Universidad de las Ciencias Informáticas

Facultad 6

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

Título: Módulo de integración con Centrales Receptora de Alarmas para el sistema de Video Vigilancia Xilema Suria

Autores:

Adonis Lima Pérez

Guillermo Alejandro Orama Fabelo

Tutor:

Ing. César Santos Sanabria

"Año 57 de la Revolución"

Junio 2015

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis que tiene por título: Módulo de integración con Centrales Receptoras de Alarma para el sistema de Video Vigilancia Xilema Suria, y reconocemos 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 ____del año _____.

Adonis Lima Pérez

_____

Firma del Autor

Guillermo Alejandro Orama Fabelo

_____

Firma del Autor

César Santos Sanabria

_____

Firma del Tutor

DATOS DE CONTACTO

Autores:

Adonis Lima Pérez

Universidad de las Ciencias Informáticas

La Habana, Cuba

e-mail: alima@estudiantes.uci.cu

Guillermo Alejandro Orama Fabelo

Universidad de las Ciencias Informáticas

La Habana, Cuba

e-mail: gaorama@estudiantes.uci.cu

Tutor:

César Santos Sanabria

Universidad de las Ciencias Informáticas

La Habana, Cuba

e-mail: csanabria@uci.cu

DEDICATORIA

A mi madre, por inculcarme los deseos de estudiar y ser alguien en la vida.

A mi padre, por darme todo su apoyo a lo largo de la carrera en todos los aspectos.

A mi abuela, por ser la persona que más me ha ayudado en todo los tiempos.

A mis tíos, por confiar en que sería un futuro ingeniero.

Adonis

A mis abuelos, por ser lo más grande que tengo en mi vida, sin ellos hoy no estuviera aquí.

A mi mama, por darme su apoyo en todo momento.

A toda mi familia, por estar siempre presente.

A mi novia Lisandra.

Guillermo

"No temo a los ordenadores; lo que temo es quedarme sin ellos"

Isaac Asimov

RESUMEN

En el Centro Geoinformática y Señales Digitales (GEYSED) de la Universidad de las Ciencias Informáticas (UCI) se desarrolla el Sistema de Video Vigilancia Xilema Suria, el cual cuenta con una serie de módulos disponibles para su correcto funcionamiento. Con los módulos incorporados al sistema Xilema Suria, este se limita solamente a la gestión de la vigilancia con cámaras. Por la necesidad que existe de expandir su alcance a nuevas áreas, tener más aceptación en el mercado del software y principalmente poder incorporar otros dispositivos de seguridad y protección, se plantea como objetivo general, desarrollar un componente de software que garantice la integración de las Centrales Receptora de Alarmas (CRA) con el sistema Xilema Suria. Posteriormente se realiza un estudio de las principales empresas y soluciones de las cuales se obtuvo características comunes como, son capaces de gestionar, monitorizar y controlar las señales emitidas por un sistema de alarmas. Estas características se emplean para obtener una solución factible a la necesidad planteada. También se valoran las tecnologías a emplear en el desarrollo de la solución propuesta, seleccionándose las que más se adecuarán al problema en cuestión. Se utiliza la metodología Rational Unified Process (RUP) para guiar el proceso de desarrollo de software, a través de los flujos de trabajo de modelado del dominio, requisitos, análisis, diseño, implementación y prueba. Al finalizar el proceso de investigación y desarrollo de la solución propuesta se tiene como principales resultados, la documentación asociada, así como la integración de las CRA con el sistema Xilema Suria.

Palabras Claves

Componente de software, CRA, dispositivos de seguridad y protección, sistema de alarmas, Xilema Suria.

ABSTRACT

At the Center for Geoinformatics and Digital Signals (GEYSED) of the University of Informatics Science (UCI) a Video Surveillance System Xilema Suria is developed. This system currently has a number of modules required for proper operation. With modules incorporated into Xilema Suria system is only limited to the management of the surveillance cameras. Because of the need to expand its reach into new areas, have more acceptance in the software market and in particular to incorporate other systems of safety and security, is the general objective to develop a software component that ensures the integration of Xilema Suria system with the Alarm Receiving Centre (CRA). Subsequently, a survey of major companies and solutions of common characteristics which are able to manage, monitor and control signals from an alarm system obtained is performed. These features are used to obtain a feasible solution to the consider question. Technology to be used is also value in the development of the proposed solution, selecting the most adapted to the problem in question. RUP (Rational Unified Process) methodology is used to guide the software development process, through the workflow domain modeling, requirements, analysis, design, implementation and testing. On completion of research and development of the proposed solution's main results, associated documentation, and integration of the CRA with Xilema Suria system.

Key Words

Software component, ARC, safety and protection devices, alarms system, Xilema Suria.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Hasta hace poco tiempo, se instalaban sistemas de seguridad y protección en lugares como fábricas, negocios, bancos, incluso hasta en hogares. Hoy en día se expande su utilización a través de la incorporación de nuevos sistemas de alarmas. Debido a la necesidad de tener mayor seguridad y protección, su empleo es cada vez más frecuente, para salvaguardar a las personas de robo y atracos, o a los bienes materiales frente a agresiones tales como sabotaje o incendio, que ocurran en un determinado local. Por tal motivo, los sistemas de alarmas deben estar conectados a una Centrales Receptora de Alarmas (CRA), la cual contará con vigilancia sistemática para garantizar su correcto funcionamiento. Estos dispositivos permitirán reducir el tiempo de ejecución de las acciones a tomar en función del problema presentado, minimizando las pérdidas o daños que puedan ocurrir. Por lo que se puede asegurar que un sistema de seguridad y protección cuenta con las características esenciales para brindar los servicios requeridos por una entidad.

Así, en presencia de un siniestro o una intrusión, en principio lo detectará, luego lo señalizará, y posteriormente iniciará las acciones encaminadas a disminuir o extinguir los efectos. Existen muchas formas de proceder, ya sea accionando mecanismos de extinción, comunicándose con una central receptora de alarmas o conectando cámaras de video vigilancia, entre otros. En conclusión, la aparición de la electrónica junto con las tecnologías de la información y las comunicaciones (TIC), ha permitido un rápido progreso en lo que se refiere a sistemas de seguridad y protección, ya que proporciona una variedad de posibilidades para crearlos. Estos se separan en cuatro grandes bloques de aplicación los cuales son robo y atraco, anti hurto, incendios y sistemas especiales los cueles están compuestos de tres partes básicas las cuales son la CRA, sensores y detectores, y sistemas de aviso y señalización. Aunque pueden ser variables según las necesidades del local a proteger y del presupuesto disponible pare ello. A partir de la breve descripción proporcionada sobre los sistemas de seguridad y protección, en la investigación se plantea la siguiente situación problemática como base para el desarrollo de una posible solución al problema presentado.

En el Centro GEYSED de la UCI se desarrolla el sistema Xilema Suria, el cual para su funcionamiento cuenta con los módulos Gestor, Visor, Grabador, Recuperador y Video Sensores. Actualmente con los módulos incorporados, el sistema Xilema Suria se limita solamente a la gestión de la vigilancia con cámaras. Por tal motivo para lograr que el sistema Xilema Suria pueda realizar una vigilancia eficiente, se

debe hacer uso de videos sensores para detectar movimiento, fuego, objetos abandonados, conteo de personas, entre otros, los cuales sobrellevan un alto consumo de recursos computacionales y exigen la utilización de hardware más avanzado. Con el propósito de minimizar el alto consumo de recursos, así como, lograr mayor aceptación en el mercado del software al expandir su alcance a nuevas áreas y principalmente poder incorporar otros dispositivos de seguridad y protección, se debe garantizar la integración del sistema Xilema Suria con los sistemas de alarmas clásicos. Con esta integración se utilizarán distintos dispositivos electrónicos compuestos por lectores de llaves electrónicas, sirenas de alta potencia, detectores de aperturas, detectores de movimiento, detectores de temperatura, detectores de presión, entre otros. Permitiendo remplazar a los video sensores por los sistemas de alarmas clásicos, en dependencia de la necesidad de los clientes, así como del presupuesto que posean para adquirir el hardware adecuado. Teniendo como prioridad lograr que el sistema Xilema Suria pueda desempeñarse exitosamente aplicando alguna de estas variantes para la seguridad y protección de las personas y bienes. Desarrollando un componente que sea funcional en el monitoreo, control y gestión de los sistemas de alarmas que están ubicados en los lugares donde esté desplegado el producto, ya que en estos momentos no cuenta con herramientas para realizar este tipo de actividades.

Partiendo de lo expuesto anteriormente se plantea el siguiente **Problema de Investigación:** ¿cómo garantizar la gestión de las señales emitidas por una CRA a través del sistema Xilema Suria?, por lo que se define como **Objeto de Estudio:** los procesos relacionados con la gestión de una CRA, enmarcado en el **Campo de Acción:** la gestión de las señales emitidas por una CRA en el sistema Xilema Suria.

A raíz del problema de investigación se traza como **Objetivo General:** desarrollar un componente de software que garantice la gestión de las señales emitidas por una CRA a través del sistema Xilema Suria, en correspondencia con el objetivo general se trazan los siguientes **Objetivos Específicos:**

1. Caracterizar los procesos relacionados con las CRA.
2. Analizar la metodología, las tecnologías y las herramientas necesarias para el desarrollo del módulo de integración con las CRA para el sistema Xilema Suria.
3. Diseñar un módulo de integración con las CRA para el sistema Xilema Suria.
4. Implementar el módulo de integración con las CRA diseñado para el sistema Xilema Suria.
5. Comprobar el funcionamiento del módulo de integración con las CRA para el sistema Xilema Suria.

Para guiar la lógica de la investigación se consideran las siguientes **Preguntas de Investigación:**

1. ¿Cómo se fundamentan los procesos relacionados con las CRA?
2. ¿Qué metodología, tecnologías y herramientas seleccionar para desarrollar el módulo de integración con las CRA para el sistema Xilema Suria?
3. ¿Garantiza el módulo la integración de las CRA con el sistema Xilema Suria?

Con el fin de solucionar el problema de investigación y dar respuesta a las preguntas científicas anteriormente planteadas, se especifican las siguientes **Tareas de la Investigación:**

1. Definición y caracterización de los procesos relacionados con las CRA.
2. Definición y caracterización de la metodología, tecnologías y herramientas seleccionada para el desarrollo del módulo de integración con las CRA para el sistema Xilema Suria.
3. Identificación de los requisitos funcionales y no funcionales asociados al módulo de integración con las CRA para el sistema Xilema Suria.
4. Realización del diseño e implementación del módulo de integración con las CRA para el sistema Xilema Suria.
5. Realización de pruebas para comprobar que la solución cumpla con las especificaciones definidas.
6. Elaboración de toda la documentación asociada al proceso de investigación.

Para guiar la investigación se aplicaron los siguientes **Métodos Científicos:**

Los Métodos Empíricos tienen como objetivo recopilar la información y descubrir un conjunto de hechos como base para verificar una hipótesis y dar respuesta a las preguntas científicas de la investigación. Se emplearon:

1. **Observación:** Es el registro visual de lo que ocurre en una situación real, permite investigar el fenómeno en su manifestación externa (Hernández León, y otros, 2002). Se utiliza para observar el funcionamiento de las CRA y los distintos sistemas de alarmas que soporta.
2. **Entrevista:** Es una conversación planificada para obtener información. Su uso constituye un medio para el conocimiento cualitativo de los fenómenos (Hernández León, y otros, 2002). Se utiliza para definir las principales funcionalidades que debe cubrir la solución que se desea implementar. **Ver Anexo 1**.

Los Métodos Teóricos tienen como objetivo la interpretación de la información empírica obtenida y el establecimiento de generalizaciones, regularidades, teorías y nuevas concepciones en la práctica. Se emplearon:

1. **Analítico-Sintético:** Este método consiste en analizar por medio de la abstracción el objetivo en el pensamiento y descomponerlo en conceptos abstractos, la formulación de dichos conceptos es la forma de lograr un nuevo conocimiento concreto (Hernández León, y otros, 2002). Se emplea para realizar un estudio de la bibliografía referente a las diferentes definiciones asociados a la investigación, permitiendo la extracción de los elementos más importantes que se relacionan con el objeto de estudio.
2. **Modelación:** Permite estudiar de forma analítica la trayectoria histórica real de los fenómenos, su evolución y desarrollo (Hernández León, y otros, 2002). Este método se utiliza fundamentalmente para modelar los artefactos que se generan a lo largo de todo el proceso de desarrollo de software de la solución, ejemplos los diagramas de casos de uso y los diagramas de despliegue.

El presente trabajo de diploma está estructurado en 3 capítulos:

CAPÍTULO 1: FUNDAMENTOS TEÓRICOS DE LA INVESTIGACIÓN

Este capítulo describe aspectos teóricos muy elementales referentes al proceso relacionado con las CRA, para argumentar la presente investigación. A partir del análisis de los aspectos teóricos descritos, se realiza una propuesta de la metodología, tecnologías y herramientas con el fin de dar respuesta a la situación problemática planteada.

CAPÍTULO 2: ANÁLISIS Y DISEÑO DEL MÓDULO

Este capítulo está enfocado al diseño de las funcionalidades para la elaboración del sistema. A partir del modelamiento del problema que existe en el proyecto, se definen requisitos funcionales y no funcionales para darle solución al mismo. Los requisitos funcionales se agrupan en casos de usos y se presenta una descripción de cada uno de ellos. Además se generan los diagramas correspondientes que los modelan, según la metodología escogida.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA DEL MÓDULO

Este capítulo muestra el modelo de implementación como resultado de lo que se desarrolló en el capítulo anterior. Está dirigido a la implementación y prueba de las funcionalidades para el desarrollo del software. El objetivo del capítulo es completar el desarrollo del sistema basándose en la arquitectura definida y al mismo tiempo se comprueba el correcto funcionamiento del sistema por medio de las pruebas correspondientes a la metodología seleccionada.

CAPÍTULO 1: FUNDAMENTOS TEÓRICOS DE LA INVESTIGACIÓN

1.1 Introducción.

En el presente capítulo se exponen las definiciones que resultan relevantes en el desarrollo de la investigación. Se realiza un estudio sobre las soluciones existentes con el fin de identificar sus principales características y funcionalidades, analizando los principales servicios que ofrecen. Se desarrolla un análisis de la problemática presentada anteriormente, para poder comprender y argumentar la necesidad actual. También se argumenta la elección y uso de las herramientas involucradas en el desarrollo de la aplicación.

1.2 Términos asociados al Dominio del Problema.

En el desarrollo de la solución propuesta se tratan varias definiciones que se reflejarán a lo largo de la investigación, los cuales son, un **sistema de seguridad y protección** no es más que, un conjunto de dispositivos colocados estratégicamente en el perímetro de un sitio específico, para detectar la presencia, irrupción, o invasión, de un individuo que no posea acceso permitido. (MaquinariaPro, 2012). Para que un sistema de seguridad y protección brinde servicios correctamente, debe estar compuesto por dos dispositivos fundamentales, los cuales son, primeramente, un **sistema de alarma**, el cual es, un elemento de seguridad pasiva que no evita una situación anormal, pero sí es capaz de advertir de ella, cumpliendo así, una función disuasoria frente a posibles problemas. (Syscom, 2014). Posteriormente para que los sistemas de alarmas maximicen su efectividad deben estar conectados a una **CRA** la cual es, un dispositivo de control, recepción y monitorización de las señales emitidas por un sistema de alarma, debido a la activación de un salto de alarma, producido por un intento de violación en una área protegida, cuando el sistema de seguridad y protección está conectado, por lo general 24 horas del día y los 365 días del año. (Gob, 2013).

Al conectar un sistema de alarma con una CRA, se puede decir que un sistema de seguridad y protección está instalado correctamente, pero antes de realizar dicha instalación habrá que tener en cuenta ciertas consideraciones, ya que definirán los resultados de su aplicación mientras esté desplegado, siendo factible para cualquier entidad que desee implementarlo. Algunas de las consideraciones a tener en cuenta son:

1. Qué se desea proteger.
2. De quién se desea proteger.
3. Situación de los objetos que se desean proteger.
4. Entorno de dichos objetos.
5. Valor de los objetos.
6. Presupuesto de que se dispone. (Xunta, 2014).

1.3 Funcionamiento del Sistema de Alarmas.

El sistema de alarma se compone de equipos eléctricos que trabajan en conjunto para proteger los puntos más vulnerables de una entidad. Cada componente realiza funciones específicas, las cuales se mencionarán a continuación detallando una breve descripción de su principal tarea:

1. Teclado: es el dispositivo que se utiliza para configurar, activar o desactivar el sistema de alarma..
2. Batería de respaldo: es el dispositivo encargado de mantener al sistema funcionando en caso de fallas de energía eléctrica.
3. Sensores y detectores: son los dispositivos utilizados para supervisar las distintas áreas de la entidad por donde puedan existir vulnerabilidades.
4. Sirenas y bocinas: son los dispositivos que proporcionan señales audiovisuales, advirtiendo de una posible intrusión. (Syscom, 2014).

Una vez que un sistema de alarma comienza a funcionar o se activa, puede tomar acciones de forma automática, dependiendo del sistema de alarma instalado. Si detecta la intrusión de una persona en un área determinada, tiene como función principal la de advertir del problema, enviando una alerta al dispositivo que gestiona, controla y monitoriza los eventos ocurridos, como por ejemplo la CRA. Además puede cumplir una función disuasoria, activando una sirena o bocina, que proporciona señales audiovisuales, advirtiendo de la ocurrencia de un posible problema en el local. (Syscom, 2014). Uno de los usos más difundidos de un sistema de alarma es mediante una conexión con la CRA, la cual se puede comunicar mediante una línea telefónica, un transmisor por radiofrecuencia, por comunicación TCP/IP o por servicios de Internet. Esto permite principalmente, reducir el tiempo de ejecución de las acciones a tomar, para evitar las pérdidas que pudieran ocasionarse.

1.4 Componentes de la Central Receptora de Alarmas.

La CRA se utiliza con el objetivo de garantizar la seguridad y protección de una entidad. Se encarga de monitorizar los sistemas de alarmas, por lo cual es un servicio que ofrece garantía, estando disponible todo el tiempo. Esto es posible debido a que dispone distintos componentes que controlan el acceso a las zonas protegidas, los cuales son:

1. Baterías: Se colocan para prevenir cualquier falta de fluido eléctrico, para manipulación intencionada o para fallo del sistema que lo suministra, asegurando de este modo el funcionamiento continuo a la central y a los detectores instalados.
2. Teclado: Está conectado a la central receptora de alarma, ubicándose generalmente en un lugar de fácil acceso para el usuario al cual le sirve para conectar y desconectar el sistema de alarma.
3. Panel de Alarma: Es la tarjeta electrónica en la que se conectan los dispositivos de entrada (sensores) y los dispositivos de salida (línea telefónica, radios, módulos de transmisión, celular GPRS, comunicación TCP/IP).
4. Microprocesador: Es el cerebro de la central receptora de alarma, recibe información continuamente del estado de los detectores y sensores instalados en el sistema de alarma, accionando las diferentes salidas en caso de incidencias.
5. Memoria EPROM: Es un chip electrónico donde se encuentran almacenadas todas las instrucciones y datos necesarios para que funcione el microprocesador.

La CRA es el dispositivo que recibe la señal eléctrica, que por algún motivo es activado por los detectores o sensores que conforman un sistema de alarmas. Al recibir la señal, que puede ser vía teléfono o por comunicación TCP/IP, los circuitos electrónicos que lleva en su interior se ponen en marcha y comunican al software de gestión que está monitorizando la CRA, la existencia de una violación. En el momento de la activación de un sistema de alarmas, ubica la posición exacta de la alarma activada, proporcionando información detallada, para que realice las acciones pertinentes ante la ocurrencia de una intrusión. También para que la CRA funcione sin ningún tipo de fallas, se debe proteger de posibles manipulaciones y sabotajes, por lo cual se debe instalar en un armario metálico muy resistente, con llave de seguridad para el acceso al dispositivo. De igual forma, debe de estar provista de una fuente de alimentación, que posibilite su funcionamiento y una batería que le proporcione autonomía ante cualquier fallo de electricidad (Gob, 2013).

1.5 Empresas y Soluciones existentes.

En la actualidad existen muchas empresas que se dedican al negocio de la seguridad y protección, el uso de sistemas de alarmas conectadas a una CRA es de vital importancia para que tengan gran aceptación en el mercado nacional e internacional, por lo que estas empresas ofrecen soluciones que integran dichos dispositivos para tener un control eficiente de sus servicios. Algunas de las empresas y soluciones a nivel mundial se encuentran:

VIOSONIC, es una empresa que provee soluciones avanzadas para respaldar servicios de control de seguridad y protección, rentables y eficaces. Cuenta con una amplia variedad de soluciones de verificación en alarmas visuales y de audio, incorpora un sistema de seguridad y protección, que cuenta con detectores y cámaras integradas, así como tecnologías avanzadas para permitir una evaluación exacta de la situación antes de responder. Posee avanzadas herramientas de administración que ofrecen nuevas oportunidades para mejorar la eficiencia y el servicio. (Visonic, 2014).

