos de esta, y los añade al listado de cámaras.<br>2.3. La aplicación actualiza la interfaz mostrando el listado de cámaras modificado. |
| **Flujos alternos** | | |
| | | |
| | **Actor** | **Aplicación** |
| | | 2.3. La aplicación muestra un mensaje informando que ninguna cámara fue encontrada. |
| **Prototipo de Interfaz:** | | |
| | | |

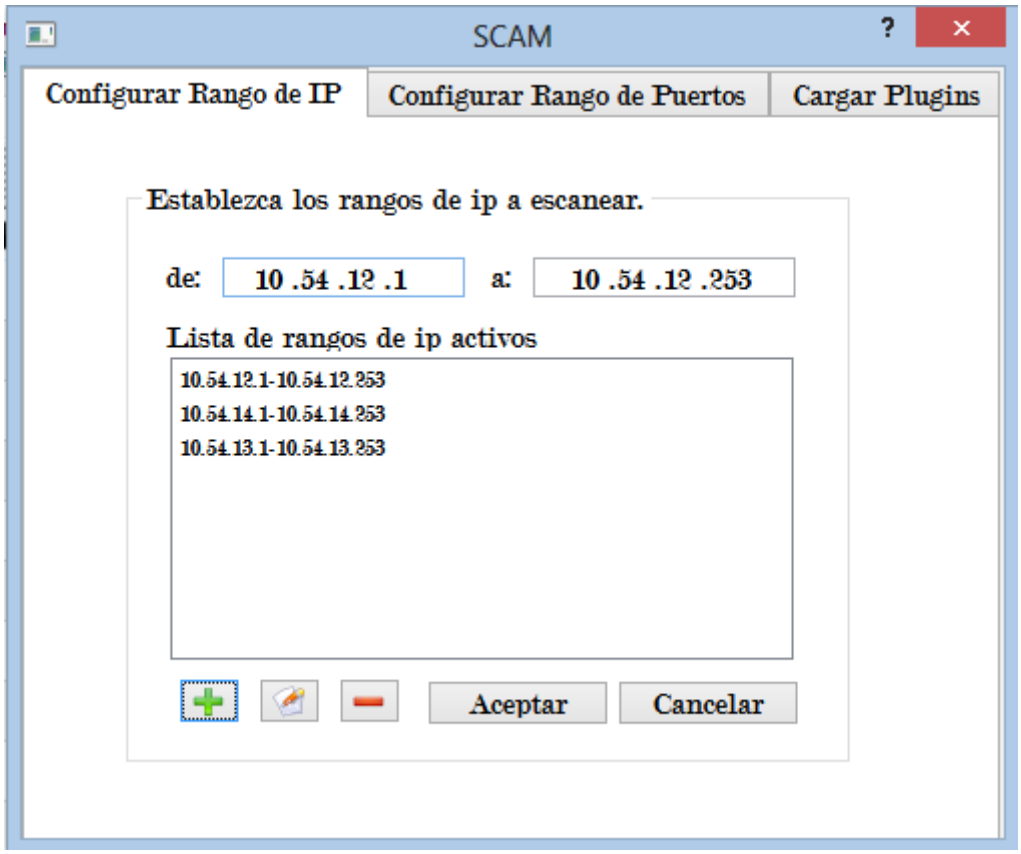



CU Configurar Rangos de direcciones IP

| | | |
|---|---|---|
| **CU** | Configurar rangos de direcciones IP | |
| **Actores** | Administrador | |
| **Resumen** | El caso de uso se inicia cuando el Administrador selecciona la opción "Configurar rangos de IP". | |
| **Referencias** | R.F3, R.F4, R.F5, R.F6 | |
| **Prioridad** | Crítico | |
| **Precondiciones** | | |
| **Postcondiciones** | Se actualiza el fichero de configuración de rangos de direcciones IP y quedan seleccionados los rangos en los que se efectuará el descubrimiento. | |
| **Flujo de eventos** | | |
| **Sección "Adicionar Rango de direcciones IP"** | | |
| | **Actor** | **Aplicación** |

| | | |
|---|---|---|
| | 1. El actor introduce los datos y presiona el botón ¨Adicionar¨. | 2. La aplicación comprueba que no existen campos vacíos.<br>3. La aplicación comprueba que el rango introducido es válido. |
| | 4. El actor selecciona la opción ¨Guardar¨ | 5. La aplicación comprueba que el rango no exista.<br>6. La aplicación actualiza el fichero de los rangos con el rango introducido por el usuario. |

