Facultad 4

Centro de Informática Industrial

Tema: Componente gráfico para la visualización de cámaras de seguridad en el HMI del SCADA SAINUX 2.0.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Oscar García Quintana.

Tutor: Ing. Henry Marcelo Cabrera Robles.

Co-Tutora: Ing. Claudia González Fernández

Curso docente: 2016-2017.

Declaración de Autoría

Declaro ser el autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste se firma la presente a los _____ días del mes de _____ del año _____.

Firma del autor

Oscar García Quintana

Firma del tutor

Ing. Henry Marcelo Cabrera Robles

Firma del tutor

Ing. Claudia María González Fernández


"El hombre se descubre cuando se mide con un obstáculo."
Antoine De Saint Exupéry

Agradecimientos

A mis padres por ser mi razón de ser, y hacer de mí una persona de bien. Gracias por estar siempre a mi lado en los buenos y malos momentos, gracias por aconsejarme y guiarme en el camino de la vida.

A mi familia por su apoyo incondicional y confianza en todo momento de la carrera.

A mis tutores por la gran ayuda que me brindaron durante el desarrollo de este trabajo, a Henry por animarme en los malos momentos y ayudarme en todo los momentos que lo necesité, a Claudia por guiarme en el transcurso de la realización de la tesis.

A todos mi amigos y compañeros del aula, a todos mis compañeros del dota, del futbol y del edificio, en especial a Miguel, Ileana, Yosvani, Jorgito y Richard por brindarme su ayuda y por compartir conmigo todos los buenos y malos momentos a lo largo de esta carrera.

A los trabajadores del laboratorio 23 de Centro de Informática Industrial por su ayuda y recomendaciones durante la elaboración de este trabajo.

Dedicatoria

Dedico esta tesis a mis padres por estar siempre a mi lado e impulsarme a ser una persona de bien. A mi madre por ser mi guía en la vida, por darme su confianza, comprensión y amor en todo momento de mi vida. A mi padre por ser mi ejemplo a seguir, por sus consejos y enseñanzas a lo largo de mi vida.

Ustedes son la razón por la que hoy me gradúo como Ingeniero en Ciencias Informáticas, tenerlos en vida es la bendición más grande que tengo, los amo.

Índice

Índice de Figuras

Índice de Tablas

Resumen

La informatización de la sociedad es una prioridad en la sociedad cuba, demostrando la voluntad política del país por acercar cada vez más las nuevas tecnologías a la población. La informatización ha alcanzado un elevado desarrollo en diversas esferas a nivel mundial, entre ellas: la automatización de procesos industriales, mejorando la calidad, el rendimiento y la eficiencia. La industria cubana del software presenta un papel protagónico en el proceso de informatización, teniendo como pilar la Universidad de las Ciencias Informáticas, esta cuenta con el Centro de Informática Industrial (CEDIN). En dicho centro, se tiene como principal línea de desarrollo los sistemas de supervisión, control y adquisición de datos (SCADA), y como producto principal el sistema SCADA SAINUX 2.0.

Dentro de los módulos del sistema SCADA SAINUX 2.0 se encuentra el módulo Interfaz Hombre Máquina (HMI), que se encarga de representar en un ordenador los procesos que ocurren en el campo, permitiendo al operador estar en contacto directo con el sistema para realizar la supervisión y el control. El HMI provee una biblioteca de componentes gráficos (CG) que permite recrear de la forma más real posible el proceso que se desea automatizar.

La presente investigación surge a partir de la necesidad de agregar nuevos componentes gráficos al sistema SCADA SAINUX 2.0, brindándole los recursos necesarios para la visualización de cámaras de seguridad en la Interfaz Hombre Máquina, para que disponga de los mecanismos necesarios para la supervisión y el control del sistema, y así minimizar el trabajo manual con las limitaciones en tiempo y recursos que implica en la industria desplegada.

Palabras claves: SCADA SAINUX 2.0, HMI, componentes gráficos.

Introducción

La automatización de los procesos industriales es uno de los mecanismos que permite el desarrollo de la sociedad, en búsqueda del aumento en los niveles de eficiencia, disminución de costos y optimización de los procesos.

A partir de la importancia que representa para el país el desarrollo de un sistema SCADA, la Universidad de las Ciencias Informáticas (UCI), posee el Centro de Informática Industrial (CEDIN) el cual se especializa en soluciones de automatización de procesos industriales. Uno de estas soluciones es el SCADA SAINUX 2.0, cuyo objetivo es brindarle al usuario el acceso completo para supervisar y controlar los procesos que ocurren en el campo.

SCADA proviene de las siglas *Supervisory Control And Data Acquisition* (Adquisición de datos y supervisión de control). Es una aplicación de control de producción, que se comunica con dispositivos de campo y controla el proceso de forma automática desde un ordenador. Proporciona información del proceso a diversos usuarios: operadores, supervisores de control de calidad, supervisión y mantenimiento.

Un sistema SCADA permite la gestión y control de cualquier sistema local o remoto gracias a una interfaz gráfica que comunica al usuario con el sistema. Un sistema SCADA es una aplicación o conjunto de aplicaciones de software especialmente diseñadas para funcionar sobre ordenadores de control de producción, con acceso a la planta mediante la comunicación digital con instrumentos y actuadores, e interfaz gráfica de alto nivel para el operador (pantallas táctiles, cursores y lápices ópticos).

Dentro de los módulos que componen un sistema SCADA se encuentra el modulo Interfaz Hombre Maquina (HMI por sus siglas en ingles). El módulo de HMI en el SCADA se encarga de representar en un ordenador, los procesos que ocurren en el campo, muestra los componentes implicados, los sensores, las estaciones remotas, y el sistema de comunicación dándole al operador total control. Este módulo es el que permite al operador estar en contacto directo con el sistema, realizar la supervisión y el control del proceso en general.

El HMI consta de dos entornos: El entorno de configuración (EC), donde los mantenedores configuran la información específica del área que se desea supervisar y diseñan los despliegues, los cuales haciendo uso de los componentes gráficos permiten simular los

procesos de campo; el entorno de visualización (EV) es donde el operador puede supervisar y controlar la configuración realizada en el EC, interactuando con los componentes gráficos para emitir control sobre el sistema, monitorear los cambios de estado de las variables, gestionar alarmas y generar reportes.

Tanto en el EC como en el EV, el HMI provee una biblioteca de CG que permiten simular los procesos de campo de una forma real. Dentro de los componentes utilizados en un proceso de automatización industrial, es necesario la visualización de cámaras de seguridad. La biblioteca de componentes gráficos del HMI SAINUX 2.0 no provee al operador el componente para dicha visualización. Esto obstaculiza la visualización de situaciones anómalas que se pudiesen presentar y que resultase, peligrosa o preocupante para el proceso a supervisar. También dificulta el control de acceso de personas no autorizadas a determinados sitios donde está desplegado el sistema.

Dada la situación problemática planteada, se llega al siguiente **problema de la investigación:** ¿Cómo permitir la visualización de cámaras de seguridad sobre la Interfaz Hombre Máquina de sistemas SCADA?

Definiéndose como objeto de estudio: Proceso de visualización de cámaras de seguridad en los sistemas SCADAs.

Delimitando como **campo de acción**: Proceso de visualización de cámaras de seguridad en la Interfaz Hombre Máquina del sistema SCADA SAINUX 2.0.

Objetivo general: Desarrollar componentes gráficos que permitan visualizar cámaras de seguridad en el entorno de visualización de la Interfaz Hombre Máquina del sistema SCADA SAINUX 2.0.

Posibles resultados:

1. Componente gráfico para la visualización de cámaras de seguridad.

2. Paleta de componentes gráficos especializada en cámaras de seguridad.

Las siguientes tareas de investigación se proponen para dar cumplimiento a la solución propuesta:

1) Elaboración del marco teórico de la investigación a través del estudio del estado del arte que existe actualmente sobre el tema.
2) Identificación y caracterización de las tecnologías y herramientas que se utilizan para la creación de componentes gráficos en la Interfaz Hombre Máquina de los sistemas SCADA.
3) Selección de las herramientas y tecnologías para la creación de un prototipo que dé solución al problema.
4) Realización del levantamiento de requisitos funcionales y no funcionales.
5) Desarrollo de componentes gráficos para la representación de cámaras de seguridad.
6) Incorporación de los componentes gráficos desarrollados al HMI del SCADA SAINUX 2.0.
7) Validación del funcionamiento de los componentes gráficos desarrollados.

Métodos de investigación.

- **Análisis-síntesis:** Se utilizó para el estudio de los conceptos relacionados con la visualización de cámaras de seguridad en los sistemas SCADA.
- **Histórico-Lógico:** Para el estudio de antecedentes y evolución de los sistemas SCADA.
- **Observación participante**: Utilizado en el seguimiento del desarrollo de componentes de visualización de cámaras de seguridad en sistemas SCADA.
- **Análisis de documentos**: Se puso en práctica para la consulta de literatura especializada relacionada con la teoría de la automatización de procesos industriales.
- **Investigación-acción:** Utilizado para la constante realización de pruebas de concepto y prototipos tanto no funcionales como funcionales.

El presente documento estará estructurado en tres capítulos.

- **Capítulo 1. Fundamentación teórica:** Este capítulo englobará conceptos y teorías asociadas al desarrollo de componentes gráficos en sistemas SCADA, para la automatización de procesos industriales.

- **Capítulo 2. Análisis y diseño**: En este capítulo se analizará y diseñará la arquitectura propuesta para la solución. Exhibiendo los términos asociados al desarrollo de la aplicación.

- **Capítulo 3. Implementación y prueba:** Este capítulo hará énfasis en la implementación, el sistema y las distintas pruebas realizadas, con el fin de comprobar su funcionamiento y robustez.

Capítulo I: Fundamentación Teórica

1. Introducción

Este capítulo hace referencia a los principales conceptos relacionados con el *Streaming* de video y los sistemas SCADA. También se hace alusión a las herramientas y tecnologías, así como, lenguaje de programación, entorno de desarrollo integrado y metodología propuesta para el desarrollo de la solución.

1.1 Conceptos asociados a la investigación

1.1.1 Componente Gráfico u Objeto Gráfico:

Representación gráfica de cada dato, en correspondencia con sus valores.

1.1.2 Despliegue:

Es un resumen esquematizado con la ventaja que permite visualizar la estructura y organización de los elementos que componen el proceso, así como el comportamiento de los mismos, los cuales pueden estar ubicados en un área geográficamente extensa para que el operador monitoree constantemente cada de uno de los componentes que conforman el proceso.

1.1.3 Streaming:

Tecnología de transmisión a través de redes de medios continuos (principalmente audio y vídeo). No existe descarga de información a un disco local. Se envía la información a través de la red y el cliente la reproduce en tiempo real al recibirla.

1.2 Sistemas SCADA

Los sistemas SCADA (del inglés *Supervisory* Control And Data *Acquisition*), son herramientas informáticas que satisfacen la necesidad de intercomunicar al operador con los elementos de control automático, para transmitir las órdenes de éste, facilitar de forma clara y simple los estados de los equipos y almacenar los datos relativos a la producción. Cuando los autómatas que forman el automatismo de la planta pertenecen al mismo proveedor, la comunicación entre estos y el sistema de supervisión y mando suele realizarse con las redes y protocolos propios del fabricante. (1)

Cada vez es más habitual que en una misma planta se hayan instalado autómatas de diferentes proveedores. En este caso, la utilización de una red específica de uno de los fabricantes no resuelve el problema, ya que la utilización de SCADAs y redes propietarias de cada uno de ellos genera una arquitectura caótica, que producirá como resultado islas incomunicadas entre sí. (2)

Un SCADA debe cumplir tres funciones principales:

- Adquisición de datos: Es la capacidad que tiene el sistema de obtener información desde el sistema de control, por ejemplo, sobre valores de temperatura o presión y a diferencia de la "supervisión" los datos son registrados o almacenados para su posterior explotación.
- Supervisión: Significa poder observar o monitorear aquello que sucede en el proceso industrial, el equipo o la maquinaria, por ejemplo, conocer (generalmente de una forma gráfica) si un motor se encuentra encendido o apagado.
- Control: Es la posibilidad de ejecutar comandos; en otras palabras, poder enviar instrucciones hacia el sistema de control, por ejemplo, ordenar al PLC (Controlador Lógico Programable por sus siglas en inglés) que encienda o apague un motor.