GUNNEBO, es una empresa que ofrece servicios seguros y efectivos, gracias a su avanzada CRA. Dicha central se encuentra totalmente automatizada mediante los más avanzados sistemas informáticos, que permiten conocer de inmediato las alarmas recibidas, el tratamiento de las mismas, así como la gestión a realizar. Sus soluciones, facilitan la resolución de las alarmas y el acceso a los bancos de datos en tiempo real, mediante los terminales de pantalla y teclado, y con terminales teleimpresores existentes en una sala de control y monitoreo. (Gunnebo, 2014).

VISEGUR, es una empresa que provee una solución integrada a la CRA, automatizada mediante avanzados sistemas informáticos, es pionera en la recepción, verificación y transmisión de señales de alarma. Opera en cuestión de segundos una vez producido un salto de alarma, dando aviso inmediato. Ofrece los servicios de detección de intrusión y atraco, detección de incendio, sabotaje de sistemas, fallos de red, control de baterías. También posee tele vigilancia de instalaciones, así como detección y control de averías de los sistemas de seguridad y protección. (Visegur, 2014).

MASTERMIND, es una plataforma de software integrado de gestión de alarmas, ofrece una solución totalmente integrada para las CRA. La gama de productos incluye la gestión de sistemas de alarmas, integración de vídeo, control de acceso, la integración del teléfono, el seguimiento de GPS y la presentación de informes. Es conocido por su fiabilidad probada y la velocidad de procesamiento de las alarmas. Cuenta de varias aplicaciones integradas en una arquitectura cliente/servidor, que gracias al uso

de una única base de datos para todos los tipos de servicios de seguridad y protección es posible acceder a la información de forma rápida y sencilla. Actualmente se implementa en las principales CRA del mundo, como la de empresas mencionadas anteriormente. (Enai, 2014).

Al presentar las principales empresas y sus soluciones, en el área de los sistemas de seguridad y protección integrados a las CRA, se puede mencionar como características generales que, son capaces de gestionar, monitorizar y controlar las señales emitidas por un sistema de alarmas, así como realizar tareas de análisis, generar alertas y reportes para diferentes tipos de intrusión. Aunque estas soluciones son privativas, es factible comentar que su estudio, sirven para fomentar una base sólida la cual puede ser aplicada en el proceso de desarrollo que se está llevando en la investigación. Extrayendo de cada uno las características más relevantes, para llevar a cabo la solución del problema. Abogando por la utilización de software libre, siendo este el principio fundamental para el proceso de desarrollo de toda la investigación.

En la investigación se propone una solución totalmente diferente a las que se desarrollan en el mundo, dado por las características del contexto donde se elabora. Esto se debe a que los sistemas de seguridad y protección como el sistema Xilema Suria por lo general son integrados a las CRA y en la solución propuesta se debe realiza el proceso inverso, o sea, garantizar la integración de las CRA con el sistema Xilema Suria. Esta peculiaridad está dada por la necesidad de reutilizar el sistema Xilema Suria como el software encargado para monitorizar, controlar y gestionar todo los sistemas de seguridad y protección conectados a la CRA. Aunque la línea base para su desarrollo, será semejante a como se manipula en las grandes empresas de seguridad y protección.

Esta forma de proceder trae consigo varias ventajas, la principal, el centro GEYSED no desarrollaría un software para dedicarlo a los procesos de gestión de la CRA, porque ya se estaría reutilizando los componentes necesarios del sistema Xilema Suria y los pondría a su disposición. Otra ventaja seria la disminución de recursos, ya sea tiempo, presupuesto o personal, debido a las facilidades propuestas por la solución de la investigación. En resumen, se dispondría de una solución totalmente integrada para procesar distintos tipos de dispositivos de seguridad y protección como, lectores de llaves electrónicas, sirenas de alta potencia, detectores de aperturas, detectores de movimiento, detectores de calor y no dependería solamente de la vigilancia con cámaras como lo hace actualmente el sistema Xilema Suria.

1.6 Metodología, Tecnologías y Herramientas para el Desarrollo de Software.

Las metodologías de desarrollo de software surgen ante la necesidad de poseer un conjunto de procedimientos, técnicas, herramientas y soporte documental a la hora de desarrollar un producto de software. Estas contribuyen a mantener el proceso de realización del producto ordenado desde que comienza hasta que termina. Actualmente, seguir una metodología de desarrollo de software, garantiza mantenerse en competencia, pues el mercado de software es muy amplio, variable y competitivo.

1.6.1 Rational Unified Process (RUP).

RUP es una metodología de desarrollo de software pesada o tradicional, destinada para satisfacer los requerimientos de los usuarios finales en tiempo y con el presupuesto establecido. Se utiliza principalmente en proyectos grandes y con requisitos de poca variación los cuales pueden ser adaptados y extendido a las necesidades de la organización que lo utilice. Realiza un modelado visual de software y permite gestionar una potente documentación y control de cambios. (Figueroa, y otros, 2009).

RUP divide el proceso de desarrollo en cuatro etapas que definen el ciclo de vida del software:

1. Inicio: El objetivo es determinar la visión del proyecto y definir lo que se desea realizar.
2. Elaboración: Etapa en la que se determina la arquitectura óptima del proyecto.
3. Construcción: Se obtiene la capacidad operacional inicial.
4. Transición: Obtener el producto acabado y definido. (Figueroa, y otros, 2009).

RUP posee tres características esenciales que lo distinguen de las restantes metodologías de desarrollo de software y lo hacen único:

1. Dirigido por casos de uso: Los casos de uso son el claro reflejo de lo que el cliente planteó que deseaba en la modelación del negocio a través de los requerimientos. A partir desde que aparecen los casos de uso, estos pasan a guiar el proceso de desarrollo, ya que los modelos que se obtienen como resultado de la realización de los flujos de trabajo están presentes en ellos.
2. Iterativo e incremental: Todas las fases se desarrollan mediante iteraciones. Una iteración incluye actividades de todos los flujos de trabajo, las cuales va refinando en cada iteración.
3. Centrado en la arquitectura: La arquitectura muestra una visión común del sistema completo. RUP se desarrolla mediante iteraciones, dando prioridad a los casos de uso arquitectónicamente significativos. (Figueroa, y otros, 2009).

Adaptándolo a las características de la investigación, la metodología de desarrollo RUP, servirá para regir el proceso de desarrollo de software en la solución propuesta. A través de los flujos de trabajo de modelado del dominio, requisitos, análisis, diseño, implementación y prueba. De este modo el punto inicial del desarrollo estará determinado por la descripción de los conceptos asociados al dominio del problema, que constituyen la entrada fundamental para la captura de requisitos. Una vez agrupados los requisitos en casos de usos se podrá realizar en primer lugar el proceso de análisis, el de diseño y por último el de implementación. Posibilitando una vez obtenidos todos estos artefactos preparar las pruebas necesarias para verificar que el sistema proporciona las características descritas en los casos de usos.

La utilización de esta metodología permitirá realizar iteraciones completas a cada caso de uso, garantizando que se prioricen las funcionalidades más complejas. Para ello, el equipo de desarrollo es responsable de especializarse en cada uno de los roles necesarios en cada flujo de trabajo. Se hace uso de la metodología de desarrollo RUP teniéndose en cuenta la poca experiencia con que cuenta el equipo de desarrollo, ya que genera una serie de artefactos que facilitarán las labores de implementación. Otro factor de peso a tener en cuenta para su elección, es la necesidad de integrar la documentación de la solución propuesta con la del sistema Xilema Suria en el cual se utiliza esta metodología.

1.6.2 Lenguaje de Modelado UML 2.0.

Los lenguajes de modelado son una guía estándar que proveen un conjunto de diagramas y mecanismos para modelar sistemas informáticos. Permite especificación, visualización, construcción y documentación de los artefactos generados de un proceso de sistema de software. El UML 2.0 es uno de los lenguajes de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. Ofrece un estándar para describir un plano del sistema o modelo, incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, también aspectos concretos, como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables. (Hamilton, y otros, 2006).

Algunas de las características representativas de UML 2.0 son:

1. Es un lenguaje de modelado de propósito general que pueden usar todos los modeladores.
2. Incluye todos los conceptos que se consideran necesarios para utilizar un proceso iterativo.
3. Se basa en construir una sólida arquitectura para resolver requisitos dirigidos por casos de uso. (Hamilton, y otros, 2006).

1.6.3 Herramienta CASE Visual Paradigm 8.0.

Las herramientas CASE son aplicaciones informáticas que facilitan completamente o en alguna de sus fases, la automatización del proceso de desarrollo de software, a través del modelado de sistemas. Favorece la calidad del sistema desarrollado, la gestión y el dominio sobre el proyecto en cuanto a su planificación, ejecución y control. Mejora la productividad y la eficacia de las áreas de desarrollo de los sistemas informáticos. Garantiza la consistencia de los procedimientos y verifica el uso de todos los elementos en el sistema diseñado. (Scribd, 2013).

Para modelar los artefactos generados en la solución de la investigación, se utilizará Visual Paradigm 8.0 porque brinda soporte para todos los diagramas que propone el lenguaje de modelado UML. El mismo propicia un conjunto de herramientas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis, el diseño, hasta la generación del código fuente de los programas y la documentación asociada.

Algunas de las características representativas de Visual Paradigm 8.0 son:

1. Diseño centrado en casos de uso, enfocado al negocio, lo que genera un software de mayor calidad.
2. Uso de un lenguaje estándar para todo el equipo de desarrollo, y facilitar la comunicación mutua.
3. Capacidades de ingeniería directa e inversa.
4. El modelo y el código permanecen sincronizados en todo el ciclo de desarrollo.
5. Entorno todo en uno para la especificación de los detalles de los casos de uso, incluyendo la especificación del modelo general y de las descripciones de los casos de uso.
6. Captura de requisitos mediante diagramas, modelado de caso de uso y análisis textual. (Paradigm, Visual, 2014).

1.6.4 Lenguaje de Programación C++11.

Para implementar la solución propuesta en la investigación, se definió la utilización del lenguaje de programación C++11, principalmente, por ser la variante en la cual han desarrollado las funcionalidades del sistema Xilema Suria. Teniendo en cuenta que C++11 es un lenguaje de programación que permite realizar aplicaciones comercializables, sin la necesidad de pagar una licencia, abogando por una soberanía tecnológica, teniendo como principio fundamental, potenciar el desarrollo de software libre. El lenguaje de programación C++11 posee múltiples características que serían aprovechadas en la implementación de la solución propuesta en la investigación.

Algunas de las características representativas del lenguaje de programación C++11 son:

1. Es un lenguaje orientado a objetos.
2. Es un lenguaje estandarizado y un mismo código fuente se puede compilar en diversas plataformas.
3. Ofrece gran cantidad de recursos, como, las formas de encapsulamiento, la sobrecarga de operadores y funciones, la herencia y el polimorfismo. (Mastermagazine, 2012).

1.6.5 Entorno de Desarrollo Integrado (IDE) QtCreator v3.2.

QtCreator v3.2 es un IDE que cuenta con herramientas diseñadas para simplificar la creación de aplicaciones e interfaces de usuario para el escritorio, embebidos y plataformas móviles. Puede reutilizar el código de manera eficiente para trabajar en múltiples plataformas con una base de código. El sistema modular de C++, las bibliotecas de clases y las herramientas que posee dicho IDE, fácilmente permite a los implementadores crear aplicaciones para una plataforma, generar y ejecutar el despliegue en otra plataforma. Cuenta con una interfaz muy cómoda, funcional y estéticamente agradable. Incorpora una herramienta de búsqueda eficaz donde puede localizar fácilmente las clases, métodos y ficheros. Cuenta con el QtDesigner que ayuda a diseñar formularios de interfaz de usuario, e incluye también la gestión y finalización del código de la solución. (Digia, 2013).

Algunas de las características representativas del entorno de desarrollo integrado QtCreator v3.2 son:

1. Posee un avanzado editor de código C++.
2. Posee una GUI integrada, un diseñador de formularios y un depurador visual.
3. Posee herramientas para proyectos y administración.
4. Presenta ayuda sensible al contexto integrado.
5. Presenta resaltado y auto-completado de código.
6. Presenta soporte para refactorización de código. (Digia, 2013).

Para la solución propuesta en la investigación se utilizará el entorno de desarrollo integrado QtCreator v3.2, definido por las características mencionadas anteriormente y principalmente por la política de migración a software libre establecida por el centro GEYSED. También porque debe existir una compatibilidad óptima entre la solución propuesta en la investigación y el sistema Xilema Suria, desarrollado en dicho IDE.

1.6.6 Tecnología de Comunicación ZeroMQ.

ZeroMQ es una tecnología que permite el intercambio de mensajes a través de comunicación TCP/IP de manera muy rápida, sencilla y compatible entre plataformas, utilizando distintos patrones de distribución que permiten realizar, balanceos de carga, redundancia de nodos y esquemas de servicios, todo esto distinto a los que normalmente se usan en los sistemas de colas de mensajes. Es una tecnología GPL y de código ampliamente abierto, utilizada en muchas empresas por la escalabilidad que proporciona. Sus características principales son, rendimiento y simplicidad. Además ofrece soporte para la mayoría de sistemas operativos como Linux y Windows. (iMatix. 2014).

Para la solución propuesta en la investigación se utilizará la tecnología de comunicación ZeroMQ a través de la clase *GestorClientProxy* que ofrece el Módulo Gestor, debido a que es una librería de mensajería que permite diseñar sistemas de comunicación. Ofrece rendimiento y compatibilidad con el lenguaje de programación seleccionado, además de ser la tecnología usada para la comunicación entre el sistema Xilema Suria y sus demás módulos.

1.7 Conclusiones Parciales.

En este capítulo se precisaron una serie de definiciones relacionados con la problemática planteada, lo cual posibilitó reflejar con claridad el significado de los principales temas a tratar durante la investigación.

1. Se profundizó acerca del objeto de estudio, precisando aspectos importantes a tener en cuenta para realizar los procesos relacionados con las CRA.
2. Se caracterizaron los sistemas de alarmas y las CRA, propiciando una amplia descripción, la cual garantiza un mejor entendimiento sobre el funcionamiento de estos dispositivos, permitiendo identificar funcionalidades a incorporar en la solución que se desea desarrollar.
3. Se analizaron las soluciones existentes evidenciando la utilidad de las CRA, lo cual posibilitó extraer características comunes en estos sistemas.
4. Se caracterizaron las tecnologías, herramientas y la metodología a utilizar en el desarrollo de la solución, enunciándose de cada una de ellas las principales características y las facilidades que ofrecen al equipo de desarrollo para dar solución a la problemática planteada.

CAPÍTULO 2: ANÁLISIS Y DISEÑO DEL MÓDULO

2.1 Introducción.

En el presente capítulo se realiza un estudio del dominio del negocio, que tiene como fin comprender las características que posee el mismo y que luego serán implementadas en la solución. También se presentan los artefactos resultantes del flujo de trabajo de la metodología seleccionada, mostrándose un correcto uso de la arquitectura y los patrones de diseño. Además se describen los requisitos funcionales, los casos de uso y actores del sistema.

2.2 Modelo del Dominio.

En el modelo de dominio se recogen los principales conceptos u objetos que conforman el entorno donde se desarrolla el negocio, en este caso las CRA, y los diferentes procesos que ocurren en el mismo. Se obtiene desde la base de conocimientos de unos pocos expertos del dominio o posiblemente del conocimiento asociado con sistemas similares al que se propone desarrollar en la solución. Es una representación de las clases conceptuales del mundo real, no de componentes de software. (Larman, 2013). A continuación se muestra el modelo de dominio el cual se representa usando el lenguaje de modelado UML, a través de un diagrama de clases donde se muestran conceptos u objetos del dominio del problema como clases conceptuales y sus relaciones.

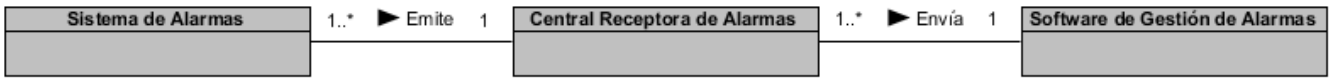

Fig. #1. Modelo del Dominio.

2.2.1 Descripción del Modelo de Dominio.

Con el objetivo de obtener un mejor entendimiento del modelo de dominio, se proporcionan las definiciones de las distintas clases identificadas.

1. **Sistema de Alarmas:** son los dispositivos eléctricos que trabajan en conjunto, ya sean sensores, detectores, sirenas y bocinas para supervisar distintas áreas de una entidad.
2. **Central Receptora de Alarmas:** es el dispositivo donde se ubica el panel de control, el cual recibe la señal eléctrica, que por algún motivo es activado por un sistema de alarmas.
3. **Software de Gestión de Alarmas:** software encargado para monitorizar, controlar y gestionar todos los sistemas de alarmas conectados a la central receptora de alarmas.

En el diagrama de la Fig. #1 se aprecia que, un sistema de alarmas comienza a funcionar cuando es activado por alguno de sus dispositivos, ya sea sensor o detector, este emite una señal eléctrica, la cual es recibida por una CRA que detecta el flujo de información y lo convierte para que el software de gestión de alarmas encargado de interpretar los datos pueda monitorizar, controlar y gestionar toda la información obtenida, para que realice las acciones pertinentes ante la activación de un dispositivo.

2.3 Especificación de los Requisitos de Software.

Para poder llevar a cabo las tareas de implementación es necesario que el equipo de desarrollo defina un conjunto de funcionalidades que debe tener la solución a desarrollar, así como las características que debe reunir el entorno donde se valla a desplegar.

2.3.1 Requisitos Funcionales del Sistema.

Los requisitos funcionales son declaraciones de los servicios que el sistema debe proporcionar, el cual se aplica a entradas particulares y situaciones donde debe mostrar correctamente los resultados deseados. Son condiciones o capacidades que debe cumplir el software, por lo que estas necesidades las plantean los clientes, evidenciando que debe tener y hacer el sistema (Somerville, 2005).

Nota: Se utiliza el prefijo RF en su nomenclatura.

RF1 Adicionar una zona: El sistema debe permitir al usuario adicionar los datos de una zona.

Entrada: número, descripción, central. (Formato: Cadena de Caracteres). Cámaras asociadas a la zona, plugins de acción asociados a la zona. (Formato: Listado).

Salida: Se adiciona en el sistema los datos de la zona.

RF2 Mostrar una zona: El sistema debe permitir al usuario mostrar los datos de una zona.

Entrada: No procede.

Salida: Se muestra en el sistema los datos de la zona seleccionada.

RF3 Eliminar una zona: El sistema debe permitir al usuario eliminar los datos de una zona.

Entrada: No procede.

Salida: Se eliminan en el sistema los datos de la zona seleccionada.

RF4 Editar una zona: El sistema debe permitir al usuario editar los datos de una zona.

Entrada: número, descripción, central. (Formato: Cadena de Caracteres). Cámaras asociadas a la zona, plugins de acción asociados a la zona. (Formato: Listado).

Salida: Se actualiza en el sistema los datos de la zona seleccionada.

RF5 Adicionar un dispositivo: El sistema debe permitir al usuario adicionar los datos de un dispositivo.
Entrada: Tipo, descripción, nombre, zona, central. (Formato: Cadena de Caracteres).
Salida: Se adiciona en el sistema los datos del dispositivo.
RF6 Mostrar un dispositivo: El sistema debe permitir al usuario mostrar los datos de un dispositivo.
Entrada: No procede.
Salida: Se muestra en el sistema los datos del dispositivo seleccionado.
RF7 Eliminar un dispositivo: El sistema debe permitir al usuario eliminar los datos de un dispositivo.
Entrada: No procede.
Salida: Se eliminan en el sistema los datos del dispositivo seleccionado.
RF8 Editar un dispositivo: El sistema debe permitir al usuario editar los datos de un dispositivo.
Entrada: Tipo, descripción, nombre, zona, central. (Formato: Cadena de Caracteres).
Salida: Se actualiza en el sistema los datos del dispositivo seleccionado.
RF9 Adicionar un plugin: El sistema debe permitir al usuario adicionar un plugin de acción a realizar.
Entrada: Ubicación del plugin de acción. (Formato: Cadena de Caracteres).
Salida: Se adiciona en el sistema el plugin de acción.
RF10 Eliminar un plugin: El sistema debe permitir al usuario eliminar un plugin de acción a realizar.
Entrada: Nombre del plugin de acción. (Formato: Cadena de Caracteres).
Salida: Se elimina del sistema el plugin de acción seleccionado.
RF11 Mostrar una alarma: El sistema debe permitir al usuario mostrar los datos asociados a una alarma.
Entrada: No procede.
Salida: Se muestra en el sistema los datos de la alarma generada.
RF12 Eliminar una alarma: El sistema debe permitir al usuario eliminar los datos de una alarma.
Entrada: No procede.
Salida: Se eliminan en el sistema los datos de la alarma generada.
RF13 Ejecutar plugins: El sistema debe permitir que se ejecuten los plugins de acción asociados a una zona cuando sea activada.
RF14 Generar alarmas: El sistema debe permitir que se generen alarmas cuando una zona sea activada.
RF15 Configurar conexión con el modulo gestor: El sistema debe permitir al usuario introducir los datos para la conexión con el Módulo Gestor.
Entrada: Dirección IP, puerto. (Formato: Cadena de Caracteres).
Salida: Se actualiza en el sistema los datos para la conexión con el modulo gestor.