**Prototipo de Interfaz:**




**Flujos alternos**

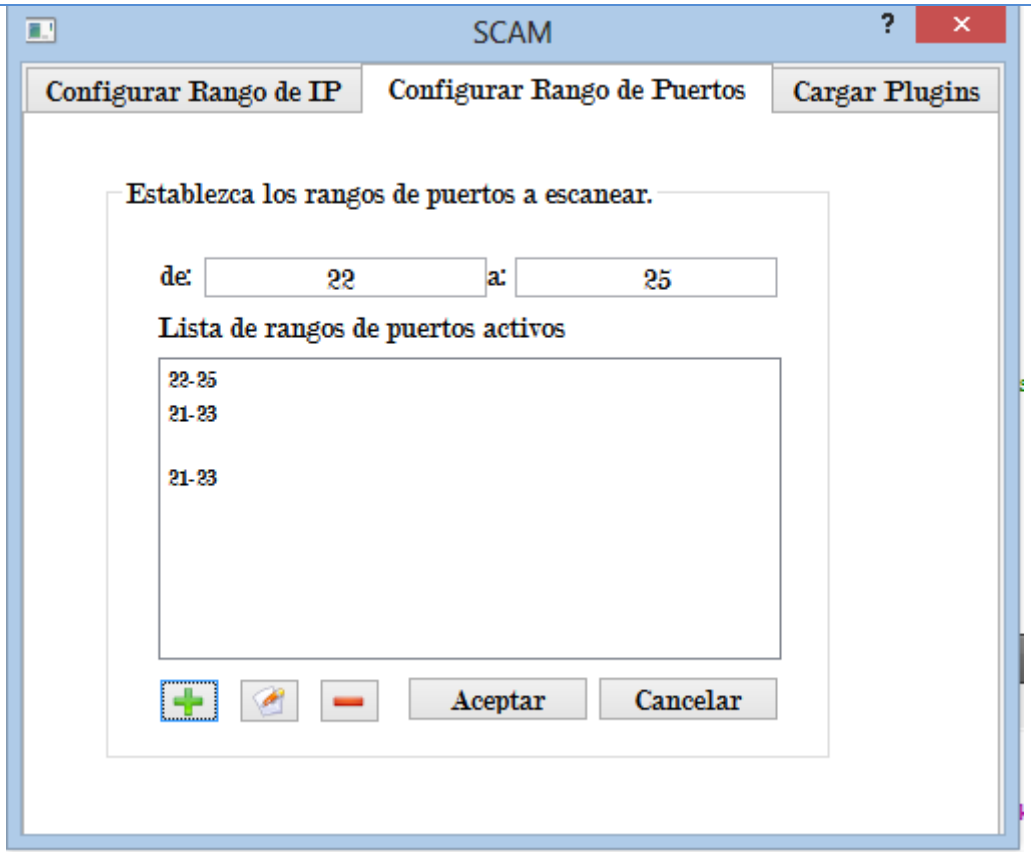
| | Actor | Aplicación |
|---|---|---|
| | | 4. La aplicación muestra un mensaje de error informando que hay campos vacíos. |

