Universidad de las Ciencias Informáticas

Facultad 6

Título: "UCI-Alert: Software para la integración de la información generada por herramientas de gestión de redes durante el proceso de gestión de fallas en la UCI".

Trabajo de Diploma

Presentado para optar por el título de

Ingeniero en Ciencias Informáticas

Autores: Gonzalo Ramón Sarmiento Carreño

Dayron José Avello Marin

Tutores: M.Sc. Maikel Yelandi Leyva Vázquez

Ing. Alieski Sarmiento Almenares

Lic. Eduardo Hermenegildo Caballero Santana

Ciudad de La Habana, junio del 2008

"Pero la juventud tiene que crear. Una juventud que no crea es una anomalía realmente."

Ernesto Che Guevara

Declaración de autoría

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

Para que así conste firmamos la presente a los _____días del mes de _____del año 2008.

Gonzalo R. Sarmiento Carreño

Firma del Autor.

Dayron José Avello Marín

Firma del Autor

Maikel Yelandi Leyva Vázquez

Firma del Tutor

Alieski Sarmiento Almenares

Firma del Tutor

Eduardo Hermenegildo C.

Firma del Tutor

Agradecimientos:

A la Revolución, por habernos dado la oportunidad de estudiar en una Universidad de excelencia donde se nos formó como buenos profesionales y como jóvenes revolucionarios consiente de nuestro presente.

A nuestros padres, quienes han sabido guiarnos por el camino correcto y han sido los principales autores de nuestro buen desempeño.

A nuestros tutores Alieski y Maykel por su asesoramiento científico, su profesionalidad y tiempo dedicado.

Al Guille, por haber sido uno de los poco que confió en nuestro trabajo.

A Keiler, por su ayuda incondicional y su apoyo incomparable.

A Daynier, por su gran ayuda brindada

A nuestros amigos, compañeros de aula, de laboratorio, de cuarto, tratar de mencionar a todos significaría exponernos al riesgo de omitir a alguien importante, todos lo son, y por ello expresamos aquí nuestro reconocimiento y gratitud, gracias por confiar en nosotros.

A todos: Muchas Gracias.

Dedicatoria:

Gonzalo:

Dedicado en especial a mis padres, pues a ellos debo lo que soy, y a ellos hago entrega de mi trabajo, pues han dedicado todos sus esfuerzos y más, para que yo siempre pudiera seguir adelante, porque han tenido la sabiduría de guiarme en la distancia a través de sabios consejos, porque me han apoyado en todas las acciones de mi vida y porque han sido los mejores padres del mundo.
A mis demás familiares, que siempre han estado preocupados por mí y me han ayudado en todo lo que fuera necesario.
A Yeilis, por apoyarme, por comprenderme y por haber compartido su vida conmigo durante estos años de Universidad, por demostrarme que el amor existe y por ser una mujer tan maravillosa, nunca te olvidaré.
A mis amigos, por ser los mejores amigos del mundo, mencionarlos a todos, correría el riesgo de omitir a alguien importante y para mí todos lo son.
A mi compañero Dayron que juntos pasamos momentos difíciles, pero que no son nada comparados con los buenos momentos y lo bien que la pasamos en muchas ocasiones, gracias por ser amigo, gracias por ser mi hermano.

Dayron:
A mis padres porque son lo más lindo que tengo y porque siempre me han apoyado en todo, además, son los que van a pagar la fiesta, nada, que los adoro.
A mi hermano porque lo quiero mucho y estoy seguro que será mucho mejor que yo en el futuro.
A la memoria de mi abuela Lala porque gracias a ella soy lo que soy hoy, y a la de mi abuelo Gallego porque espero tener tantas mujeres como él, a ellos, siempre los llevo conmigo.
A mi abuela Amada por ser tan buena y especial conmigo, y a mi abuelo Chepe porque espero algún día ser tan inteligente como él pero sin sus defectos.
A mi tía Annie porque siempre ha sido un ejemplo a seguir, porque la quiero mucho, porque se sacrifica por mi familia y porque me mandó el pantalón para la exposición.
A mi tía Judith porque parece que al final no seré chofer de taxi como ella decía, nada, a ella también la quiero mucho.
A mi primo Ernestico porque ojalá sea un gran medico en el futuro.
A mi tío Jose porque siempre me ha encaminado por el camino de las ciencias.
A mi grupo del pre porque con ellos viví muy buenos momentos y son los mejores amigos que tengo.
A mi amigo Lester porque más que un amigo ha sido como un hermano para mí.
A mi amigo Gonzalo porque también lo quiero como un hermano, además de que me ha aguantado mucho durante la tesis.
A los socios de la cancha porque con ellos estuve la mayor parte del tiempo y en esa cancha dejo mis momentos más felices y tristes.
A los socios del cuarto porque me soportaron durante 5 años y créanme que eso no es nada fácil.
A las profesoras Taymara, Maybel y Yanelis por haberme dado una oportunidad cuando menos la merecía, espero nunca haberle fallados.
A los que nunca confiaron en mí por ser mi fuente de inspiración, mi título de oro es para ellos.
A mi novia porque quisiera ser muy feliz con ella.
A todos los que por una razón u otra se me han olvidado en esta lista y siempre los tengo presente.
A todos los que están aquí hoy conmigo, por haberme demostrado que realmente son mis amigos.
A todos mis familiares y amigos……para ustedes esta tesis.

Resumen

Hoy en nuestros días el uso de las redes telemáticas se ha hecho extensivo, garantizar una gestión eficiente de las fallas permite un uso extensivo de las mismas. En nuestra Universidad está presente la red de información más grande de Cuba por lo que es de vital importancia mantener una correcta gestión de fallas, para ello se cuenta con dos software que permiten llevar a cabo dicha tarea, el EPICenter y el Whats Up. El primero se encarga de gestionar los dispositivos y el segundo de monitorearlos y notificar sobre las fallas ocurridas, por lo que surge la necesidad de crear un sistema que integre la información de estos software. Con este objetivo se crea UCI-Alert, un sistema que se desarrolló en un ambiente multiplataforma, utilizando el lenguaje java para su implementación, el NetBeans como IDE de desarrollo y el PostgreSQL para el trabajo con bases de datos.

Palabras claves: redes telemáticas, gestión de fallas, EPICenter, Whats Up, UCI-Alert.

Índice

Introducción

La información es la materia prima del conocimiento, el cual a medida que han pasado los años se ha ido convirtiendo en la principal fuente de riquezas de los países, ganando grandes perspectivas de desarrollo. Las Tecnologías de la Información y las Comunicaciones (TIC) han jugado un papel primordial en la evolución de la distribución de la información a lo largo y ancho del globo terráqueo, pues han revolucionado desde las formas más primitivas de comunicación hasta alcanzar el desarrollo informático actual, gracias al empleo de tecnologías de información vía satélite y el gran uso de la red mundial de información, a través de la Internet.

Las redes de información han posibilitado el intercambio de conocimientos, estudios y experiencias de especialistas de diferentes ramas del saber a nivel mundial, mediante el envío de sonido, voz, texto e imagen, haciendo posible mantener actualizados a los científicos, empresarios y en general a todas las comunidades en los últimos avances del conocimiento, con el fin de impulsar la investigación básica y aplicada de punta, para que de esta forma las naciones puedan elevar su nivel de desarrollo en los diferentes entornos.

Las redes de información o también llamadas redes telemáticas nos brindan múltiples servicios con notables facilidades y beneficios para el desarrollo de las nuevas tecnologías. Entre estos servicios se destacan:

1. Correo electrónico
2. Transmisión de archivos
3. Consulta a páginas Web
4. Comercio electrónico
5. Publicaciones periódicas en texto completo
6. Boletines electrónicos
7. Sistemas de conferencias
8. Foros electrónicos

En la actualidad, el desarrollo de la tecnología de redes ha experimentado un gran salto cualitativo, ocasionando que la cantidad de recursos destinados para su administración sea cada vez mayor y

sean más complejos, lo cual conlleva a la evolución conjunta de acciones destinadas a optimizar dicha administración, estas acciones se encuentran agrupadas dentro de la Gestión de Redes.

"La Gestión de Redes tiene como propósito la utilización y coordinación de los recursos para planificar, organizar, mantener, supervisar, evaluar, y controlar los elementos de las redes de comunicaciones para adaptarse a la calidad de servicio necesaria, a un determinado costo." (1). Esta se encuentra dividida en 5 áreas funcionales:

- Gestión de Fallas
- Gestión de Configuración
- Gestión de Contabilidad
- Gestión de Desempeño
- Gestión de Seguridad

Dentro de estas áreas que componen la Gestión de Redes, una de las más importantes es la Gestión de Fallas, la cual nos brinda la posibilidad de obtener notificaciones de errores, registrar dichas notificaciones, así como comprobar estados y disponibilidad de los recursos de red para trazar y detectar fallas. Grandes locales con una amplia estructura de red necesitan una adecuada Gestión de Fallas.

La Universidad de las Ciencias Informáticas(UCI) cuenta con más de 7000 computadoras conectadas entre sí a través de 12000 puntos de acceso al medio, las cuales están distribuidas en 135 laboratorios docentes y de producción, cuenta además con 250 switch; por lo que nuestra red es considerada la más grande de nuestro país. De lo anterior expuesto se puede deducir que es primordial mantener una correcta gestión de fallas en nuestra Universidad. Para ello se cuenta con diferentes software especializados, tales como el "Whats up" y el "EPICenter".

Las versiones existentes de estos software en la UCI son utilizadas para monitorear el estado de la red y para la administración de los switch capa 3; cuando ocurre alguna falla, el "Whats up" es el encargado de informar sobre la misma, ya sea por correo electrónico, por beeper o de forma gráfica, dicha notificación es un poco limitada, puesto que solamente informa la dirección IP del dispositivo afectado; el "EPICenter", dedicado a la administración de los switch capa 3 se encarga de registrar información específica de estos dispositivo como por ejemplo: la última vez que falló el dispositivo, la

cantidad de veces anteriores que ha fallado, fecha, hora y el tiempo que estuvo caído, una descripción del evento ocurrido, además de que este software registra todas las trazas de los equipos administrados las cuales son muy útil para determinar las posibles causas del problema; en la actualidad la información especificada por el "EPICenter" no es reflejada cuando el "Whats Up" notifica sobre una determinada falla.

Teniendo en cuenta lo antes expuesto el **problema científico** de la investigación es: *La falta de integración de las informaciones generadas por el distinto software durante el proceso de gestión de fallas de la red en la UCI.*

Siendo el **objeto de estudio** de este trabajo**:** *El proceso de gestión de redes telemáticas*, cuyo **campo de acción** está enmarcado en: *El proceso de gestión de fallas de las redes telemáticas en la Universidad de las Ciencias Informáticas.*

Para darle solución a esta problemática se ha propuesto como **objetivo general**: *Desarrollar un sistema que integre la información generada por los software durante el proceso de gestión de fallas en la UCI.* Del cual se derivan los **objetivos específicos** siguientes**:**

- *Obtener los requisitos del sistema.*
- *Diseñar el sistema.*
- *Implementar el sistema*
- *Validar la calidad del sistema desarrollado.*

Para resolver el problema científico y alcanzar el cumplimiento de los objetivos se propusieron las siguientes **tareas científicas:**

1. *Estudio del negocio donde se implantará el sistema.*
2. *Realización de entrevistas con los clientes para identificar las funcionalidades que tendría el sistema.*
3. *Realización de las descripciones textuales de los CUS.*
4. *Realización de los diagramas de clases del diseño.*
5. *Identificación de las clases persistentes.*
6. *Realización de la BD del sistema.*

7. *Realización de los diagramas de secuencia.*
8. *Realización del diagrama de clases persistentes.*
9. *Realización del modelo de datos.*
10. *Realización del diagrama de despliegue.*
11. *Realización de los diagramas de componentes.*
12. *Implementación del sistema.*
13. *Realización de las pruebas de unidad a lo implementado.*

Para una mejor organización del trabajo, se decidió estructurar el contenido dividido en capítulos como se muestra a continuación:

Capítulo 1: Fundamentación Teórica. Se presenta una descripción más profunda de aspectos fundamentales para el desarrollo del sistema, temas como la Gestión de Redes, elementos que la componen, Gestión de Fallas, etc. Además ofrece una breve descripción de distintos software que se utilizan a nivel mundial para el proceso de gestión de fallas en la red, mostrado sus características, ventajas y deficiencias. Se describen las tendencias y tecnologías actuales a tener en cuenta para implementar el sistema, y se justifica el por qué del uso de las escogidas para el desarrollo de éste.

Capítulo 2: Características del Sistema.
Abarca lo relacionado al funcionamiento del negocio. En este caso se emplea un modelo de dominio, en el cual se realiza un análisis de cada uno de los conceptos y entidades presentes en el negocio. Además describen los elementos imprescindibles para una exitosa solución: como son los requerimientos funcionales y no funcionales. Se definen y describen los casos de uso del sistema al igual que los actores que los inician.

Capítulo 3: Análisis y Diseño del sistema. Aborda lo relacionado al flujo de trabajo de análisis y diseño, mostrando los diagramas correspondientes al mismo como son: las vistas lógicas, los diagramas de clases del diseño y los diagramas de interacción para cada caso de uso correspondiente, se refiere además a los patrones de diseño y de arquitectura utilizados para el desarrollo del sistema. También se definen las clases persistentes y el modelo de datos.