RF16 Configurar ubicación de la carpeta plugins: El sistema debe permitir al usuario introducir la ubicación donde se almacenan los plugins.
Entrada: Ubicación. (Formato: Cadena de Caracteres).
Salida: Se actualiza en el sistema la ubicación donde se almacenan los plugins.
RF17 Configurar ubicación de la base de datos: El sistema debe permitir al usuario introducir la ubicación donde se encuentra la base de datos.
Entrada: Ubicación. (Formato: Cadena de Caracteres).
Salida: Se actualiza en el sistema la ubicación donde se encuentra la base de datos.
RF18 Configurar plugin de CRA: El sistema debe permitir al usuario editar los datos de configuración de un plugin de central.
Entrada: No procede.
Salida: Se actualiza en el sistema los datos de configuración.

2.3.2 Requisitos no Funcionales del Sistema.

Los requisitos no funcionales se denominan a las restricciones de los servicios ofrecidos por el sistema, donde se incluyen las restricciones de tiempo, de estándares, de patrones y sobre el proceso de desarrollo del producto. Son las propiedades emergentes como: la fiabilidad, tiempo de respuesta, la usabilidad, eficiencia, soporte y la capacidad de almacenamiento. Dichas propiedades constituyen accesorios fundamentales para la interfaz, la rapidez y los elementos que representan un sistema factible y seguro, además de especificar los requisitos básicos que deberá cumplir el sistema para un correcto funcionamiento. Los requisitos no funcionales representan los datos que hacen posible un mejor entendimiento de las funcionalidades específicas de cualquier sistema y hacen más entendible los resultados obtenidos (Somerville, 2005).

Nota: Se utiliza el prefijo RnF en su nomenclatura.

Requisitos de Usabilidad:

RnF1 El sistema debe hacer uso de botones, imágenes o textos que indiquen de modo intuitivo la función que realiza.

Requisitos de Disponibilidad:

RnF2 El sistema debe estar accesible las 24 horas del día, garantizando que se pueda monitorizar las alarmas generadas, por la activación de los dispositivos conectados a la CRA.

Requisitos de Ambiente:

RnF3 El sistema para que funcione, debe ser capaz de emitir una alarma, por lo que requiere al menos un dispositivo conectado a una zona de la CRA.

Restricciones del Diseño:

RnF4 El sistema estará implementado con el lenguaje de programación C++11, utilizando el framework Qt v5.4, en el IDE QtCreator v3.2.

Requisitos de Interfaz:

RnF5 El sistema contará con las interfaces gráficas para el usuario implementadas para concebirse en un ambiente sencillo y de navegación fácil para el usuario.

Requisitos de Hardware:

RnF6 Procesador Dual Core a 2.20 GHz o superior.

RnF7 Disco Duro de 40 GB o superior.

RnF8 Memoria RAM de 2 GB o superior.

Requisitos de Software:

RnF9 El sistema debe funcionar en la plataforma para GNU/Linux.

2.4 Modelo del Sistema.

Luego de identificados los requisitos funcionales que tendrá la solución propuesta, se agruparon en Casos de Uso (CU), los cuales se representan en el Diagrama de Casos de Uso del Sistema (DCUS). Un CU se define como una secuencia de acciones que el sistema proporciona para los actores. Los mismos representan un punto medio, de entendimiento para los desarrolladores del sistema, de ahí la importancia de una buena descripción de los mismos. Proporcionan la entrada fundamental para el análisis, diseño y las pruebas del sistema.

2.4.1 Definición de los Actores del Sistema.

Para crear un CU, primero se deben identificar los diferentes tipos de actores que se utilizan en la solución propuesta. Un actor es quien se comunica con el sistema y es externo al mismo. Es importante indicar que un actor y un usuario no son la misma cosa. Un usuario normal puede jugar un número de papeles diferentes cuando utiliza un sistema, por lo tanto un actor representa una clase de entidades externas, no siempre personas, que lleva a cabo un papel. (Pressman, 2001).

Tabla. #1. Actores del Sistema.

Actor	Descripción
Administrador	Posee el permiso adecuado para realizar diversas tareas en el sistema. Es el actor que da inicio a las funcionalidades Gestionar Zonas, Gestionar Dispositivos, Administrar Plugins, Administrar Alarmas, Configuración General y Configurar Plugin de CRA.
Panel de Control	Es el encargado de emitir al sistema la señal de activación de los distintos dispositivos ubicados en las zonas. Es el actor que da inicio a las funcionalidades Ejecutar Plugins y Generar Alarmas.

2.4.2 Definición de los Casos de Uso del Sistema.

Un Caso de Uso del Sistema (CUS) es un fragmento de una funcionalidad que ofrece la solución para aportar un resultado de valor para sus actores. En ellos se agrupan algunos requisitos funcionales, se especifica las acciones o actividades que caracterizan la interacción actor-sistema, mostrando un flujo de eventos que describen gráficamente los procesos y condiciones indispensables para entender la correlación existente entre los diversos requisitos funcionales. Un CU es una descripción de los pasos o las actividades que deberán realizarse para llevar a cabo algún proceso, y representa una secuencia de iteraciones que se desarrollarán entre un sistema y sus actores, en respuesta a un evento que inicia un actor principal sobre el propio sistema. (Pressman, 2001). Para la solución propuesta se identificaron los siguientes CU que engloban todos los requisitos funcionales previamente definidos:

Tabla. #2. Definición de los Casos de Uso.

CU1 Gestionar Zonas	**CU2 Gestionar Dispositivos**	**CU3 Administrar Plugins**
RF1 Adicionar una zona. **RF2** Mostrar una zona. **RF3** Eliminar una zona. **RF4** Editar una zona.	**RF5** Adicionar un dispositivo. **RF6** Mostrar un dispositivo. **RF7** Eliminar un dispositivo. **RF8** Editar un dispositivo.	**RF9** Adicionar un plugin. **RF10** Eliminar un plugin.
CU4 Administrar Alarmas	**CU5 Ejecutar Plugins**	**CU6 Generar Alarmas**
RF11 Mostrar una alarma. **RF12** Eliminar una alarma.	**RF13** Ejecutar plugins.	**RF14** Generar alarmas.

CU7 Configuración General	**CU8 Configurar Plugin de CRA**
RF15 Configurar conexión con el modulo gestor. **RF16** Configurar ubicación de la carpeta plugins. **RF17** Configurar ubicación de la base de datos.	**RF18** Configurar plugin de CRA.

2.4.3 Diagrama de Casos de Uso del Sistema.

Un Diagrama de Casos de Usos del Sistema (DCUS) representa gráficamente los procesos y sus interacciones con los actores. Sirven para mostrar la interacción que ocurre entre los actores principales con las diversas funcionalidades que engloba un sistema determinado. Muestra cómo puede o cómo debe actuar el sistema antes diversas situaciones y cómo debe responder ante las acciones que realiza un actor determinado, ya sea cumpliendo con las funcionalidades principales o las respuesta antes incidentes que necesiten de una acción rápida y correcta por parte del software, mostrando a los desarrolladores qué características y estructura deben implementar para darle una acertada solución a los diversos inconvenientes. (Pressman, 2001). A continuación se muestra el DCUS diseñado para la solución propuesta.

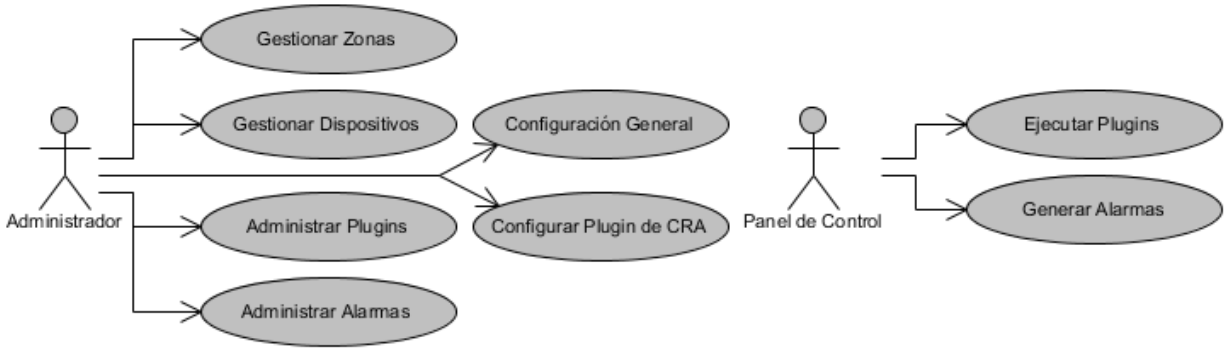

Fig. #2. Diagrama de Casos de Usos del Sistema.

2.4.4 Especificación de los Casos de Uso del Sistema.

A continuación se presenta la descripción textual del CU Gestionar Zonas por ser considerado arquitectónicamente significativo. Las descripciones de los CU Gestionar Dispositivos, Administrar Plugins, Administrar Alarmas, Ejecutar Plugins, Generar Alarmas, Configuración General y Configurar Plugin de CRA pueden encontrarse en los anexos del presente trabajo. **Ver Anexo 2**.

Tabla. #3. Descripción del CU Gestionar Zonas.

Nombre del CU1	Gestionar Zonas
Objetivo	Manipular los datos de una zona.
Actores	Administrador: (Inicia) adiciona, muestra, elimina y edita los datos de una zona.
Resumen	El CU inicia cuando el administrador selecciona la operación de adicionar, mostrar, eliminar o editar los datos de una zona. El CU termina cuando los datos de la zona se actualizan en el sistema.

Complejidad	Alta
Prioridad	Crítico
Precondiciones	Para Mostrar, Eliminar o Editar: La zona ha sido adicionada. La zona ha sido seleccionada.
Postcondiciones	Mostrar la lista de zonas actualizada.

Flujo de Eventos	
Actor	**Sistema**
1: El administrador selecciona cual operación sobre las zonas que se encuentran en la interfaz principal desea realizar: A: **Adicionar una zona.** B: **Mostrar una zona.** C: **Eliminar una zona.** D: **Editar una zona.**	2: El sistema, en dependencia de la operación seleccionada por el administrador, muestra la interfaz correspondiente: A: Adicionar una zona: ir a la sección "**Adicionar una zona**" B: Mostrar una zona: ir a la sección "**Mostrar una zona**" C: Eliminar una zona: ir a la sección "**Eliminar una zona**" D: Editar una zona: ir a la sección "**Editar una zona**"
Sección A: "Adicionar una zona"	
Flujo Básico: Adicionar los datos satisfactoriamente.	
Actor	**Sistema**
1: El administrador indica al sistema adicionar una zona.	2: El sistema muestra la interfaz para adicionar una zona.
3: El administrador introduce los datos de la zona que desea adicionar: número, descripción, central, listado de cámaras asociadas y listado de plugins de acción asociados.	
4: El administrador indica al sistema aceptar los datos introducidos.	5: El sistema verifica que ningún campo obligatorio haya quedado vacío.
	6: El sistema genera de forma automática un identificador para la zona: que será incremental respecto a la última zona adicionada.

	7: El sistema adiciona los datos de la nueva zona y finaliza el CU.
Flujo Alterno: Cancelar los datos introducidos.	
Actor	**Sistema**
4.1: El administrador indica al sistema cancelar los datos introducidos y finaliza el CU.	
Flujo Alterno: Existen campos obligatorios vacíos.	
Actor	**Sistema**
	5.1: El sistema emite un mensaje “Existen campos obligatorios vacíos”.
Sección B: “Mostrar una zona”	
Flujo Básico: Mostrar los datos satisfactoriamente.	
Actor	**Sistema**
1: El administrador debe seleccionar una zona de la lista de zonas que se muestra en la interfaz principal.	
2: El administrador indica al sistema mostrar la zona.	3: El sistema verifica que se haya seleccionado la zona.
	4: El sistema muestra una interfaz con los datos de la zona y finaliza el CU.
Flujo Alterno: No se ha seleccionado la zona.	
Actor	**Sistema**
	3.1: El sistema emite un mensaje “Debe seleccionar una zona de la lista de zonas”.
Sección C: “Eliminar una zona”	
Flujo Básico: Eliminar los datos satisfactoriamente.	
Actor	**Sistema**
1: El administrador debe seleccionar una zona de la lista de zonas que se muestra en la interfaz principal.	
2: El administrador indica al sistema eliminar la zona.	3: El sistema verifica que se haya seleccionado la zona.
	4: El sistema emite un mensaje de confirmación “Si

	elimina la zona, elimina los dispositivos asociados a esta".
5: El administrador indica al sistema la confirmación que si desea eliminar la zona.	
	6: El sistema elimina los datos de la zona y finaliza el CU.
Flujo Alterno: No se ha seleccionado la zona.	
Actor	**Sistema**
	3.1: El sistema emite un mensaje "Debe seleccionar una zona de la lista de zonas".
Flujo Alterno: No eliminar la zona.	
Actor	**Sistema**
5.1: El administrador indica al sistema la confirmación que no desea eliminar la zona.	
Sección D: "Editar una zona"	
Flujo Básico: Editar los datos satisfactoriamente.	
Actor	**Sistema**
1: El administrador debe seleccionar una zona de la lista de zonas que se muestra en la interfaz principal.	
2: El administrador indica al sistema editar la zona.	3: El sistema verifica que se haya seleccionado la zona.
	4: El sistema muestra la interfaz para editar la zona.
5: El administrador modifica los datos de la zona que desea editar: número, descripción, central, listado de cámaras asociadas o listado de plugins de acción asociados.	
6: El administrador indica al sistema aceptar los datos modificados.	7: El sistema verifica que ningún campo obligatorio haya quedado vacío.
	8: El sistema actualiza los datos de la zona y finaliza el CU.
Flujo Alterno: No se ha seleccionado la zona.	
Actor	**Sistema**

		3.1: El sistema emite un mensaje “Debe seleccionar una zona de la lista de zonas”.
Flujo Alterno: Cancelar los datos modificados.		
Actor		**Sistema**
6.1: El administrador indica al sistema cancelar los datos modificados y finaliza el CU.		
Flujo Alterno: Existen campos obligatorios vacíos.		
Actor		**Sistema**
		7.1: El sistema emite un mensaje “Existen campos obligatorios vacíos”.
Relaciones	**CU Incluidos**	No Procede.
	CU Extendidos	No Procede.
Requisitos no funcionales	RnF1, RnF5.	
Asuntos pendientes	No Procede.	

2.5 Definición de la Arquitectura de Software.

La arquitectura de software es una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema. La vista arquitectónica es una vista abstracta, aportando el más alto nivel de comprensión y la supresión o diferimiento del detalle inherente a la mayor parte de las abstracciones. Es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos, el ambiente y los principios que orientan su diseño y evolución (Reynoso, 2005).

2.5.1 Patrones Arquitectónicos.

Los patrones arquitectónicos representan la descripción de un problema particular y recurrente de diseño, que aparece en contextos de diseño específico, y presenta un esquema genérico demostrado con éxito para su solución. El esquema de solución se especifica mediante la descripción de los componentes que la constituyen, sus responsabilidades y desarrollos, así como también la forma como estos colaboran entre sí. (Reynoso, 2005).

2.5.2 Arquitectura de Pizarra.

Esta arquitectura pertenece a los estilos centrados en datos, donde son sistemas en los cuales cierto número de clientes accede y actualiza datos compartidos de un repositorio de manera frecuente. En esta arquitectura hay dos componentes principales, una estructura de datos que representa el estado actual y una colección de componentes independientes que operan sobre él. El sistema Xilema Suria está diseñado siguiendo una arquitectura de Pizarra. Esta consta de varios elementos funcionales conocidos como agentes y un instrumento de control conocido como pizarra. Los agentes normalmente son programas especializados en una tarea concreta, su comportamiento básico es examinar la pizarra, realizar su tarea y escribir sus conclusiones en la pizarra. Así, otro agente puede trabajar sobre los resultados arrojados por el primero. La pizarra tiene un doble papel, por una parte, coordina a los distintos agentes y por otra, facilita su intercomunicación. (Peraza, 2013).

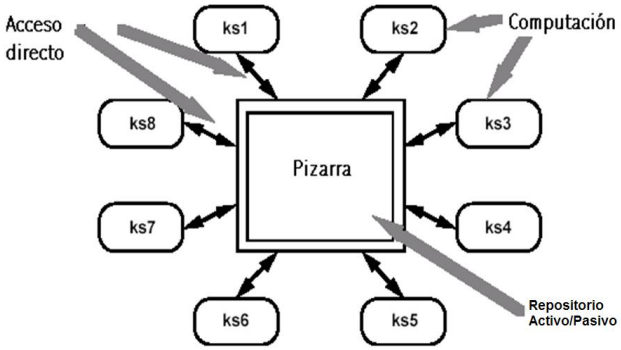

Fig. #3. Arquitectura de Pizarra del sistema Xilema Suria.

En el sistema Xilema Suria, la pizarra es el módulo Gestor, el cual es la parte fundamental de la aplicación, mientras que los restantes módulos como el Visor, Grabador, Recuperador y Video Sensores funcionan como agentes, encargándose de realizar tareas específicas. En tal caso también se encontrará como agente el módulo de integración con CRA, el cual es la solución propuesta en la investigación.

2.5.3 Arquitectura de Plugins.

Esta arquitectura pertenece a los estilos basados en componentes, el cual se enfoca en la descomposición del diseño en componentes funcionales o lógicos que expongan interfaces de comunicación bien definidas. Un componente es un objeto de software específicamente diseñado para cumplir con cierto propósito. En esta arquitectura un plugin es una aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy específica. Esta aplicación adicional es ejecutada

por la aplicación principal e interactúa por medio de la API. Dicha arquitectura se ve de forma independiente, se enfoca en explicar cómo se puede diseñar una aplicación con el fin de soportar varios plugin permitiendo que la misma se extienda en tiempo de ejecución mediante la carga dinámica de módulos o clases que no conoce durante la compilación. (Mayer, 2006).

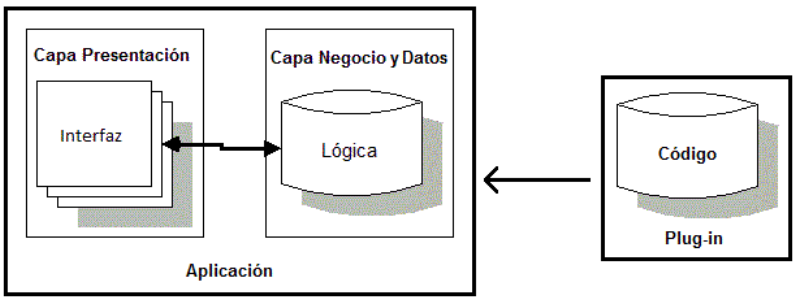

Fig. #4. Arquitectura de Plugin del Módulo CRA.

En el desarrollo del módulo de integración con las CRA para el sistema Xilema Suria, se aplica la arquitectura de plugins, debido a que las principales funcionalidades con las que debe contar el sistema, se basan en permitirle insertar, eliminar o ejecutar cada uno de los archivos que contienen el código fuente para realizar acciones vitales en el funcionamiento de la solución. No es más que administrar el paquete de plugin que contendrá todos los archivos (.so) para Linux, que tendrán también código fuente pero que será cargado por la aplicación en caso que sea necesario o si el propio administrador lo solicita.

2.5.4 Arquitectura en Capas.

Esta arquitectura pertenece a los estilos de llamada y retorno donde, los datos son pasados como parámetros y el manejador principal proporciona un ciclo de control sobre las subrutinas. Reflejan la estructura del lenguaje de programación y permite al diseñador del software construir una estructura de programa relativamente fácil de modificar y ajustar. En esta arquitectura cada capa proporciona servicios a la capa superior y se sirve de las prestaciones que le brinda la inferior. Al dividir un sistema en N-capas, cada capa puede tratarse de forma independiente, sin tener que conocer los detalles de las demás. La división de un sistema en capas facilita el diseño modular, permite la construcción de sistemas débilmente acoplados, lo que significa que si se minimiza las dependencias entre capas, resulta más fácil sustituir la implementación de una capa sin afectar al resto del sistema (Reynoso, 2005).