| | | |
|---|---|---|
| | | 5. La aplicación muestra un mensaje de error informando que el rango introducido no es válido.<br>6. La aplicación muestra un mensaje de error informando que el rango ya existe. |
| **Sección "Eliminar Rango de direcciones IP"** | | |
| | **Actor** | **Aplicación** |
| | 1. El actor selecciona un rango de IP y marca la opción "Eliminar". | 2. La aplicación muestra un mensaje de confirmación<br>3. Elimina el rango de IP. |
| **Prototipo de Interfaz:** | | |
| | | |
| **Sección "Modificar Rango de direcciones IP"** | | |
| | **Actor** | **Aplicación** |
| | 1. El actor selecciona un rango de IP y marca la opción "Modificar rango de IP". | 2. La aplicación muestra un cuadro de texto con el rango seleccionado. |
| | 3. El actor introduce los datos y da ¨Aceptar¨. | 4. El sistema comprueba que el campo no está vacío.<br>5. El sistema comprueba que el rango introducido es válido. |
| | 6. El actor selecciona la opción ¨Guardar¨ | 7. El sistema comprueba que el rango no exista.<br>8. El sistema actualiza el fichero de los rangos con el rango modificado por el usuario. |
| **Prototipo de Interfaz:** | | |
| | | |
| **Flujos Alternos** | | |
| **Acción del Actor** | **Respuesta de la Aplicación** | |

| | |
|---|---|
| | 4. El sistema muestra un mensaje de error informando que hay campos vacíos.<br>5. El sistema muestra un mensaje de error informando que el rango introducido no es válido.<br>6. El sistema muestra un mensaje de error informando que el rango ya existe. |

CU Configurar Rangos de Puertos

| | | |
|---|---|---|
| **CU** | Configurar rangos de Puertos | |
| **Actores** | Administrador | |
| **Resumen** | El caso de uso se inicia cuando el Administrador selecciona la opción "Configurar Rangos de Puertos". | |
| **Referencias** | R.F7, R.F8, R.F9, R.F10 | |
| **Prioridad** | - | |
| **Precondiciones** | | |
| **Postcondiciones** | Se actualiza el fichero de configuración de rangos de Puertos y quedan seleccionados los rangos en los que se efectuará el descubrimiento. | |
| **Flujo de eventos** | | |
| **Sección "Adicionar Rangos de Puertos"** | | |
| | **Actor** | **Aplicación** |
| | 1. El actor introduce los datos y presiona el botón ¨Adicionar¨. | 2. La aplicación comprueba que no existen campos vacíos.<br>3. La aplicación comprueba que el rango introducido es válido. |
| | 4. El actor selecciona la opción ¨Guardar¨ | 5. La aplicación comprueba que el rango no exista.<br>6. La aplicación actualiza el fichero de los rangos con el rango introducido por el usuario. |
| **Prototipo de Interfaz:** | | |




| Flujos alternos | | |
|---|---|---|
| | **Actor** | **Aplicación** |
| | | 4. La aplicación muestra un mensaje de error informando que hay campos vacíos.<br>5. La aplicación muestra un mensaje de error informando que el rango introducido no es válido.<br>6. La aplicación muestra un mensaje de error informando que el rango ya existe. |
| **Sección "Eliminar Rango de Puertos"** | | |
| | **Actor** | **Aplicación** |
| | 1. El actor selecciona un rango de Puertos y marca la opción "Eliminar". | 2. La aplicación muestra un mensaje de confirmación<br>3. Elimina el rango de IP. |