Capítulo 4 Implementación y pruebas. En este capítulo se analizan los diagramas de componentes, se ejemplifican con códigos de algunas clases el uso de los patrones de arquitectura y diseño. Se realizan pruebas de caja negra para comprobar que se cumple con el objetivo planeado en cuanto a su funcionamiento.

Capítulo 1: Fundamentación Teórica.

1.1 Introducción:

En este capítulo se realiza un análisis acerca de las tendencias de la gestión de redes telemáticas, objeto de estudio de la investigación, centrándose en lo relacionado con la gestión de fallas de las redes. Se muestran los aspectos más importantes de distintos software desarrollados en el mundo para el proceso de gestión de fallas de las redes de información. Además se establece la metodología de trabajo a seguir, así como las técnicas y herramientas que se aplicarán durante la investigación.

1.2 Gestión de Redes Telemáticas

En nuestros días, el desarrollo de las tecnologías ha permitido que el uso de las redes telemáticas se haga extensivo, expandiéndose a todas las ramas de la sociedad. Debido a su complejidad, las redes de información necesitan de varios factores para su funcionamiento, los cuales necesitan estar integrados para lograr un eficiente uso de las redes, por tal razón surge la Gestión de Red.

1.2.1 ¿Qué es la Gestión de Redes?

Se entiende por gestión de red al conjunto de elementos de control y supervisión de los recursos que permiten que la comunicación tenga lugar sobre la red, así como asegurar un servicio continuo a los usuarios finales además de que garantiza que se incremente el desempeño de una red con el empleo de la mejor tecnología, recursos humanos adecuados y herramientas integradas que automaticen las operaciones de gestión.

"La gestión de redes incluye el despliegue, integración y coordinación del hardware, software y los elementos humanos para monitorizar, probar, sondear, configurar, analizar, evaluar y controlar los recursos de la red para conseguir los requerimientos de tiempo real, desempeño operacional y calidad de servicio a un precio razonable" (2).

1.2.2 Elementos de un Sistema de Gestión de Redes

Durante el proceso de gestión intervienen diversos elementos fundamentales como son el gestor, el agente, el protocolo de gestión, la Base de Gestión de Información (MIB, por sus siglas en inglés) y el protocolo de gestión los cuales se interrelacionan empleando el modelo gestor-agente.

El gestor es quien envía las instrucciones que dirigen las operaciones de gestión y recibe notificaciones y respuestas. Este es implementado en una estación de gestión en la cual se debe contar con la MIB del dispositivo en gestión y con una interfaz visual.

El agente es quien da respuesta a las instrucciones enviadas por el gestor, esta operación se realiza accediendo a la MIB para manipular los objetos relacionados con la petición realizada. El agente se encuentra ubicado en el dispositivo gestionado.

La MIB es quien contiene al conjunto de objetos gestionados, los cuales representan a los recursos de la red que permiten algún tipo de gestión en una forma abstracta. La MIB se encuentra ubicada en el dispositivo a gestionar y una referencia de ésta es necesaria en el gestor.

El protocolo de gestión es quien define las reglas que permite la comunicación entre los diferentes dispositivos. En la actualidad SNMP (Protocolo Simple de Gestión de Red), forma parte del modelo de gestión de Internet, y CMIP (Common Management Information Protocol), es parte del modelo de gestión OSI son los protocolos predominantes.

1.2.3 Áreas Funcionales

La Gestión de Redes se divide en 5 áreas fundamentales:

Gestión de fallas: Se encarga de generar alarmas con información específica del error ocurrido, registrar dichas notificaciones y analizar el estado de los recursos de red para trazar e identificar fallas.

Gestión de configuración: Se encuentra dividido en actividades de inicialización, instalación y abastecimiento lo que permite recoger información acerca de la configuración y el estado en

demanda, de esta forma se brindan facilidades de inventario y además permite que se anuncien los cambios de configuración mediante notificaciones relevantes.

Gestión de contabilidad: Su objetivo principal es recolectar información de contabilidad y procesarla con el propósito de realizar cobros y facturaciones. Estas actividades establecen un límite contable para que un conjunto de costos se combinen con recursos múltiples y se utilicen en un contexto de servicio.

Gestión de desempeño: Se encarga de ordenar la información lo que permite determinar la carga del sistema y de la red bajo condiciones naturales y artificiales, además de que proporcionando estadísticas lo que permite actividades de planeación de configuración.

Gestión de seguridad: Se relaciona con 2 aspectos fundamentales de la seguridad del sistema: la gestión de seguridad misma, la cual necesita supervisar y controlar la disponibilidad de facilidades de seguridad al igual que reportar amenazas y rupturas en la seguridad. La seguridad de la gestión, la cual requiere la capacidad para autentificar usuarios y aplicaciones de gestión, con el propósito de garantizar la confidencialidad e integridad de intercambios de operaciones de gestión, así como impedir accesos no autorizados a la información.

1.2.4 Importancia de la Gestión de Redes.

"Las redes de comunicaciones han evolucionado con el paso del tiempo ante la necesidad de satisfacer las demandas de los diferentes servicios de telecomunicaciones, que día a día necesitan un mayor ancho de banda y una mejor calidad de servicio para las nuevas aplicaciones que se han venido desarrollando hasta la actualidad" (1)

Su uso se ha extendido a todos los renglones de la sociedad, su correcto funcionamiento depende de la aplicación adecuada de los sistemas de gestión de red, la implantación de estos se hace importante por las siguientes razones:

- Los sistemas de información son necesarios para el desarrollo de la sociedad y están soportados sobre redes.
- El incremento del volumen de información manejada es cada vez mayor y se encuentra más disperso.

- Las nuevas tecnologías necesitan de una gestión cada vez más especializada, que le permita el empleo eficiente de sus recursos de telecomunicaciones.
- El adecuado empleo de las tecnologías de gestión de redes permite mejorar la eficiencia, disponibilidad y el rendimiento de las mismas, aumentar la relación calidad/costo en el diseño de las redes, así como aumentar la satisfacción de los usuarios por el servicio de red proporcionado.

1.2.5 Deficiencias en la Gestión de Redes

La creciente complejidad de las redes y su distribución geográfica conllevan al surgimiento de diversas dificultades a la hora de gestionar de forma eficaz las redes y sistemas. Durante el proceso de gestión surgen problemas en acciones tales como el acceso a Internet, la seguridad de la red, las aplicaciones cliente-servidor, exigencias de los usuarios, la proliferación de aplicaciones distribuidas, los crecientes costos de los servicios, etc. Las principales limitaciones que hoy en día presentan los sistemas de gestión de red vienen dadas por las deficiencias propias del protocolo SNMP y de las MIB, en la actualidad se trabaja por crear versiones más eficientes de las mismas para así lograr mayores resultados durante el proceso de gestión de red.

1.3 Gestión de Fallas

La gestión de fallas es una de las áreas fundamentales dentro de la gestión de redes ya que es la encargada de interactuar directamente con los errores que se producen en la red. ***"Se ocupa de mantener un funcionamiento correcto de la red, tratando de protegerla de los fallos que puedan aparecer en el sistema en su conjunto o en los elementos que lo componen"*** (3).

Sus objetivos principales son identificar los fallos lo más rápidamente posible e identificar su causa para corregirlo, es decir que esta área se encarga de la detección, aislamiento y eliminación de los fallos, entiéndase por fallo a toda desviación del conjunto de objetivos operacionales, servicios o funcionales del sistema.

1.3.1 Funciones de la Gestión de Fallas

- Guardar un *log* de errores significativos, para esto deben usarse criterios para evitar la sobrecarga y permitir obtener información referente a fallos concretos.

- Anticiparse a posibles errores, para lograr esto se debe poner límite a los números de disparos para los determinados valores monitorizados en la red.
- Aislamiento y diagnóstico de fallos, para ello debe proporcionar un conjunto de herramientas para realizar pruebas.
- Proporcionar una interfaz de usuario efectiva, ya que es importante que las tareas de detección y aislamiento se hagan de manera rápida.

1.4 Tendencias de las Aplicaciones de Software para la Gestión de Fallas

1.4.1 Whats Up

Software realizado por la compañía Ipswich, esta herramienta realiza un eficaz monitoreo en tiempo real y sin límites lo que permite una rápida identificación de los problemas, es fácil de usar lo que permite una interacción eficiente con el usuario, monitorea elementos vitales de la red generando alarmas cuando encuentra alguna dificultad, estas notificaciones son enviadas por beeper o por email, este herramienta brinda la posibilidad de mostrar en una página web un reporte completo del estado de la red el cual puede ser accedido fácilmente de cualquier PC.

"En un mercado abrumado por la complejidad, WhatsUp establece un nuevo estándar en herramientas de monitorización de red con su exclusiva combinación de capacidad de uso y funcionalidad, bajo coste de propiedad y retorno rápido de la inversión" (4). Brinda los siguientes servicios:

- Despliegue gráfico del estado de los componentes que han sido monitoreados.
- Confirmación de inicio y terminación de una conexión realizada en la red.
- Monitoreo constante de muchos de los elementos de la red.
- Iniciación de alarmas y la notificación digital de las mismas.

Limitaciones:

La principal limitación que presenta es que es un software de fabricación privada por lo que es necesario pagar para obtener una versión el mismo, versiones de este software cuestan en el mercado mundial alrededor de 1 000 dólares, no es muy eficiente a la hora de leer las MIB que no sean definidas por su fabricante.

En la UCI se utiliza esta herramienta en la versión WhatsUp Gold v11 pero no se paga la licencia.

1.4.2 EPICenter

EPICenter es una aplicación potente y fácil de usar que trabaja con un amplio margen de estadísticas en tiempo real, brinda muchas facilidades a la hora de gestionar los switch capa 3, este software permite una rápida configuración lo que facilita el análisis y la supervisión del estado de la red, ofrece un amplio conjunto de herramientas de gestión que son muy útiles para sus usuarios, presenta una gran número de reportes lo que facilita el manejo de las fallas y estos pueden ser accedidos a través de la web.

Este software fue creado por la compañía Extreme Network y desarrollado sobre java, el cual presenta las siguientes funcionalidades:

- Control eficiente de la red.
- Gestión inteligente de los dispositivos de red.
- Multiplataforma.
- Soporta el acceso de múltiples usuarios.
- Permite acceso web a su base de datos de fallas.

Limitaciones:

A pesar de ser una ponderosa herramienta para gestionar los dispositivos de red no es muy eficiente a la hora de manejar aquellos dispositivos que no sean creados por su fabricante, genera un gran volumen de información referente a las fallas lo que puede limitar la búsqueda de datos relevantes, es un software propietario y su precio es alrededor de los 4 600 dólares.

Esta herramienta es utilizada en la UCI, pero no se paga la licencia.

1.4.3 Nagios

Es un sistema especializado en el monitoreo de equipos diseñado para informar proactivamente al cliente de sus posibles problemas de red, escrito en C y publicado bajo la licencia GNU, permite obtener datos e interpretarlos con el propósito de tomar mejores decisiones, está constituido por un

núcleo que es la interfaz de usuario y por plugins que pueden ser diseñados en varios lenguajes y que representan los ojos y oídos del sistema.

"Nagios es diseñado para permitir a plugins devolver datos de funcionamiento opcionales además de datos de estado normal, así como permitirle pasar aquellos datos de funcionamiento a programas externos para su tratamiento. Con ello podemos obtener datos extras de estados de servicios y de hosts" (5).

Su funcionamiento consiste básicamente en una arquitectura cliente-servidor que realiza de manera constante el chequeo de recursos y servicios sobre sistemas clientes, dentro de sus funcionalidades principales se pueden encontrar:

- Amplia gestión de servicios.
- Monitorización de recursos de sistemas.
- Diferentes notificaciones de errores por tipo de contacto.
- Definición de acciones reactivas ante las fallas detectadas.
- Visión rápida y sencilla de los elementos gestionados.

Limitaciones:

Su configuración inicial suele ser complicada ya que se crean un gran número de opciones y parámetros, no es muy efectivo a la hora de gestionar dispositivos de red ya que su formato de configuración es basado en plantillas lo que hace muy pesado el acceso a estos dispositivos.

Esta herramienta es utilizada en la UCI, está configurada para controlar los servicios (http, smtp, ftp, etc.), pero no para monitorear los switch.

1.4.4 Webalaizer

"Es una herramienta de análisis de servidores rápida, fiable y fácil de usar, esta aplicación fue desarrollada en el lenguaje C y distribuida bajo la licencia pública de GNU, realiza un control constante de los log y brinda los reportes en formato HTML lo que facilita el acceso a los mismos, permite un monitoreo rápido de la red ya que mide y analiza el movimiento dentro la aplicación analizada lo que facilita tomar mejores decisiones avaladas por la audiencia" (6).

Fue diseñado fundamentalmente para monitorear aplicaciones web, presenta diversas funcionalidades como:

- Manejo rápido de los registros almacenados.
- Soporta múltiples lenguajes.
- Capacidad ilimitada para guardar los log.
- Generación de reportes a través de script.

Limitaciones:

Esta aplicación genera un gran número de información lo que provoca que en los reportes aparezca información no útil, además de que no es recomendable utilizarlo para gestionar dispositivos de red ya que su propósito fundamental es monitorear los servidores web.