1.2.1 Módulos de un SCADA

Los módulos que permiten las actividades de supervisión, adquisición y control de datos son los siguientes:

- **Configuración:** Permite al usuario definir el entorno de trabajo de su SCADA adaptándolo a la aplicación particular que se desea desarrollar.
- **Interfaz gráfico del operador:** Proporciona al operador las funciones de control y supervisión de la planta. El proceso se representa mediante gráficos representativos almacenados en el ordenador de procesos y generados desde el editor incorporado en el SCADA o importados desde otra aplicación durante la configuración del paquete.
- **Módulo de proceso:** Ejecuta las acciones de mando reprogramadas a partir de los valores actuales de variables leídas.
- **Gestión y archivos de datos:** Almacena y procesa ordenadamente los datos, de forma que otra aplicación o dispositivo pueda tener acceso a ellos.

- **Comunicaciones:** Proporciona la transferencia de información entre la planta y la arquitectura hardware que soporta el SCADA, y entre ésta y el resto de elementos informáticos de gestión.

1.2.2 El módulo Interfaz Hombre-Máquina

La Interfaz Hombre-Máquina es el aparato encargado de presentar los datos a un operador (humano) a través de la cual éste controla el proceso. La industria de HMI nació esencialmente de la necesidad de estandarizar la manera de monitorear y de controlar múltiples sistemas remotos, controladores lógicos programables y otros mecanismos de control. Aunque un PLC realiza automáticamente un control pre-programado sobre un proceso, normalmente se distribuyen a lo largo de toda la planta, haciendo difícil recoger los datos de manera manual, los sistemas SCADA se encargan de hacerlo de manera automática.

1.3 Sistemas homólogos

En la actualidad existe una gran variedad de sistemas SCADA que poseen una Interfaz Hombre-Máquina que incluyen una extensión para la supervisión y control de los procesos mediante video vigilancia. Entre los sistemas que se pueden encontrar que cumplen estas características, todos tienen solución con tecnologías bajo licencia propietaria, entre ellos se encuentran:

- ✓ *SCADAdvantage*: Posee un módulo de video vigilancia llamado Downhole Video. Ofrece a los operadores de petróleo y gas una manera de ver los problemas de pozo en tiempo real en lugar de inferir o adivinar sobre ellos, mejorando y agilizando la toma de decisiones. Está basado en tecnologías de Microsoft, como ODBC y OPC(4).
- ✓ *System SIMATIC*: Este sistema scada contiene un módulo de vigilancia llamado WinCC OA VIDEO*,* este permite integrar funciones completas de gestión de vídeo al scada. Esto permite al usuario scada configurar el hardware y el software de vídeo igual que el hardware clásico de automatización. WinCC OA VIDEO se utiliza para visualizar los componentes de vídeo y las cámaras de control y las cámaras PTZ ya sea mediante un panel de control o un teclado y un ratón. Debido a esto, es posible monitorear y controlar exhaustivamente dispositivos complejos(6).

- ✓ *Logwatc*h: Es una solución de monitorización de vídeo autónomo completo que se puede integrar en sistemas scada. El paquete incluye varias interfaces de control de procesos basados en estándares de la industria tales como OPC y Modbus TCP (tecnologías de Microsoft). Estas interfaces permiten el control completo del sistema de vídeo a través de SCADA. Longwatch se compone de dos componentes: el Centro de control de vídeo (VCC) y Motores Longwatch Vídeo (Ives).

Al realizar un análisis de las funcionalidades de los módulos de video vigilancia anteriormente expuestos en busca de una solución homóloga, se arrojó como resultado que a pesar de que los mismos poseen rasgos sugestivos, no cumplen con los requisitos necesarios para ser tomados como solución de la presente investigación. La principal limitante de estos sistemas es que están basados en tecnologías privativas pertenecientes a *Microsoft*, mientras que el sistema SCADA SAINUX 2.0, está basado en tecnologías libres.

1.4 Streaming de video

Se conoce como *Streaming* de video a la transmisión continua e ininterrumpida de vídeo a través de una red, donde, ésta se transmite desde una computadora que funge como servidor, hacia una o varias computadoras que actúan como clientes, recibiendo es *stream* de video enviado desde el servidor. Se puede observar que se menciona un nuevo concepto *Stream*; se puede definir como *Stream* a la información que se envía del origen a uno o varios destinos; mientras que *Streaming* sería la acción de transmitir esa información.(8)

1.4.1 Tipos de Streaming de video

El proceso de *Streaming* se puede dividir en dos categorías, en función de cómo se obtiene la información a difundir: *Streaming* en directo o bajo demanda.

- El *Streaming* en directo es aquel que transmite eventos que están sucediendo justo en el momento de la difusión. Por ejemplo, la transmisión de conciertos o de clases son eventos que típicamente se difunden usando este tipo de *Streaming*. La transmisión de radio y televisión por Internet también tiene estas características, aunque en ocasiones parte de la información que se difunde no parte de un evento en directo (por ejemplo, un programa que ha sido grabado previamente, pero que se va a difundir en un momento determinado).(1)

- En un *Stream* de video bajo demanda, la transmisión del medio empieza desde el inicio del evento a ser reproducido para cada uno de los clientes. El medio a transmitir puede estar ya preparado desde el comienzo del proceso en un fichero comprimido. En este caso no representa una ventaja adicional el disponer de posibilidad de realizar multidifusión en la red, ya que cada cliente recibe una parte distinta del *Stream* y por lo tanto un paquete de datos diferente. (1)

1.5 Protocolos de comunicación

Para realizar *Streaming* se pueden utilizar muchos protocolos de comunicación de red, Entre los más utilizados está el protocolo UDP (del inglés *User Datagram Protocol*), este envía paquetes por la red sin esperar confirmación de entrega al destinatario, esto permite que la transmisión sea más rápida y fluida (1).

Otro protocolo utilizado para el *Streaming* de video es el RTSP (*Real Time Streaming Protocol*), es un protocolo de la capa de aplicación utilizado para controlar la entrega de múltiples flujos multimedia sincronizados, por ejemplo, flujos de audio y vídeo relacionados. Por lo general se utiliza como un control remoto de red para servidores multimedia. El protocolo está destinado a seleccionar canales de entrega (como UDP, UDP de multidifusión y el protocolo de control de transferencias) y para seleccionar un mecanismo de entrega basado en el protocolo de tiempo real (RTP).(10)

1.5.1 Funcionamiento RTP

RTP, es un protocolo de transporte en tiempo real que proporciona las condiciones adecuadas para que transmisión de datos, tales como audio y video, puedan efectuarse. RTP transporta las señales de audio y vídeo de forma codificada mediante paquetes RTP que contienen una cabecera RTP, seguida de esta, las señales de audio y vídeo. Un paquete RTP pasa por la capa UDP, que le añade una cabecera UDP. El conjunto es traspasado a la capa IP, donde se le agrega una cabecera IP. Entonces, el datagrama IP es encaminado hacia el destino. En recepción, el paquete es entregado a la aplicación adecuada(12). RTP (y RTSP) han sido diseñados para ser independientes del protocolo de la capa de transporte o capa de red utilizada.

RTP fue creado con los siguientes propósitos:

- Lograr separación entre datos de control de los de datos multimedia (audio y video).
- Funcionar independiente del protocolo de la capa transporte o red utilizado.
- Escalable.
- Seguro al soportar opciones de cifrado.

1.6 Elementos claves del Streaming de videos

Para que el streaming de video funcione son fundamentales tres partes, comenzando desde dónde se transmite la información, es decir, el servidor; a quién se le enviará la información, es decir, los clientes; y cómo transmitir la información.(1)

1.6.1 Servidor

Es el encargado de transmitir la información a partir de una fuente, el servidor genera datagramas que se envían a través de la red. Al hablar de servidor se hace referencia tanto a hardware, como al posible software que éste incluya.

1.6.2 Cliente

Es cada uno de los dispositivos receptores de la información que se transmite, este puede ser un ordenador o un teléfono móvil.

1.6.3 Medio

Es a través de dónde se enviará la información, si se profundiza un poco más y se llega al nivel en donde se realiza, se podría decir que es en la capa 4 del modelo OSI, es decir, la capa de transporte; para realizar streaming se utilizan datagramas (UDP), permitiendo que la transmisión sea más rápida y fluida.

1.7 Funcionamiento de la tecnología *Streaming*

Esta tecnología funciona de forma muy simple: Primeramente, se establece una conexión entre el cliente (ordenador personal) y el servidor desde el cual se va realizar la transmisión. Una vez establecida la conexión, el servidor empieza a enviar el fichero por partes. El

fichero es recibido por el cliente y va reconstruyendo un buffer[1], en el cual almacena la información.

El buffer se empieza a llenar poco a poco y va mostrando parte del archivo en un reproductor integrado, mientras continua realizando la carga del archivo en memoria. Toda esta operación está completamente sincronizada, para que la reproducción del contenido sea continua, mientras se realiza la descarga.

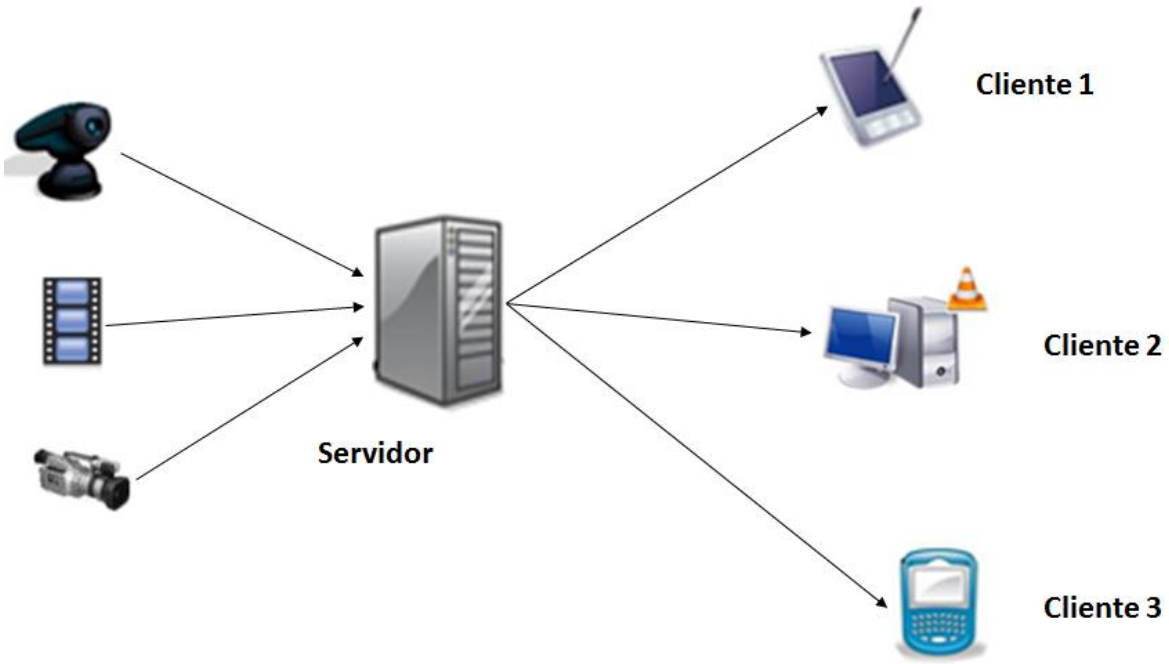

Figura 1 Streaming de video

Fuente: Elaboración propia

1.8 Principales usos de la tecnología *Streaming*

Entre los principales usos que se le puede dar a la tecnología *Streaming*, tenemos los siguientes:

Radio por Internet: Debido al asombroso alcance de Internet, cada día millones de usuarios de aplicaciones *Streaming* están utilizando esta tecnología para masificar la radio en vivo, permitiendo a millones de personas establecer sus propias producciones y compartirlas en la red.

[1]Es un espacio de la memoria reservado para el almacenamiento temporal de información, mientras que está esperando ser procesada.

Televisión por Internet: Al igual que la radio, se está utilizando con mayor frecuencia este tipo de aplicaciones para distribuir producciones en video, surgiendo nuevos talentos y obteniéndose seguidores para diferentes tipos de programas de televisión en vivo.

1.9 Herramientas y tecnologías

Para lograr el desarrollo del componente para la visualización de cámaras de seguridad, se hace necesario tener conocimientos de las tecnologías a las que se recurrirá para alcanzar el objetivo propuesto. Con el objetivo de lograr una correcta integración con el HMI, se utilizaron las siguientes herramientas para el desarrollo de la solución propuesta, las mismas se encuentran definidas en las dependencias del proyecto SCADA SAINUX 2.0.