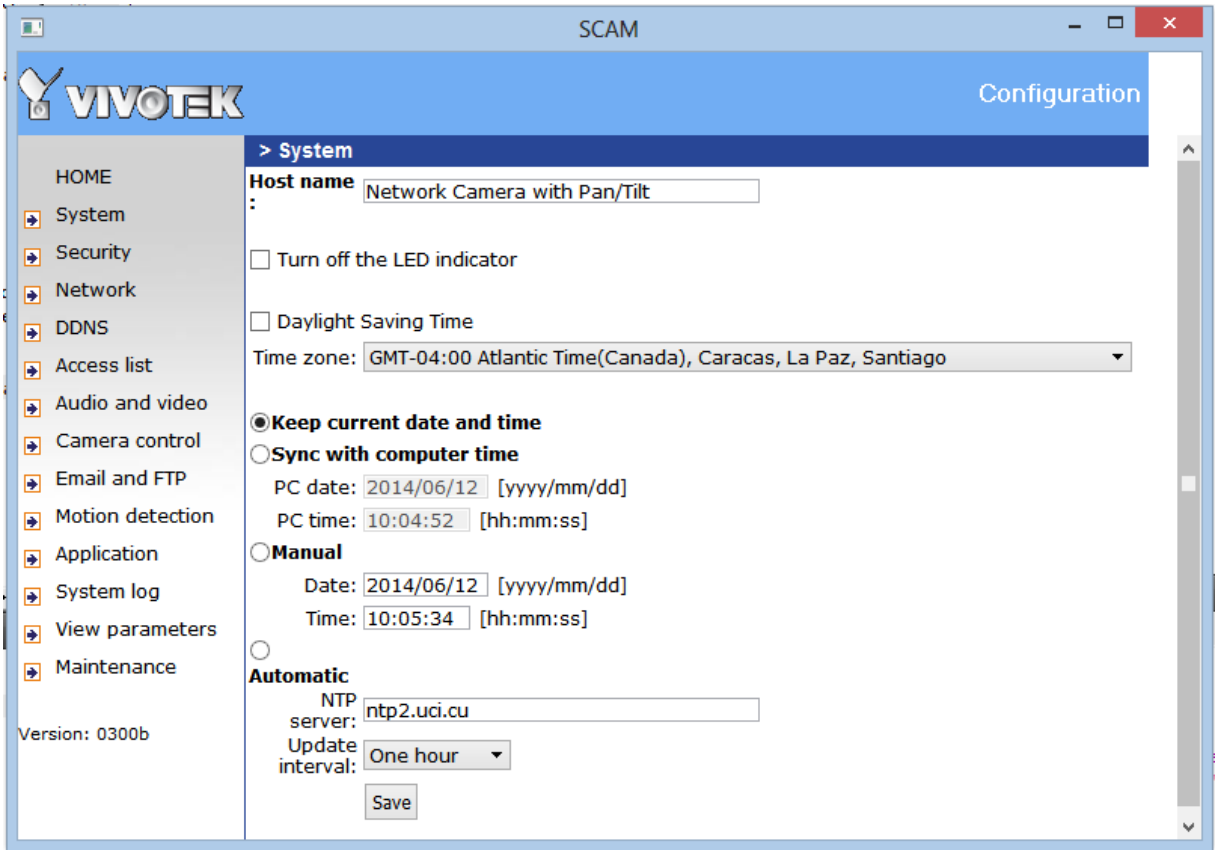
| | | |
|---|---|---|
| | | |
| **Sección "Modificar Rango de Puertos"** | | |
| | **Actor** | **Aplicación** |
| | 1. El actor selecciona un rango de Puerto y marca la opción "Modificar rango de Puerto". | 2. La aplicación muestra un cuadro de texto con el rango seleccionado. |
| | 3. El actor introduce los datos y da ¨Aceptar¨. | 4. El sistema comprueba que el campo no está vacío.<br><br>5. El sistema comprueba que el rango introducido es válido. |
| | 6. El actor selecciona la opción ¨Guardar¨ | 7. El sistema comprueba que el rango no exista.<br><br>8. El sistema actualiza el fichero de los rangos con el rango modificado por el usuario. |

| **Flujos Alternos** | |
|---|---|
| **Acción del Actor** | **Respuesta de la Aplicación** |
| | 4. El sistema muestra un mensaje de error informando que hay campos vacíos.<br>5. El sistema muestra un mensaje de error informando que el rango introducido no es válido.<br>6. El sistema muestra un mensaje de error informando que el rango ya existe. |

CU Modificar Datos del dispositivo

| | |
|---|---|
| **CU** | Modificar Datos del dispositivo |
| **Actores** | Administrador |
| **Resumen** | El caso de uso se inicia cuando el Administrador selecciona la opción "Modificar Datos del dispositivo". |
| **Referencias** | R.F11 |
| **Prioridad** | - |
| **Precondiciones** | |
| **Postcondiciones** | Se actualiza el fichero de configuración de las cámaras IP. |
| **Flujo de eventos** | |
| **Sección "Modificar datos de la cámara"** | |

| | Actor | Aplicación |
|---|---|---|
| | 1. El actor selecciona una cámara y elije la opción “Modificar datos de la cámara”. | 2. La aplicación muestra una ventana pidiendo el usuario y otra para la contraseña para acceder a la cámara.<br>3. La aplicación muestra una ventana donde estará cargada la página de configuración de la cámara. |
| **Prototipo de Interfaz:** | | |