Esta herramienta se utiliza en la UCI.

1.4.5 EventWatch

Aplicación especializada en el manejo de fallas que fue desarrollado bajo la experiencia acumulada durante todos estos años en el manejo de fallas lo que demuestra su probada efectividad, realiza un monitoreo en tiempo real ya que corre como un servicio de Windows y siempre se mantiene analizando la base de datos de eventos, es fácil de instalar y configurar ya que se instala el mismo y demora menos de una hora, presenta una interfaz amigable a los usuarios, permite una reparación eficiente de los errores ya que los soluciona antes de que el usuario se detecte la falla, presenta su propia base de datos de dispositivos lo que permite una mayor compatibilidad con las MIB de distintos fabricantes, utiliza el método de correlación para caracterizar y eliminar las fallas.

Esta herramienta que fue desarrollada por la compañía TAVVE, la cual corre sobre los sistemas operativos Windows, Sun Solaris 2.6, IBM AIX 2.6, este software se puede adquirir gratis y sus principales funcionalidades son:

- Automatizar el monitoreo y la gestión de fallas.
- Soportar múltiples protocolos.
- Reportar el estado actual de los dispositivos de red de forma web.

- Mostrar diferentes gráficas que reflejen el estado de la red.
- Notificar los reportes por las vías tradicionales (email, beeper, etc.).

Limitaciones:
No corre sobre sistema operativo Linux, lo que podría ser un limitante a la hora de utilizarse en un local donde los servidores estén montados en dicho sistema operativo.

1.5 Metodologías de Desarrollo

Como ya se ha demostrado anteriormente la gestión de fallas juega un papel importante en el buen desempeño de las redes telemáticas por lo que se necesitan aplicaciones de software confiable y de calidad que permitan su desarrollo y mantenimiento. En los últimos años se han venido publicando estándares, notaciones y procesos de desarrollo de software que fijan las buenas prácticas para el desarrollo de aplicaciones de software.

Se estableció un estudio entre las metodologías más utilizadas:

1.5.1 Metodología XP.

Fig. 1 Metodología XP

La Programación Extrema (XP) es una metodología ligera de software que se basa en la simplicidad, la comunicación, realimentación o reutilización del código desarrollado. Es una de las metodologías de desarrollo de software utilizada para proyectos de corto plazo.

"La Programación Extrema asume que la planificación nunca será perfecta, y que variará en función de cómo varíen las necesidades del negocio. Por tanto, el valor real reside en obtener rápidamente un plan inicial, y contar con mecanismos de feedback que permitan conocer con precisión dónde estamos. Como es lógico, la planificación es iterativa: un representante del negocio decide al comienzo de cada iteración qué características concretas se van a implementar" (7).

La metodología XP tiene como objetivo mantener el software lo más sencillo posible, no se invierte ningún esfuerzo en hacer un desarrollo que en el futuro pueda tener valor ya que no se tiene referencia de cómo será el software.

1.5.2 Metodología RUP

Cuando se enfrenta la tarea de desarrollar un producto de software, se hace indispensable tener básicamente una metodología para su desarrollo. El "Proceso Unificado" es una de las metodologías que más se ajusta a las necesidades actuales de desarrollo de software ya que representa el resultado. Brinda la posibilidad de crear un modelo iterativo e incremental, ajustado al entorno cambiante de los requisitos en muchos proyectos, es centrado en la arquitectura y guiado por los casos de uso. Incluye artefactos (que son los productos tangibles del proceso como por ejemplo, el modelo de casos de uso, el código fuente, etc.) y roles (papel que desempeña una persona en un determinado momento, una persona puede desempeñar distintos roles a lo largo del proceso).

"El "Proceso Unificado" es más que un simple proceso; es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas de software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyectos" (8).

RUP elimina los errores cometidos en las iteraciones previas, lo cual permite desarrollar un proceso con calidad. Para eso se agrupan las actividades en grupos lógicos definiéndose flujos de trabajo principales divididos en 4 fases. Los 6 primeros flujos son conocidos como flujos de ingeniería y los tres últimos como de apoyo, donde todos y cada uno de ellos cobran vital importancia en el proceso de desarrollo de software.

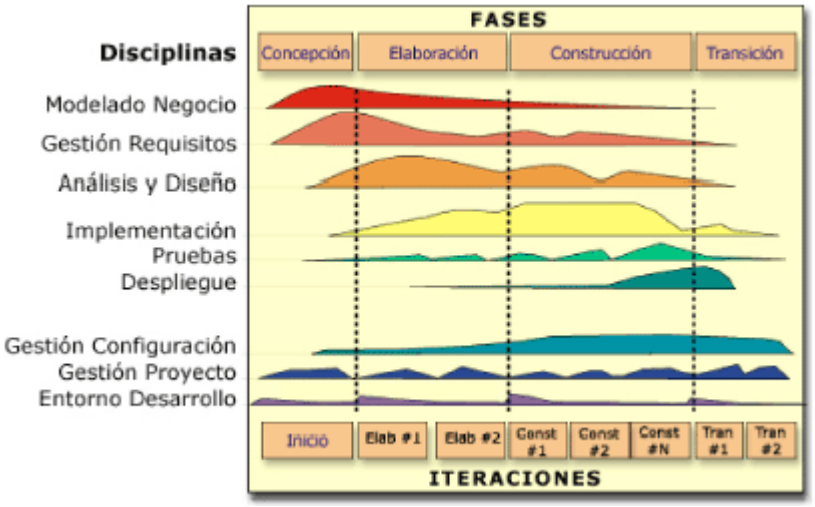


Fig. 2 RUP en dos dimensiones.

1.5.3 Fundamentación de la Metodología de Desarrollo seleccionada: Metodología RUP

La metodología escogida para el desarrollo del sistema fue RUP pues es la más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. Sus características de crear un modelo iterativo e incremental permite la adaptación a los diferentes cambios que se podrían producir en el sistema. A pesar de que RUP es un proceso de desarrollo de software genérico, se concibió en gran medida para el desarrollo de sistemas basados en programación orientada a objetos y se suele emplear esta metodología en proyectos de programación en lenguajes como Java, obteniendo muy buenos resultados. RUP genera gran volumen de información, lo que posibilita un mayor entendimiento tanto con el cliente como entre los miembros del equipo de desarrollo; respaldando de alguna manera las fallas, contradicciones y dudas que puedan surgir durante el desarrollo. La gran cantidad de artefactos que se generan contribuyen a un mejor entendimiento del problema.

1.6 Lenguaje y Herramientas utilizadas

Un buen uso del lenguaje de programación, así como la utilización de las mejores técnicas y herramientas garantiza una buena calidad del software a desarrollar, para su selección se analizaron

diferentes características como portabilidad, multiplataforma y facilidad de uso, después de este análisis se escogieron los que más se acerquen a los requerimientos de la plataforma.

1.6.1 ¿Rational Rose o Visual Paradigm?

Para el uso de las herramientas CASE se realizó un estudio y análisis entre las más utilizadas:

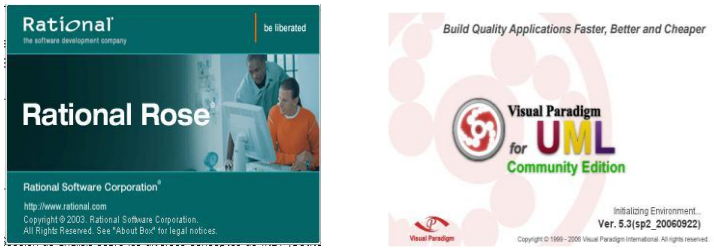


Fig. 3 Rational Rose y Visual Paradigm

Rational Rose brinda soporte a Unified Modeling Language (UML). Es un entorno de modelado que permite generar código a partir de modelos Ada, ANSI C++, C++, CORBA, Java/J2EE, Visual C++ y Visual Basic. Ofrece un lenguaje de modelado común que agiliza la creación del software. Es un software propietario. Visual Paradigm es una herramienta CASE que utiliza "UML": como lenguaje de modelado. Visual Paradigm-UML soporta los últimos estándares de anotaciones de JAVA y UML, provee soporte para la generación de código y la ingeniería inversa para Java. Además, Visual Paradigm -UML se integra con Eclipse, Borland, JBuilder, NetBeans IDE/Sun ONE, IntelliJ IDEA, Oracle JDeveloper y BEA WebLogic Workshop para soportar las fases de implementación en el desarrollo de software. Tiene dentro de sus características que es multiplataforma, portable y posee gran facilidad de uso. Debido a que Rational Rose es una herramienta CASE de software propietario y según lo planteado anteriormente, pues se decidió utilizar el Visual Paradigma, ya que es una herramienta de fácil integración con JAVA que es el lenguaje que se utilizará para desarrollar el sistema.

1.6.2 Fundamentación de la Herramienta CASE seleccionada: Visual Paradigm

Fue escogida la herramienta Visual Paradigm por ser un sistema multiplataforma y poseer gran facilidad de uso, y aunque es un software propietario, la UCI paga la licencia. Además agrupa

acciones muy útiles para el desarrollo del sistema, pues posee importantes relaciones de asociación con el lenguaje de programación y el IDE propuesto para la realización del sistema, por ejemplo brinda soporte para la generación de código y la ingeniería inversa para Java y se integra con Eclipse y NetBeans para soportar las fases de implementación en el desarrollo de software.

1.6.3 Características Principales de Java

Java es un lenguaje multiplataforma de cuarta generación, orientado a objetos, se caracteriza por su sencillez ya que ofrece toda la funcionalidad de un lenguaje potente pero sin las particularidades sofisticadas que provocan confusiones, ***"otro aspecto de la simplicidad de Java es que nada en Java es realmente nuevo. Si observa el conjunto de funciones de Java junto a la historia de la informática, descubrirá que todo procede de algún otro lugar"*** (9).

Presenta diversas funcionalidades que permiten el trabajo en la red, pues se construyó con extensas capacidades de interconexión TCP/IP, posee bibliotecas que permiten interactuar con protocolos como HTTP Y FTP. También ***"Java proporciona una colección de clases para su uso en aplicaciones de red, que permiten abrir sockets y establecer y aceptar conexiones con servidores o clientes remotos, facilitando así la creación de aplicaciones distribuidas"*** (10).

Constituye un sistema robusto. ***"Java fue diseñado para crear software altamente fiable. Para ello proporciona numerosas comprobaciones en compilación y en tiempo de ejecución"*** (10). Presenta características indiferentes a la arquitectura, pues Java está diseñado para aplicaciones que podrían ser ejecutadas en los diferentes ambientes de red.

1.6.4 Características Principales de Perl

PERL es un lenguaje interpretado que tiene varias utilidades, pero está principalmente orientado a la búsqueda, extracción y formateado de ficheros de tipo texto. También es muy usado para manejo y gestión de procesos.

"Algunas de las ventajas del uso del lenguaje PERL son las siguientes:

- ***Construcción de pequeños programas que pueden ser usados como filtros para obtener información de ficheros, realizar búsquedas, ...***

- ***Se puede utilizar en varios entornos, como puede ser Windows 95, OS/2,..., sin realizar cambios de código, siendo únicamente necesario la introducción del intérprete PERL correspondiente a cada sistema operativo.***
- ***También es uno de los lenguajes más utilizados en la programación de CGI scripts, que son guiones o scripts que utilizan el interface CGI (Common Gateway Interface), para intercambio de información entre aplicaciones externas y servicios de información.***
- ***El mantenimiento y depuración de un programa en PERL es mucho más sencillo que la de cualquier programa en C."*** (11)

Una de sus principales deficiencias radica en que no soporta polimorfismo completo.

1.6.5 Características Principales de Phyton

Phyton es uno de los lenguajes de script más frecuentemente empleados, se destaca por estar orientado a objetos de principio a fin. Su sintaxis emplea tabuladores para marcar bloques de código, destaca por la claridad y legibilidad de sus programas.

Contiene una estructura minimalista, ya que todo el lenguaje está desarrollado a partir de unos componentes básicos, los cuales también pueden ser modificados.

1.6.6 Fundamentación del Lenguaje de Programación seleccionado: Java

Fue escogido Java como lenguaje de programación, por ser un lenguaje multiplataforma, orientado a objetos, capaz de soportar las tres características propias del paradigma orientado a objetos: encapsulación, herencia y polimorfismo. Presenta propiedades de gran utilidad para el desarrollo del sistema, como por ejemplo sus clases, que permiten interactuar con protocolos HTTP y FTP, el soporte que brinda para sincronización de múltiples hilos de ejecución, por la seguridad, fiabilidad e integridad del lenguaje, todos estos son aspectos fundamentales y de gran utilidad para sistemas basados en conexiones de red. Además la Dirección de Informatización de la UCI propone este lenguaje para el desarrollo de aplicaciones de escritorio.

1.6.7 NetBeans vs Eclipse

En lo relacionado al IDE escogido para programar es evidente establecer una comparación entre estos dos grandes IDEs gratuitos.

Existen dos grandes diferencias, por ejemplo Eclipse posee un buen soporte de refactorización de cual carece NetBeans, pues sólo cuenta con un pequeño módulo gratuito con pocas refactorizaciones.