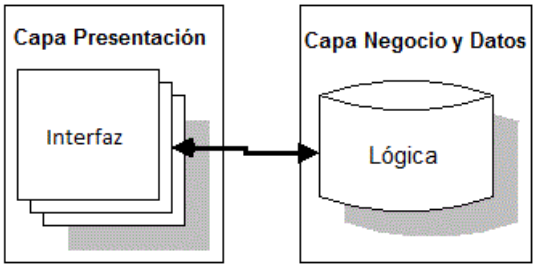

Fig. #5. Arquitectura de 2 Capas del Módulo CRA.

Para la implementación del módulo de integración con CRA para el sistema Xilema Suria se emplea una arquitectura de 2 capas, variante donde se divide la presentación del negocio y datos. En la capa presentación se tratan los aspectos gráficos, la parte visual del sistema, con el objetivo de realizar la interacción con el administrador. El uso de esta capa permite la captura de datos ingresados por el usuario y visualiza la información solicitada. Mientras que en la capa negocio y datos, estarán definidas e implementadas las operaciones que debe realizar el sistema, así como el acceso a la base de datos. El uso de esta capa permite mantener una comunicación con la capa presentación.

2.6 Patrones de Diseño de Software.

Los patrones de diseño ayudan a estructurar las clases y los objetos para guiar todos los procesos de creación de software. Componen soluciones concretas a problemas que se presentan durante el diseño de una aplicación. Entre los patrones de diseño más utilizados se encuentran los patrones GRASP y los patrones GoF. (Ferro, 2013).

2.6.1 Patrones GRASP empleados.

Los patrones GRASP describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable. (Ferro, 2013).

1. **Experto:** Este patrón garantizará que cada clase cumpla con sus responsabilidades, según la información que contenga y las funcionalidades que se deseen implementar. (Bertran, 2013). En la solución propuesta, dicho patrón se evidencia en las clases *Zona*, *Dispositivo*, *Alarma* las cuales contienen la información necesaria para cumplir sus responsabilidades.

2. **Controlador:** Permite asignar la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. Se hace ventajoso llevar el control de la aplicación en una sola clase, es un objeto en particular, en una aplicación, que lleva el manejo de la misma. (Bertran, 2013). En la solución propuesta, dicho patrón se evidencia en la clase *Central,* la cual se encarga de recibir la información adicionada desde las interfaces y la envía a las distintas clases según sea necesario.
3. **Creador:** Plantea la necesidad de asignarle a una clase la responsabilidad de crear una instancia de otra clase siempre y cuando agregue los objetos de la clase, los contenga, registre las instancias de estos objetos y los utilice específicamente. (Bertran, 2013). En la solución propuesta, dicho patrón se evidencia en la clase *Central,* la cual es responsable de la construcción de objetos o sea, es capas de asignarle datos a los constructores.
4. **Bajo Acoplamiento:** El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases. Permite asignar las responsabilidades de forma tal que las clases se comuniquen con el menor número de clases que sea posible. (Bertran, 2013). En la solución propuesta, dicho patrón se evidencia en las clases *Zona*, *Dispositivo*, *Alarma, Conexión*, *Accion_Interface y Central_Interface* asegurando como máximo que dependan de una solamente, por lo cual de producirse una modificación, tenga la mínima repercusión posible en el resto, así fortalece la reutilización y disminuye la dependencia.
5. **Alta Cohesión:** La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. (Bertran, 2013). En la solución propuesta, dicho patrón se evidencia en todas las clases, asegurando que solo cuenten con las responsabilidades que deban asumir y la información contenida están estrechamente relacionadas de manera coherente, por lo cual esto facilita que sea flexible a futuros cambios.

2.6.2 Patrones GoF empleados.

Los patrones GoF son soluciones técnicas basadas en POO. Se clasifican en patrones estructurales, creacionales y de comportamiento. (Ferro, 2013).

1. **Fachada:** Patrón estructural que provee de una interfaz unificada y simple para acceder a una interfaz o grupo de interfaces de un subsistema. (Larman, 2013). En la solución propuesta, dicho patrón se evidencia en la clases *Accion_Interface y Central_Interface,* las cuales proporcionan una interfaz unificada que representan un subsistema y permite acceder a funcionalidades de un conjunto de clases.

2.7 Modelo del Diseño.

El modelo de diseño es un modelo físico y concreto, da forma al sistema y debe ser mantenido durante todo el desarrollo del ciclo de vida del software. Se centra en la realización de los CU a través de los requisitos. Representa el centro de atención al final de la fase de elaboración y es el comienzo de las iteraciones de construcción, este representa un plano del modelo de implementación garantizando de esta manera una arquitectura sólida y estable. El objetivo final del flujo de trabajo de diseño es producir un modelo lógico del sistema a implementar. (Jacobson, 2000).

2.7.1 Diagrama de Clases del Diseño.

El Diagrama de Clases del Diseño (DSD) describe gráficamente las especificaciones y las relaciones entre las clases del diseño. Se utiliza para modelar una vista de diseño estática de un sistema, modelar el vocabulario del sistema, modelar las colaboraciones o modelar esquemas. Por lo cual son importantes no sólo para visualizar, especificar y documentar modelos estructurales, sino también para construir sistemas ejecutables, aplicando ingeniería directa e inversa. (Jacobson, 2000). A continuación se muestra el DSD correspondiente a la solución propuesta, donde se representan las clases utilizadas para el desarrollo del mismo teniendo en cuenta la estructura del patrón arquitectónico en 2 capas.

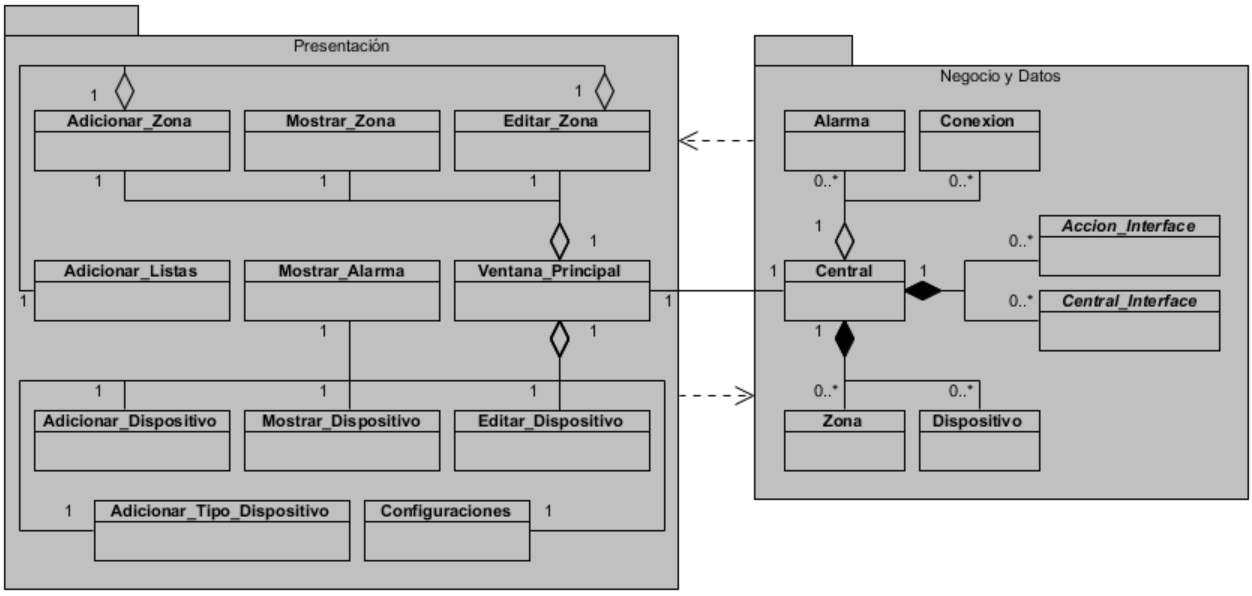

Fig. #6. Diagrama de Clases del Diseño Reducido.

2.7.2 Descripción del Diagrama de Clases del Diseño.

En el diagrama de la Fig. #6 se aprecia que, las clases son agrupadas por paquetes, respetando el patrón arquitectónico bajo el cual se desarrolla el sistema. En la capa de Presentación se encuentran las clases *Ventana_Principal, Adicionar_Zona, Mostrar_Zona, Editar_Zona, Adicionar_Dispositivo, Mostrar_ Dispositivo, Editar_ Dispositivo, Adicionar_Tipo_Dispositivo, Adicionar_Listas, Mostrar_Alarmas* y *Configuraciones,* que en conjunto conforman la interfaz del sistema. Esta capa interactúa con el administrador durante los procesos de gestión y administración de la CRA. En la capa de Negocio y Datos se encuentran las clases que se encargan de procesar la información brindada por las interfaces, como también las funcionalidades más importantes del sistema y la encargada del almacenamiento de la información. Un ejemplo es la clase *Central* la cual se encarga de ser intermediaria entre las interfaces y las entidades controlando todo el flujo de información que manipula el sistema. Las clases *Zona, Dispositivo, Alarma, Accion_Interface y Central_Interface* implementan las funcionalidades que las identifican a cada una de ellas mediante la obtención de los datos brindados por las interfaces. La clase *Conexion* es la encargada de implementar las funcionalidades para el acceso a la base de datos, así como las consultas a realizar. Este DCD solo contiene las clases y sus relaciones, con más detalle puede encontrarse en los anexos del presente trabajo. **Ver Anexo 4**.

2.7.3 Diagrama de Secuencia del Diseño.

El Diagrama de Secuencia (DS) es una variante del diagrama de interacción, que muestra los objetos como líneas de vida y sus interacciones en el tiempo, representadas como mensajes dibujados con flechas desde la línea de origen, hasta la línea de destino. Los elementos que se deben encontrar en este diagrama son las instancias de los actores, los objetos del diseño y las transmisiones de mensaje que representen el flujo de eventos para cada caso de uso. (Jacobson, 2000). A continuación se muestra el DS correspondiente al CU Gestionar Zona de la solución propuesta, donde se representan los flujos de los procesos útiles para definir acciones que se puedan realizar en el sistema durante la implementación de cada CU. Los restantes DS de los CU Gestionar Dispositivos, Administrar Plugins, Administrar Alarmas, Ejecutar Plugins y Generar Alarmas pueden encontrarse en los anexos del presente trabajo. **Ver Anexo 5**.

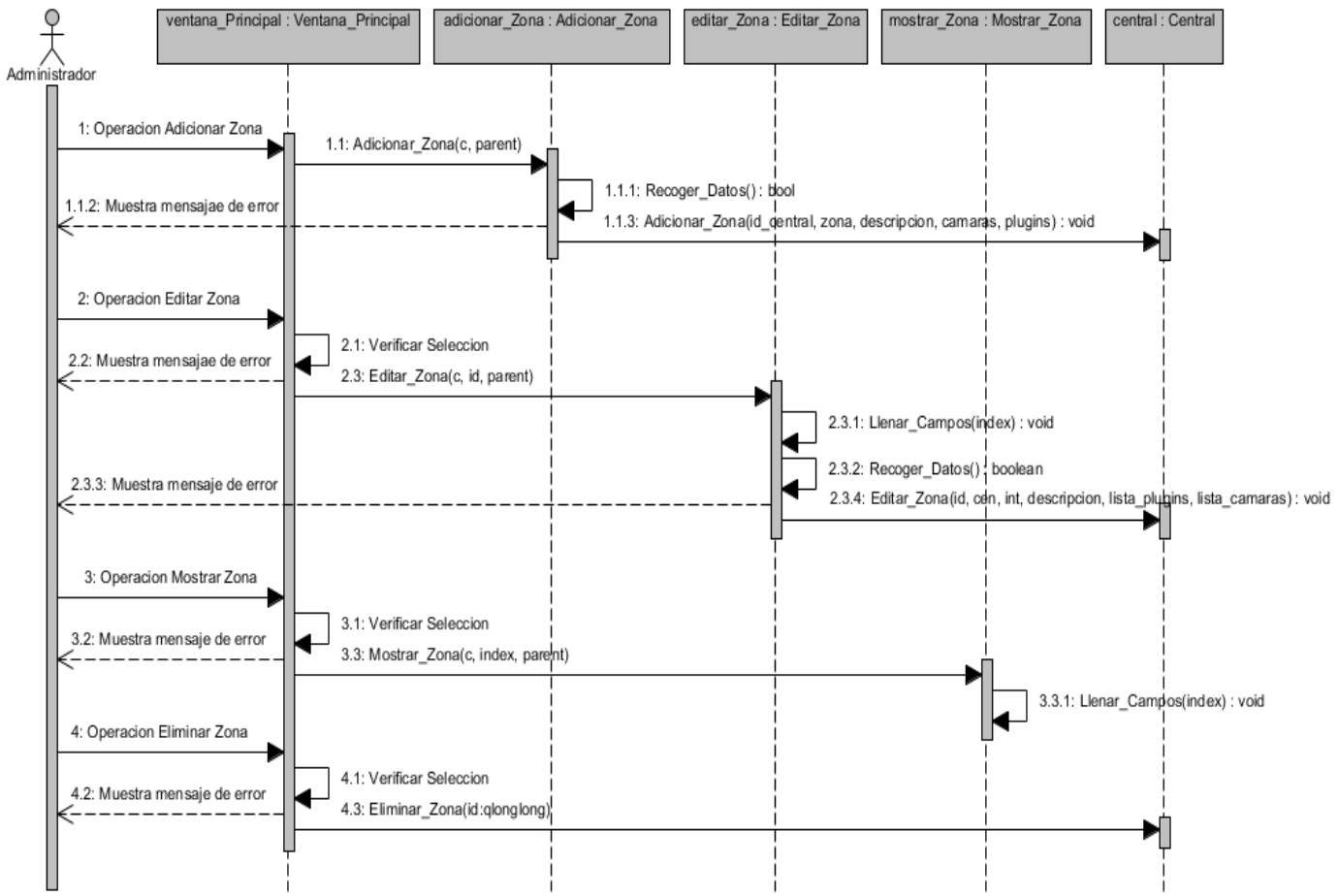

Fig. #7. Diagrama de Secuencia CU Gestionar Zonas.

2.8 Conclusiones Parciales.

En este capítulo se definieron los temas relacionados con el análisis y el diseño de la solución propuesta, del cual se obtuvo una serie de artefactos que fueron analizados detalladamente para facilitar la comprensión del desarrollo del sistema.

1. Se realizó una modelación de la situación problemática, obteniéndose un modelo del dominio el cual esclarece cómo funciona el proceso que se desea automatizar y se describe brevemente para facilitar una mejor comprensión en su desarrollo.
2. Se especificaron los requisitos que tendrá la solución, obteniéndose un total de 18 requisitos funcionales y 9 requisitos no funcionales, que establecen los límites y restricciones correspondientes para que la solución funcione correctamente.

3. Se identificaron los actores que interactúan con cada uno de los casos de usos que se definieron, obteniéndose el diagrama de casos de uso del sistema, lo cual facilitó la realización de las especificaciones, permitiendo a los desarrolladores conocer de forma detallada el flujo de cada una de las funcionalidades.
4. Se estableció la línea base de la arquitectura que tendrá la solución propuesta, empleándose el patrón arquitectónico en Capas específicamente en su variante de 2 capas, el cual definió una estructura lógica para el sistema, asegurando una mejor organización e interrelación entre sus componentes.
5. Se identificaron los patrones de diseño GRASP y GoF utilizados en el desarrollo de la aplicación, permitiendo facilitar la asignación de responsabilidades, logrando un diseño de software que sirva de apoyo a la implementación de la solución.
6. Se realizó el diagrama de clases de diseño, obteniendo con esto una visión estática del sistema y los diagramas de secuencias del diseño, mostrándose en cada uno de ellos las relaciones entre las clases por medio de funciones, ambos facilitaron un mejor entendimiento de cómo funciona la solución propuesta.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA DEL MÓDULO

3.1 Introducción.

En el presente capítulo se muestran los resultados de la implementación de los diferentes componentes del sistema, en términos de artefactos como el diagrama de despliegue y el diagrama de componentes, pertenecientes al flujo de trabajo en cuestión. Posteriormente se realizan pruebas al sistema para lograr erradicar errores que puedan ser introducidos en la implementación de la solución, y a la vez, para comprobar que el producto final cumple con los requisitos establecidos en la primera fase de la investigación. En el contenido del capítulo se exponen diferentes elementos asociados a la construcción y a la validación del sistema que permiten reflejar con claridad las actividades realizadas durante esta etapa.

3.2 Modelo de Implementación.

El modelo de implementación describe cómo los elementos del modelo de diseño se implementan en términos de componentes sin descartar la dependencia que se manifiesta entre éstos. Por lo cual, propone cómo se deben organizar los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación, sin dejar de tener en cuenta el lenguaje de programación utilizado. Tiene como objetivos implementar las clases de diseño como componentes, asignar los componentes a los nodos, probar los componentes individualmente e integrar los componentes en un sistema ejecutable (Jacobson, 2000).

3.3 Diagrama de Componentes.

El diagrama de componentes permite describir los elementos físicos del sistema y sus relaciones. Los componentes representan todos los tipos de elementos de software que entran en la fabricación de aplicaciones informáticas, los cuales pueden ser simples archivos, paquetes o bibliotecas cargadas dinámicamente. Debido a que la complejidad de una aplicación se suele dividir por partes, representándose la estructura a alto nivel de los diferentes subsistemas y sus relaciones que intervienen en la solución de una funcionalidad. (Jacobson, 2000). A continuación se muestra el diagrama de componentes correspondiente a la solución propuesta, donde se encuentran los paquetes de componentes y las relaciones entre ellos.

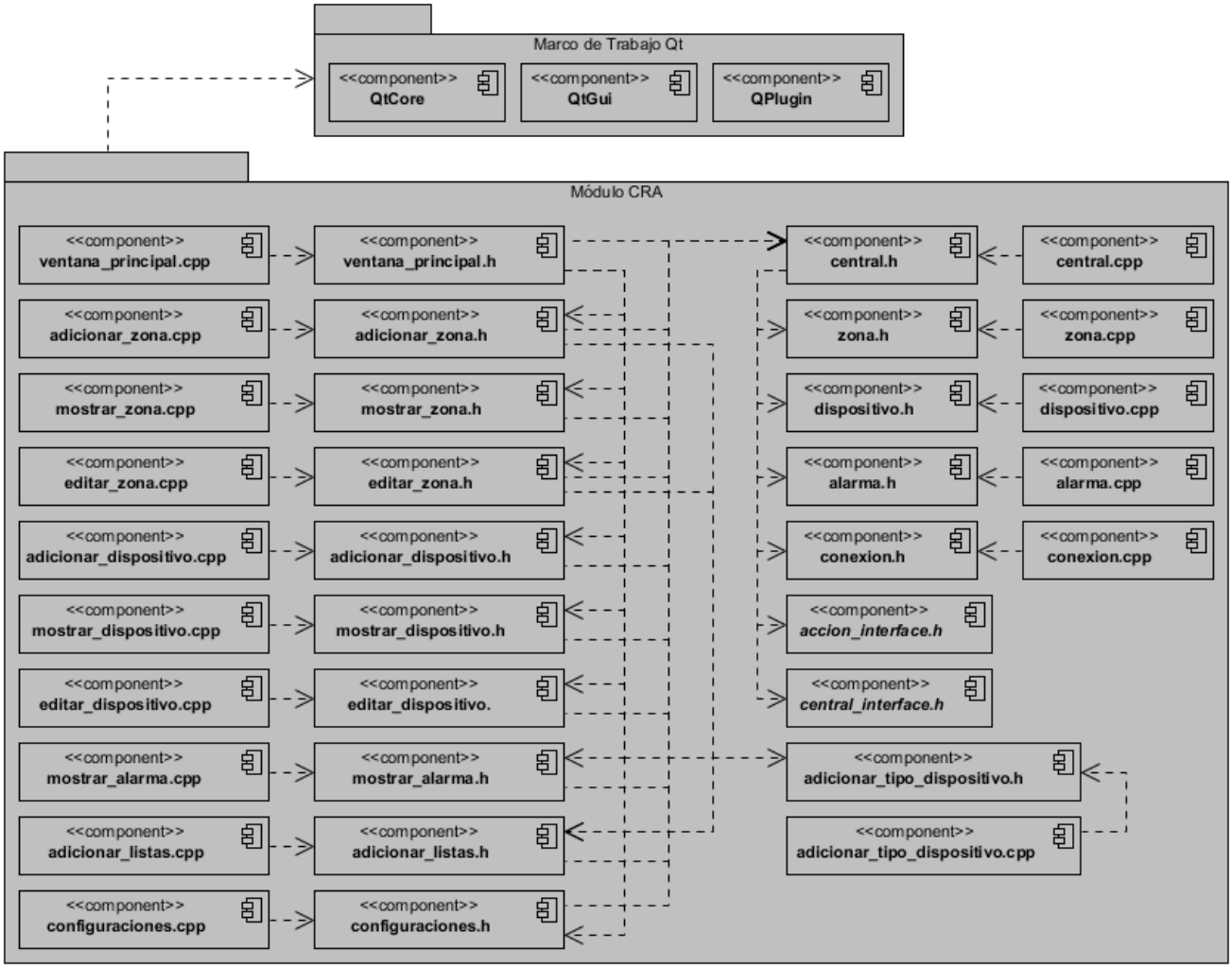

Fig. #8. Diagrama de Componentes.