Como metodología de desarrollo se utilizó AUP en su variante UCI, como marco de trabajo Qt, utilizando el lenguaje de programación C++ y el entorno de desarrollo integrado (IDE por sus siglas en inglés *Integrated Development Environment*) Qt Creator. Para la captura del flujo de video originado por las cámaras se utilizó la biblioteca de desarrollo *libvlc,* esta se puede integrar en una aplicación con el objetivo de obtener las capacidades multimedia que posee la biblioteca. Debido al que el reproductor multimedia VLC, está basado en la biblioteca vlc, la aplicación será capaz de obtener las características de este reproductor en su totalidad(14).

Conclusiones parciales

Con el estudio de los conceptos asociados al *streaming* de video, se ha obtenido una visión más clara de las características a tener en cuenta durante el diseño e implementación de un componente gráfico de cámaras de seguridad.

Por lo que se llega a las siguientes conclusiones:

- ✓ La descripción del objeto de estudio ayudó a comprender el funcionamiento de las cámaras de seguridad en los sistemas SCADA.
- ✓ Se caracterizaron las herramientas y tecnologías a utilizar en el desarrollo de la aplicación enunciándose de cada una de ellas las principales características y las facilidades que ofrecen al equipo de desarrollo, favoreciendo la integración del componente con el sistema SCADA SAINUX 2.0.

Capítulo II: Análisis y diseño de la solución

2. Introducción

Este capítulo tiene como objetivo describir los procesos de análisis y diseño. Se presentan los requisitos funcionales y no funcionales por los que se regirá el desarrollo de la solución propuesta. A partir de estos requisitos, se determinan las historias de usuarios, se muestran los diagramas que describen el funcionamiento del sistema, se diseña y se describe una arquitectura y se exponen los patrones utilizados para el desarrollo de la solución propuesta.

2.1 Modelo de Dominio

Utilizando la notación de UML, el modelo de dominio es utilizado para la representación de las clases así como su jerarquía y las relaciones entre ellas.

Es un artefacto de la disciplina de análisis que puede utilizarse para capturar y expresar el entendimiento ganado en un área bajo análisis como paso previo al diseño de un sistema, ya sea de software o de otro tipo. (16)

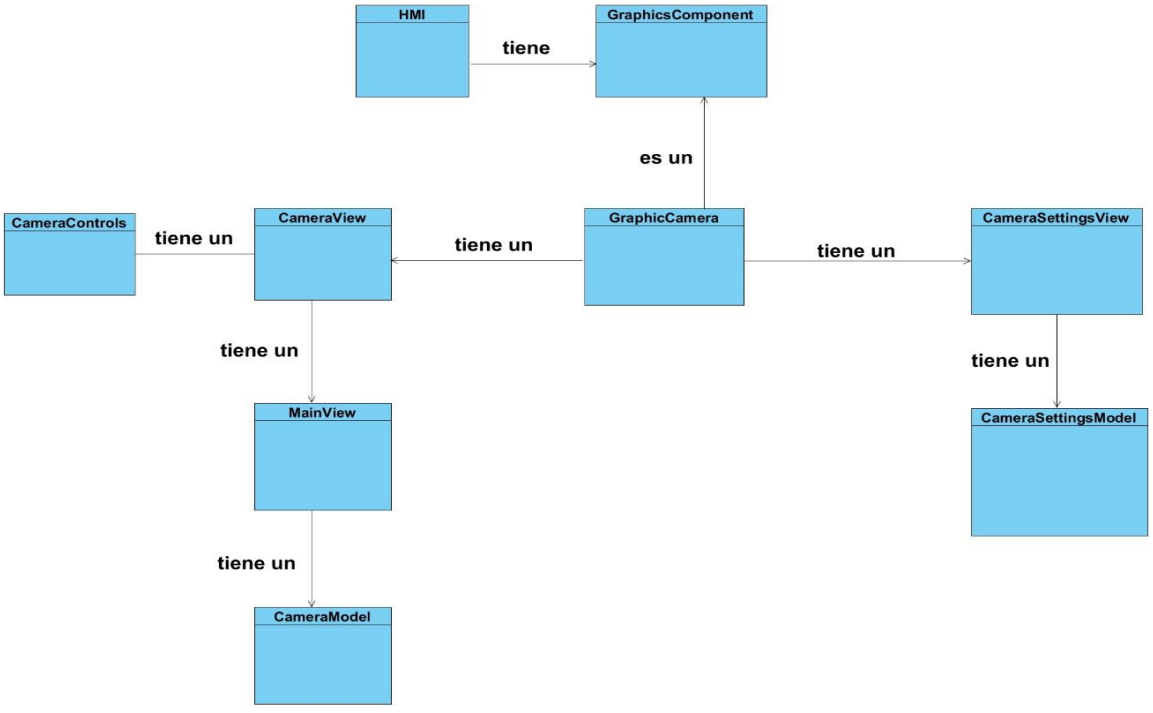

Figura 2 Modelo de Dominio

Fuente: Elaboración propia

Para el diseño del modelo de dominio antes presentado se tuvo en cuenta las siguientes descripciones:

- **HMI:** es el módulo que se encarga de representar al usuario u operador lo que sucede en el proceso.
- **GraphicsComponent:** representación abstracta de algún dispositivo del proceso que se está supervisando.
- **GraphicCamera:** Es la representación del componente tanto en el entorno de configuración (despliegue) como en el entorno de ejecución.
- **CameraView:** Es la clase encargada de apoyar la visualización del componente gráfico.
- **CameraWidget:** Es la clase encargada de contener un grupo de cámaras de seguridad (*MainView*), con los controles asociados para su manejo.
- **MainView:** Es la clase encargada visualizar el contenido de la cámara de seguridad.
- **CameraControls:** Es la clase encarda de mostrar la interfaz con los controles para el manejo de las cámaras de seguridad.
- **CameraSettingView:** Es la clase en encargada de mostrar la interfaz pertinente para la configuración de las cámaras de seguridad.
- **CameraModel:** Es la clase que guarda los datos que deben persistir del componente y los controles de la biblioteca de vlc para el manejo del *Streaming*.
- **CameraSettingModel:** Es la clase encargada de guardar los datos referentes a la configuración de las cámaras de seguridad.

2.1.1 Propuesta del sistema a desarrollar

La solución que se plantea es la de desarrollar un componente gráfico para la visualización de cámaras de seguridad que permita realizar las tareas de vigilancia al proceso industrial a supervisar, con el objetivo, de tener imágenes en tiempo real de posibles fallas en el campo.

El componente contendrá un visualizador de hasta 8 cámaras, donde el operador podrá monitorear el proceso supervisado, ajustando los controles de video y grabando el contenido, para un posterior procesamiento.

2.2 Requisitos

Un requisito es una “condición o capacidad que necesita el usuario para resolver un problema o conseguir un objetivo determinado”. También se aplica a las condiciones que debe cumplir o poseer un sistema o uno de sus componentes, para satisfacer un contrato, una norma o una especificación. Los requisitos pueden verse como, una declaración abstracta de alto nivel de un servicio que el sistema debe proporcionar o una definición matemática detallada y formal de una función del sistema. (18) Una vez identificados los requerimientos que debe tener el software, se procede a clasificarlos, dichos requerimientos se pueden clasificar en Funcionales (los que forman parte del funcionamiento del sistema) y no Funcionales.

A continuación, se muestran los requisitos identificados para el desarrollo de la solución, clasificándose de acuerdo a la metodología seleccionada.

2.2.1 Requisitos Funcionales

RF1. El sistema debe permitir definir nombre del componente.

RF2. El sistema debe permitir configurar color de fondo del componente.

RF3. El sistema debe permitir configurar la opacidad del componente.

RF4. El sistema debe permitir configurar la dimensión de ancho del componente.

RF5. Configurar la dimensión de altura del componente.

RF6. El sistema debe permitir configurar color del borde del componente.

RF7. El sistema debe permitir configurar la posición del componente.

RF8. El sistema debe permitir a rotación del componente.

RF9. El sistema debe permitir que se elimine el componente.

RF10. El sistema debe permitir la configuración del protocolo de comunicación de las cámaras de seguridad.

RF11. El sistema debe permitir la configuración de la dirección IP o URL de las cámaras de seguridad.

RF12. El sistema debe permitir la configuración del nombre de las cámaras de seguridad.

RF13. El sistema debe permitir la configuración del puerto de comunicación de las cámaras de seguridad.

RF14. El sistema debe permitir la configuración de las cámaras de seguridad.

RF15. El sistema debe permitir detener la reproducción de las cámaras de seguridad.

RF16. El sistema debe permitir reproducir una cámara de seguridad.

RF17. El sistema debe permitir grabar el contenido de las cámaras de seguridad.

RF18. El sistema debe permitir el cambio de brillo de las cámaras de seguridad.

RF19. El sistema debe permitir el cambio de contraste de las cámaras de seguridad.

RF20. El sistema debe permitir el cambio de saturación de las cámaras de seguridad.

RF21. El sistema debe permitir el cambio de volumen de las cámaras de seguridad.

2.2.2 Requisitos No Funcionales

Son restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. Los requerimientos no funcionales a menudo se aplican al sistema en su totalidad. Normalmente apenas se aplican a características o servicios individuales del sistema. Estos, como su nombre sugiere, son aquellos requerimientos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes de éste como la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento, entre otros. De forma alternativa, definen las restricciones del sistema como la capacidad de los dispositivos de entrada/salida y las representaciones de datos que se utilizan en las interfaces del sistema.(7)

Software

- El sistema debe funcionar en el Sistema Operativo GNU/Linux en su distribución Debian 8.
- Qt en su versión 5.3.

Hardware

- Memoria RAM: 2 Gigabytes.
- Microprocesador: Intel(R) Core(Tm) 2 Duo.

2.3 Historias de usuario

Una historia de usuario describe una funcionalidad que, por sí misma, aporta valor al usuario. Se compone de una descripción escrita de la historia, usada como recordatorio y para planificar, conversaciones acerca de la historia que sirven para aclarar los detalles y un criterio de aceptación (idealmente automatizado) que permita determinar cuándo la historia ha sido completada.

Tabla 1: *Historia de Usuario #10*

Fuente: Elaboración Propia

Historia de Usuario	
Número: 10	**Nombre de Historia:** Configurar el protocolo de comunicación.
Programador: Oscar García Quintana	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Tiempo estimado:** 0.6
Nivel de Complejidad: Alta	**Tiempo real:** 0.6
Descripción: El sistema proporciona una interfaz para configurar el protocolo de comunicación de las cámaras de seguridad.	

Tabla 2 Historia de Usuario #11

Fuente: Elaboración propia

Historia de Usuario	
Número: 11	**Nombre de Historia:** Configurar la dirección *URL* de las cámaras de seguridad.
Programador: Oscar García Quintana	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Tiempo estimado:** 0.6
Nivel de Complejidad: Alta	**Tiempo real:** 0.6
Descripción: El sistema proporciona una interfaz para configurar la URL o dirección IP de las cámaras de seguridad.	

Tabla 3 Historia de Usuario #13

Fuente: Elaboración Propia

Historia de Usuario	
Número: 13	**Nombre de Historia:** Configurar el puerto de comunicación de las cámaras de seguridad.
Programador: Oscar García Quintana	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Tiempo estimado:** 0.4
Nivel de Complejidad: Alta	**Tiempo real:** 0.4

Descripción: El sistema proporciona una interfaz para configurar el puerto de comunicación de las cámaras de seguridad.

La descripción de las demás historias de usuario se encuentra en el Anexo 1.

2.4 Planificación del desarrollo

Esta fase tiene como propósito establecer un acuerdo entre clientes y desarrolladores sobre el menor tiempo en que la mayor cantidad de historias de usuarios puedan ser utilizadas.

2.4.1 Estimación de esfuerzos e iteraciones por historia de usuario

Tabla 4 Estimación de esfuerzos e iteraciones

Fuente: Elaboración Propia

Historias de Usuario	Tiempo de Estimado (horas)	Iteración	Duración total de las iteraciones (horas)
Definir nombre del objeto	24	1	192
Configurar color de fondo del componente.	48		
Configurar la opacidad del componente.	48		
Configurar la dimensión del ancho del componente.	24		
Configurar la dimensión del alto del componente.	24		
Configurar el color del borde del componente.	24		

Configurar la posición del componente.	48	2	312
Permitir la rotación del componente.	48		
Eliminar el componente.	48		
Configurar protocolo de comunicación.	72		
Configurar la dirección IP o URL de las cámaras de seguridad.	72		
Configurar el nombre de las cámaras de seguridad.	24		
Configurar el puerto de comunicación de las cámaras de seguridad.	48	3	336
Configurar las cámaras de seguridad.	72		
Detener una cámara de seguridad.	48		
Reproducir una cámara de seguridad.	48		
Grabar contenido.	48		
Configurar brillo.	24		

Configurar contraste.	24		
Configurar saturación	24		
Configurar volumen.	48		

2.4.2 Plan de iteraciones

Para realizar las Historias de Usuario se llevó a cabo un plan que consta de tres iteraciones las cuales son expuestas a continuación.