Eclipse requiere la utilización de muchos plugins para la realización de determinadas acciones, no sucediendo lo mismo con NetBeans, el cual ya viene pre-configurado con muchas funcionalidades y presenta una configuración operacional más sencilla que Eclipse, por ejemplo en lo relacionado a la interfaz gráfica NetBeans viene pre configurado con un editor de interfaz gráfica (como cualquier IDE) y esto ya resulta ser una ventaja, pues por otro lado Eclipse requiere de plugins para dichas funciones.

1.6.8 Fundamentación del IDE seleccionado: Netbeans 6.0

Fue escogido este IDE por sus distintas características que apoyaban el desarrollo del sistema. Mediante NetBeans es posible diseñar aplicaciones con sólo arrastrar y soltar objetos sobre la interfaz de un formulario, facilitando el trabajo de usuarios familiarizados con entornos como .NET. Otras características que esta versión incluye son: mejor edición del código, capacidades de navegación e inspección, historia local, soporte integrado para Subversion, y mayores capacidades de personalización integradas en la distribución estándar.

Además, se puede encontrar una interfaz totalmente rediseñada y el editor es importantemente más rápido que en sus versiones anteriores, también nos proporciona de una nueva coloración por sintaxis. El mobility pack [1]ya se encuentra integrado al IDE y utiliza el nuevo modelo de diseñador visual de NetBeans. También en comparación con otros IDE es más fácil de manipular y más rápido para el desarrollo y compilación de las aplicaciones.

[1] NetBeans Mobility Pack permite escribir, probar y depurar aplicaciones para la tecnología Java

1.7 Sistemas Gestores de Base Datos (SGBD)

Una base de datos es un conjunto de información que se puede acceder por medios informáticos y sobre el que se puede realizar diferentes acciones (insertar nuevos datos, selección de datos, actualización y el borrado de registros, combinación con otras bases de datos, generación de informes impresos, etc.).

Los SGBD ofrecen un control centralizado de la información, entre sus principales objetivos se encuentran:

- Evitar la redundancia de los datos.
- Mejorar los mecanismos de seguridad y privacidad de los datos.
- Mantener la integridad de los datos realizando las validaciones necesarias.
- Mejorar la eficacia de acceso a los datos.

1.7.1 PostgreSQL

"PostgreSQL es un Sistema Gestor de Bases de Datos Objeto-Relacionales (ORDBMS) que ha sido desarrollado de varias formas desde 1977. Comenzó como un proyecto denominado Ingres en la Universidad Berkeley de California. Está ampliamente considerado como el sistema de bases de datos de código abierto más avanzado del mundo. Posee muchas características que tradicionalmente sólo se podían ver en productos comerciales de alto calibre…" (12).

Presenta diferentes características como la herencia, funciones y restricciones. PostgreSQL brinda un gran número de funcionalidades que normalmente sólo aparecían en las bases de datos comerciales tales como DB2 u Oracle.

Soporta diferentes lenguajes como C, C++, Java, Python, PHP entre otros. Por otro lado, la velocidad de respuesta que ofrece este gestor quizás parezca no muy eficiente cuando probamos con bases de datos relativamente pequeñas, pero hay que tener en cuenta que esta velocidad es la que mantiene al gestionar bases de datos con grandes volúmenes de información.

1.7.2 Oracle

Oracle es básicamente una herramienta cliente/servidor para la gestión de bases de datos. Es un producto altamente vendido a nivel mundial, aunque la gran potencia que tiene y su elevado precio provocan que generalmente se utilice en grandes empresas y multinacionales. En el desarrollo de páginas web pasa lo mismo: como es un sistema muy caro no está tan extendido como otras bases de datos, por ejemplo, Access, MySQL, SQL Server, etc. Tiene el inconveniente que es un software privado.

1.7.3 MySQL

MySQL es un sistema de gestión de bases de datos relacional, licenciado bajo la GPL[2] de la GNU. Su diseño hace posible soportar una gran carga de forma muy eficiente. Fue creada por la empresa sueca MySQL AB, que mantiene el copyright del código fuente del servidor SQL, así como también de la marca.

Este gestor de bases de datos es, probablemente, el gestor más usado en el mundo del software libre, debido a su gran rapidez y facilidad de uso. Esta gran aceptación es debida, en parte, a que existen infinidad de librerías y otras herramientas que permiten su uso a través de gran cantidad de lenguajes de programación, además de su fácil instalación y configuración. Dejó de ser software libre, pues fue comprado por Sun Microsystem.

1.7.4 Fundamentación del Sistema Gestor de Base de Datos (SGBD) seleccionado: PostgreSQL

El gestor de bases de datos que se pretende utilizar para el sistema es PostgreSQL porque además de cumplir con las cuatros libertades del software libre, y sus características en las cuales incluye funcionalidades orientadas a operaciones con redes, es un sistema que se caracteriza por su robustez, escalabilidad y cumplimiento de los estándares SQL. En PostgreSQL el tamaño máximo de la base de dato es ilimitada; el de una tabla asciende a 32 TB, el de una fila a 1.6 TB y el de un campo de dato a 1GB. Estas características son de gran importancia pues el sistema que se desea implementar debe registrar los log de mayor importancia que ocurren en los switch capa 3, los

[2] General Public License.

cuales atendiendo a la cantidad de dispositivos y los posibles eventos que podrían ocurrir se necesitaría en una base de datos capaz de almacenar un gran volumen de información. Además la Dirección de Informatización de la UCI propone este SGBD para el manejo de bases de datos.

1.8 Conclusiones

En este capítulo se mostraron diferentes aspectos relacionados con la Gestión de Redes, haciendo énfasis en la Gestión de Fallas. Se realizó un análisis de las distintas aplicaciones utilizadas en el mundo para realizar la Gestión de Fallas. La metodología escogida para el desarrollo del sistema fue RUP, como herramienta CASE se utilizó Visual Paradigm, como lenguaje de programación se seleccionó Java, el IDE escogido fue Netbeans 6.0 y como SGBD PostgreSQL.

Capítulo 2 Características del sistema

2.1 Introducción

En este capítulo se realiza una descripción del objeto de estudio. Se utiliza el modelo de dominio para describir los procesos del negocio, se definen las funcionalidades del software a través de los requisitos funcionales y no funcionales; los actores y el diagrama de casos de usos del sistema, finalmente se realiza la descripción de los casos de usos.

2.2 Modelo del Dominio

El proceso de modelamiento del negocio permite obtener una visión de los procesos, roles y responsabilidades en los modelos de casos de uso y de objetos. En el negocio actual los flujos de información solo corresponden a eventos, es imposible determinar subsistemas y establecer reglas de funcionamiento; es decir, que se evidencia una poca estructuración de los procesos del negocio, por tales motivos se decidió hacer un modelo de dominio; además porque el objetivo fundamental del sistema a desarrollar es almacenar y mostrar información.

Esta técnica tiene la potencialidad de ahorrar cantidad de trabajo a los desarrolladores, así como la de evitar errores, entre otras cosas. Así el modelo del dominio tiene como objetivo fundamental la comprensión y descripción de las clases más importantes. Resume todo lo que tenga importancia conceptual para el entorno en que desarrolle el sistema y las relaciones que existan entre estos conceptos.

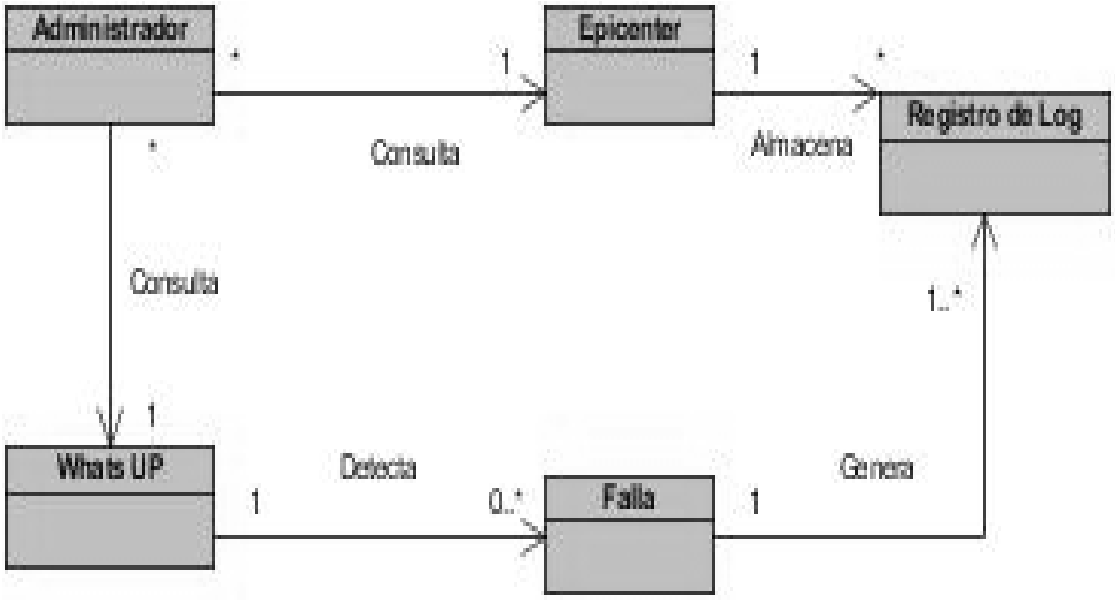


Fig. 4 Modelo de Dominio.

2.3 Requerimientos Funcionales

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir. Ellos permiten determinar, de una manera clara, responsabilidades que se propone el sistema.

R1: Verificar Estado

1.1 Detectar equipo caído

1.2 Notificar equipo caído

1.3 Detectar cambio de estado del equipo caído.

1.4 Notificar cambio de estado.

R2: Almacenar Registro de Falla.

R3: Registrar Notificación

3.1 Buscar registro de falla

3.2 Enviar registro de falla a las direcciones especificadas.

3.3 Guardar notificación.

R4: Gestionar Equipo

4.1 Insertar Equipo.

4.2 Mostrar Equipo.

4.3 Eliminar Equipo.

R5: Gestionar Usuario

5.1 Insertar usuario.

5.2 Modificar usuario.

5.3 Eliminar usuario.

5.4 Visualizar listado de usuarios.

R6: Autenticar Usuario

6.1 Verificar usuario y contraseña.

6.2 Otorgar privilegios.

R7: Actualizar Registro

R8: Mostrar Registro

R9 Gestionar Notificantes

9.1 Adicionar notificante.

9.2 Eliminar notificante.

9.3 Mostrar notificante.

2.4 Requerimientos No Funcionales

"Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen que el producto sea atractivo, usable, rápido o confiable. Normalmente están vinculados a requerimientos funcionales, es decir una vez que sepamos lo que el sistema debe hacer, podemos determinar cómo ha de comportarse, qué cualidades debe tener o cuán rápido o grande debe ser" (8).

Software:

Es necesario disponer de sistemas operativos Linux, Windows 95 o superior para la instalación de la aplicación. También debe tener instalado Java Runtime Enviroment (JRE) versión 1.5 o superior, además del gestor de bases de datos PostgreSQL 8.1.

Hardware:

Para el desarrollo y funcionamiento del sistema se necesitan máquinas con los siguientes requerimientos:

- Procesador Pentium IV o superior
- 1GB de memoria RAM o superior
- Memoria física superior a 500MB

Seguridad:

- Confidencialidad: la información manejada estará protegida de acceso no autorizado.
- Disponibilidad: la información siempre estará disponible y se podrán obtener los datos deseados en un momento dado.
- Autenticidad: el sistema presenta autenticación de usuario para su acceso.

Interfaz:

- Interfaz simple de usar.
- Amigable.
- Adaptable.
- Legible.

Restricciones en el diseño y la implementación

- Lenguaje de programación: Java
- Herramientas de desarrollo: SGBD PostgreSQL

Usabilidad

- El sistema brindará facilidad de uso para los usuarios finales
- El sistema va dirigido a técnicos medios, especialistas e ingenieros de la rama informática que tengan conocimientos o estén asociados a las redes telemáticas.

2.5 Definición de los Actores del Sistema

"Pueden representar el rol que juega una o varias personas, un equipo o un sistema automatizado" (8).

Tabla1: Actores del sistema

Actores del sistema	Justificación
Usuario	Es un usuario que generaliza el rol de autenticación al sistema.
EPICenter	Es un sistema externo encargado de guardar los registros de log correspondiente a las distintas fallas.
What's Up	Es un sistema externo encargado de detectar y notificar ante la presencia de fallas.
Servidor de correo	Es quien permite enviar notificación por correo electrónico.
Invitado	Es un usuario a quien la aplicación sólo le permite mostrar datos.
Administrador	Es el encargado de administrar la aplicación.

2.6 Casos de Usos del Sistema

Son artefactos que describen el comportamiento del sistema desde el punto de vista del usuario. Además establece acuerdos entre desarrolladores y clientes sobre los requisitos que debe cumplir el sistema.

Tabla2: Casos de Usos del Sistema

Cod.	Nombre de Caso de Uso	Justificación
1	Verificar Estado	Necesario para identificar cuando ha ocurrido una falla en un dispositivo.
2	Registrar Notificación	El sistema debe informar sobre la ocurrencia de fallas.
3	Almacenar Registro de Fallas	Una de las principales funcionalidades del sistema es guardar información referente a las fallas.
4	Gestionar Equipo	Es muy importante para el sistema ya que permite insertar, eliminar y mostrar los equipos.
5	Gestionar Usuario	Necesario porque le permite al administrador insertar, modificar y eliminar usuario.