3.3.1 Descripción del Diagrama de Componentes.

En el diagrama de la Fig. #8 se representan los elementos físicos y sus relaciones por los que están compuesto la solución propuesta. A continuación se explican los paquetes representados y los componentes que contiene la representación:

1. **Marco de Trabajo Qt:** En este paquete se encuentran los componentes, *QtCore* que es el núcleo de todas las funcionalidades como métodos que incluye el marco de trabajo, *QtGui* que es el encargado de todos los eventos relacionados con formularios e interfaces para representar o visualizar datos y *QPlugin* que es utilizado para la gestión de los plugins de acción a realizar o de modelo de central que se adicionaran o eliminaran.

2. **Módulo CRA:** En este paquete se encuentran los principales componentes físicos y las dependencias que existen entre estos, los cuales conforman el sistema que pertenece a la solución propuesta. Tiene como objetivo principal brindar una descripción de las interacciones existentes entre los principales componentes que integran la solución, así como sus dependencias más significativas.

3.4 Diagrama de Despliegue.

El diagrama de despliegue es un tipo de diagrama UML que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo. 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. Es útil para modelar el hardware empleado en las implementaciones de sistemas y las relaciones entre sus componentes, lo cual representa la capa estructural que define el comportamiento de los diversos dispositivos de hardware y lenguaje de programación. (Jacobson, 2000). A continuación se muestra el diagrama de despliegue correspondiente a la solución propuesta, donde se muestra la disposición física de los diferentes elementos, también conocidos como nodos, que intervienen en el proceso y las relaciones existentes entre ellos.

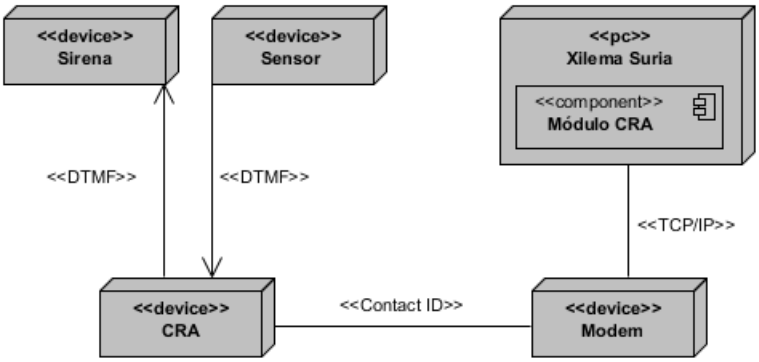

Fig. #9. Diagrama de Despliegue.

3.4.1 Descripción del Diagrama de Despliegue.

En el diagrama de la Fig. #9 se aprecia que, para la solución propuesta se cuenta básicamente con 5 nodos, los cuales se describirán a continuación:

1. **Sensor:** dispositivo encargado de monitorear un área determinada, puede ser de movimiento, de apertura, de temperatura, de presión, entre otros.
2. **Sirena:** dispositivo encargado de ofrecer una señal de audio ante la activación de un sensor.

3. **CRA:** dispositivo encargado de recibir el pulso eléctrico emitido por la activación de un sensor.
4. **Modem:** dispositivo encargado de interpretar las señal analógica emitida por la CRA y convertirla en señal digital para que se entienda en el leguaje de computadoras.
5. **Xilema Suria:** software de video vigilancia donde estará integrado el módulo CRA.

Cuando un *Sensor* se activa, emite un pulso eléctrico a la *CRA* en formato DTMF, el cual es un sistema de marcación por tono a través de una línea telefónica, la *CRA* al recibir la señal emite un pulso eléctrico a una *Sirena* con el mismo sistema, para que ofrezca una señal sonora de la activación del *Sensor,* la *CRA* a la vez también codifica el DTMF con el protocolo ContactID, para permitir la comunicación con un *Modem* a través de una PSTN, que básicamente es una red telefónica y pueda interpretar la información enviada. El *Modem* al recibir la información en DTFM convierte el codificado en Ethernet TCP/IP con el protocolo de telefonía SIP, esto le permitirá conectarse a una computadora y enviarle la información para que el sistema *Xilema Suria,* el cual tendrá integrado el *Módulo CRA,* se encargue de gestionar todo el proceso por el cual pasa el sistema de seguridad y protección, al finalizar el *Módulo CRA* envía una orden al sistema *Xilema Suria* de realizar las acciones que tiene definida ante la activación de un *Sensor*.

3.5 Pruebas de Software.

Las pruebas de software se realizan con la finalidad de descubrir errores, tanto en la lógica interna como en funciones externas del mismo. Estas son un elemento crítico para garantizar la calidad del software, aunque no la aseguran, si detectan un grupo de debilidades que la afectan directamente. Por lo cual caracterizan el control que se enmarca en comprobar el correcto funcionamiento y las posibles respuestas que el producto despliega ante posibles situaciones. Son básicamente un conjunto de actividades dentro del desarrollo de software, dependiendo del tipo, estas actividades podrán ser implementadas en cualquier momento de dicho proceso de desarrollo. (Pressman, 2001).

3.5.1 Estrategia de Pruebas.

Una estrategia para pruebas de software integra las técnicas de diseño de casos de prueba en una serie de pasos bien planificados que llevan a la evaluación correcta del software. Es una parte fundamental del proceso de validación y verificación del software. La verificación es una actividad la cual asegura que las distintas partes del software cumple con la función para la cual fueron diseñadas, o sea se encarga de revisar el funcionamiento de los módulos del software, mientras que la validación se encarga de comprobar que los módulos verificados cumplen con los requisitos definidos. (Universidad, 2014).

La estrategia de pruebas proporciona la descripción de los pasos que hay que llevar a cabo como parte de la prueba, cuándo se deben planificar y realizar esos pasos, y cuánto esfuerzo, tiempo y recursos se van a requerir. Una estrategia de pruebas contiene:

1. Planificación de la prueba.
2. Diseño de casos de prueba.
3. Ejecución de las pruebas.
4. Agrupación y evaluación de datos

3.5.2 Niveles de Pruebas.

Según la metodología de desarrollo de software RUP utilizada en la solución propuesta, define la fase de pruebas como, una actividad en la cual un sistema o componente es ejecutado bajo unas condiciones o requerimientos específicos, los resultados son observados, registrados, y evaluados. (Pressman, 2001). RUP propone una serie de niveles de pruebas, los cuales nos muestran diferentes ángulos para verificar y validar un producto de software, tales como, Nivel de Unidad, Nivel de Integración, Nivel de Sistema y Nivel de Aceptación.

Para verificar y validar la solución propuesta se realizaran primeramente pruebas a Nivel de Unidad, ya que este tipo de pruebas son ejecutadas normalmente por el equipo de desarrollo, básicamente consisten en la ejecución de actividades que le permitan verificar al desarrollador que los componentes unitarios están codificados, soportando el ingreso de datos erróneos o inesperados y demostrando así la capacidad de tratar errores de manera controlada. Posteriormente se aplicarán pruebas a Nivel de Integración, debido a que este tipo de pruebas también son ejecutadas por el equipo de desarrollo y consisten en la comprobación de los elementos del software que interactúan entre sí, funcionen de manera correcta, o sea, validar las conexiones e integración entre dos o más componentes de software.

3.5.3 Pruebas Unitarias: Método de Pruebas.

El método de pruebas a realizar para las pruebas unitarias es el de Caja Negra, el cual se centra en los requisitos funcionales del software y permiten probar con un conjunto de condiciones de entrada o valores de prueba. Las pruebas de caja negra pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto. Para preparar los casos de pruebas hacen falta un número de datos que ayuden a la ejecución de las funcionalidades y que permitan que el sistema se ejecute en todas sus variantes.

3.5.4 Pruebas Unitarias: Técnica de Pruebas.

Para diseñar los casos de prueba se hará uso de la técnica de Partición Equivalente, que divide los campos de entrada de un programa en variables de equivalencia con juegos de datos de entrada y salida. Las variables de equivalencia representan un conjunto de estados válidos y no válidos para las condiciones de entrada de un programa. Se definen dos tipos de variables de equivalencia, las válidas, que representan entradas válidas al programa, y las no válidas, que representan valores de entrada erróneos, aunque pueden existir valores no relevantes a los que no sea necesario proporcionar un valor real de dato.

3.5.5 Pruebas Unitarias: Casos de Pruebas.

Caso de Prueba #1 "CU Gestionar Zonas".

Tabla. #4. Sección 1: Adicionar una Zona.

Nombre de Sección			**Descripción de Sección**	
Sección 1: Adicionar una zona.			El administrador selecciona la opción "Adicionar" e introduce los datos de la zona que desea adicionar.	
Sección 1: Descripción de las Variables que Intervienen.				
No.	**Variable**	**Clasificación**	**Valor Nulo**	**Descripción**
1	Número	Combo Box	No	Alfanumérico de 1 a 7 caracteres.
2	Descripción	Text Box	No	Alfanumérico de 1 a 255 caracteres.
3	Central	Combo Box	No	Alfanumérico de 1 a 255 caracteres.
4	Listado de Cámaras	List Widget Item	SI	Alfanumérico de 1 a 255 caracteres.
5	Listado de Plugins	List Widget Item	SI	Alfanumérico de 1 a 255 caracteres.

Tabla. #5. Sección 1: Escenarios de Prueba.

Escenario	**Descripción**	**1**	**2**	**3**	**4**	**5**	**Respuesta del Sistema**	**Flujo Central**
EC.1. Adicionar los datos de la zona satisfactoriamente.	El administrador indica al sistema adicionar una zona e inserta los datos de la zona correctamente.	C	C	C	C	C	El sistema genera de forma automática un identificador para la zona.	Paso 1: Clic en la opción "Adicionar Zona". Paso 2: Llenar los campos. Paso 3: Clic en el botón "Adicionar".
		C	C	C	V	C		
		C	C	C	C	V		
		C	C	C	V	V	El sistema adiciona los datos de la nueva zona.	
EC.2. Adicionar los	El administrador	V	C	C	C	C	El sistema verifica que	Paso 1: Clic en la

datos de la zona con campos vacíos.	indica al sistema adicionar una zona e inserta los datos de la zona con campos vacíos.	C	V	C	C	C	ningún campo obligatorio haya quedado vacío.	opción “Adicionar Zona”. Paso 2: Dejar los campos vacíos. Paso 3: Clic en el botón “Adicionar”.
		C	C	V	C	C		
		V	V	V	C	C	El sistema emite un mensaje “Existen campos obligatorios vacíos”.	
EC.3. Adicionar los datos de la zona se cancela.	El administrador inserta los datos de la zona correctamente o con campos vacíos e indica al sistema cancelar los datos introducidos.	Para todas las combinaciones de datos.					El sistema cierra la interfaz para adicionar los datos de la zona.	Paso 1: Clic en la opción “Adicionar Zona”. Paso 2: Llenar los campos o dejarlos vacíos. Paso 3: Clic en el botón “Cancelar”.

Tabla. #6. Sección 2: Mostrar una zona.

Nombre de Sección			**Descripción de Sección**	
Sección 2: Mostrar una zona.			El administrador debe seleccionar una zona de la lista de zonas que se muestra en la interfaz principal y selecciona la opción “Mostrar”.	
Sección 2: Descripción de las Variables que Intervienen.				
No.	**Variable**	**Clasificación**	**Valor Nulo**	**Descripción**
1	Zona	List Widget Item	No	Alfanumérico de 1 a 255 caracteres.

Tabla. #7. Sección 2: Escenarios de Prueba.

Escenario	**Descripción**	**1**	**Respuesta del Sistema**	**Flujo Central**
EC.1. Mostrar los datos de la zona satisfactoriamente.	El administrador indica al sistema mostrar la zona.	C	El sistema muestra una interfaz con los datos de la zona.	Paso 1: Clic para seleccionar una zona de la lista de zonas. Paso 2: Clic en la opción “Mostrar Zona”.
EC.2. Mostrar los datos de la zona	El administrador indica al sistema	V	El sistema verifica que se haya seleccionado la zona.	Paso 1: No seleccionar una zona de la lista de zonas.

incorrectamente.	mostrar la zona.		El sistema emite un mensaje “Debe seleccionar una zona de la lista de zonas”.	Paso 2: Clic en la opción “Mostrar Zona”.
EC.3. Mostrar los datos de la zona se cancela.	El administrador indica al sistema mostrar la zona.	C	El sistema muestra una interfaz con los datos de la zona.	Paso 1: Clic para seleccionar una zona de la lista de zonas. Paso 2: Clic en la opción “Mostrar Zona”. Paso 3: Clic en el botón “Aceptar”.

Tabla. #8. Sección 3: Eliminar una zona.

Nombre de Sección			**Descripción de Sección**	
Sección 3: Eliminar una zona.			El administrador debe seleccionar una zona de la lista de zonas que se muestra en la interfaz principal y selecciona la opción “Eliminar”.	
Sección 3: Descripción de las Variables que Intervienen.				
No.	**Variable**	**Clasificación**	**Valor Nulo**	**Descripción**
1	Zona	List Widget Item	No	Alfanumérico de 1 a 255 caracteres.

Tabla. #9. Sección 3: Escenarios de Prueba.

Escenario	**Descripción**	**1**	**Respuesta del Sistema**	**Flujo Central**
EC.1. Eliminar los datos de la zona satisfactoriamente.	El administrador indica al sistema eliminar la zona.	C	El sistema elimina los datos de la zona.	Paso 1: Clic para seleccionar una zona de la lista de zonas. Paso 2: Clic en la opción “Eliminar Zona”.
EC.2. Eliminar los datos de la zona incorrectamente.	El administrador indica al sistema eliminar la zona.	V	El sistema verifica que se haya seleccionado la zona. El sistema emite un mensaje “Debe seleccionar una zona de la lista de zonas”.	Paso 1: No seleccionar una zona de la lista de zonas. Paso 2: Clic en la opción “Eliminar Zona”.

Tabla. #10. Sección 4: Editar una zona.

Nombre de Sección	**Descripción de Sección**
Sección 4: Editar una zona.	El administrador debe seleccionar una zona de la lista de

			zonas que se muestra en la interfaz principal y selecciona la opción “Editar”.	
Sección 4: Descripción de las Variables que Intervienen.				
No.	**Variable**	**Clasificación**	**Valor Nulo**	**Descripción**
1	Número	Combo Box	No	Alfanumérico de 1 a 7 caracteres.
2	Descripción	Text Box	No	Alfanumérico de 1 a 255 caracteres.
3	Central	Combo Box	No	Alfanumérico de 1 a 255 caracteres.
4	Listado de Cámaras	List Widget Item	SI	Alfanumérico de 1 a 255 caracteres.
5	Listado de Plugins	List Widget Item	SI	Alfanumérico de 1 a 255 caracteres.

Tabla. #11. Sección 4: Escenarios de Prueba.

Escenario	**Descripción**	**1**	**2**	**3**	**4**	**5**	**Respuesta del Sistema**	**Flujo Central**
EC.1. Editar los datos de la zona satisfactoriamente.	El administrador indica al sistema editar una zona y actualiza los datos de la zona correctamente.	C	C	C	C	C	El sistema actualiza los datos de la zona.	Paso 1: Clic para seleccionar una zona de la lista de zonas. Paso 2: Clic en la opción “Editar Zona”. Paso 3: Llenar los campos. Paso 4: Clic en el botón “Aceptar”.
		C	C	C	V	C		
		C	C	C	C	V		
		C	C	C	V	V		
EC.2. Editar los datos de la zona incorrectamente.	El administrador indica al sistema editar una zona.	**Zona**					El sistema verifica que se haya seleccionado la zona. El sistema emite un mensaje “Debe seleccionar una zona de la lista de zonas”.	Paso 1: No seleccionar una zona de la lista de zonas. Paso 2: Clic en la opción “Editar Zona”.
		V						
EC.3. Editar los datos de la zona con campos	El administrador indica al sistema editar una zona y	V	C	C	C	C	El sistema verifica que ningún campo obligatorio haya quedado vacío.	Paso 1: Clic para seleccionar una zona de la lista de
		C	V	C	C	C		
		C	C	V	C	C		

vacíos.	actualiza los datos de la zona con campos vacíos.	V	V	V	C	C	El sistema emite un mensaje “Existen campos obligatorios vacíos”.	zonas. Paso 2: Clic en la opción “Editar Zona”. Paso 3: Dejar los campos vacíos. Paso 4: Clic en el botón “Guardar”.
EC.4. Editar los datos de la zona se cancela.	El administrador actualiza los datos de la zona correctamente o con campos vacíos e indica al sistema cancelar los datos editados.	Para todas las combinaciones de datos.					El sistema cierra la interfaz para editar los datos de la zona.	Paso 1: Clic en la opción “Adicionar Zona”. Paso 2: Llenar los campos o dejarlos vacíos. Paso 3: Clic en el botón “Cancelar”.

3.5.6 Pruebas Unitarias: Resultados de los Casos de Pruebas.

Para la ejecución de la técnica Partición Equivalente perteneciente al método de Caja Negra se realizó un diseño de casos de pruebas (CP) para cada CU del sistema. Obteniéndose un total de 7 casos de pruebas, los cuales arrojaron los siguientes resultados:

Para la 1era iteración se obtuvieron 6 no conformidades, las mismas se muestra en la siguiente tabla:

Tabla. #12. Resultados de la 1era Iteración de los Casos de Pruebas.

No.	Descripción	CP	Clasificación
1	Al hacer clic en la opción “Mostrar Zona”, el sistema siempre muestra la interfaz con los datos de la primera zona, sin tener en cuenta la zona seleccionada.	#1	Baja
2	Al hacer clic en la opción “Editar Zona”, el sistema muestra la interfaz para editar los datos de una zona, pero no guarda los cambios realizados.	#1	Baja
3	Al adicionar y editar un dispositivo, el sistema admite campos obligatorios vacíos.	#2	Baja
4	Al hacer clic en la opción “Eliminar Dispositivo”, el sistema elimina todos los dispositivos de la lista de dispositivos.	#2	Baja
5	Al adicionar un plugin en el sistema, no se almacena en la ubicación definida por	#3	Media

	defecto para los plugins de acción.		
6	Al mostrar una alarma, el sistema no muestra los campos hora y fecha en la cual se generó la alarma.	#4	Baja

Para la 2da iteración se obtuvieron 2 no conformidades, las mismas se muestra en la siguiente tabla:

Tabla. #13. Resultados de la 2da Iteración de los Casos de Pruebas.

No.	Descripción	CP	Clasificación
1	El sistema no ejecuta los plugins de acción asociados a una zona, cuando esta se activa.	#5	Alta
2	El sistema genera las alarmas sin que se registren todos sus datos.	#6	Alta

Para la 3era iteración no se obtuvieron no conformidades, quedando verificado y validado el sistema.

3.5.7 Pruebas de Integración. Tipo de Pruebas.

La necesidad de realizar pruebas de integración viene dada por el hecho de que los módulos que forman un programa, suelen fallar cuando trabajan de forma conjunta, aunque previamente se haya demostrado que funcionan correctamente de manera individual. Por lo cual la prueba de integración, es una prueba sistemática para construir la arquitectura del software, mientras que al mismo tiempo, se aplican las pruebas para descubrir errores asociados a la interfaz, asegurándonos que los módulos que están relacionados se ejecuten correctamente.

Para la solución propuesta el tipo de prueba de integración que se realizará será la Incremental Descendente, por ser la recomendada para integrar los módulos en su forma general. En esta prueba se integran los módulos moviéndose hacia abajo por jerarquía de control, comenzando por el módulo principal. Los módulos subordinados al módulo principal se van incorporando en la estructura o sea, se empieza con los módulos de nivel superior, y se verifica que llaman a los de nivel inferior de manera correcta, con los parámetros correctos.

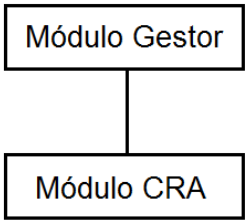

Fig. #10. Representación de la prueba de integración Incremental Descendente.