- **Iteración 1**: Para esta iteración se desarrollan las HU 1, 2,3,4,5 y 6 las cuales se encargan de definir el nombre del componente, la configuración del color y las dimensión de ancho y alto garantizado la representación gráfica del componente.
- **Iteración 2**: En esta iteración se desarrollarán las HU 7,8,9,10,11 y 12 las cuales se encargan de configurar puerto, dirección y protocolo de comunicación del componente.
- **Iteración 3**: En esta iteración se desarrollarán las HU desde la 13 hasta la 21, estas se encargan de realizar todas las operaciones referentes al manejo de las cámaras de seguridad asociadas al componente gráfico.

2.5 Diseño de clases

Un diagrama de clases es una representación gráfica que sirve para representar la estructura de un sistema que será implementado utilizando un lenguaje orientado a objetos. Los diagramas de clases se realizan en la fase de diseño del software después de la fase de requisitos. La idea de estos diagramas es representar las clases que tendrá el sistema, así como su contenido y sus relaciones con otras clases. La implementación de sistemas medianamente grandes no sería abordable sin este tipo de diagramas, y aunque fuera abordable se tardaría mucho más y sería más fácil cometer errores(20). El diseño de clase para el componente gráfico cámara de seguridad se muestra a continuación.

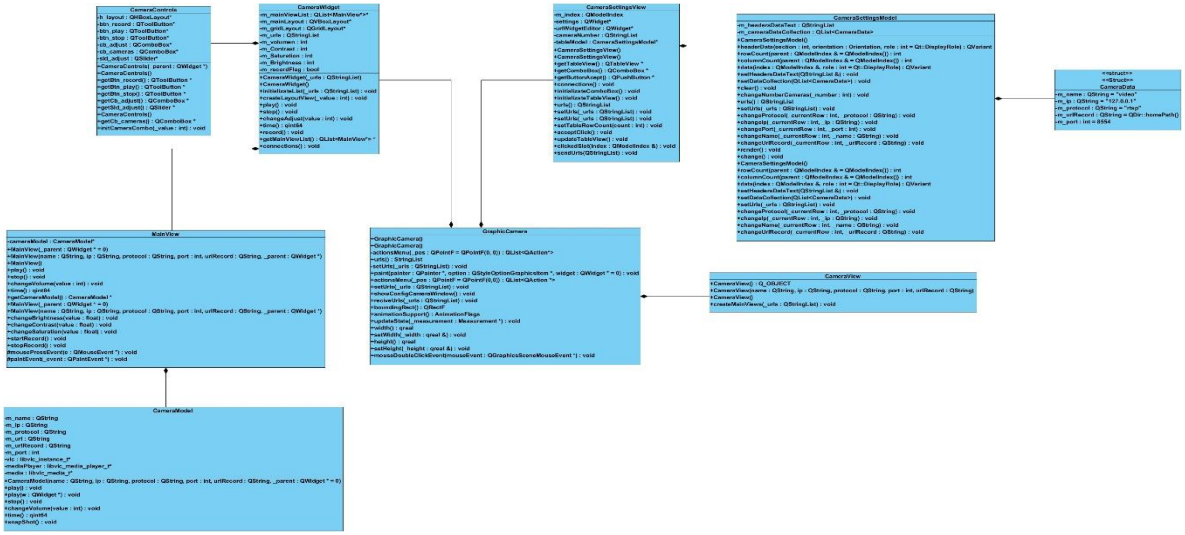

Figura 3 Diseño de Clases

Fuente: Elaboración Propia

2.6 Patrón de Arquitectura

El patrón de arquitectura utilizado es Modelo/Vista, este es una especificación del patrón Modelo-Vista-Delegado. El patrón modelo-vista proporciona una solución que utiliza una arquitectura más versátil. El modelo-vista elimina los problemas de consistencia de datos que pueden ocurrir con los *widgets* estándar. También facilita el uso de más de una vista de los mismos datos porque un modelo puede transmitirse a muchas vistas. Dado que las clases de vista no conocen la estructura de sus datos, se debe proporcionar un contenedor para que sus datos se ajusten a la interfaz. Una vez que la vista recibe un puntero a un modelo, leerá y mostrará su contenido y será su editor(15).

Figura 4 Patrón de arquitectura Modelo-Vista

Fuente: Elaboración Propia

Modelo:

- La clase CameraSettingsModel es la encargada de manejar los datos referentes a la configuración de las cámaras de seguridad.

Vista:

- La clase *CameraSettingView* es la encargada de presentar una interfaz al usuario donde podrá visualizar y editar los datos.

2.8 Patrones de Diseño

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Los patrones de diseño tienen como objetivo, proporcionar catálogos de elementos reutilizables en el diseño de sistemas software. Evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente. Formalizar un vocabulario común entre diseñadores. Estandarizar el modo en que se realiza el diseño.

Existen diversas maneras de agrupar los patrones de diseño. Quizá la más extendida es agruparlos según su propósito (24). En este caso tendríamos las siguientes categorías:

- **Patrones creacionales:** Procuran independizar al sistema de cómo sus objetos son creados y/o representados.
- **Patrones de comportamiento:** Se centran en los algoritmos y en la asignación de responsabilidades e interacción entre los objetos.
- **Patrones estructurales:** Se refieren a cómo las clases y los objetos son organizados para conformar estructuras más complejas.

Los patrones GRASP utilizados fueron:

- **Experto:** Asigna una responsabilidad a la clase que cuenta con la información necesaria para cumplirla. Este patrón se encuentra en la implementación de todas las clases, ya que las clases en si tienen las responsabilidades que les corresponde en función de la información que manejan.
- **Creador**: Guía la asignación de responsabilidades relacionadas con la creación de objetos. Se aplica en todos los casos donde una clase tiene la responsabilidad de crear una nueva instancia de la otra. Un ejemplo donde se utiliza este patrón es en la clase *CameraWidget* encargada de crear una o varias instancias de la clase *MainView*.
- **Alta Cohesión:** Asigna una responsabilidad de modo que la cohesión siga siendo alta. La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Este patrón se evidencia en todas las clases ya que ellas almacenan la información necesaria para trabajar, sin involucrar otras clases.

- **Bajo Acoplamiento**: El uso de los patrones Experto y Creador contribuyen al bajo acoplamiento entre las clases del sistema. La intención es tener las clases lo menos acopladas entre sí, de esta forma en caso de producirse una modificación en alguna de ellas, la repercusión en el resto de clases sería mínima. Este patrón se evidencia en todas las clases porque existe la mínima dependencia entre ellas para realizar modificaciones sin alterar su comportamiento.

2.9 Conclusiones Parciales

En este capítulo se trataron temas referentes al análisis y diseño del componente para la visualización de cámaras de seguridad, lo que permitió arribar a las siguientes conclusiones:

- ✓ La modelación del dominio permitió establecer un punto de partida para lograr un correcto diseño del sistema.
- ✓ Se estableció la arquitectura que tendrá el componente a desarrollar empleándose el patrón arquitectónico modelo-vista.
- ✓ Con la realización del diagrama de clases, se obtuvo una visión más clara del sistema en términos de implementación.

Capítulo III: Implementación y prueba

3. Introducción

En este capítulo se pondrán en práctica la implementación y pruebas que detectarán posibles fallas del componente gráfico ya diseñado. Una vez corregidos los errores podrán ser desplegados para su utilización. Se expondrá el estándar de codificación utilizado para el desarrollo de la solución, así como su respectivo diagrama de componentes y de despliegue.

3.1 Modelo de Implementación

La implementación constituye una de las fases más importantes del desarrollo de *software*. En ella se toman como punto de partida los resultados obtenidos en el diseño, implementándose el sistema en términos de componentes como ficheros de código binario, código fuente, *scripts*[2] y ejecutables. Su importancia radica en la obtención de un sistema ejecutable, siendo esto uno de los principales objetivos en el desarrollo de *software.*

3.2 Diagrama de Componentes

Los diagramas de componentes representan la estructura física del código. Cuando Visual Paradigm genera código, los diagramas de componentes representan la ubicación de los archivos de código fuente para sus clases. Al realizar ingeniería inversa en un proyecto ya existente, los diagramas de componentes pueden ayudarle a establecer relaciones entre cada diagrama de clases y los archivos de código fuente.

[2] Es un documento que contiene instrucciones, escritas en códigos de programación.

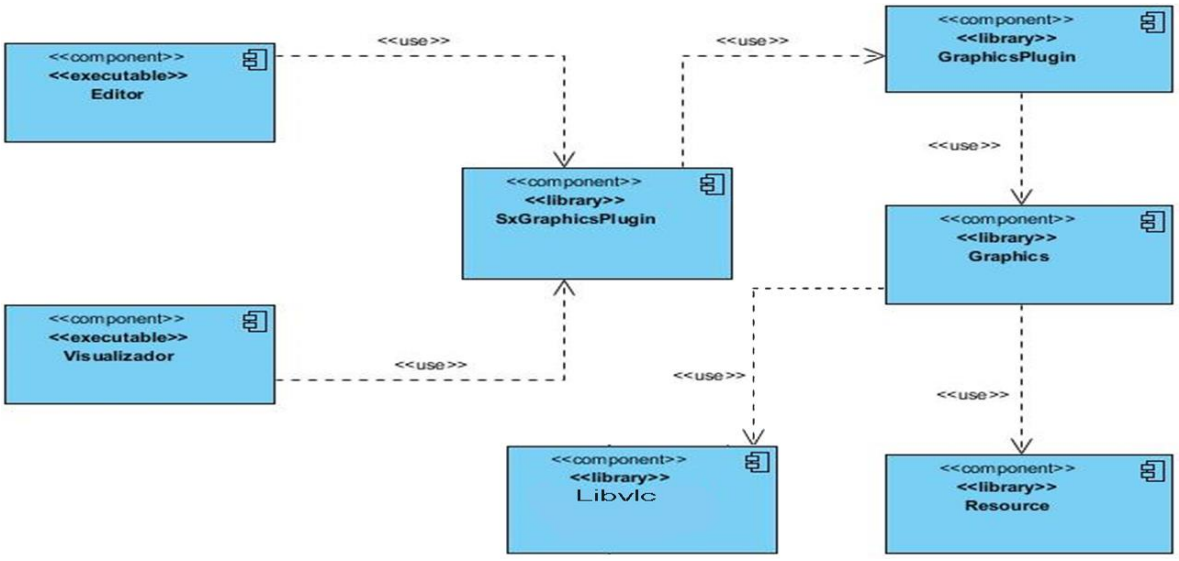

Figura 5 Diagrama de Componentes

Fuente: Elaboración Propia

El diagrama de componentes antes representado está estructurado por ejecutables y bibliotecas:

- **Editor:** Es la aplicación donde se trabaja en el entorno de configuración para representar los procesos del campo
- **Visualizador:** Es una aplicación donde se supervisa el área configurada para interactuar con los componentes gráficos.
- **Biblioteca Graphics Plugins:** En esta biblioteca se almacenan los Plugins de los objetos gráficos.
- **Biblioteca Graphics:** Aquí están almacenados objetos gráficos que se utilizan para representar lo componentes gráficos, ya sea de forma simple o componentes complejos.
- **SxGraphics Plugins**: Es una interfaz de la biblioteca Graphics Plugins específica para los componentes de SAINUX.
- **Resource**: Biblioteca donde se encuentran agrupadas cada una de las entidades encargadas de brindar las imágenes, los iconos y los diseños de los componentes.

3.3 Diagrama de Despliegue

Un diagrama de despliegue muestra las relaciones físicas entre los componentes de hardware (nodos) y software (artefactos) en el sistema final, es decir, la configuración de

los elementos de procesamiento en tiempo de ejecución y los componentes software (procesos y objetos que se ejecutan en ellos). En el contexto del diagrama de despliegue, una asociación representa una ruta de comunicación entre los nodos. (22)

Características del diagrama de despliegue:

Describen la arquitectura física del sistema durante la ejecución, en términos de:

- Procesadores
- Dispositivos
- Componentes de software
- Describen la topología del sistema: la estructura de los elementos de hardware y el software que ejecuta cada uno de ellos.

El siguiente diagrama muestra el diagrama de despliegue para el sistema SCADA SAINUX 2.0.