6	Autenticar Usuario	Permite mantener la seguridad del sistema.
7	Actualizar Registro	Permite al administrador modificar campos del registro de fallas.
8	Mostrar Registros	Necesario porque permite mostrar los registros de las fallas ocurridas.
9	Gestionar Notificantes	Permite adicionar, eliminar y mostrar los usuarios a los cuales se desea notificar.

2.7 Diagrama de Casos de Uso del Sistema

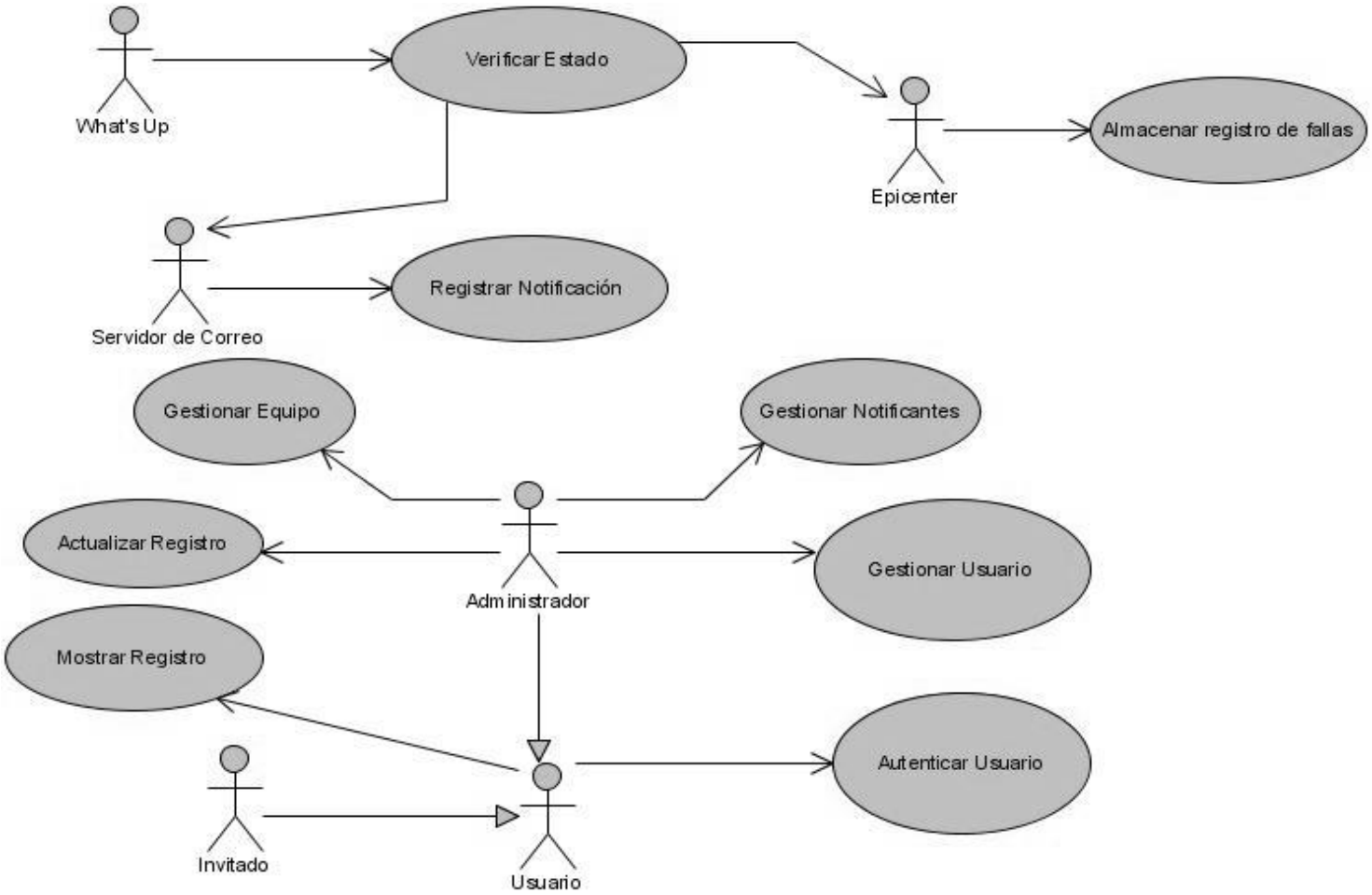


Fig. 5 Diagrama de Casos de Uso del Sistema.

2.8 Descripción de los Casos de Uso del Sistema

Tabla 3: Descripción del CU: Verificar Estado

Caso de Uso:	Verificar Estado	
Actores:	What's Up (Inicia).	
Propósito:	Detectar, reconocer e informar sobre el estado de un equipo.	
Resumen:	Cuando cambia el estado de un dispositivo el actor inicializa el caso de uso creando un fichero donde se especifica dicho estado. El sistema analiza el fichero y según su información realiza determinadas acciones, terminado así el caso de uso.	
Referencia:	R1,R2,R3	
Precondiciones:	Que el fichero se haya creado producto al cambio de estado de algún dispositivo de red, conteniendo la información de dicho cambio, que el sistema tenga registrado su dirección IP y los destinatarios a los cuales se les desea notificar.	
Flujo Normal de Eventos		
Acción del Actor		Respuesta del Sistema
1. El What´s Up crea un fichero especificando el estado del dispositivo. 2. El EPICenter facilita el registro de log del dispositivo afectado. 3. El Servidor de Correo envía la notificación a los distintos usuarios.		1.1 El sistema lee el fichero y muestra en una ventana el estado de los equipos. 1.2 El sistema accede al EPICenter para adquirir el registro de log del dispositivo afectado. 2.1 El sistema procede a guardar el registro de log. 2.2 El sistema le comunica al servidor de correo las direcciones de los usuarios a los cuales se les desea notificar sobre la falla. 3.1 El sistema procede a guardar la notificación realizada.
Flujos Alternos		

Acción del Actor	Respuesta del Sistema.
Poscondiciones:	Se determina el estado de los dispositivos.
Prioridad:	Crítico

Tabla 4: Descripción del CU: Almacenar Registro de Fallas

Caso de Uso:	Almacenar Registro de Fallas
Actores:	EPICenter. (Inicia)
Propósito:	Guardar el registro de las fallas que ocurren en los dispositivos.
Resumen:	Cuando ocurre alguna falla en el dispositivo el actor inicializa el caso de uso brindando al sistema el registro de log de dicho dispositivo.
Referencia:	R2
Precondiciones:	Que el sistema acceda al EPICenter.
Poscondiciones:	Se guarda el registro de log.
Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema
1. El EPICenter brinda los datos relacionados con la falla ocurrida en el dispositivo.	1.1 El sistema recibe el registro de log y le adiciona nueva información. 1.2 El sistema procede a guardar el registro de log terminando así el caso de uso.
Flujos Alternos	
Acción del Actor	Respuesta del Sistema
Prioridad:	Crítico

Tabla 5: Descripción del CU: Registrar Notificación

Caso de Uso:	Registrar Notificación.
Actores:	Servidor de Correo (Inicia).
Propósito:	Registrar la información notificada y a los usuarios notificados de la falla ocurrida.

Resumen:	Cuando ocurre alguna falla en el dispositivo el actor inicializa el caso de uso notificando sobre la falla ocurrida a los usuarios.
Referencia:	R3
Precondiciones:	Que falle el dispositivo y que el sistema tenga registrado las direcciones de los usuarios.
Poscondiciones:	Se guarda el registro de la notificación.

Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema
1. El Servidor de Correo envía las notificaciones a los distintos usuarios.	1.1 El sistema guarda las notificaciones enviadas finalizando el caso de uso.

Flujos Alternos	
Acción del Actor	Respuesta del Sistema

Prioridad:	Crítico

Tabla 6: Descripción del CU: Gestionar Equipo

Caso de Uso:	Gestionar Equipo
Actores:	Administrador(Inicia)
Propósito:	Permite adicionar, mostrar y eliminar los equipos.
Resumen:	El caso de uso se inicia cuando el administrador selecciona una de las siguientes opciones: Adicionar Equipo: El Administrador inserta la dirección del equipo a gestionar, el sistema verifica dicha dirección, si es correcta la guarda y muestra el siguiente mensaje: “El equipo ha sido adicionado correctamente”, si no es correcta se muestra: “La dirección ip no es correcta”, si el dispositivo ya ha sido adicionado se muestra un mensaje de: “Ya existe esa dirección ip”. Eliminar Equipo: El sistema muestra una lista con todos los dispositivos, el Administrador selecciona una dirección, el sistema muestra el siguiente mensaje: “Equipo eliminado correctamente” y elimina

	dicha dirección de la lista de equipos.
Referencia:	R4.1, R4.2
Precondiciones:	El usuario debe estar autenticado como Administrador.
Poscondiciones:	El sistema adiciona, muestra y elimina las direcciones de los dispositivos a gestionar

Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema
1. El Administrador selecciona realizar alguna de las opciones: - Adicionar Equipo. - Eliminar Equipo.	1.1 El sistema, según la operación que se seleccionó, hace lo siguiente: - Si el Administrador desea adicionar algún equipo, ir a sección: “Adicionar Equipo”. - Si el Administrador decide eliminar algún equipo, ir a sección: “Eliminar Equipo”.\
Sección: Adicionar Equipo.	
Acción del Actor	Respuesta del Sistema
2. El Administrador escribe la dirección IP del equipo a adicionar.	1.2 Muestra la interfaz que permite adicionar equipo. 2.1 El sistema verifica que no hayan campos vacíos. 2.2 El sistema verifica que la dirección IP no esté en la lista de dispositivos. 2.3 El sistema verifica que la dirección IP sea la correcta. 2.4 El sistema adiciona el equipo y muestra el siguiente mensaje: “El equipo ha sido adicionado correctamente”. 2.5 ir al paso 1.2 finalizando así el caso de uso.
Flujo Alterno: Adicionar Equipo	
Acción del Actor	Respuesta del Sistema
	2.1.1 El sistema detecta que el campo no ha sido llenado y muestra el siguiente mensaje: “No ha escrito una dirección IP”.

	2.1.2 Ir al paso 1.2 finalizando así el caso de uso.. 2.2.1 El sistema comprueba que la dirección IP ya está en la lista de dispositivos y muestra un mensaje de: “Ya existe esa dirección IP”. 2.2.2 Ir al paso 1.2 finalizando así el caso de uso. 2.3.1 El sistema comprueba que la dirección IP no es correcta y muestra un mensaje de: “La dirección ip no es correcta”. 2.3.2 Ir al paso 1.2 finalizando así el caso de uso.
Sección: Eliminar Equipo	
Acción del Actor	Respuesta del Sistema
2. El Administrador selecciona la dirección a eliminar.	1.2 Muestra la interfaz que contiene todos los dispositivos adicionados. 2.1 Busca y elimina de la lista de equipos la dirección seleccionada. 2.2 Muestra un mensaje de: “Equipo eliminado correctamente”. 2.3 Ir al paso 1.2 finalizando así el caso de uso.
Prioridad:	Crítico

Tabla 7: Descripción del CU: Gestionar Usuario

Caso de Uso:	Gestionar Usuario
Actores:	Administrador (Inicia).
Propósito:	Permite adicionar, mostrar y eliminar los usuarios del sistema, así como cambiar la contraseña de los mismos.
Resumen:	El caso de uso se inicia cuando el administrador selecciona una de las siguientes opciones: Adicionar Usuario: El Administrador inserta el nombre de usuario, la contraseña, verifica la contraseña y selecciona un tipo de usuario, finalmente el sistema muestra el siguiente mensaje: “Se ha agregado correctamente el ’tipo de usuario’: ’nombre usuario’ ” y se agrega a la

	base de datos. Eliminar Usuario: El sistema muestra a todos sus usuarios, el Administrador selecciona el que desea eliminar, el sistema muestra el siguiente mensaje: "Usuario eliminado correctamente" y elimina dicha dirección de la base de datos. Cambiar Contraseña: El Administrador selecciona el usuario al cual se desea cambiar la contraseña, adiciona la nueva contraseña y confirma la misma, el sistema muestra un mensaje de: "Se ha cambiado la contraseña correctamente" y actualiza la base de datos.
Referencia:	R5.1, R5.2, R5.3
Precondiciones:	El usuario debe estar autenticado como Administrador.
Poscondiciones:	El sistema adiciona, muestra y elimina los distintos usuarios, ya sean administradores o invitados.

Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema
1. El Administrador selecciona realizar alguna de las opciones: - Adicionar Usuario. - Eliminar Usuario. - Cambiar Contraseña	1.1 El sistema, según la operación que se seleccionó, hace lo siguiente: - Si el Administrador desea adicionar algún usuario, ir a sección: "Adicionar Usuario". - Si el Administrador decide eliminar algún usuario, ir a sección: "Eliminar Usuario". - Si el Administrador decide cambiar la contraseña de algún usuario, ir a sección: "Cambiar Contraseña
Sección: Adicionar Usuario	
Acción del Actor	Respuesta del Sistema
2. El Administrador inserta el nombre del usuario, la contraseña, verifica la misma y selecciona el tipo de usuario.	1.2 Muestra la interfaz que permite adicionar usuario. 2.1 El sistema verifica que los campos no estén vacíos y que se haya seleccionado algún usuario. 2.2 El sistema verifica que la contraseña tenga 5 o más dígitos.