3.5.8 Pruebas de Integración. Casos de Pruebas.

Caso de Prueba #1 "Obtener Listado de Cámaras".

Tabla. #14. Resultado del Caso de Prueba Obtener Listado de Cámaras.

No.	Módulo Gestor	Módulo CRA
1		Envía una petición para obtener el listado de cámaras que se encuentren registradas en el Módulo Gestor.
2	Recibe la petición solicitada y envía el listado de cámaras al Módulo CRA satisfactoriamente.	

Caso de Prueba #2 "Mandar a Grabar Cámaras".

Tabla. #15. Resultado del Caso de Prueba Mandar a Grabar Cámaras.

No.	Módulo Gestor	Módulo CRA
1		Envía una petición para ordenar a grabar una cámara pasándole como parámetro el identificador de la misma.
2	Recibe la petición solicitada junto con el identificador de la cámara y la ordena a grabar.	

3.6 Conclusiones Parciales.

En este capítulo se definieron los temas relacionados con la implementación y las pruebas realizadas a la solución propuesta, del cual se obtuvo una serie de artefactos que fueron analizados detalladamente para facilitar la comprensión del despliegue y la validación de los requisitos del sistema.

1. Se realizó el diagrama de componentes, el cual permitió describir los elementos físicos del sistema y sus relaciones, garantizando una descripción detallada de su estructura.
2. Se realizó el diagrama de despliegue, el cual proporciona una vista para modelar el hardware empleado en las implementaciones de sistemas y las relaciones entre sus componentes.
3. Se realizó las pruebas de software para verificar y validar el funcionamiento del sistema, aplicándose a nivel unitario, el método de caja negra en su técnica de partición equivalente y a nivel de integración, el tipo incremental descendente. Estas pruebas arrojaron en total 8 no conformidades, las cuales fueron resueltas para alcanzar los resultados esperados en el desarrollo de la solución propuesta.

CONCLUSIONES GENERALES

El cumplimiento de las actividades propuestas durante la investigación del presente trabajo de diploma, hizo posible que los objetivos trazados al inicio se lograran desarrollar con éxito, permitiendo obtener una solución factible a la necesidad presentada. Lo cual conllevó arribar a las siguientes conclusiones:

1. La caracterización y revisión de la fundamentación teórica de los procesos relacionados a las CRA, permitió afirmar que las soluciones que existen hoy, de alguna manera tributan a la solución a desarrollar en la investigación, pero al ser privativas, no resuelven la problemática planteada, por lo cual se expresó la necesidad del desarrollo de la propuesta.
2. Las herramientas y tecnologías utilizadas en el desarrollo de la aplicación son las definidas por el centro GEYSED, favoreciendo con esto la integración entre la solución desarrollada y el sistema Xilema Suria, lo cual garantiza futuras actualizaciones del sistema.
3. La arquitectura de 2 capas utilizada en la implementación de la solución facilitó, la reducción de las dependencias existentes entre las clases, garantizando un mejor soporte al sistema y brindando una mejor organización durante el desarrollo del software.
4. Los artefactos generados a lo largo del proceso de desarrollo de la solución, servirán para que otro grupo de desarrolladores obtengan un mejor entendimiento de la estructura implementada, facilitando la realización ante posibles modificaciones o la adición de nuevas funcionalidades.
5. La utilización de herramientas libres para el desarrollo de la solución contribuyó a ampliar las posibilidades de mercado, además con esta característica propuso alcanzar la soberanía tecnológica que el país viene buscando desde hace varios años.

RECOMENDACIONES

Durante el desarrollo de la solución propuesta surgieron ideas que podrían implementarse en versiones posteriores del sistema, buscando con esto que se logren agregar nuevas funcionalidades o modificar las existentes, para lo cual se recomienda:

1. Desarrollar variantes de plugins de acciones, ya que para validar la solución propuesta se implementó solamente una acción, la cual es Grabar Cámaras.
2. Desarrollar variantes de plugins de modelos de central, ya que para validar la solución propuesta se implementó solamente un modelo de central, el cual es PC1832.

REFERENCIAS BIBLIOGRÁFICAS

MaquinariaPro. 2013. MaquinariaPro. Evolución en los sistemas de seguridad. [En línea] 2013. [Citado el: 10 de septiembre de 2014.] http://www.maquinariapro.com/sistemas/sistema-de-seguridad.html.

Xunta, Edu. 2014. Xunta de Galicia. Sistemas de Seguridad. [En línea] 2014. [Citado el: 10 de septiembre de 2014.] http://www.edu.xunta.es/centros/iescelanova/system/files/SISTEMAS+DE+SEGURIDAD1.doc.

Syscom. 2014. Syscom. Sistemas de Alarma Electrónica. [En línea] 2014. [Citado el: 10 de septiembre de 2014.] http://www.syscom.com.mx/que_es_alarma.htm

Gob, Interior. 2013. Gobierno De España, Ministerio Del Interior. Orden INT/316/2011, de 1 de febrero. [En línea] 2013. [Citado el: 10 de septiembre de 2014.] http://www.interior.gob.es/web/servicios-al-ciudadano/normativa/ordenes-int/orden-int-316-2011-de-1-de-febrero#CAPII

Visonic. 2014. Visonic. Centrales Receptoras de Alarmas. [En línea] 2014. [Citado el: 15 de septiembre de 2014.] http://www.visonic.com/spain/es/central-monitoring-stations.

Gunnebo. 2014. Gunnebo. Central Receptora de Alarma. [En línea] 2014. [Citado el: 15 de septiembre de 2014.] http://www2.gunnebo.com/es/products/cra/Pages/default.aspx.

Visegur. 2014. Visegur. Central Receptora. [En línea] 2014. [Citado el: 15 de septiembre de 2014.] http://www.visegur.com/central-receptora-alarmas-madrid.

Enai. 2014. Enai. Plataforma de software integrado MASterMind. [En línea] 2014. [Citado el: 15 de septiembre de 2014.] http://www.enai.eu/software-cra-2/mastermind.

Rodríguez, Gisela Sú. 2013. Repositorio Institucional. Repositorio Institucional. [En línea] 2013. [Citado el: 10 de octubre de 2014.] http://repositorio_institucional.uci.cu/jspui/handle/ident/TD_05581_12.

Figueroa, Roberth G., Solís, J. y Cabrera, Armando A. Camilo. 2009. Metodologías Tradicionales VS. Metodologías Ágiles. Loja, Ecuador: Universidad Técnica Particular de Loja, 2009.

Scribd. 2013. Scribd. Herramientas Case. [En línea] 2013. [Citado el: 15 de octubre de 2014.] http://www.scribd.com/doc/3062020/CapituloI-HERRAMIENTAS-CASE.

iMatix. 2014. ØMQ. Learn the Basics. [En línea] 2014. [Citado el: 10 de noviembre de 2014.] http://zeromq.org/intro:read-the-manual.

Hamilton, Kim y Russell, Miles. 2006. Learning UML 2.0. s.l.: O'Reilly, ISBN: 0-596-00982-8, 2006.

Digia, Oyj. 2014. QT. QT. [En línea] 2014. [Citado el: 10 de noviembre de 2014.] http://qt.digia.com.

Paradigm, Visual. 2014. Visual Paradigm. Visual Paradigm. [En línea] 2013. [Citado el: 15 de octubre de 2014.] http://www.visual-paradigm.com.

Mayer, Johannes. 2006. Ligthweight Plug-in Based Aplication Development, 2006.

Jacobson, Ivar. 2000. El Proceso Unificado de Desarrollo de Software. Madrid : s.e., 2000

Mastermagazine. 2012. Mastermagazine. Definición de C++. [En línea] 2012. [Citado el: 5 de noviembre de 2014.] http://www.mastermagazine.info/termino/4124.php.

Hernández León, Rolando Alfredo y Sayda Coello González. 2002. El paradigma cuantitativo de la investigación científica. La Habana: Universitaria, 2002. ISBN 959-16-0343-6.

Larman, Craig. 2013. [En línea] 2013. [Citado el: 10 de febrero de 2015.] http://is.ls.fi.upm.es/docencia/is2/documentacion/ModeloDominio.pdf.

Somerville, Ian. 2005. Ingeniería de Software. Madrid: Séptima Edición, 2005.

Pressman, Roger S. 2001. Ingeniería de Software, Un enfoque práctico. Madrid: 5ta edición, 2001.

Reynoso, Carlos. 2005. Introducción a la Arquitectura de Software. Buenos Aires: s.n., 2005.

Peraza, Breissy. 2013. Repositorio Institucional. Repositorio Institucional. [En línea] 2015. [Citado el: 15 de marzo de 2015.] http://repositorio_institucional.uci.cu/jspui/bitstream/ident/8417/1/TD_07057_13.pdf.

Ferro, Doina. 2013. Repositorio Institucional. Repositorio Institucional. [En línea] 2015. [Citado el: 20 de marzo de 2015.] http://repositorio_institucional.uci.cu/jspui/bitstream/ident/8559/1/TD_07085_13.pdf.

Bertran, Javier. 2013. Repositorio Institucional. Repositorio Institucional. [En línea] 2015. [Citado el: 25 de marzo de 2015.] http://repositorio_institucional.uci.cu/jspui/bitstream/ident/8412/1/TD_07052_13.pdf.

BIBLIOGRAFÍA

Figueroa, Roberth G., Solís, J. y Cabrera, Armando A. Camilo. 2009. Metodologías Tradicionales VS. Metodologías Ágiles. Loja, Ecuador: Universidad Técnica Particular de Loja, 2009.

Hernández León, Rolando Alfredo y Sayda Coello González. 2002. El paradigma cuantitativo de la investigación científica. La Habana: Universitaria, 2002. ISBN 959-16-0343-6.

Somerville, Ian. 2005. Ingeniería de Software. Madrid: Séptima Edición, 2005.

Jacobson, Ivar. 2000. El Proceso Unificado de Desarrollo de Software. Madrid : s.e., 2000.

Pressman, Roger S. 2001. Ingeniería de Software, Un enfoque práctico. Madrid: 5ta edición, 2001.

Reynoso, Carlos. 2005. Introducción a la Arquitectura de Software. Buenos Aires: s.n., 2005.

Hamilton, Kim y Russell, Miles. 2006. Learning UML 2.0. s.l.: O'Reilly, ISBN: 0-596-00982-8, 2006.

Mayer, Johannes. 2006. Ligthweight Plug-in Based Aplication Development, 2006.

Digia, Oyj. 2014. QT. QT. [En línea] 2014. [Citado el: 10 de noviembre de 2014.] http://qt.digia.com.

iMatix. 2014. ØMQ. Learn the Basics. [En línea] 2014. [Citado el: 10 de noviembre de 2014.] http://zeromq.org/intro:read-the-manual.

Paradigm, Visual. 2014. Visual Paradigm. Visual Paradigm. [En línea] 2013. [Citado el: 15 de octubre de 2014.] http://www.visual-paradigm.com.

Larman, Craig. 2013. [En línea] 2013. [Citado el: 10 de febrero de 2015.] http://is.ls.fi.upm.es/docencia/is2/documentacion/ModeloDominio.pdf.

Enai. 2014. Enai. Plataforma de software integrado MASterMind. [En línea] 2014. [Citado el: 15 de septiembre de 2014.] http://www.enai.eu/software-cra-2/mastermind.

Scribd. 2013. Scribd. Herramientas Case. [En línea] 2013. [Citado el: 15 de octubre de 2014.] http://www.scribd.com/doc/3062020/CapituloI-HERRAMIENTAS-CASE.

Mastermagazine. 2012. Mastermagazine. Definición de C++. [En línea] 2012. [Citado el: 5 de noviembre de 2014.] http://www.mastermagazine.info/termino/4124.php.

Universidad, Nacional Asistida y a Distancia. 2014. Ingeniería de Software. Capítulo 9, Estrategias de Prueba del Software [En línea] 2014. [Citado el: 20 de abril de 2015.] http://datateca.unad.edu.co/contenidos/301404/301404_ContenidoEnLinea/captulo_9_estrategias_de_prueba_del_software.html.

Rodríguez, Gisela Sú. 2013. Repositorio Institucional. Repositorio Institucional. [En línea] 2013. [Citado el: 10 de octubre de 2014.] http://repositorio_institucional.uci.cu/jspui/handle/ident/TD_05581_12.

Bertran, Javier. 2013. Repositorio Institucional. Repositorio Institucional. [En línea] 2015. [Citado el: 25 de marzo de 2015.] http://repositorio_institucional.uci.cu/jspui/bitstream/ident/8412/1/TD_07052_13.pdf.

Peraza, Breissy. 2013. Repositorio Institucional. Repositorio Institucional. [En línea] 2015. [Citado el: 15 de marzo de 2015.] http://repositorio_institucional.uci.cu/jspui/bitstream/ident/8417/1/TD_07057_13.pdf.

Ferro, Doina. 2013. Repositorio Institucional. Repositorio Institucional. [En línea] 2015. [Citado el: 20 de marzo de 2015.] http://repositorio_institucional.uci.cu/jspui/bitstream/ident/8559/1/TD_07085_13.pdf.

Xunta, Edu. 2014. Xunta de Galicia. Sistemas de Seguridad. [En línea] 2014. [Citado el: 10 de septiembre de 2014.] http://www.edu.xunta.es/centros/iescelanova/system/files/SISTEMAS+DE+SEGURIDAD1.doc.

Syscom. 2014. Syscom. Sistemas de Alarma Electrónica. [En línea] 2014. [Citado el: 10 de septiembre de 2014.] http://www.syscom.com.mx/que_es_alarma.htm

Gob, Interior. 2013. Gobierno De España, Ministerio Del Interior. Orden INT/316/2011, de 1 de febrero. [En línea] 2013. [Citado el: 10 de septiembre de 2014.] http://www.interior.gob.es/web/servicios-al-ciudadano/normativa/ordenes-int/orden-int-316-2011-de-1-de-febrero#CAPII

Visonic. 2014. Visonic. Centrales Receptoras de Alarmas. [En línea] 2014. [Citado el: 15 de septiembre de 2014.] http://www.visonic.com/spain/es/central-monitoring-stations.

Gunnebo. 2014. Gunnebo. Central Receptora de Alarma. [En línea] 2014. [Citado el: 15 de septiembre de 2014.] http://www2.gunnebo.com/es/products/cra/Pages/default.aspx.

Visegur. 2014. Visegur. Central Receptora. [En línea] 2014. [Citado el: 15 de septiembre de 2014.] http://www.visegur.com/central-receptora-alarmas-madrid.

GLOSARIO DE TÉRMINOS

GPRS: General Packet Radio Service o Servicio General de Paquetes vía Radio.

TCP: Transmission Control Protocol o Protocolo de Control de Transmisión.

IP: Internet Protocol o Protocolo de Internet.

EPROM: Erasable Programmable Read-Only Memory o Memoria de Solo Lectura Programable Borrable.

GPS: Global Positioning System o Sistema de Posicionamiento Global.

UML: Unified Modeling Language o Lenguaje Unificado de Modelado.

CASE: Computer Aided Software Engineering o Ingeniería de Software Asistida por Computadora.

C++: Lenguaje de Programación Multiparadigma.

IDE: Integrated Development Environment o Entorno de Desarrollo Integrado.

GUI: graphical user interface o interfaz gráfica de usuario

GPL: General Public License o Licencia Pública General.

Framework: Marco de Trabajo.

Plugin: Add-On o Complemento.

API: Application Programming Interface o Interfaz Programada de la Aplicación.

so: Shared Objects u Objetos Compartidos

GRASP: General Responsibility Assignment Software Patterns o Patrones Generales de Software para Asignar Responsabilidades.

GoF: Gang of Four o Grupo de Cuatro.

POO: Programación Orientada a Objetos.

DTMF: Dual Tone Multi Frequency o Sistema de Marcación por Tonos Multifrecuencial.

ContactID: Protocolo de comunicación para señales digitales de alarmas.

PSTN: Public Switched Telephone Network o Red Telefónica Pública Conmutada.

Ethernet: Estándar de redes de área local para computadores.

SIP: Session Initiation Protocol o Protocolo de Inicio de Sesiones.

ANEXOS

Anexo 1: Entrevista.

Fecha: Diciembre 2014

Entrevistados: Reynier Pupo Gómez (Jefe del proyecto Video Vigilancia), César Santos Sanabria (Arquitecto del proyecto Video Vigilancia), Yuniesky Orlando Vasconcelo Mir (Especialista del proyecto Sistemas de Gestión para CRA).

Preguntas de la entrevista:

¿Considera importante el desarrollo de un módulo de integración con CRA para el sistema Xilema Suria?

¿Qué beneficios traería para el centro GEYSED una solución de este tipo?

¿Qué requisitos funcionales y no funcionales considera importantes a desplegar por la solución?

Resultados generales de la entrevista:

1. Una solución como la del módulo de integración con CRA para el sistema Xilema Suria se consideraría muy importante, debido a QUE permitiría reutilizar el sistema Xilema Suria como el software encargado para monitorizar, controlar y gestionar todo los sistemas de seguridad y protección conectados a la CRA.
2. Algunos de los beneficios que traería para el centro GEYSED una solución de este tipo son:
   No desarrollaría una solución para dedicarlo a los procesos de gestión de la CRA, porque ya se estaría reutilizando los componentes necesarios del sistema Xilema Suria y los pondría a su disposición, por lo cual existirá una disminución de recursos, ya sea tiempo, presupuesto o personal, debido a las facilidades propuestas por la solución de la investigación.
3. Algunos de los requisitos mencionados para el desarrollo de la solución son:
   Insertar, mostrar, eliminar y editar dispositivos, Mostrar y eliminar alarmas, Insertar y eliminar plugin, Ejecutar plugins.
   El sistema debe hacer uso de botones, imágenes o textos que indiquen de modo intuitivo la función que realiza.
   El sistema debe estar accesible las 24 horas del día, garantizando que se puedan monitorizar las alarmas generadas, por la activación de los dispositivos conectados a la CRA.
   El sistema para que funcione, debe emitir una alerta al módulo Gestor, por lo que requiere al menos un dispositivo conectado a la CRA.

Anexo 2: Especificación de los Casos de Uso del Sistema.

Tabla. #16. Descripción del CU Gestionar Dispositivos.

Nombre del CU2	**Gestionar Dispositivos**
Objetivo	Manipular los datos de un dispositivo.
Actores	Administrador: (Inicia) adiciona, muestra, elimina y edita los datos de un dispositivo.
Resumen	El CU inicia cuando el administrador selecciona la operación de adicionar, mostrar, eliminar o editar los datos de un dispositivo. El CU termina cuando los datos del dispositivo se actualizan en el sistema.
Complejidad	Alta
Prioridad	Crítico
Precondiciones	Para Mostrar, Eliminar o Editar: El dispositivo ha sido adicionado. El dispositivo ha sido seleccionado.
Postcondiciones	Mostrar la lista de dispositivos actualizada.
Flujo de Eventos	
Actor	**Sistema**
1: El administrador selecciona cual operación sobre los dispositivos que se encuentran en la interfaz principal desea realizar: A: **Adicionar un dispositivo.** B: **Mostrar un dispositivo.** C: **Eliminar un dispositivo.** D: **Editar un dispositivo.**	2: El sistema, en dependencia de la operación seleccionada por el administrador, muestra la interfaz correspondiente: A: Adicionar un dispositivo: ir a la sección "**Adicionar un dispositivo**" B: Mostrar un dispositivo: ir a la sección "**Mostrar un dispositivo**" C: Eliminar un dispositivo: ir a la sección "**Eliminar un dispositivo**" D: Editar un dispositivo: ir a la sección "**Editar un dispositivo**"
Sección A: "Adicionar un dispositivo"	
Flujo Básico: Adicionar los datos satisfactoriamente.	
Actor	**Sistema**
1: El administrador indica al sistema adicionar un dispositivo.	2: El sistema verifica que exista alguna zona a la cual adicionarle un dispositivo.