## Anexo 3. Diagramas de secuencia

### *Diagrama de Secuencia del CU Configurar Rango de IP*

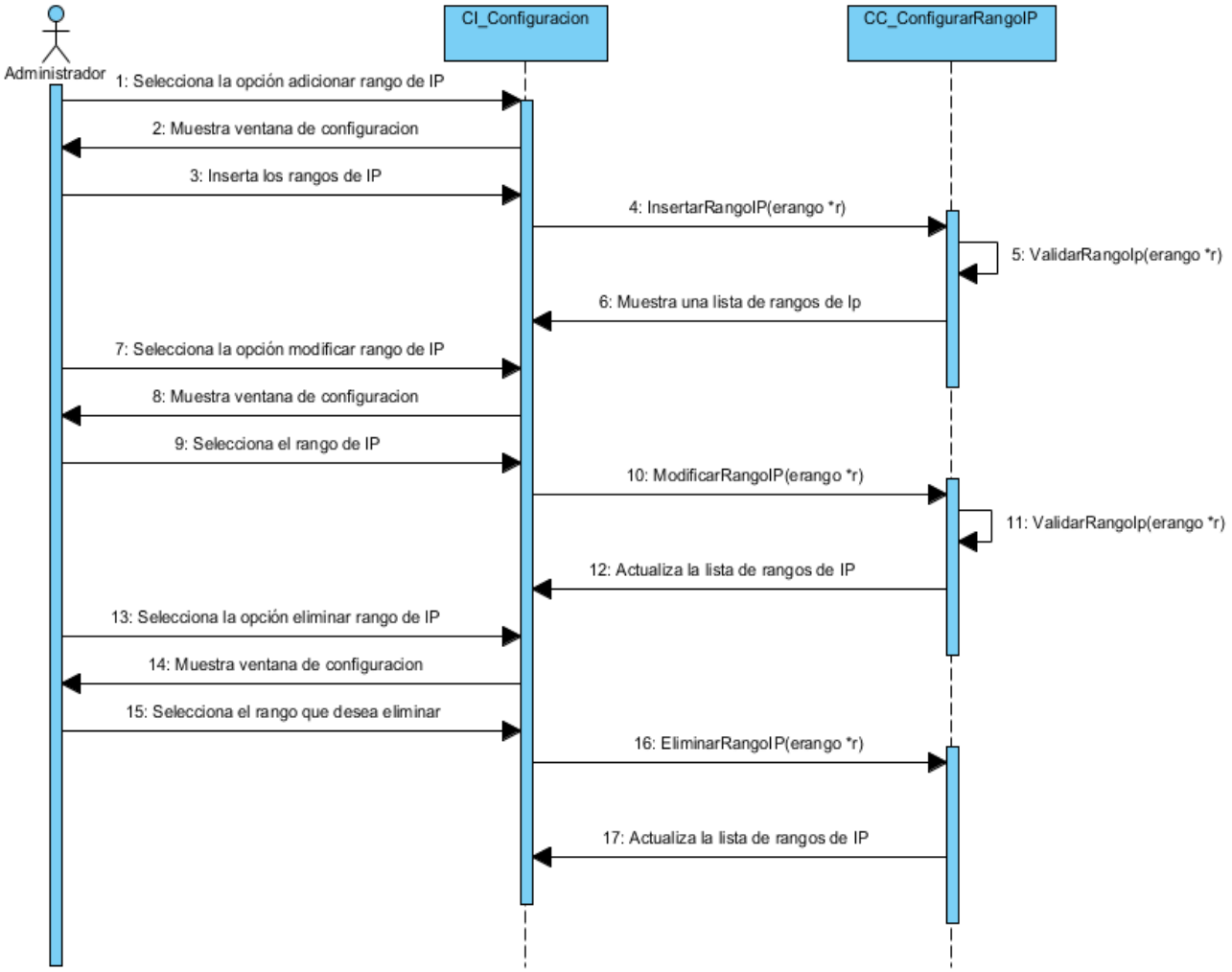


***Diagrama de Secuencia del CU Configurar