	2.3 El sistema controla que al confirmar contraseña coincida con la anterior. 2.4 El sistema adiciona el equipo y muestra un mensaje de: “Se ha agregado correctamente el ’tipo de usuario’: ’nombre usuario’ ”. 2.5 Ir al paso 1.2 finalizando así el caso de uso.
Flujo Alterno: Adicionar Usuario	
Acción del Actor	Respuesta del Sistema
	2.1.1 El sistema detecta que algunos de los campos no han sido llenados y muestra el siguiente mensaje: “No debe dejar campos vacíos”. En caso de que todos estén llenos pero no se haya seleccionado ningún usuario pues se muestra el siguiente mensaje: “Debe seleccionar algún usuario”. 2.1.2 Ir al paso 1.2 finalizando así el caso de uso. 2.2.1 El sistema detecta que la contraseña introducida tiene menos de 5 dígitos y muestra el siguiente mensaje “Por medidas de seguridad la contraseña debe tener 5 o más caracteres”. 2.2.2 ir al paso 1.2 finalizando así el caso de uso. 2.3.1 El sistema comprueba que ha surgido un error a la hora de confirmar la contraseña y muestra el siguiente mensaje: “Error al confirmar contraseña”. 2.3.2 ir al paso 1.2 finalizando así el caso de uso.
Sección: Eliminar Usuario	
Acción del Actor	Respuesta del Sistema
2. El Administrador selecciona el usuario	1.2 El sistema muestra la interfaz que contiene todos los usuarios del sistema, ya sean Administradores o Invitados. 2.1 El sistema verifica que se haya seleccionado

a eliminar.	algún usuario. 2.2 El sistema verifica que el usuario seleccionado no es “Administrador”. 2.3 Elimina el usuario y muestra el siguiente mensaje de: “Usuario eliminado correctamente”. 2.4 Ir al paso 1.2 finalizando así el caso de uso.
Flujo Alterno: Eliminar Usuario	
Acción del Actor	Respuesta del Sistema
	2.1.2 El sistema detecta que no se ha seleccionado ningún usuario y muestra el siguiente mensaje de: “Debe seleccionar un usuario”. 2.1.2 Ir al paso 1.2 finalizando así el caso de uso. 2.2.1 El sistema detecta que el usuario seleccionado es “Administrador” y muestra un mensaje de: “Administrador es el usuario por defecto del sistema, no puede ser eliminado”. 2.2.2 Ir al paso 1.2 finalizando así el caso de uso.
Sección: Cambiar Contraseña	
Acción del Actor	Respuesta del Sistema
2. El Administrador selecciona el usuario, introduce la nueva contraseña y luego la confirma.	1.2 El sistema muestra una interfaz que contiene todos los usuarios, además de campos para introducir la nueva contraseña y luego confirmarla. 2.1 El sistema verifica que se haya seleccionado un usuario y que los otros campos se hayan llenado. 2.2 El sistema verifica que la contraseña introducida tenga 5 o más dígitos. 2.3 El sistema actualiza la nueva contraseña del usuario y muestra el siguiente mensaje de: “Se ha cambiado la contraseña correctamente”. 2.4 Ir al paso 1.2 finalizando así el caso de uso.
Flujo Alterno: Cambiar Contraseña	

Acción del Actor	Respuesta del Sistema
	2.1.2 El sistema reconoce que no se ha seleccionado ningún usuario y muestra el siguiente mensaje: “Debe seleccionar algún usuario”. En caso de que si haya sido seleccionado, pero algunos de los otros campos han quedado en blanco, el sistema muestra un mensaje de: “Debe llenar todos los campos”. 2.1.2 Ir al paso 1.2 finalizando así el caso de uso. 2.2.1 El sistema detecta que la contraseña introducida tiene menos de 5 dígitos y muestra un mensaje de: “Por medidas de seguridad la contraseña debe tener 5 o más caracteres”. 2.2.2 ir al paso 1.2 finalizando así el caso de uso.
Prioridad:	Crítico

Tabla 8: Descripción del CU: Actualizar Registro

Caso de Uso:	Actualizar Registro
Actores:	Administrador (Inicia)
Propósito:	Actualizar datos en el registro de log almacenado.
Resumen:	El caso de uso comienza cuando el Administrador accede al registro de log para actualizar los campos relacionados con la descripción del problema y la solución que éste propone.
Referencia:	R6.
Precondiciones:	Que se haya almacenado en la base datos el registro que se quiere actualizar.
Poscondiciones:	Se actualiza el registro de log seleccionado.
Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema
1. El Administrador selecciona la opción de: Actualizar Registro.	1.1El sistema muestra una interfaz con los campos necesarios para actualizar el registro

	seleccionado.
2. El Administrador inserta el número del evento que desea actualizar y luego escribe su descripción del problema y la posible solución.	2.1 El sistema verifica que el campo relacionado con el número de evento no esté vacio. 2.2 El sistema verifica que el número de evento insertado se encuentre en la base de datos. 2.3 El sistema comprueba que se haya llenado al menos uno de los campos relacionados con la descripción del administrador y la posible solución. 2.4 El sistema actualiza la base de datos y muestra un mensaje de: “Se ha actualizado correctamente el evento seleccionado”. 2.5 Ir al paso 1.1 finalizando así el caso de uso.
Flujos Alternos	
Acción del Actor	Respuesta del Sistema
	2.1.1 El sistema detecta que no se ha introducido ningún número de evento y muestra el siguiente mensaje: “Debe señalar un número de evento”. 2.1.2 Ir al paso 1.1 finalizando así el caso de uso. 2.2.1 El sistema comprueba que el número de evento no se encuentra almacenado en la base de datos y muestra un mensaje de: “No existe ese número de evento”. 2.2.2 Ir al paso 1.1 finalizando así el caso de uso. 2.3.1 El sistema reconoce que se ha introducido un número de evento que se encuentra en la base de datos, pero se ha dejado sin llenar los otros dos campos, por lo que se muestra el siguiente mensaje: “No se ha actualizado ninguna información”. 2.3.2 Ir al paso 1.1 finalizando así el caso de uso.

Prioridad:	Crítico

Tabla 9: Descripción del CU: Autenticar Usuario

Caso de Uso:	Autenticar Usuario	
Actores:	Usuario (Inicia)	
Propósito:	Permitir a los distintos usuarios acceder al sistema	
Resumen:	El caso de uso comienza cuando el usuario intenta acceder al sistema y este le muestra una interfaz que le permitirá entrar sus datos.	
Referencia:	R7.	
Precondiciones:	Que el usuario acceda al sistema.	
Poscondiciones:	El usuario accede al sistema.	
Flujo Normal de Eventos		
Acción del Actor		Respuesta del Sistema
1. El usuario intenta acceder a la aplicación. 2. El usuario inserta su usuario y contraseña.		1.1 El sistema muestra una interfaz que le permitirá introducir los datos necesarios para acceder al sistema 2.1 El sistema verifica que no hayan campos vacíos. 2.2 El sistema comprueba que el usuario esté en la base de datos. 2.3 El sistema verifica que la contraseña se corresponda con la del usuario que se esté autenticando 2.4 Se le da acceso al usuario al sistema y se les brinda los privilegios que le correspondan, terminado así el caso de uso.
Flujos Alternos		
Acción del Actor		Respuesta del Sistema
		2.1.1 El sistema detecta que el campo relacionado con el usuario no ha sido llenado y

	muestra el siguiente mensaje: “Debe autenticarse con un usuario”, en caso de que el campo relacionado con la contraseña no haya sido llenado muestra un mensaje de: “Por medidas de seguridad debe existir una contraseña”. 2.1.2 Ir al paso 1.1 finalizando así el caso de uso. 2.2.1 El sistema reconoce que el usuario no se encuentra en la base de datos y muestra un mensaje de: “Usuario no existe”. 2.2.2 Ir al paso 1.1 finalizando así el caso de uso. 2.3.1 El sistema reconoce que la contraseña introducida no se corresponde con la del usuario que se está autentificando y muestra un mensaje de: “Contraseña Incorrecta”. 2.3.2 Ir al paso 1.1 finalizando así el caso de uso.
Prioridad:	Crítico

Tabla 10: Descripción del CU: Mostrar Registro

Caso de Uso:	Mostrar Registro
Actores:	Usuario (Inicia).
Propósito:	Mostrar los registros de log almacenados en la base de datos así como las notificaciones realizadas por el sistema.
Resumen:	El Caso de Uso se inicia cuando el usuario selecciona una de las siguientes opciones: Inicio: Se muestra el estado de los dispositivos gestionados y el último registro de log del dispositivo con fallas. Registro de log: Muestra los registros de log correspondientes a los datos que introduce el usuario para realizar la búsqueda. Notificaciones: Muestra las notificaciones realizadas por el sistema correspondientes a los datos que introduce el usuario para realizar la búsqueda.

Referencia:	R8.1, R8.2, R8.3
Precondiciones:	Que haya registros de log y notificaciones almacenados en la base de datos.
Poscondiciones:	El sistema muestra los registros de log y las notificaciones.

Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema
1. El usuario selecciona realizar alguna de las opciones: - Inicio. - Registro de log. - Notificaciones.	1.1 El sistema, según la operación que se seleccionó, hace lo siguiente: - Si el usuario desea ver el estado de los dispositivos, ir a sección: “Inicio”. - Si el usuario decide buscar los registros de log almacenados, ir a sección: “Registro de log”. - Si el usuario decide buscar las notificaciones realizadas por el sistema, ir a sección: “Notificaciones”.
Sección: Inicio	
Acción del Actor	Respuesta del Sistema
2. El usuario selecciona un dispositivo.	1.2 Muestra la interfaz que contiene el estado de todos los dispositivos. 2.1 El sistema verifica que el estado del dispositivo sea “caído” y muestra el último registro de log almacenado en la base de datos referente a ese dispositivo. 2.2 Ir al paso 1.2 dando por culminado el caso de uso.
Flujo Alterno: Inicio	
Acción del Actor	Respuesta del Sistema
	2.1.1 El sistema reconoce que el estado del dispositivo seleccionado no es “caído” y muestra el siguiente mensaje: “Este equipo está funcionando correctamente”.

	2.1.2 Ir al paso 1.2 dando por culminado el caso de uso.
Sección: Registro de log	
Acción del Actor	Respuesta del Sistema
2. El usuario introduce los datos por el que desea realizar la búsqueda. 3. El usuario selecciona un campo de la tabla	1.2 El sistema muestra la interfaz que contiene los campos necesarios para realizar la búsqueda. 2.1 El sistema realiza una búsqueda de los registros de log que estén almacenados, cuyos parámetros correspondan con los datos introducidos por el usuario, luego muestra el resultado en una tabla. 3.1 El sistema verifica que se haya seleccionado un campo y lo muestra debajo de la tabla. 3.2 Finaliza el caso de uso.
Flujo Alterno: Registro log	
Acción del Actor	Respuesta del Sistema
	3.1.2 El sistema detecta que no se ha seleccionado ningún campo de la tabla y finaliza así el caso de uso.
Sección: Notificaciones	
Acción del Actor	Respuesta del Sistema
2. El usuario introduce los datos por el que desea realizar la búsqueda. 3. El usuario selecciona un campo de la tabla.	1.2 El sistema muestra la interfaz que contiene los campos necesarios para realizar la búsqueda. 2.1 El sistema realiza una búsqueda de las notificaciones que se han realizado, cuyos parámetros correspondan con los datos introducidos por el usuario, luego muestra el resultado en una tabla. 3.1 El sistema verifica que se haya seleccionado

	un campo y lo muestra debajo de la tabla. 3.2 finaliza el caso de uso.
Flujo Alterno: Notificaciones	
Acción del Actor	Respuesta del Sistema
	3.1.2 El sistema detecta que no se ha seleccionado ningún campo de la tabla y finaliza así el caso de uso.
Prioridad:	Crítico

Tabla 11: Descripción del CU: Gestionar Notificante.