	3: El sistema muestra la interfaz para adicionar un dispositivo.
3: El administrador introduce los datos del dispositivo que desea adicionar: nombre, descripción, tipo, zona y central.	
4: El administrador indica al sistema aceptar los datos introducidos.	5: El sistema verifica que ningún campo obligatorio haya quedado vacío.
	6: El sistema genera de forma automática un identificador para el dispositivo: que será incremental respecto al último dispositivo adicionado.
	7: El sistema adiciona los datos del nuevo dispositivo y finaliza el CU.
Flujo Alterno: No existen zonas en el sistema.	
Actor	**Sistema**
	2.1: El sistema emite un mensaje “Debe adicionar una zona para poder adicionar dispositivos”.
Flujo Alterno: Cancelar los datos introducidos.	
Actor	**Sistema**
4.1: El administrador indica al sistema cancelar los datos introducidos y finaliza el CU.	
Flujo Alterno: Existen campos obligatorios vacíos.	
Actor	**Sistema**
	5.1: El sistema emite un mensaje “Existen campos obligatorios vacíos”.
Sección B: “Mostrar un dispositivo”	
Flujo Básico: Mostrar los datos satisfactoriamente.	
Actor	**Sistema**
1: El administrador debe seleccionar un dispositivo de la lista de dispositivos que se muestra en la interfaz principal.	
2: El administrador indica al sistema mostrar el dispositivo.	3: El sistema verifica que se haya seleccionado el dispositivo.
	4: El sistema muestra una interfaz con los datos del

	dispositivo y finaliza el CU.
Flujo Alterno: No se ha seleccionado el dispositivo.	
Actor	**Sistema**
	3.1: El sistema emite un mensaje “Debe seleccionar un dispositivo de la lista de dispositivos”.
Sección C: “Eliminar un dispositivo”	
Flujo Básico: Eliminar los datos satisfactoriamente.	
Actor	**Sistema**
1: El administrador debe seleccionar un dispositivo de la lista de dispositivos que se muestra en la interfaz principal.	
2: El administrador indica al sistema eliminar el dispositivo.	3: El sistema verifica que se haya seleccionado el dispositivo.
	4: El sistema emite un mensaje de confirmación “Esta seguro que desea eliminar el dispositivo”.
5: El administrador indica al sistema la confirmación si eliminar el dispositivo.	
	6: El sistema elimina los datos del dispositivo y finaliza el CU.
Flujo Alterno: No se ha seleccionado el dispositivo.	
Actor	**Sistema**
	3.1: El sistema emite un mensaje “Debe seleccionar un dispositivo de la lista de dispositivos”.
Flujo Alterno: No eliminar el dispositivo.	
Actor	**Sistema**
5.1: El administrador indica al sistema la confirmación no eliminar el dispositivo.	
Sección D: “Editar un dispositivo”	
Flujo Básico: Editar los datos satisfactoriamente.	
Actor	**Sistema**
1: El administrador debe seleccionar un dispositivo de la lista de dispositivos que se muestra en la interfaz principal.	

2: El administrador indica al sistema editar el dispositivo.	3: El sistema verifica que se haya seleccionado el dispositivo.
	4: El sistema muestra la interfaz para editar el dispositivo.
5: El administrador modifica los datos del dispositivo que desea editar: nombre, descripción, tipo, zona o central.	
6: El administrador indica al sistema aceptar los datos modificados.	7: El sistema verifica que ningún campo obligatorio haya quedado vacío.
	8: El sistema actualiza los datos del dispositivo y finaliza el CU.
Flujo Alterno: No se ha seleccionado el dispositivo.	
Actor	**Sistema**
	3.1: El sistema emite un mensaje “Debe seleccionar un dispositivo de la lista de dispositivos”.
Flujo Alterno: Cancelar los datos modificados.	
Actor	**Sistema**
6.1: El administrador indica al sistema cancelar los datos modificados y finaliza el CU.	
Flujo Alterno: Existen campos obligatorios vacíos.	
Actor	**Sistema**
	7.1: El sistema emite un mensaje “Existen campos obligatorios vacíos”.
Relaciones / **CU Incluidos**	No Procede.
CU Extendidos	No Procede.
Requisitos no funcionales	RnF1, RnF5.
Asuntos pendientes	No Procede.

Tabla. #17. Descripción del CU Administrar Plugins.

Nombre del CU3	Administrar Plugins
Objetivo	Manipular los plugins de acciones del sistema o de modelo de central.
Actores	Administrador: (Inicia) adiciona y elimina los plugins.

Resumen	El CU inicia cuando el administrador selecciona la operación de adicionar o eliminar los plugins. El CU termina cuando los plugins de acciones o de modelo de central se actualizan en el sistema.
Complejidad	Media
Prioridad	Crítico
Precondiciones	Para Eliminar: El plugin ha sido adicionado. El plugin ha sido seleccionado.
Postcondiciones	Mostrar la lista de plugins actualizada.

Flujo de Eventos	
Actor	**Sistema**
1: El administrador selecciona cual operación sobre los plugins de acciones que se encuentran en la interfaz principal desea realizar: A: **Adicionar un plugin.** B: **Eliminar un plugin.**	2: El sistema, en dependencia de la operación seleccionada por el administrador, muestra la interfaz correspondiente: A: Adicionar un plugin: ir a la sección "**Adicionar un plugin**" B: Eliminar un plugin: ir a la sección "**Eliminar un plugin**"
Sección A: "Adicionar un plugin"	
Flujo Básico: Adicionar los plugins satisfactoriamente.	
Actor	**Sistema**
1: El administrador indica al sistema adicionar un plugin.	2: El sistema muestra la interfaz para seleccionar la ubicación del plugin.
3: El administrador selecciona el plugin que desea adicionar.	
4: El administrador indica al sistema aceptar el plugin introducido.	5: El sistema verifica que el archivo seleccionado sea del tipo correcto.
	6: El sistema verifica que no exista un plugin con el mismo nombre previamente adicionado.
	7: El sistema adiciona el nuevo plugin y finaliza el CU.
Flujo Alterno: Cancelar la ubicación del plugin introducido.	
Actor	**Sistema**
4.1: El administrador indica al sistema cancelar la	

ubicación del plugin introducido y finaliza el CU.		
Flujo Alterno: Tipo de plugin incorrecto.		
Actor		**Sistema**
		5.1: El sistema emite un mensaje “El archivo seleccionado no es del tipo plugin correcto”.
Flujo Alterno: Existe el plugin.		
Actor		**Sistema**
		6.1: El sistema emite un mensaje “Existe el plugin en el sistema”.
Sección B: “Eliminar un plugin”		
Flujo Básico: Eliminar los plugins satisfactoriamente.		
Actor		**Sistema**
1: El administrador debe seleccionar un plugin de la lista de plugins que se muestra en la interfaz principal.		
2: El administrador indica al sistema eliminar el plugin.		3: El sistema verifica que se haya seleccionado el plugin.
		4: El sistema emite un mensaje de confirmación “Esta seguro que desea eliminar el plugin”.
5: El administrador indica al sistema la confirmación si eliminar el plugin.		
		6: El sistema elimina el plugin y finaliza el CU.
Flujo Alterno: No se ha seleccionado el plugin.		
Actor		**Sistema**
		3.1: El sistema emite un mensaje “Debe seleccionar un plugin de la lista de plugins”.
Flujo Alterno: No eliminar el plugin.		
Actor		**Sistema**
5.1: El administrador indica al sistema la confirmación no eliminar el plugin.		
Relaciones	**CU Incluidos**	No Procede.
	CU Extendidos	No Procede.
Requisitos no funcionales	RnF1, RnF5.	

Asuntos pendientes	No Procede.

Tabla. #18. Descripción del CU Administrar Alarmas.

Nombre del CU4	**Administrar Alarmas**
Objetivo	Manipular los datos de una alarma.
Actores	Administrador: (Inicia) muestra y elimina los datos de una alarma.
Resumen	El CU inicia cuando el administrador selecciona la operación de mostrar o eliminar datos de una alarma. El CU termina cuando los datos de la alarma se actualizan en el sistema.
Complejidad	Media
Prioridad	Crítico
Precondiciones	Para Mostrar o Eliminar: La alarma ha sido generada. La alarma ha sido seleccionada.
Postcondiciones	Mostrar la lista de alarmas actualizada.
Flujo de Eventos	
Actor	**Sistema**
1: El administrador selecciona cual operación sobre las alarmas que se encuentran en la interfaz principal desea realizar: A: **Mostrar una alarma.** B: **Eliminar una alarma.**	2: El sistema, en dependencia de la operación seleccionada por el administrador, muestra la interfaz correspondiente: A: Mostrar una alarma: ir a la sección "**Mostrar una alarma**" B: Eliminar una alarma: ir a la sección "**Eliminar una alarma**"
Sección A: "Mostrar una alarma"	
Flujo Básico: Mostrar los datos satisfactoriamente.	
Actor	**Sistema**
1: El administrador debe seleccionar una alarma de la lista de alarmas que se muestra en la interfaz principal.	
2: El administrador indica al sistema mostrar la alarma.	3: El sistema verifica que se haya seleccionado la alarma.
	4: El sistema muestra una interfaz con los datos de la alarma y finaliza el CU.

Flujo Alterno: No se ha seleccionado la alarma.	
Actor	**Sistema**
	3.1: El sistema emite un mensaje “Debe seleccionar una alarma de la lista de alarmas”.
Sección B: “Eliminar una alarma”	
Flujo Básico: Eliminar los datos satisfactoriamente.	
Actor	**Sistema**
1: El administrador debe seleccionar una alarma de la lista de alarmas que se muestra en la interfaz principal.	
2: El administrador indica al sistema eliminar la alarma.	3: El sistema verifica que se haya seleccionado la alarma.
	4: El sistema emite un mensaje de confirmación “Esta seguro que desea eliminar la alarma”.
5: El administrador indica al sistema la confirmación si eliminar la alarma.	
	4: El sistema elimina los datos de la alarma y finaliza el CU.
Flujo Alterno: No se ha seleccionado la alarma.	
Actor	**Sistema**
	3.1: El sistema emite un mensaje “Debe seleccionar una alarma de la lista de alarmas”.
Flujo Alterno: No eliminar la alarma.	
Actor	**Sistema**
5.1: El administrador indica al sistema la confirmación no eliminar la alarma.	
Relaciones **CU Incluidos**	No Procede.
CU Extendidos	No Procede.
Requisitos no funcionales	RnF1, RnF5.
Asuntos pendientes	No Procede.

Tabla. #19. Descripción del CU Ejecutar Plugins.

Nombre del CU5	Ejecutar Plugins
Objetivo	Realizar el conjunto de acciones asociadas a la activación de un dispositivo.

Actores	Panel de Control: (Inicia) ejecutar plugins	
Resumen	El CU inicia cuando el panel de control emite al sistema una señal de activación de un dispositivo ubicado en una zona. El CU termina cuando el sistema ejecuta los plugins asociados a la zona en cuestión o a la central.	
Complejidad	Baja	
Prioridad	Auxiliar	
Precondiciones	Un dispositivo ha sido activado.	
Postcondiciones	Realiza las acciones asociadas a la zona que contiene el dispositivo activado o a la central.	
Flujo de Eventos		
Flujo Básico: Ejecutar plugins satisfactoriamente.		
Actor		**Sistema**
1: El panel de control emite al sistema una señal de activación de un dispositivo ubicado en una zona.		2: El sistema recibe la señal emitida.
		3: El sistema determina de qué zona proviene la señal emitida y si ha sido anteriormente adicionada.
		4: El sistema ejecuta los plugins asociados a la zona o a la central y finaliza el CU.
Flujo Alterno: La zona no se ha adicionado.		
Actor		**Sistema**
		3.1: El sistema emite un mensaje “Se ha recibido una alarma de una zona no registrada, por favor actualice el sistema”.
Relaciones	**CU Incluidos**	No Procede.
	CU Extendidos	No Procede.
Requisitos no funcionales	RnF2, RnF3.	
Asuntos pendientes	No Procede.	

Tabla. #20. Descripción del CU Generar Alarmas.

Nombre del CU6	**Generar Alarmas**
Objetivo	Registrar los eventos de activación de los dispositivos asociados a las zonas.
Actores	Panel de Control: (Inicia) generar alarmas.

Resumen	El CU inicia cuando el panel de control emite al sistema una señal de activación de un dispositivo ubicado en una zona. El CU termina cuando el sistema genera los datos asociados a la alarma.	
Complejidad	Baja	
Prioridad	Auxiliar	
Precondiciones	Un dispositivo ha sido activado.	
Postcondiciones	Generar una alarma con los datos asociados.	
Flujo de Eventos		
Flujo Básico: Generar alarmas satisfactoriamente.		
Actor		**Sistema**
1: El panel de control emite al sistema una señal de activación de un dispositivo ubicado en una zona.		2: El sistema recibe la señal emitida.
		3: El sistema determina de qué zona proviene la señal emitida y si ha sido anteriormente adicionada, así como la hora y la fecha de la activación del dispositivo.
		4: El sistema genera la alarma con los datos asociados y finaliza el CU.
Flujo Alterno: La zona no se ha adicionado.		
Actor		**Sistema**
		3.1: El sistema emite un mensaje “Se ha recibido una alarma de una zona no registrada, por favor actualice el sistema”.
Relaciones	**CU Incluidos**	No Procede.
	CU Extendidos	No Procede.
Requisitos no funcionales	RnF2, RnF3.	
Asuntos pendientes	No Procede.	

Tabla. #21. Descripción del CU Configuración General.

Nombre del CU7	Configuración General
Objetivo	Introducir los datos para la configuración del sistema.
Actores	Administrador: (Inicia) configuración general.
Resumen	El CU inicia cuando el administrador selecciona la operación de configuración general. El CU termina cuando los datos de configuración se

	actualizan en el sistema.	
Complejidad	Baja	
Prioridad	Auxiliar	
Precondiciones	Estado del Módulo Gestor en “Disponible”	
Postcondiciones	Actualizar los datos de configuración en el sistema.	
Flujo de Eventos		
Flujo Básico: Introducir los datos satisfactoriamente.		
Actor		**Sistema**
1: El administrador selecciona la operación configuración general del menú Configuraciones		2: El sistema muestra la interfaz para configurar los datos del sistema.
3: El administrador introduce los datos de configuración: dirección IP y puerto, ubicación de los plugins y ubicación de la BD.		
4: El administrador indica al sistema aceptar los datos introducidos.		5: El sistema verifica que ningún campo obligatorio haya quedado vacío.
		6: El sistema actualiza los datos de configuración y finaliza el CU.
Flujo Alterno: Cancelar los datos introducidos.		
Actor		**Sistema**
5.1: El administrador indica al sistema cancelar los datos introducidos y finaliza el CU.		
Flujo Alterno: Existen campos obligatorios vacíos.		
Actor		**Sistema**
		6.1: El sistema emite un mensaje “Existen campos obligatorios vacíos”.
Relaciones	**CU Incluidos**	No Procede.
	CU Extendidos	No Procede.
Requisitos no funcionales	RnF1, RnF5.	
Asuntos pendientes	No Procede.	

Tabla. #22. Descripción del CU Configurar Plugin de CRA.

Nombre del CU8	**Configurar Plugin de CRA**
Objetivo	Editar los datos de configuración de un plugin de central.

Actores	Administrador: (Inicia) configurar plugin de CRA.	
Resumen	El CU inicia cuando el administrador selecciona la operación de configurar plugin de CRA. El CU termina cuando los datos de configuración se actualizan en el sistema.	
Complejidad	Baja	
Prioridad	Auxiliar	
Precondiciones	Debe existir algún plugin de CRA configurable.	
Postcondiciones	Actualizar la configuración de un plugin de CRA.	
Flujo de Eventos		
Flujo Básico: Configurar Plugin de CRA satisfactoriamente.		
Actor		**Sistema**
1: El administrador selecciona en el apartado Centrales del menú Configuraciones un plugin de CRA a editar.		2: El sistema muestra la interfaz de configuración definida por el plugin de CRA.
3: El administrador introduce los datos de configuración definidos por el plugin de CRA.		
4: El administrador indica al sistema aceptar los datos introducidos.		5: El sistema actualiza los datos de configuración y finaliza el CU.
Flujo Alterno: Cancelar los datos introducidos.		
Actor		**Sistema**
4.1: El administrador indica al sistema cancelar los datos introducidos y finaliza el CU.		
Relaciones	**CU Incluidos**	No Procede.
	CU Extendidos	No Procede.
Requisitos no funcionales	RnF1, RnF5.	
Asuntos pendientes	No Procede.	

Anexo 3: Modelo de Datos.

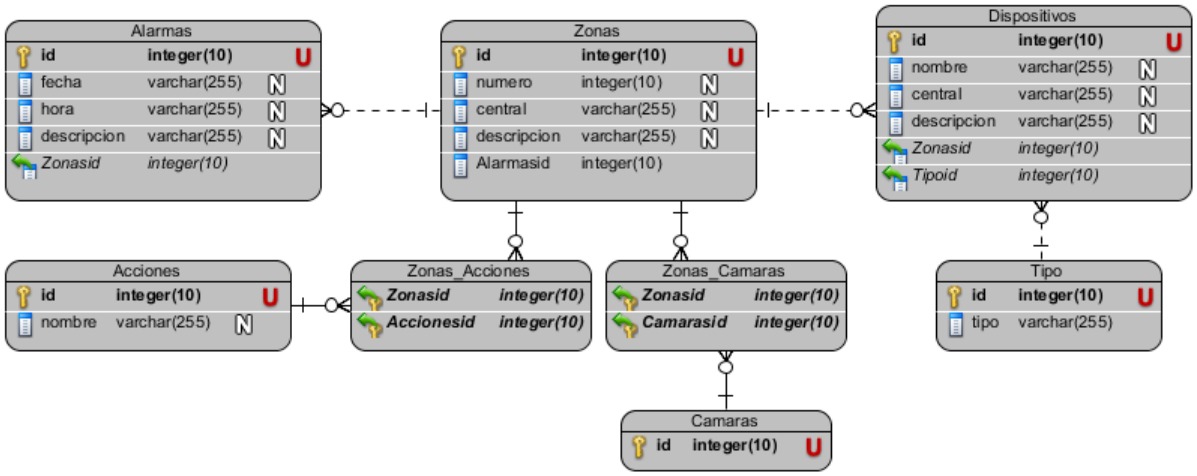

Fig. #11. Diagrama del Modelo de Datos.

Anexo 4: Diagrama de Clases del Diseño.

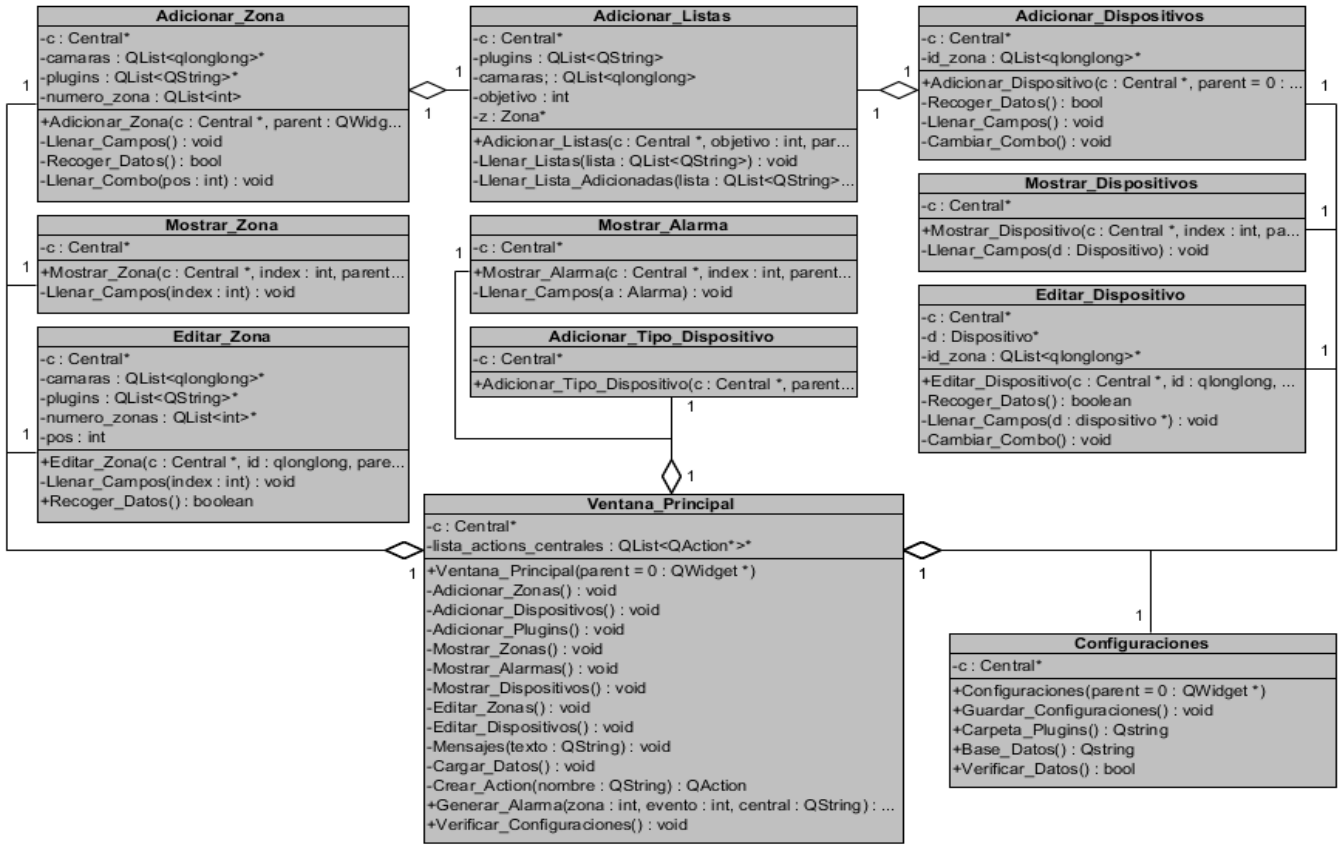

Fig. #12. Diagrama de Clases del Diseño Capa Presentación.