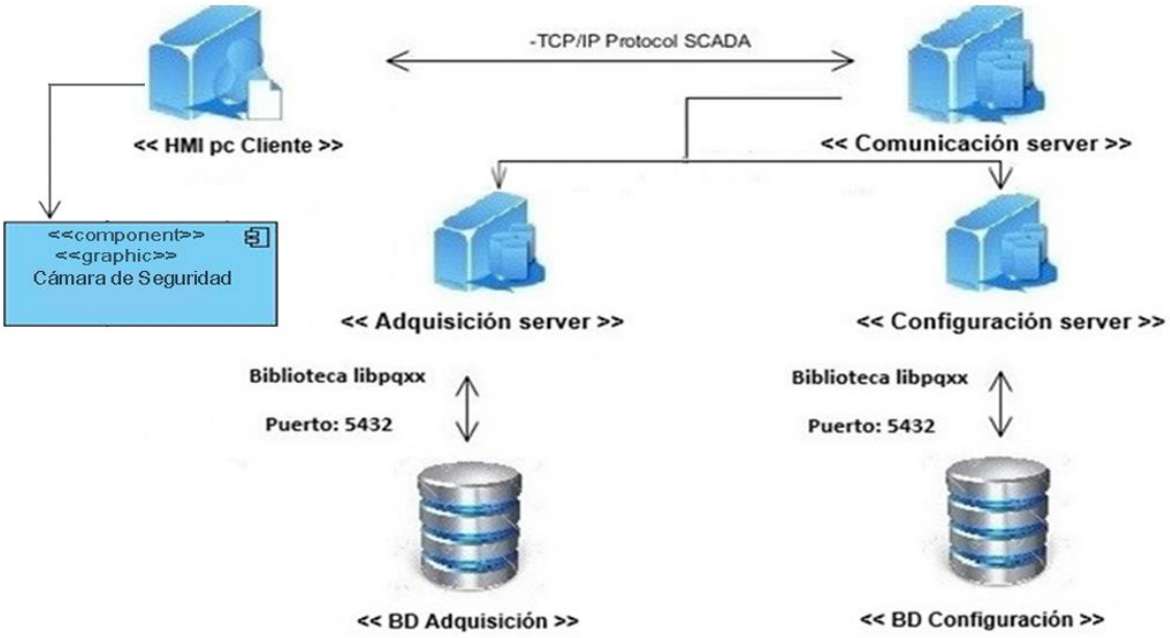

Figura 6 Diagrama de Despliegue

Fuente: Elaboración Propia

- **PC HMI:** En esta pc donde se instala el módulo HMI, donde se encuentran desplegados el Ambiente Configuración y el Ambiente Edición, o solo uno de ellos.

- **Servidor Adquisición y Servidor Configuración:** Estos módulos se ejecutan en distintas pc que pueden servir como servidores poniendo en evidencia la arquitectura distribuida que presenta el SCADA SAINUX 2.0.
- **Adquisición y Configuración:** Los módulos Adquisición y Configuración se encuentran instalados con sus respectivas Bases de Datos conectados a través de los puertos 5432, utilizando la biblioteca libpqxx[3].

  La conexión entre estos módulos se realiza utilizando el protocolo de comunicación SCADA desarrollado por el centro.

3.4 Estándar de Codificación

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Si bien los programadores deben implementar un estándar de forma prudente, éste debe tender siempre a lo práctico. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiese escrito todo el código de una sola vez. Al comenzar un proyecto de software, se debe establecer un estándar de codificación para asegurarse de que todos los programadores del proyecto trabajen de forma coordinada. Cuando el proyecto de software incorpore código fuente previo, o bien cuando realice el mantenimiento de un sistema de software creado anteriormente, el estándar de codificación debería establecer cómo operar con la base de código existente.

Usar técnicas de codificación sólidas y realizar buenas prácticas de programación con vistas a generar un código de alta calidad es de gran importancia para la calidad del software y para obtener un buen rendimiento. Si se aplica de forma prolongada un estándar de codificación bien definido, se utilizan técnicas de programación apropiadas y posteriormente se efectúan revisiones del código de rutinas, caben muchas posibilidades de que un proyecto de software se convierta en un sistema de software fácil de comprender y de mantener.

Como la solución propuesta en este trabajo es parte del sistema SCADA SAINUX 2.0 el estándar de codificación utilizado fue definido por el proyecto:

- Es importante especificar el nombre del autor, para ello se utilizan la etiqueta ***@autor***.

[3] La biblioteca libpqxx, es la encargada de la comunicación con las base de datos.

- El código será escrito en inglés y la documentación en español.
- Las variables y funciones comienzan con letra minúscula. Cada palabra consecutiva en el nombre comienza con letra mayúscula.

  Ejemplo: void createObjectModel ();

- Los atributos de las clases deben empezar con ***m***_ seguido del nombre del atributo, en el caso de atributos compuestos, la inicial de la segunda palabra debe comenzar con mayúscula.

  Ejemplo: m_posScreen

- Los parámetros que recibe una función debe empezar con _ (guion bajo).

  Ejemplo: void paintRuntime (QPainter *_painter, const QStyleOptionGraphicsItem *_option, QWidget *_widget);

- Todas las funcionalidades y atributos deben seguir el siguiente formato: para documentar **@*brief Nombre del método***.
- Ninguna función debe tener más de 200 líneas.
- Los valores de los numerativos deben ser con letras mayúsculas.
- Las secciones *public*, *protected* y *private* son declaradas en el orden expuesto.

3.5 Despliegues del sistema

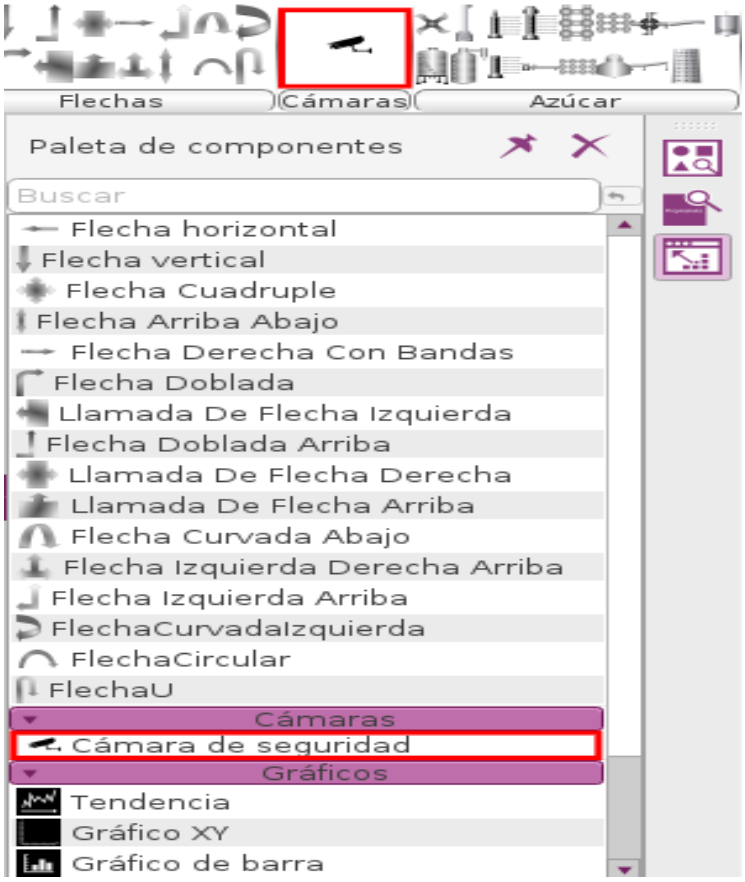

Figura 7 Paleta del Componente.

Fuente: Elaboración propia

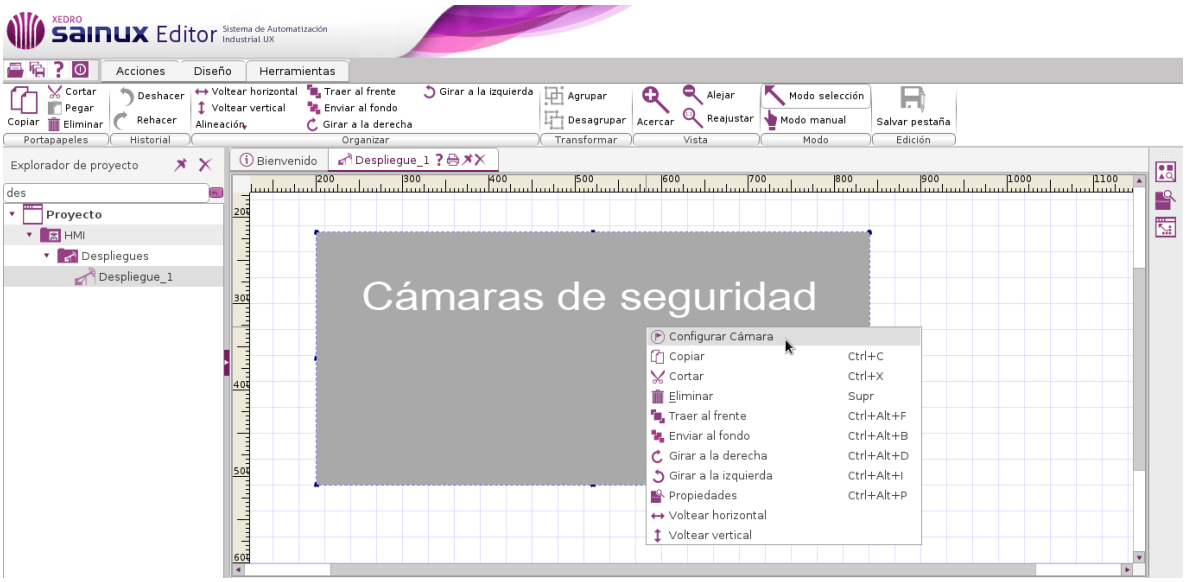

Figura 8 Menú de acceso a la configuración del componente.

Fuente: Elaboración propia

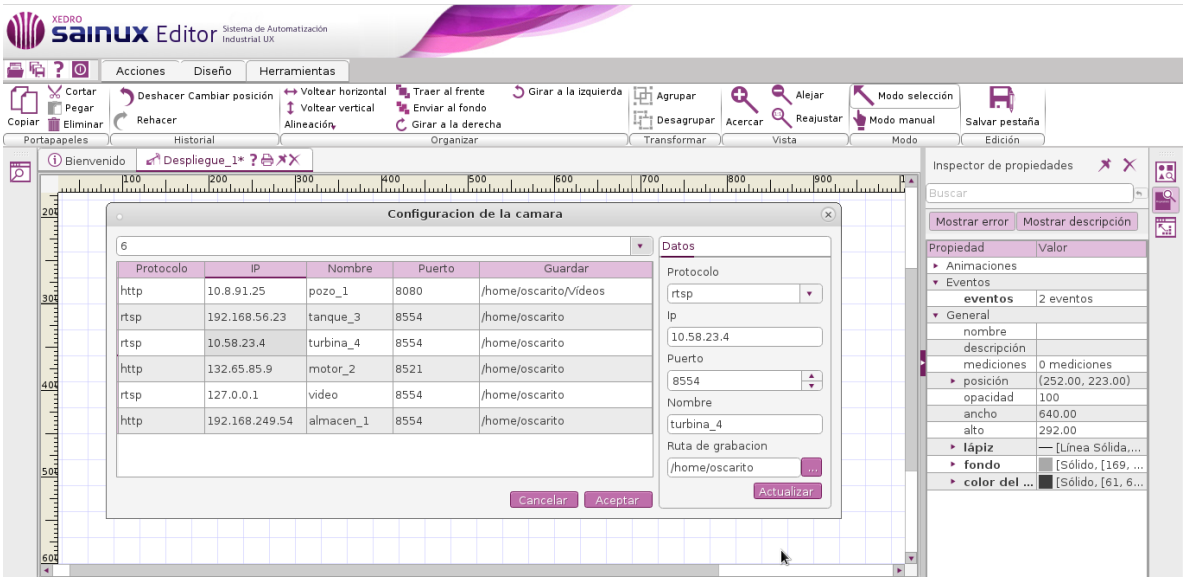

Figura 9 Ventana de configuración del componente.