Caso de Uso:	Gestionar Notificantes
Actores:	Administrador (Inicia).
Propósito:	Permite adicionar, mostrar y eliminar los usuarios a los cuales se les desea informar.
Resumen:	El Caso de Uso se inicia cuando el administrador selecciona una de las siguientes opciones: Adicionar Notificante: El Administrador selecciona que desea realizar la notificación por correo electrónico, escribe la dirección, si se desea informar por beeper el Administrador selecciona esta opción e introduce el número, finalmente el sistema muestra el siguiente mensaje: "Se notificará a:'direcion de correo electrónico' " y se agrega a la base de datos. Eliminar Notificante: El sistema muestra todas las direcciones de correo, el Administrador selecciona la que desea eliminar, el sistema muestra el siguiente mensaje: "Usuario eliminado correctamente" y elimina dicha dirección de la base de datos.
Referencia:	R9.1, R9.2
Precondiciones:	El usuario debe estar autenticado como Administrador.
Poscondiciones:	El sistema adiciona, muestra y elimina las direcciones de correo electrónico a las cuales se desea notificar.
Flujo Normal de Eventos	

Acción del Actor	Respuesta del Sistema
1. El Administrador selecciona realizar alguna de las opciones: - Adicionar Notificante. - Eliminar Notificante.	1.1 El sistema, según la operación que se seleccionó, hace lo siguiente: - Si el Administrador desea adicionar una dirección de correo electrónico o un número de beeper, ir a sección: “Adicionar Notificante”. - si el Administrador decide eliminar alguna dirección de correo, ir a sección: “Eliminar Notificante”.
Sección: Adicionar Notificante.	
Acción del Actor	Respuesta del Sistema
2. El Administrador selecciona que desea realizar la notificación por correo electrónico y por beeper, luego introduce los datos requeridos	1.2 Muestra la interfaz que permite adicionar la dirección de correo electrónico y el número de beeper. 2.1 El sistema verifica que no hayan campos vacios. 2.2 El sistema verifica que la dirección de correo electrónico este correcta. 2.3 El sistema adiciona la dirección de correo electrónico, el número de beeper y muestra un mensaje de: “Se notificará a:’direcion de correo electrónico’ ”. 2.4 Ir al paso 1.2.
Flujo Alterno: Adicionar Notificante	
Acción del Actor	Respuesta del Sistema
	2.1.1 El sistema detecta que no se ha introducido una dirección de correo electrónico y muestra el siguiente mensaje: “Debe introducir una dirección de correo electrónico”. En caso de que este campo se haya llenado correctamente pero no se ha introducido ningún número de beeper se muestra un mensaje de: “El usuario a notificar no

	presenta beeper". 2.1.2 Ir al paso 1.2. 2.2.1 El sistema comprueba que la dirección de correo electrónico no es correcta y muestra un mensaje de: "Debe escribir una dirección de correo válida". 2.2.2 ir al paso 1.2
Sección: Eliminar Notificante	
Acción del Actor	Respuesta del Sistema
2. El Administrador selecciona la dirección a eliminar.	1.2 El sistema muestra la interfaz que contiene todas las direcciones de correo electrónico. 2.1 El sistema verifica que se haya seleccionado algún usuario. 2.2 Elimina el usuario de la base de datos y muestra un mensaje de: "Usuario eliminado correctamente". 2.3 Ir al paso 1.2.
Flujo Alterno: Eliminar Notificante	
Acción del Actor	Respuesta del Sistema
	2.1.2 El sistema detecta que no se ha seleccionado ningún usuario y muestra un mensaje de: "Debe seleccionar un usuario". 2.1.2 Ir al paso 1.2.
Prioridad:	Crítico

2.9 Conclusiones

Para una mejor comprensión del problema, en este capítulo se identificaron conceptos, que fueron relacionados mediante un diagrama de modelo de dominio. En el mismo se muestran de forma general los distintos objetos existentes en el negocio. Además se describieron los requisitos que el sistema debe cumplir para su correcto y eficaz funcionamiento, analizando así las distintas funcionalidades del mismo, determinadas en 9 casos de uso, de los cuales se realizó una descripción detallada que permite un mayor entendimiento acerca de lo que debe realizar el sistema.

Capítulo 3 Análisis y Diseño del Sistema

3.1 Introducción.

Este capítulo se refiere a lo relacionado con traducir los requisitos a una descripción de cómo desarrollar el sistema, se muestra el modelado visual de su desarrollo, se presenta la arquitectura, se adapta el diseño para hacerlo corresponder con el ambiente de implementación, se muestra el modelo físico de la base de datos y se realiza el modelo de despliegue.

3.2 Modelo de Análisis.

Luego de haberse definido los requerimientos funcionales del sistema los diagramas de clases del análisis pasan a dar una descripción más específica de lo que es la estructura de las clases que lo conforman, igual quedan definidas las relaciones existentes entre ellas. El Modelo de Análisis ayuda a descomponer la implementación en partes más sencillas que puedan ser desarrolladas por diferentes equipos. En el caso particular del desarrollo de nuestro sistema no se consideró relevante la idea de realizar el modelo de análisis, pues ya se tiene experiencia en el modelado con UML y se decidió como mejor opción dedicar el esfuerzo en el diseño de las clases que se iban a implementar.

3.3 Modelo de Diseño.

"El modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar. Sirve de abstracción de la implementación y es utilizada como entrada fundamental de las actividades de implementación" (8).

3.3.1 Arquitectura.

"La arquitectura de software de un sistema de programas o computación es la estructura de las estructuras del sistema, la cual comprende los componentes del software, las propiedades de esos componentes visibles externamente y las relaciones entre ellos" (13). La arquitectura no es un software operacional, en otras palabras es la representación que provee al ingeniero se software de analizar la efectividad del diseño para la consecución de los requisitos fijados, además permite analizar las diferentes variantes arquitectónicas y ayuda a disminuir los posibles riesgos que pudiera correr el desarrollo de software

3.3.1.1 Patrón Modelo Vista Controlador.

Es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos:

Modelo es el objeto que representa los datos del programa. Maneja los datos y controla todas sus transformaciones. El Modelo no tiene conocimiento específico de los Controladores o de las Vistas, ni siquiera contiene referencias a ellos. Es el propio sistema el que tiene encomendada la responsabilidad de mantener enlaces entre el Modelo y sus Vistas, y notificar a las Vistas cuando cambia el Modelo.

Vista es el objeto que maneja la presentación visual de los datos representados por el Modelo. Genera una representación visual del Modelo y muestra los datos al usuario. Interactúa con el Modelo a través de una referencia al propio Modelo.

Controlador es el objeto que proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el Modelo. Cuando se realiza algún cambio, entra en acción, bien sea por cambios en la información del Modelo o por alteraciones de la Vista. Interactúa con el Modelo a través de una referencia al propio Modelo.

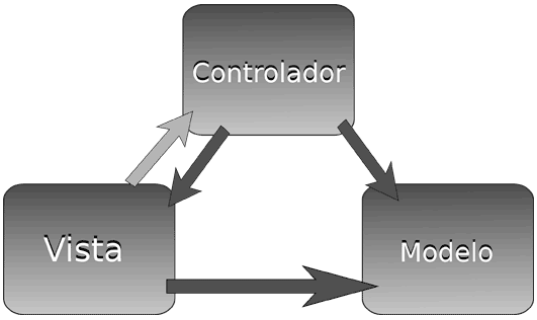


Fig. 6 Modelo Vista Controlador.

El modelo no tiene que acceder ni a la vista ni al controlador, sin embargo la vista debe tener la posibilidad de acceder al modelo (para representarlo) y al controlador (para enviar las peticiones que hace el usuario). El controlador ha de acceder al modelo (para conocer los datos y eventualmente pedir el cambio de estos) y a la vista para indicar los cambios en los datos.

Presenta algunos inconvenientes, como son:

- La complejidad va creciendo más rápido que el tamaño de la aplicación.
- Problemas al conectar las distintas capas.
- Acceso ineficiente, pues es necesario varias llamadas al modelo para actualizar los datos.
- La vista y el controlador son específicos para una plataforma.

3.3.1.2 Arquitectura 3 Capas

La arquitectura de tres capas se basa en la división en el nivel de acceso a datos, nivel de lógica de negocio y nivel de presentación o aplicación.

Capa de presentación: esta es la que el usuario puede ver en su ordenador, es donde se tratan los datos que se van a mostrar. Se intenta que en esta capa haya el mínimo de procesamiento. Esta capa se comunicará solamente con la capa de negocio.

Capa de negocios: en esta capa esta la lógica, se recibe las peticiones del usuario, y tras ejecutar una acción se le envía las respuestas del proceso. Esta capa se comunica como se ha dicho con la de presentación, la cual le envía peticiones y esta le responde con los resultados. Y también se comunica con la capa de datos, para pedirle datos.

Capa de datos: es donde se accede a los datos. Se hace referencia a uno o más gestores de BD que realizan el almacenamiento, modificación y consulta de los datos. Recibe peticiones desde la capa de negocios.

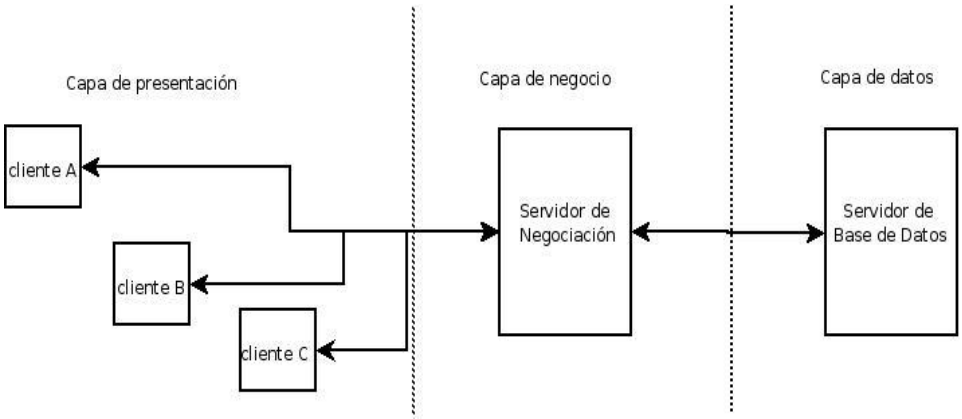


Fig. 7 Arquitectura 3 Capas

3.3.1.3 Justificación de la arquitectura seleccionada: 3 Capas.

Después de haber realizado un análisis y estudio basado en la arquitectura de 3 capas, se alega que esta es la adecuada para la implementación del sistema debido a las diferentes características que se presentan a continuación:

Brinda reusabilidad, pues por ejemplo podría ocurrir la necesidad de cambiar de repositorio de datos, estos cambios se concentrarían en la capa de acceso de datos, pero no en las otras dos.

Mejor calidad en los sistemas, como las aplicaciones son construidas en unidades separadas, estas pueden ser probadas independientemente y con mucho más detalle, esto conduce a obtener un producto mucho más sólido.

Permite el desarrollo de aplicaciones robustas debido al encapsulamiento.

La separación de roles en tres capas, permite que se haga de una forma sencilla el proceso de reemplazar o modificar una capa sin afectar a los módulos restantes..

También el modelo de 3 capas propone un ambiente para la construcción y ejecución de aplicaciones de avanzada que muy probablemente reemplazará a los sistemas actuales y además permite la modificación del sistema en períodos de tiempo reducidos, incluso cuando es necesario agregar características especiales a las aplicaciones.

Los ambientes de tres capas pueden incrementar el tráfico en la red y requerir más balance de carga y tolerancia a las fallas.

En el sistema se identificaron dentro de la capa presentación las siguientes clases: Principal, PModificar_Registros, Autenticación, PAdicionar_Equipo, PEliminar_Equipo, PNotificar, PEliminar_Notificante, PAdicionar_Usuario, PEliminar_Usuario, PCambiar_Contraseña, PInicio, PBuscar_Notificacion, PRegistro_Log. Dentro de la capa de negocio: Control_Monitoreo, Monitorear_Dispositivo, JDBC_Adaptado. Finalmente, dentro de la capa de datos se identificó la clase Acceso_BaseDatos.

3.3.2 Patrones de diseño.

Los patrones de diseño constituyen la base para la búsqueda de soluciones a situaciones o problemas comunes que podrían aparecer en el desarrollo de software. Estos son capaces de identificar clases, instancias, colaboraciones, roles y responsabilidades. Los patrones de diseño tienen como objetivo principal capturar buenas prácticas que brindan la posibilidad de mejorar la calidad del diseño de sistemas, definiendo objetos que soporten roles útiles en un contexto específico.

"Un patrón de diseño es:

- **una solución estándar para un problema común de programación**
- **una técnica para flexibilizar el código haciéndolo satisfacer ciertos criterios**
- **un proyecto o estructura de implementación que logra una finalidad determinada**
- **un lenguaje de programación de alto nivel**
- **una manera más práctica de describir ciertos aspectos de la organización de un programa"** (8)

3.3.2.1 Patrones GRASP.

"Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones" (8).

En el desarrollo del sistema se aplicaron los patrones básicos GRASP:

-**Experto:**

Su función es asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad.

-Creador

Su función es asignarle a la clase B la responsabilidad de crear una instancia de clase A. B es un creador de los objetos A.

-Alta Cohesión

Su función es asignar una responsabilidad de modo que la cohesión siga siendo alta. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen mucho trabajo

-Bajo Acoplamiento

Su función es asignar una responsabilidad para mantener bajo acoplamiento. Las clases deben comunicarse con un número pequeño de clases tanto como sea posible.

-Controlador

Asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase que represente características particulares como por ejemplo la de ser un sistema global.

3.3.2.2 Patrones GoF. Patrón Observador.

Dentro de los patrones GoF se hizo un estudio del patrón Observador, el cual es usado en programación para observar el estado de un objeto en un programa. **"Define una dependencia uno- a-muchos entre objetos, de forma que cuando un objeto cambia de estado se notifica y se actualizan automáticamente todos los objetos"** (8).

Al objeto que notifica a los demás de su cambio de estado se le llama sujeto y a los objetos que se actualizan al ser notificados de los cambios del sujeto se les llama observadores.

Entre los principales objetivos se encuentran:

1. Tratar de reducir al mínimo el acoplamiento entre las clases a las que pertenecen los objetos sujetos y observadores para así aumentar la reusabilidad de dichas clases.

   Para ello se utilizan las clases abstractas Sujeto y Observador que crean un primer nivel de abstracción en el que se incluyen todas las dependencias entre clases para que las clases concretas que heredan de ellas sean lo más independientes posibles.
2. Brindar la posibilidad de presentar un número ilimitado de objetos observadores que están pendientes a un cierto objeto sujeto.