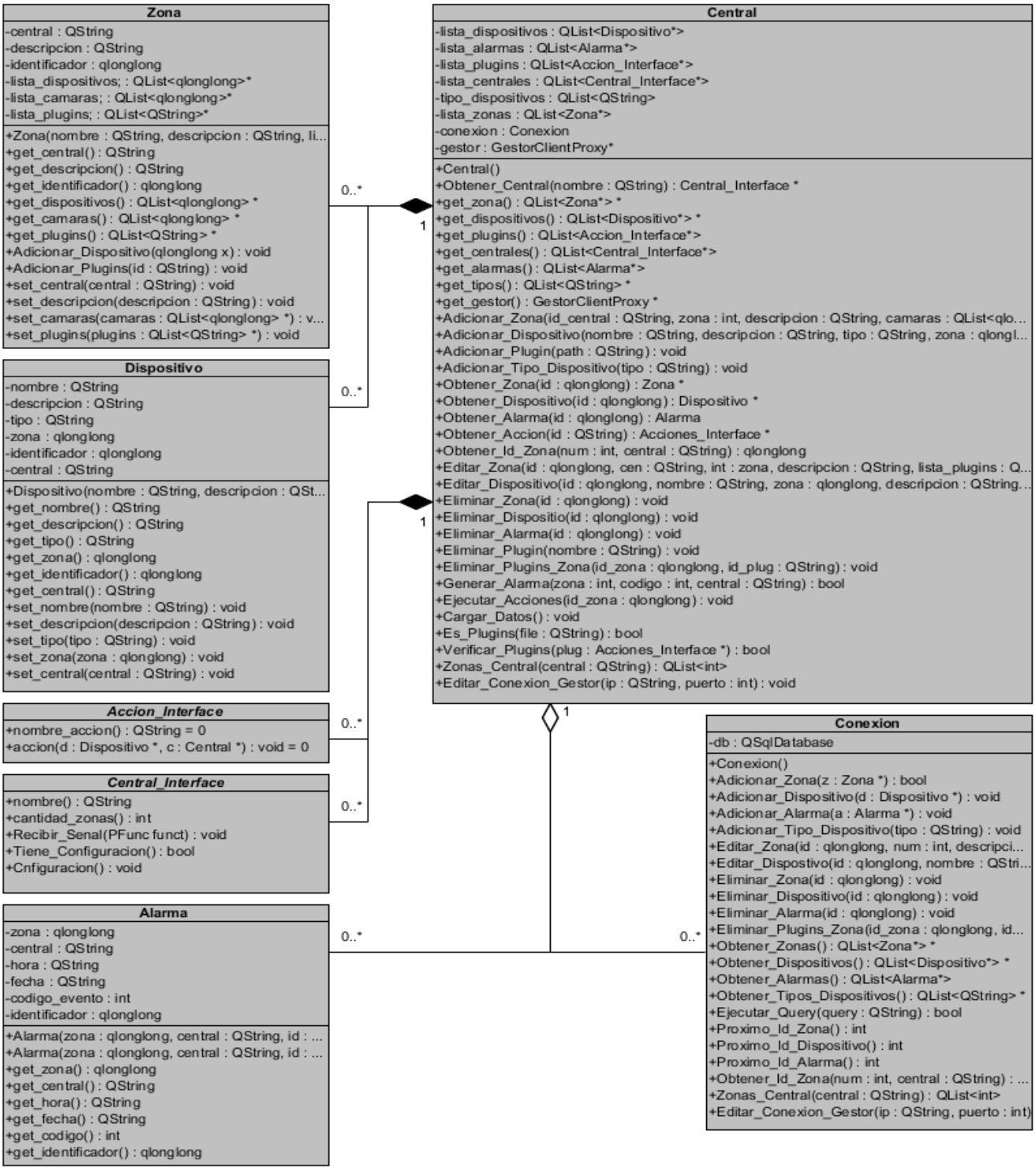

Fig. #13. Diagrama de Clases del Diseño Capa Negocio y Datos.

Anexo 5: Diagrama de Secuencia.

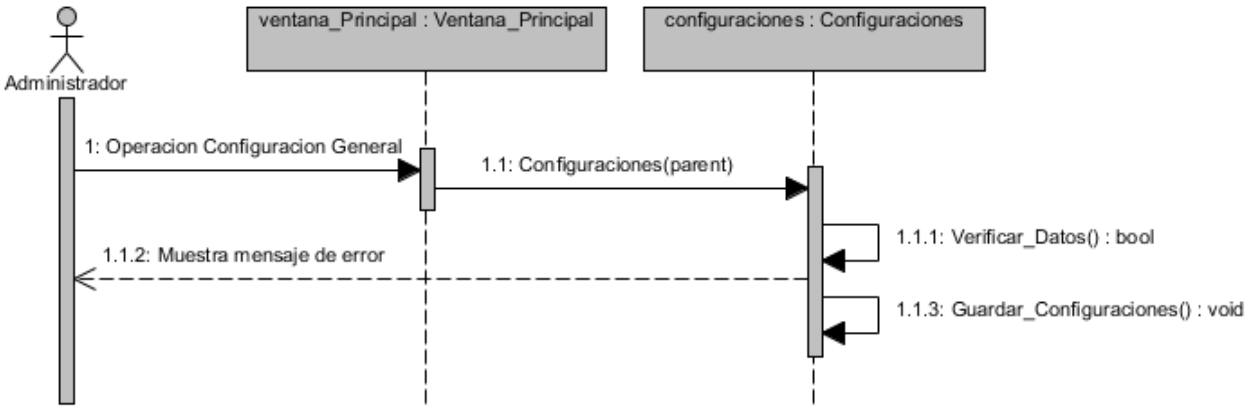

Fig. #14. Diagrama de Secuencia CU Configuración General.

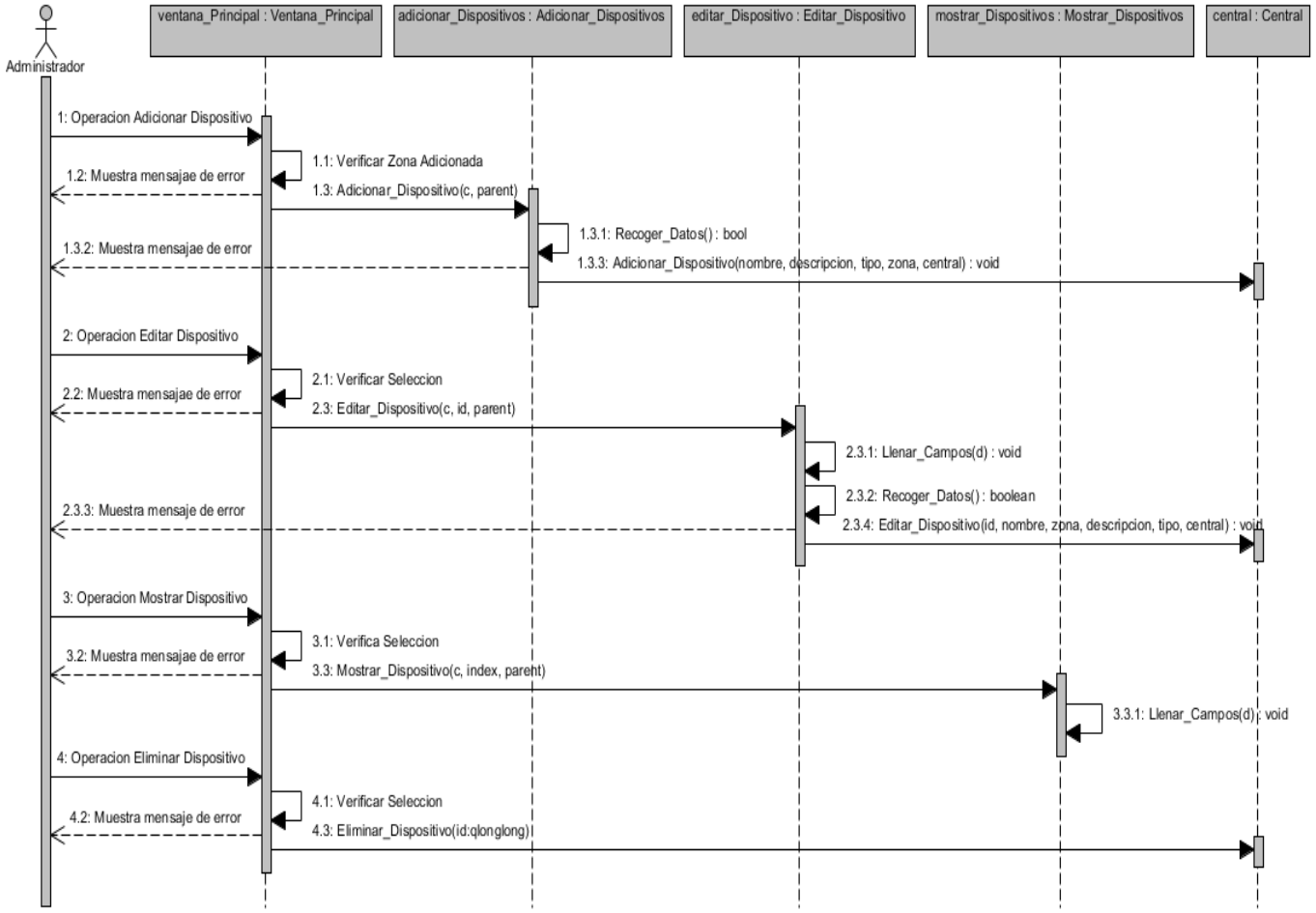

Fig. #15. Diagrama de Secuencia CU Gestionar Dispositivos.

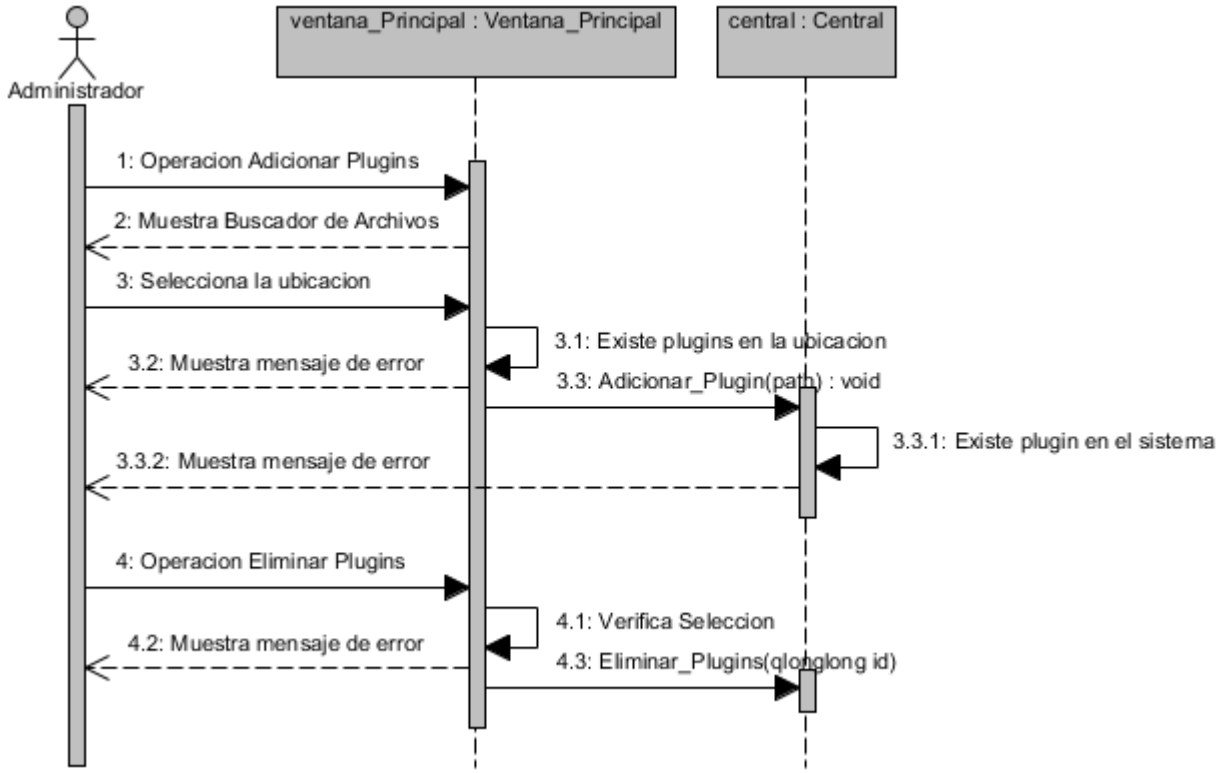

Fig. #16. Diagrama de Secuencia CU Administrar Plugins.

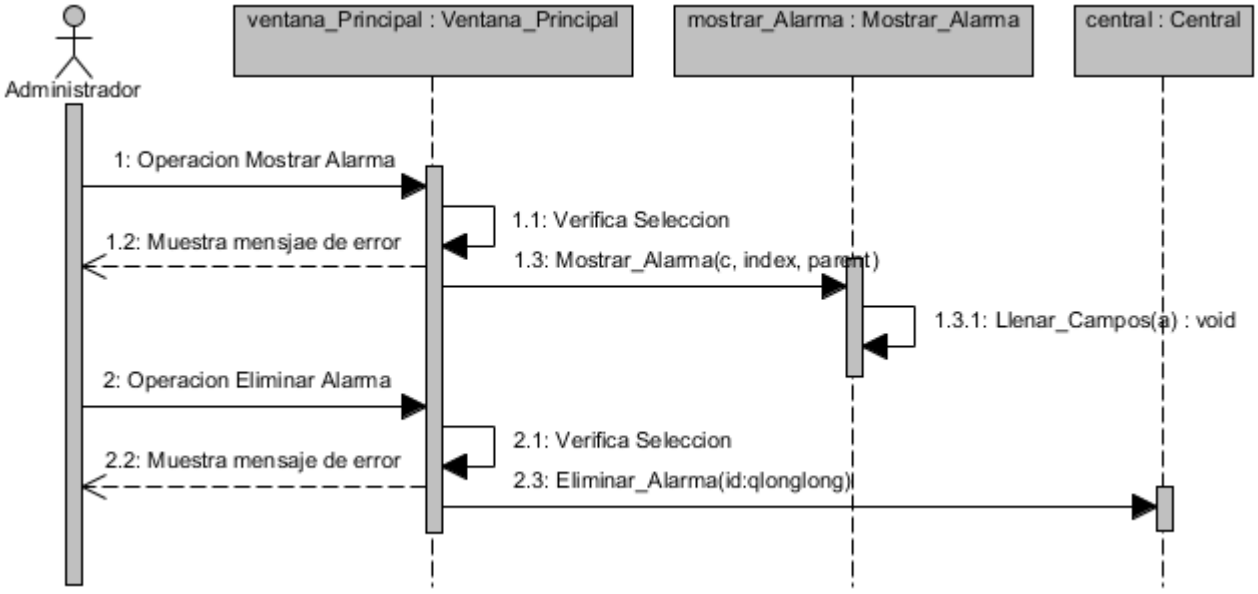

Fig. #17. Diagrama de Secuencia CU Administrar Alarmas.

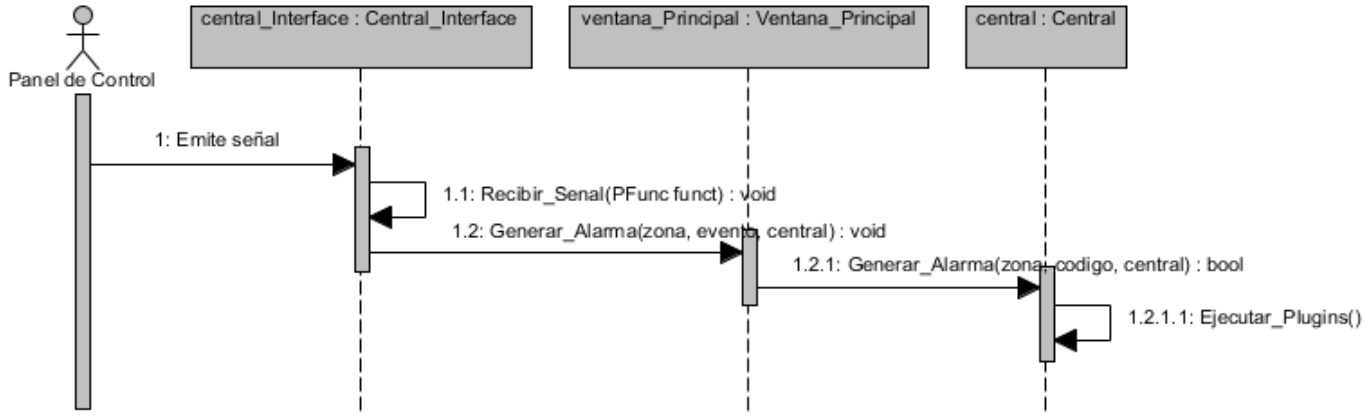

Fig. #18. Diagrama de Secuencia CU Ejecutar Plugins.

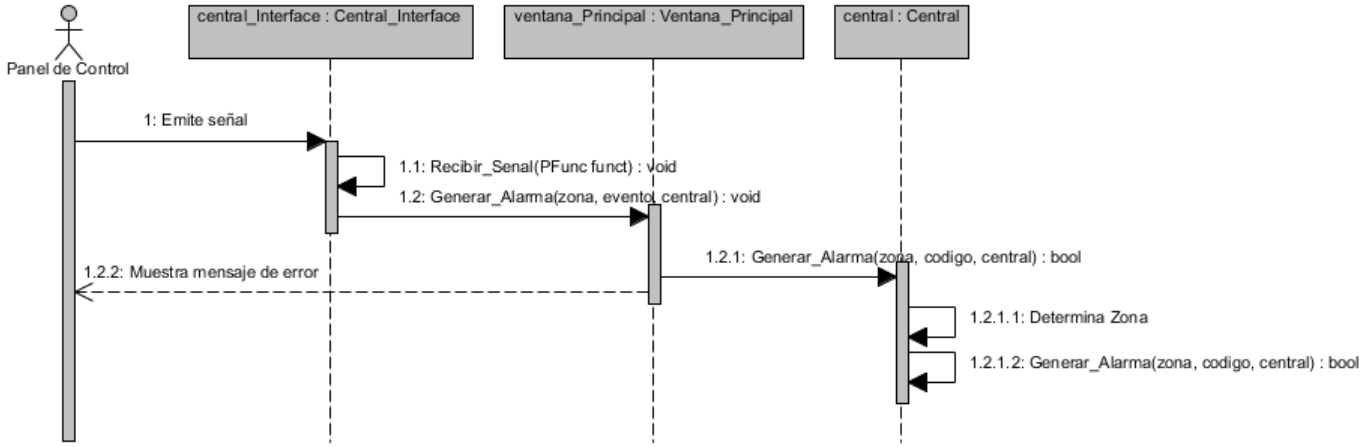

Fig. #19. Diagrama de Secuencia CU Generar Alarmas.

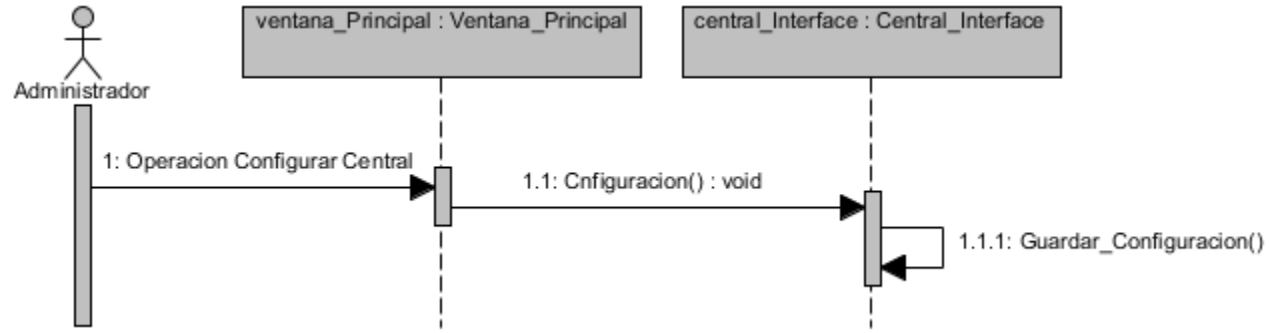

Fig. #20. Diagrama de Secuencia CU Configurar Plugin de CRA.