Fuente: Elaboración propia

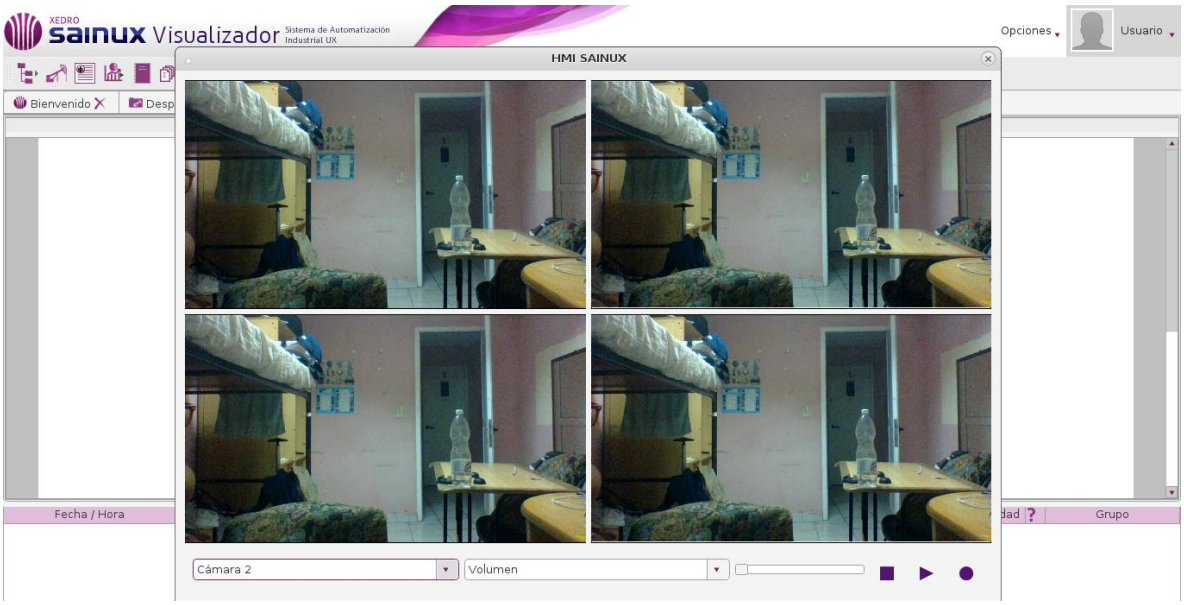

Figura 10 Ventana de reproducción del componente

Fuente: Elaboración propia

3.6 Pruebas

La fase de pruebas del sistema tiene como objetivo, verificar el sistema software para comprobar si este cumple sus requisitos. Dentro de esta fase pueden desarrollarse distintos tipos de pruebas, en función de los objetivos de las mismas. Algunos tipos son: pruebas funcionales, pruebas de usabilidad, pruebas de rendimiento y pruebas de seguridad.

Los procesos de aseguramiento de calidad de un producto de software suelen dividirse en lo que respecta a su componente analítico en pruebas estáticas y dinámicas. La diferencia fundamental entre estos tipos de pruebas, radica en que, las pruebas estáticas se centran en evaluar la calidad con la que se está generando la documentación del proyecto, por medio de revisiones periódicas, mientras que las pruebas dinámicas, requieren de la ejecución del software con el fin de medir el nivel de calidad con la que este fue codificado y el nivel de cumplimiento en relación con la especificación del sistema.(23)

3.6.1 Tipos de Pruebas

- ✓ ***Pruebas de regresión***

Las pruebas de regresión intentan verificar que los cambios realizados no han introducidos nuevos defectos y que el resto de la aplicación sigue funcionando correctamente(24). Estas

pueden ser desarrolladas de forma manual, sin embargo, la mejor manera es diseñar y construir scripts de pruebas que puedan ejecutarse de forma automática(25).

- ✓ ***Pruebas de aceptación***

El uso de cualquier producto de software tiene que estar justificado por las ventajas que ofrece. Uno de los instrumentos para esta determinación es la prueba de aceptación. En esta prueba se evalúa el grado de calidad del software con relación a todos los aspectos relevantes para que el uso del producto se justifique.(26)

Tabla 5 Caso de Prueba #10

Fuente: Elaboración Propia

Caso de Prueba Aceptación	
Número: 10	**Historia de usuario: 10**
Nombre: Configurar el protocolo de comunicación.	
Descripción: Comprobar que se puede configurar el protocolo	
Condiciones de Ejecución: El usuario debe comprobar que el protocolo de comunicación sea configurable.	
Entradas/ Pasos de Ejecución: - Seleccionar el Componente. - Clic derecho encima del componente - Selecciona la opción de configurar cámara.	
Resultado esperado: El usuario pudo configurar el protocolo de comunicación.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 6 Caso de Prueba #11

Fuente: Elaboración Propia

Caso de Prueba Aceptación	
Número: 11	**Historia de usuario: 11**
Nombre: Configurar la dirección IP de las cámaras de seguridad.	
Descripción: Comprobar que se pueda configurar la dirección de IP.	
Condiciones de Ejecución: El usuario debe comprobar que la dirección IP sea configurable.	
Entradas/ Pasos de Ejecución: - Seleccionar el Componente. - Clic derecho encima del componente - Selecciona la opción de configurar cámara.	
Resultado esperado: El usuario pudo configurar la dirección IP de las cámaras de seguridad.	
Evaluación de la prueba: Pruebas satisfactoria.	

Tabla 7 Caso de Prueba #13

Fuente: Elaboración Propia

Caso de Prueba Aceptación	
Número: 13	**Historia de usuario: 13**

Nombre: Configurar puerto de comunicación de las cámaras de seguridad.
Descripción: Comprobar que se puede configurar el puerto de comunicación de las cámaras de seguridad.
Condiciones de Ejecución: El usuario debe comprobar que el puerto de comunicación sea configurable.
Entradas/ Pasos de Ejecución: - Seleccionar el Componente. - Clic derecho encima del componente. - Selecciona la opción de configurar cámara.
Resultado esperado: El usuario pudo configurar el puerto de comunicación de las cámaras de seguridad.
Evaluación de la prueba: Prueba satisfactoria.

La descripción de las restantes pruebas de aceptación se encuentra en el Anexo 2.

3.6.2 Diseño de Casos de Prueba

Tabla 8 Diseño de Caso de Prueba

Fuente: Elaboración Propia

Pruebas del Sistema	HU	Iteración	NC	Cerrada
	22	1ra	7	7
		2da	4	4
		3ra	1	1

Una vez realizados los casos de Pruebas para un total de 21 historias de usuarios, con tres iteraciones se eliminaron las 12 No Conformidades.

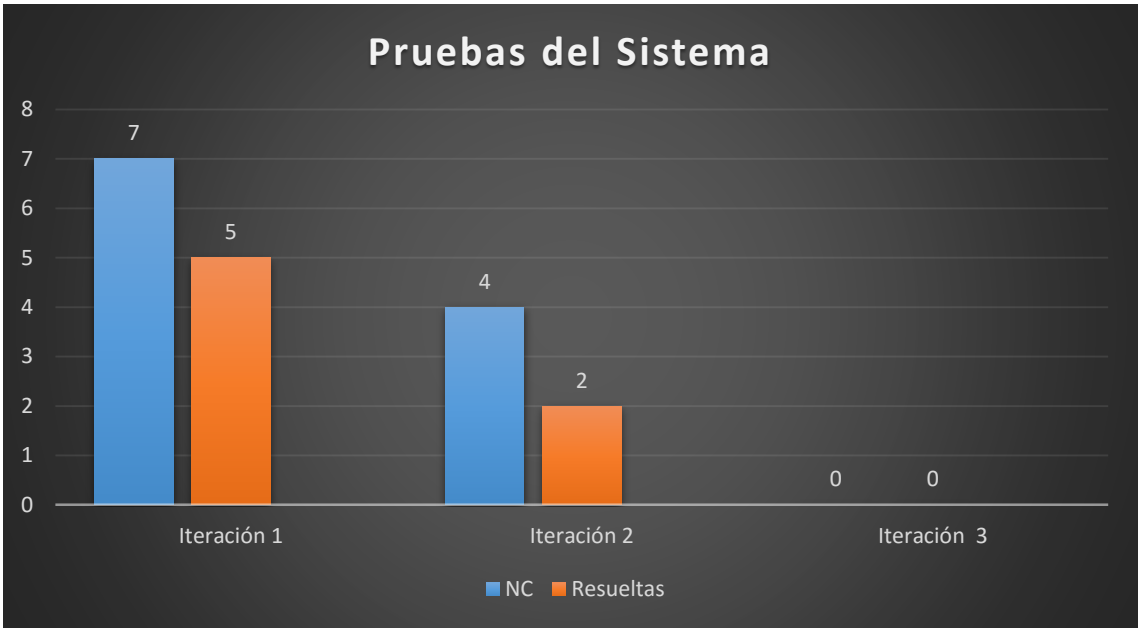

Figura 11 Iteraciones de casos de pruebas.

Fuente: Elaboración Propia

3.6.3 Resultados de las pruebas

Durante la fase de prueba se realizaron tres iteraciones al Componente para la visualización de cámaras de seguridad. En la primera iteración las principales no conformidades detectadas fueron la incapacidad de redimensionar el componente y la incorrecta grabación de video de una cámara de seguridad en el disco duro. Estas inconformidades fueron corregidas y luego se volvió a realizar otra iteración de las pruebas para el caso de uso Grabar Contenido, arrojando resultados satisfactorios.

Durante la segunda iteración se detectó, que los controles de ajuste de video no modificaban la imagen según la escala brindada. Esto provocaba que el operador no pudiese ajustar la imagen acorde a su necesidad. El problema radicaba en que los controles no se encontraban en el rango establecido por la biblioteca libvlc.

3.7 Conclusiones Parciales

En el presente capítulo se realizó una descripción de la implementación y las pruebas realizadas al sistema. En el mismo se generaron los artefactos necesarios para la

implementación y las pruebas del sistema, por lo que se puede arribar a las siguientes conclusiones:

- ✓ La realización del diagrama de componentes permitió entender la relación entre los componentes que conforman el sistema.
- ✓ Mediante las pruebas ejecutadas al sistema, se lograron encontrar errores funcionales en el sistema.
- ✓ Obteniendo resultados satisfactorios en la última iteración de pruebas se puede concluir que el sistema no presenta errores funcionales.

Conclusiones generales

Con la realización de este trabajo se desarrolló un componente gráfico para contribuir a la visualización de cámaras de seguridad sobre la Interfaz Hombre Máquina del sistema SCADA SAINUX 2.0. De esta forma se le da cumplimiento al objetivo propuesto al inicio de la investigación, además se comprobó que:

- ✓ El estudio de las principales formas de comunicación que utilizan las cámaras de vigilancias existentes y cómo estas se manifiestan en la actualidad permitió llevar a cabo el desarrollo del módulo.

- ✓ La metodología de desarrollo de *software* utilizada permitió la correcta definición por parte del cliente de las Historias de Usuario, logrando así ver con claridad los requisitos funcionales.

- ✓ La representación de los diagramas: modelo del dominio, clases del diseño, diagrama de componentes y de despliegue, permitieron establecer un punto de partida para el desarrollo de la solución de forma estructurada, siguiendo debidamente las pautas establecidas por la metodología AUP-UCI para el proceso de desarrollo de software.
- ✓ Con la implementación de la solución, Componente gráfico para la visualización de cámaras de seguridad en el HMI del SCADA SAINUX 2.0, se eliminan los problemas descritos en la situación problemática.

- ✓ La realización de las pruebas de aceptación, permitieron detectar los posibles defectos antes de desplegar la solución informática. Además, esto demostró el cumplimiento adecuado de las especificaciones realizadas y la obtención de un producto final con calidad.

Recomendaciones

Se recomienda seguir ampliando la gama de componentes gráficos que representan cámaras de seguridad.

Incorporar el componente gráfico en el visualizador Web que se está desarrollado en el Centro de Informática Industrial.

Adicionarle un mecanismo de tratamiento de video al componente de cámara de seguridad.

Bibliografía

2. SCADAs para redes multiautómatas - - tecnicaindustrial.es. [online]. [Accessed 25 November 2016]. Available from: http://www.tecnicaindustrial.es/tifrontal/a-1482-SCADAs-redes-multiautomatas.aspx

3. Video Monitoring & Surveillance - Wireless Mesh Network Communication Solutions | ABB Wireless (Communication Networks). [online]. [Accessed 12 June 2017]. Available from: http://new.abb.com/network-management/communication-networks/wireless-networks/solutions/video-monitoring-surveillanceUsing video in oil and gas operations offers increased visibility into field operations, boosts operational efficiency and enhances worker, as well as environmental, safety.