Rango de Puerto***

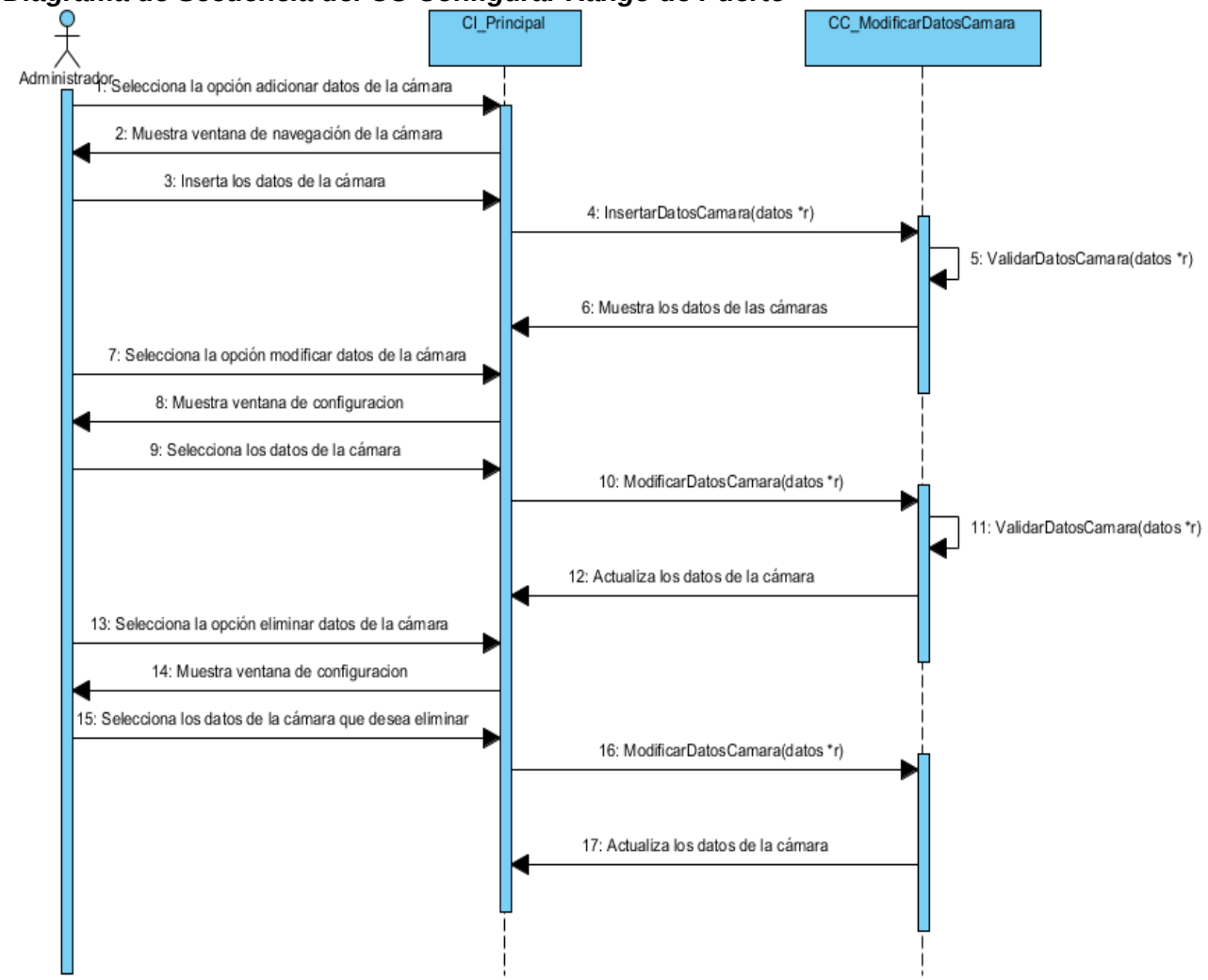
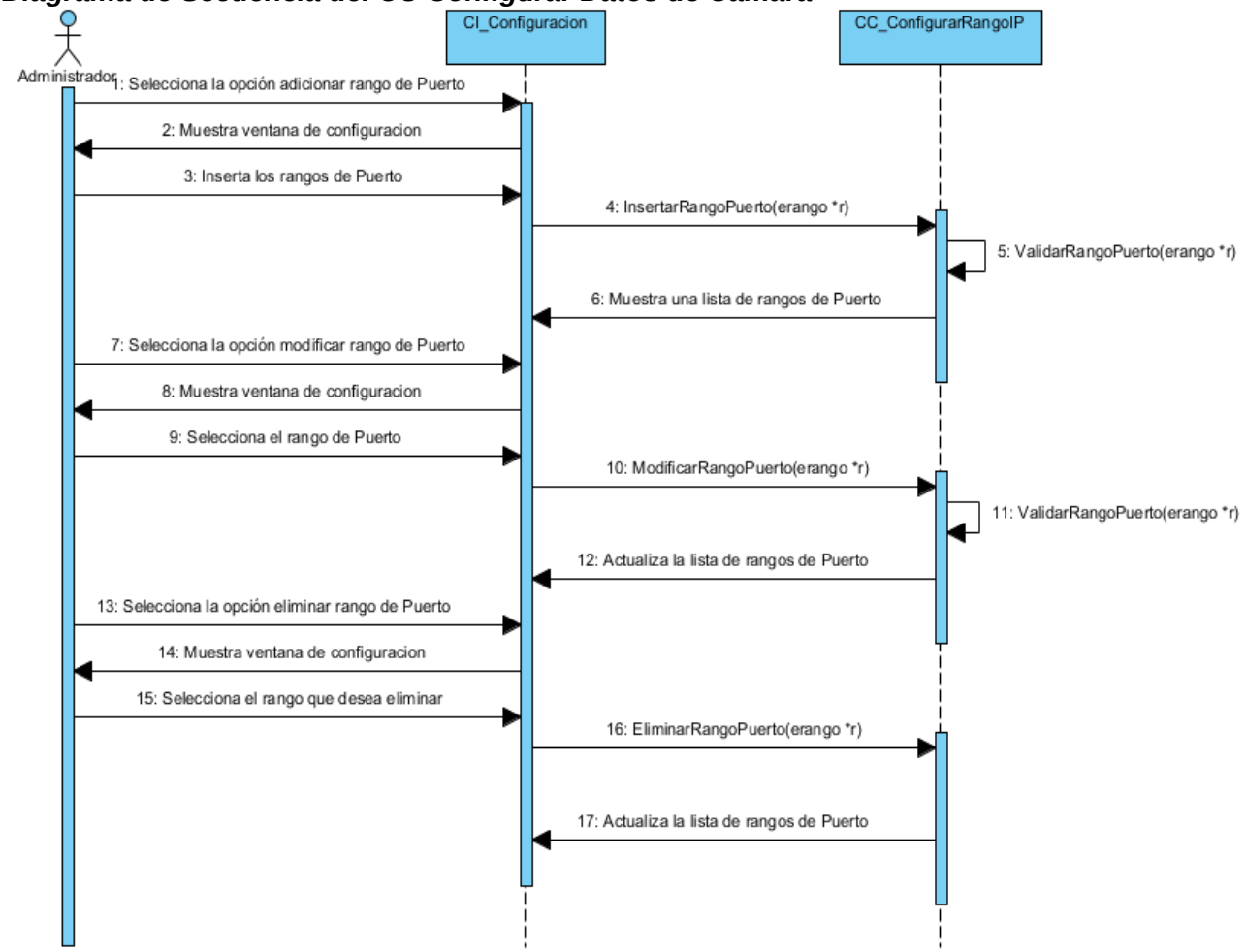


***Diagrama de Secuencia del CU Configurar Datos de Cámara***



## Glosario

**CASE:** Las herramientas CASE (Ingeniería de Software Asistida por Computadora) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste del mismo en términos de tiempo y de dinero.

1