4. SIMATIC WinCC Open Architecture Video - Integrated Video management - HMI Software - Siemens. [online]. [Accessed 13 June 2017]. Available from: http://w3.siemens.com/mcms/human-machine-interface/en/visualization-software/simatic-wincc-open-architecture/wincc-oa-options/wincc-oa-video/pages/default.aspx

5. Axel Omar Meza Arrecis. *IMPLEMENTACIÓN DE PLATAFORMA PARA STREAMING DE VÍDEO EN TIEMPO REAL, A PARTIR DE TECNOLOGÍAS LIBRES*. Universidad de San Carlos de Guatemala, [no date].

6. Real-Time Streaming Protocol (RTSP) session helper (rtsp). [online]. [Accessed 25 November 2016]. Available from: http://help.fortinet.com/fos50hlp/54/Content/FortiOS/fortigate-system-administration-54/Session%20Helpers/Real-Time%20Streaming%20Protocol%20(RTSP).htm

7. *Microsoft Word - RTP_ES_EFORT.doc - RTP_ES_EFORT.pdf* [online]. [Accessed 8 June 2017]. Available from: http://www.efort.com/media_pdf/RTP_ES_EFORT.pdf

8. libVLC - VideoLAN Wiki. [online]. [Accessed 11 June 2017]. Available from: https://wiki.videolan.org/LibVLC/

9. Modelo de Dominio | Tecnología y Synergix. [online]. [Accessed 5 March 2017]. Available from: https://synergix.wordpress.com/2008/07/10/modelo-de-dominio/

10. Microsoft PowerPoint - 2-requisitos.ppt [Modo de compatibilidad] - 2-requisitos.pdf. [online]. [Accessed 5 March 2017]. Available from: https://www.infor.uva.es/~mlaguna/is1/apuntes/2-requisitos.pdf

11. Diagrama de Clases - Instinto Binario. [online]. [Accessed 6 March 2017]. Available from: http://instintobinario.com/diagrama-de-clases/

12. Sparx Systems - Tutorial UML 2 - Diagrama de Despliegue. [online]. [Accessed 11 May 2017]. Available from: http://www.sparxsystems.com.ar/resources/tutorial/uml2_deploymentdiagram.html

13. PRUEBAS DE SOFTWARE. *PRUEBAS DE SOFTWARE* [online]. [Accessed 11 May 2017]. Available from: https://pruebasdelsoftware.wordpress.com/Ingenieria de Software con énfasis en pruebas.

14. Pruebas de Regresión. *Softqanetwork.com* [online]. 14 May 2008. [Accessed 14 June 2017]. Available from: http://www.softqanetwork.com/pruebas-de-regresionComo norma general, cualquier aplicación debe probarse a fondo. Lamentablemente, en la práctica, las prueba son insuficientes y a menudo pasadas por alto. Las pruebas de regresión intentan verifica…

15. Pruebas Funcionales y de Regresión con Selenium | SG. [online]. [Accessed 14 June 2017]. Available from: https://sg.com.mx/revista/43/pruebas-funcionales-y-regresion-selenium

16. La prueba de aceptación es la prueba más importante para los productos software. [online]. [Accessed 13 May 2017]. Available from: http://www.pruebasdesoftware.com/pruebadeaceptacion.htm

17. MONROY, Martín E., ARCINIEGAS, José L. and RODRÍGUEZ, Julio C. Modelo Ontológico para Contextos de uso de Herramientas de Ingeniería Inversa. *Información tecnológica*. 2016. Vol. 27, no. 4, p. 165–174. DOI 10.4067/S0718-07642016000400018.

18. 3. Técnicas para Identificar Requisitos Funcionales y No Funcionales - Metodología Gestión de Requerimientos. [online]. [Accessed 5 March 2017]. Available from: https://sites.google.com/site/metodologiareq/capitulo-ii/tecnicas-para-identificar-requisitos-funcionales-y-no-funcionales

19. Requerimientos funcionales y no funcionales. [online]. [Accessed 5 March 2017]. Available from: https://es.scribd.com/doc/37187866/Requerimientos-funcionales-y-no-funcionales

20. UML_clase_05_UML_paquetes.pdf. [online]. [Accessed 6 March 2017]. Available from: http://www.codecompiling.net/files/slides/UML_clase_05_UML_paquetes.pdf

21. 06Patrones - 06Patrones.pdf. [online]. [Accessed 29 March 2017]. Available from: http://siul02.si.ehu.es/~alfredo/iso/06Patrones.pdf

22. Model/View Tutorial | Qt Widgets 5.9. [online]. [Accessed 13 June 2017]. Available from: http://doc.qt.io/qt-5/modelview.html

23. *Model/View Tutorial | Qt Widgets 5.9* [online]. [Accessed 13 June 2017]. Available from: http://doc.qt.io/qt-5/modelview.html

24. Runtime software. [online]. [Accessed 14 June 2017]. Available from: https://www.reliance-scada.com/en/products/reliance4/runtime-software

25. WinCC Runtime Software - Automation Software - Siemens. [online]. [Accessed 14 June 2017]. Available from: http://w3.siemens.com/mcms/automation-software/en/tia-portal-software/wincc-tia-portal/wincc-tia-portal-runtime/pages/default.aspx

ANEXO 1 Historias de usuarios.

Tabla 9 Historia de usuario #1

Fuente: Elaboración Propia

Historia de Usuario	
Número: 1	**Nombre de Historia:** Definir nombre del objeto
Programador: Oscar García Quintana	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Tiempo estimado:** 0.4
Nivel de Complejidad: Alta	**Tiempo real:** 0.4
Descripción: El sistema proporciona una interfaz al cliente donde puede escribir el nombre del objeto, el mismo no debe exceder de 32 caracteres, ni puede contener comillas simples (') y and per se and (&).	

Tabla 10 Historia de usuario #2

Fuente: Elaboración Propia

Historia de Usuario	
Número: 2	**Nombre de Historia:** Configurar el color de fondo del componente.
Programador: Oscar García Quintana	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Tiempo estimado:** 0.4

Nivel de Complejidad: Alta	**Tiempo real:** 0.4
Descripción: El sistema permite configurar el color de fondo del componente.	

Tabla 11 Historia de usuario #3

Fuente: Elaboración Propia

Historia de Usuario	
Número: 3	**Nombre de Historia:** Configurar la opacidad del componente.
Programador: Oscar García Quintana	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Tiempo estimado:** 0.4
Nivel de Complejidad: Alta	**Tiempo real:** 0.4
Descripción: El sistema permite configurar la opacidad del componente en un rango de 0 a 100.	

Tabla 12 Historia de usuario #4

Fuente: Elaboración Propia

Historia de Usuario	
Número: 4	**Nombre de Historia:** Configurar la dimensión del ancho del componente.

Programador: Oscar García Quintana	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Tiempo estimado:** 0.2
Nivel de Complejidad: Alta	**Tiempo real:** 0.2
Descripción: El sistema proporciona una interfaz al cliente donde puede configurar el ancho del componente.	

Tabla 13 Historia de usuario #5

Fuente: Elaboración Propia

Historia de Usuario	
Número: 5	**Nombre de Historia:** Configurar la dimensión del alto del componente.
Programador: Oscar García Quintana	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Tiempo estimado:** 0.2
Nivel de Complejidad: Alta	**Tiempo real:** 0.2
Descripción: El sistema proporciona una interfaz al cliente donde puede configurar el alto del componente.	

Tabla 14 Historia de usuario #6

Fuente: Elaboración Propia

Historia de Usuario	
Número: 6	**Nombre de Historia:** Configurar el color del borde del componente.
Programador: Oscar García Quintana	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Tiempo estimado:** 0.2
Nivel de Complejidad: Alta	**Tiempo real:** 0.2
Descripción: El sistema permite al cliente cambiar el color del borde del componente.	

Tabla 15 Historia de usuario #7

Fuente: Elaboración Propia

Historia de Usuario	
Número: 7	**Nombre de Historia:** Configurar la posición del componente.
Programador: Oscar García Quintana	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Tiempo estimado:** 0.6
Nivel de Complejidad: Alta	**Tiempo real:** 0.6
Descripción: El sistema permite al cliente configurar la posición del componente, en el eje X y el eje Y en un rango (0; 2147483647).	

Tabla 16 Historia de usuario #8

Fuente: Elaboración Propia

Historia de Usuario	
Número: 8	**Nombre de Historia:** Permitir la rotación del componente.
Programador: Oscar García Quintana	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Tiempo estimado:** 0.6
Nivel de Complejidad: Alta	**Tiempo real:** 0.6
Descripción: Establece un ángulo de rotación. Definir donde localizar el centro de rotación. El centro de rotación se puede localizar en distintas posiciones, en la parte inferior del widget, en la parte superior, en el centro, a la derecha o a la izquierda.	

Tabla 17 Historia de usuario #9

Fuente: Elaboración Propia

Historia de Usuario	
Número: 9	**Nombre de Historia:** Permitir eliminar el componente
Programador: Oscar García Quintana	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Tiempo estimado:** 0.4
Nivel de Complejidad: Alta	**Tiempo real:** 0.4

Descripción: El sistema permite al usuario eliminar el componente una vez añadido al despliegue.

Tabla 18 Historia de usuario #12

Fuente: Elaboración Propia

Historia de Usuario	
Número: 12	**Nombre de Historia:** Configurar el nombre de las cámaras de seguridad.
Programador: Oscar García Quintana	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Tiempo estimado:** 0.2
Nivel de Complejidad: Alta	**Tiempo real:** 0.2
Descripción: El sistema proporciona una interfaz para configurar el nombre de las cámaras de seguridad.	

Tabla 19 Historia de usuario #14

Fuente: Elaboración Propia

Historia de Usuario	
Número: 14	**Nombre de Historia:** Configurar las cámaras de seguridad.

Programador: Oscar García Quintana	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Tiempo estimado:** 0.6
Nivel de Complejidad: Alta	**Tiempo real:** 0.6
Descripción: El sistema, mediante una interfaz, permite al cliente configurar las cámaras de seguridad en su totalidad.	

Tabla 20 Historia de usuario #15

Fuente: Elaboración Propia

Historia de Usuario	
Número: 15	**Nombre de Historia:** Detener una cámara de seguridad.
Programador: Oscar García Quintana	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Tiempo estimado:** 0.4
Nivel de Complejidad: Alta	**Tiempo real:** 0.4
Descripción: El sistema permite detener la reproducción de una cámara de seguridad.	

Tabla 21 Historia de usuario #16

Fuente: Elaboración Propia

Historia de Usuario	
Número: 16	**Nombre de Historia:** Reproducir una cámara de seguridad.
Programador: Oscar García Quintana	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Tiempo estimado:** 0.4
Nivel de Complejidad: Alta	**Tiempo real:** 0.4
Descripción: El sistema permite reproducir una cámara de seguridad.	

Tabla 22 Historia de usuario #17

Fuente: Elaboración Propia

Historia de Usuario	
Número: 17	**Nombre de Historia:** Grabar contenido.
Programador: Oscar García Quintana	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Tiempo estimado:** 0.6
Nivel de Complejidad: Alta	**Tiempo real:** 0.6
Descripción: El sistema permite al cliente grabar el contenido de las cámaras de seguridad.	

Tabla 23 Historia de usuario #18

Fuente: Elaboración Propia

Historia de Usuario	
Número: 18	**Nombre de Historia:** Configurar brillo.
Programador: Oscar García Quintana	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Tiempo estimado:** 0.2
Nivel de Complejidad: Alta	**Tiempo real:** 0.2
Descripción: El sistema proporciona una interfaz al cliente donde puede configurar el brillo de una cámara de seguridad.	

Tabla 24 Historia de usuario #19

Fuente: Elaboración Propia

Historia de Usuario	
Número: 19	**Nombre de Historia:** Configurar contraste.
Programador: Oscar García Quintana	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Tiempo estimado:** 0.4
Nivel de Complejidad: Alta	**Tiempo real:** 0.4

Descripción: El sistema proporciona una interfaz al cliente donde puede configurar el contraste de una cámara de seguridad.

Tabla 25 Historia de usuario #20

Fuente: Elaboración Propia

Historia de Usuario	
Número: 20	**Nombre de Historia:** Configurar saturación.
Programador: Oscar García Quintana	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Tiempo estimado:** 0.4
Nivel de Complejidad: Alta	**Tiempo real:** 0.4
Descripción: El sistema proporciona una interfaz al cliente donde puede configurar la saturación de una cámara de seguridad.	

Tabla 26 Historia de usuario #21

Fuente: Elaboración Propia

Historia de Usuario	
Número: 21	**Nombre de Historia:** Configurar volumen.
Programador: Oscar García Quintana	**Iteración Asignada:** 1

Prioridad en Negocio: Alta	**Tiempo estimado:** 0.6
Nivel de Complejidad: Alta	**Tiempo real:** 0.6
Descripción: El sistema proporciona una interfaz al cliente donde puede configurar el volumen de una cámara de seguridad.	

ANEXO 2: Pruebas de Aceptación

Tabla 27 PA# 1

Fuente: Elaboración Propia

Caso de Prueba Aceptación	
Número: 1	**Historia de usuario: 1**
Nombre: Definir nombre del componente.	
Descripción: Comprobar que el componente se nombra de forma correcta sin utilizar números o símbolos.	
Condiciones de Ejecución: El usuario debe comprobar que se puede configurar el nombre del componente.	
Entradas/ Pasos de Ejecución: - Seleccionar el componente. - Clic derecho con el ratón. - Seleccionar la opción Propiedades. - Cambiar el nombre.	
Resultado esperado: El nombre ha sido cambiado.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 28 PA# 2

Fuente: Elaboración Propia

Caso de Prueba Aceptación	
Número: 2	**Historia de usuario: 2**
Nombre: Configurar el color de fondo del componente.	
Descripción: Comprobar que al componente puede configurarse su color de fondo.	
Condiciones de Ejecución: El usuario debe comprobar que se cumple la propiedad de color de fondo para el componente.	
Entradas/ Pasos de Ejecución: - Seleccionar el componente. - Abrir el inspector de propiedades. - Seleccionar opción de color de fondo.	
Resultado esperado: La opción de color de fondo para el componente es configurable.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 29 PA #3

Fuente: Elaboración Propia

Caso de Prueba Aceptación	
Número: 3	**Historia de usuario: 3**
Nombre: Configurar la opacidad del componente.	
Descripción: Comprobar que al componente puede configurarse su opacidad.	

Condiciones de Ejecución: El usuario debe comprobar que se cumple la propiedad de opacidad para el componente.
Entradas/ Pasos de Ejecución: - Seleccionar el componente. - Abrir el inspector de propiedades. - Seleccionar opción de opacidad.
Resultado esperado: La opción de opacidad para el componente es configurable.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 30 PA #4

Fuente: Elaboración Propia

Caso de Prueba Aceptación	
Número: 4	**Historia de usuario: 4**
Nombre: Configurar la dimensión de ancho del componente.	
Descripción: Comprobar que se puede configurar la dimensión de ancho del componente, definido en el rango (- 2147483647; 2147483647).	
Condiciones de Ejecución: El usuario debe comprobar que el objeto puede ser configurado en ancho.	
Entradas/ Pasos de Ejecución: - Seleccionar el objeto.	

- Con el clic izquierdo seleccionar uno de los puntos donde puede redimensionarse para agrandarlo o achicarlo.
Resultado esperado: El componente se redimensiona sin perder calidad.
Evaluación de la prueba: Una vez realizada la primera iteración detectaron varias no conformidades, a las cuales se les da solución y se realiza otra iteración que arrojó una evaluación satisfactoria.

Tabla 31 PA #5

Fuente: Elaboración Propia

Caso de Prueba Aceptación	
Número: 5	**Historia de usuario: 5**
Nombre: Configurar la dimensión de alto del componente.	
Descripción: Comprobar que se puede configurar la dimensión del alto del componente, definido en el rango (- 2147483647; 2147483647).	
Condiciones de Ejecución: El usuario debe comprobar que el objeto puede ser configurado en alto.	
Entradas/ Pasos de Ejecución: - Seleccionar el objeto. - Con el clic izquierdo seleccionar uno de los puntos donde puede redimensionarse para agrandarlo o achicarlo.	
Resultado esperado: El componente se redimensiona sin perder calidad.	

Evaluación de la prueba: Una vez realizada la primera iteración detectaron varias no conformidades, a las cuales se les da solución y se realiza otra iteración que arrojó una evaluación satisfactoria.

Tabla 32 PA #6

Fuente: Elaboración Propia

Caso de Prueba Aceptación	
Número: 6	**Historia de usuario: 6**
Nombre: Configurar color del borde del componente.	
Descripción: Comprobar que el color del borde pueda ser configurable para diferenciarlo de los demás atributos del componente.	
Condiciones de Ejecución: El usuario debe comprobar que el color del borde pueda ser cambiado por otro color.	
Entradas/ Pasos de Ejecución: - Seleccionar el componente. - Abrir inspector de propiedades. - Seleccionar opción Color de borde. - Seleccionar color del panel de colores.	
Resultado esperado: El color del borde puede ser cambiado por otro cualquiera.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 33 PA #7

Fuente: Elaboración Propia

Caso de Prueba Aceptación	
Número: 7	**Historia de usuario: 7**
Nombre: Configurar la posición del componente.	
Descripción: Comprobar que el objeto se traslada sobre cualquier punto, definido en el rango (- 2147483647; 2147483647).	
Condiciones de Ejecución: El usuario debe comprobar que el componente se puede mover sobre el despliegue.	
Entradas/ Pasos de Ejecución: - Seleccionar el componente en el menú contextual. - Ubicarlo en el despliegue. - Seleccionar el componente y trasladarlo sobre cualquier punto.	
Resultado esperado: El componente se traslada sobre el despliegue en cualquier punto.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 34 PA #8

Fuente: Elaboración Propia

Caso de Prueba Aceptación	
Número: 8	**Historia de usuario: 8**

Nombre: Configurar la rotación del componente.
Descripción: Comprobar que el objeto puede rotar.
Condiciones de Ejecución: El usuario debe comprobar que el componente se puede rotar sobre el despliegue.
Entradas/ Pasos de Ejecución: - Seleccionar el componente en el menú contextual. - Ubicarlo en el despliegue. - Seleccionar el componente y rotarlo.
Resultado esperado: El componente rota en el despliegue en cualquier punto.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 35 PA #9

Fuente: Elaboración Propia

Caso de Prueba Aceptación	
Número: 9	**Historia de usuario: 9**
Nombre: Permitir Eliminar el componente.	
Descripción: Comprobar que el componente se elimine.	
Condiciones de Ejecución: El usuario debe comprobar que el componente sea eliminado del sistema	

Entradas/ Pasos de Ejecución: - Seleccionar el Componente. - Clic derecho encima del componente - Selecciona la opción de eliminar el componente. Otra vía: - Presionar la tecla Delete en el teclado y se elimina el componente.
Resultado esperado: El usuario pudo eliminar el componente sobre cualquiera de las dos opciones.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 36 PA #12

Fuente: Elaboración Propia

Caso de Prueba Aceptación	
Número: 12	**Historia de usuario: 12**
Nombre: Configurar el nombre de las cámaras de seguridad.	
Descripción: Comprobar que se puede configurar el nombre de las cámaras de seguridad.	
Condiciones de Ejecución: El usuario debe comprobar que el nombre sea configurable.	
Entradas/ Pasos de Ejecución:	

- Seleccionar el Componente. - Clic derecho encima del componente. - Selecciona la opción de configurar cámara.
Resultado esperado: El usuario pudo configurar nombre de las cámaras de seguridad.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 37 PA #14

Fuente: Elaboración Propia

Caso de Prueba Aceptación	
Número: 14	**Historia de usuario: 14**
Nombre: Configurar las cámaras de seguridad.	
Descripción: Comprobar que se pueden configurar las cámaras de seguridad.	
Condiciones de Ejecución: El usuario debe comprobar que las cámaras de seguridad sean configurables.	
Entradas/ Pasos de Ejecución: - Seleccionar el Componente. - Clic derecho encima del componente. - Selecciona la opción de configurar cámara.	
Resultado esperado: El usuario pudo configurar las cámaras de seguridad.	

Evaluación de la prueba: Prueba satisfactoria.

Tabla 38 PA #15

Fuente: Elaboración Propia

Caso de Prueba Aceptación	
Número: 15	**Historia de usuario: 15**
Nombre: Detener las reproducción de las cámaras de seguridad	
Descripción: Comprobar que se detenga la reproducción de una cámara de seguridad.	
Condiciones de Ejecución: El usuario debe comprobar que detiene la reproducción de una cámara de seguridad.	
Entradas/ Pasos de Ejecución: - Visualizar el componente en el ambiente de ejecución. - Seleccionar la cámara - Clic izquierdo en el botón detener.	
Resultado esperado: El usuario pudo detener la reproducción de la cámara.	
Evaluación de la prueba: Prueba Satisfactoria.	

Tabla 39 PA #16

Fuente: Elaboración Propia

Caso de Prueba Aceptación

Número: 16	**Historia de usuario: 16**
Nombre: Reproducir una cámara de seguridad.	
Descripción: Comprobar que se reproduzca una cámara de seguridad.	
Condiciones de Ejecución: El usuario debe comprobar que se reproduce una cámara de seguridad.	
Entradas/ Pasos de Ejecución: - Visualizar el componente en el ambiente de ejecución. - Seleccionar la cámara - Clic izquierdo en el botón reproducir.	
Resultado esperado: El usuario pudo reproducir una cámara de seguridad.	
Evaluación de la prueba: Prueba satisfactoria**.**	

Tabla 40 PA #17

Fuente: Elaboración Propia

Caso de Prueba Aceptación	
Número: 17	**Historia de usuario: 17**
Nombre: Grabar el contenido de las cámaras de seguridad.	
Descripción: Comprobar que se pueda grabar el contenido de las cámaras de seguridad.	

Condiciones de Ejecución: El usuario debe comprobar que se grabó el contenido de una cámara de seguridad.
Entradas/ Pasos de Ejecución: - Visualizar el componente en el ambiente de ejecución. - Seleccionar la cámara - Clic izquierdo en el botón grabar.
Resultado esperado: El usuario pudo grabar el contenido de una cámara de seguridad.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 41 PA #18

Fuente: Elaboración Propia

Caso de Prueba Aceptación	
Número: 18	**Historia de usuario: 18**
Nombre: Cambiar el brillo de una cámara de seguridad.	
Descripción: Comprobar que se puede cambiar el brillo de una cámara de seguridad.	
Condiciones de Ejecución: El usuario debe comprobar que el brillo de una cámara es ajustable.	
Entradas/ Pasos de Ejecución: - Visualizar el componente en el ambiente de ejecución. - Seleccionar la cámara. - Seleccionar la opción ¨ Brillo ¨en el combo box.	

- Ajustar el slider al brillo deseado.
Resultado esperado: El usuario pudo cambiar el brillo de una cámara de seguridad.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 42 PA #19

Fuente: Elaboración Propia

Caso de Prueba Aceptación	
Número: 19	**Historia de usuario: 19**
Nombre: Cambiar el contraste de una cámara de seguridad.	
Descripción: Comprobar que se puede cambiar el contraste de una cámara de seguridad.	
Condiciones de Ejecución: El usuario debe comprobar que el contraste de una cámara es ajustable.	
Entradas/ Pasos de Ejecución: - Visualizar el componente en el ambiente de ejecución. - Seleccionar la cámara. - Seleccionar la opción ¨ Contraste ¨en el combo box. - Ajustar el slider al contraste deseado.	
Resultado esperado: El usuario pudo cambiar el contraste de una cámara de seguridad.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 43 PA #20

Fuente: Elaboración Propia

Caso de Prueba Aceptación	
Número: 20	**Historia de usuario: 20**
Nombre: Cambiar la saturación de una cámara de seguridad.	
Descripción: Comprobar que se puede cambiar la saturación de una cámara de seguridad.	
Condiciones de Ejecución: El usuario debe comprobar que la saturación de una cámara es ajustable.	
Entradas/ Pasos de Ejecución: - Visualizar el componente en el ambiente de ejecución. - Seleccionar la cámara. - Seleccionar la opción ¨Saturación¨ en el combo box. - Ajustar el slider a la saturación deseado.	
Resultado esperado: El usuario pudo cambiar la saturación de una cámara de seguridad.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 44 PA #21

Fuente: Elaboración Propia

Caso de Prueba Aceptación

Número: 21	**Historia de usuario: 21**
Nombre: Cambiar el volumen de una cámara de seguridad.	
Descripción: Comprobar que se puede cambiar el volumen de una cámara de seguridad.	
Condiciones de Ejecución: El usuario debe comprobar que el volumen de una cámara es ajustable.	
Entradas/ Pasos de Ejecución: - Visualizar el componente en el ambiente de ejecución. - Seleccionar la cámara. - Seleccionar la opción ¨Volumen¨ en el combo box. - Ajustar el slider al volumen deseado.	
Resultado esperado: El usuario pudo cambiar el volumen de una cámara de seguridad.	
Evaluación de la prueba: Prueba satisfactoria.	