   Para ello la clase abstracta Sujeto presenta una lista de objetos observadores a los que notificar en caso de que suceda algún evento relevante.

Sus características de aplicabilidad se pueden apreciar a la hora de que cuando un objeto requiere cambiar otros y no es conocido cuántos objetos necesitan cambiarse y ademas cuando un objeto deberia ser capaz de notificar a otros sin hacer suposiciones sobre quiénes son dichos objetos. Es decir no se quiere que los objetos estén fuertemente acoplados.

3.3.2.3 Justificación del Patrón seleccionado: Observador.

Fue aplicado este patrón de diseño por sus características, pues corresponden al desarrollo del sistema, por ejemplo es usado como un detector de eventos, lo cual es una propiedad muy interesante en términos del desarrollo de aplicaciones en tiempo real. UCI-Alert ante el cambio de estado de un dispositivo, provocado por la ocurrencia de un evento en el proceso de monitoreo en tiempo real, entre sus funciones debe realizar distintas acciones que se ejecutan atendiendo a la notificación de dicho cambio de estado.

3.3.3 Diagramas de Clases y de Interacción del Diseño.

Los diagramas de clases del diseño representan a las clases del diseño y las relaciones entre ellas. Las clases del diseño constituyen una abstracción de una o varias clases en la implementación del sistema, dependiendo del lenguaje de programación. Los diagramas de interacción representan una relación entre varios objetos así como los mensajes que puedan enviarse entre ellos, los diagramas de secuencia representan un tipo de diagrama de interacción en el cual se destaca el orden temporal de los mensajes.

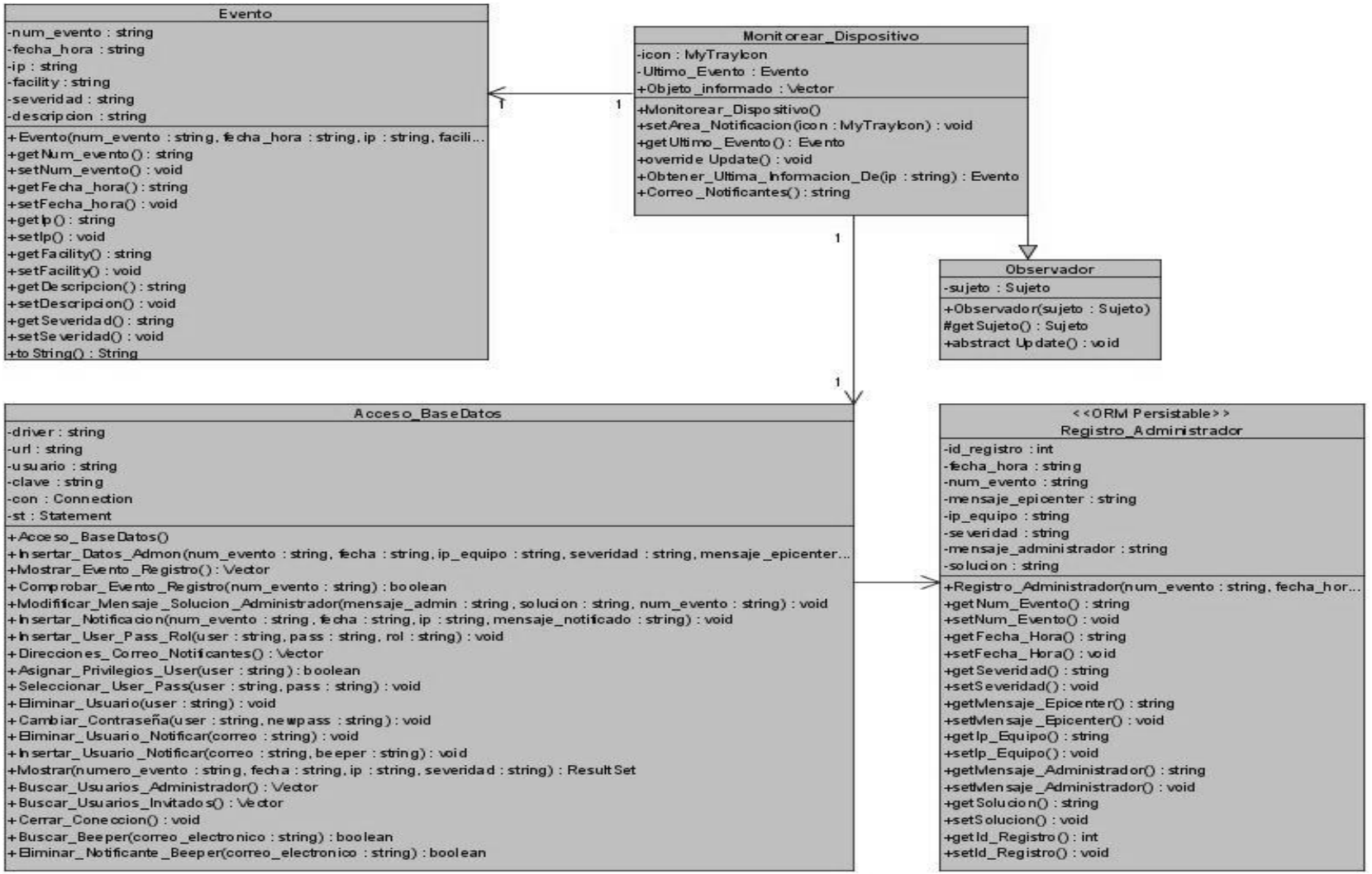


Fig. 8 Diagrama de Clases del Diseño: Almacenar Registro de Fallas.

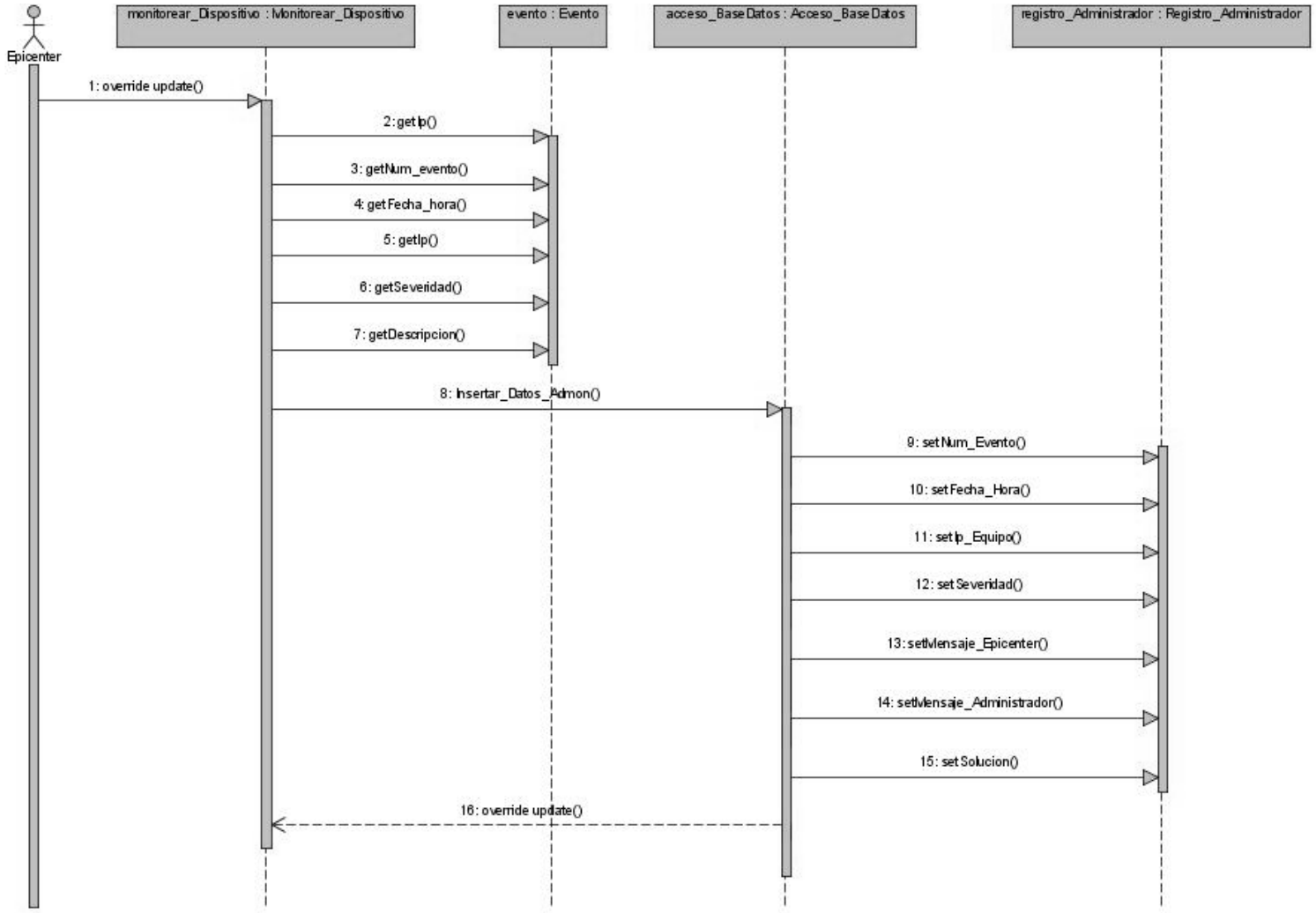


Fig. 9 Diagrama de Secuencia: Almacenar Registro de Fallas.

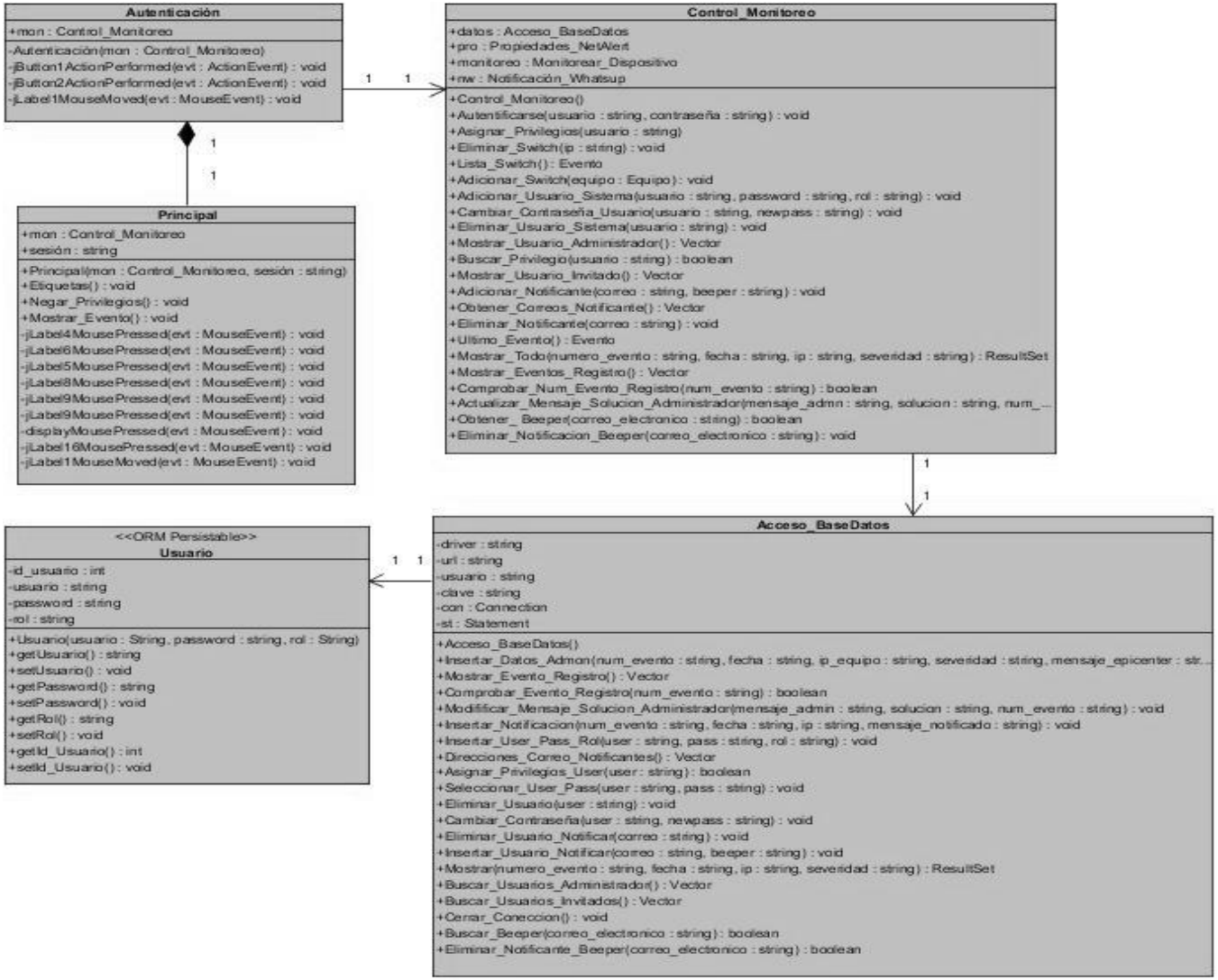


Fig. 10 Diagrama de Clases del Diseño: Autenticar Usuario.

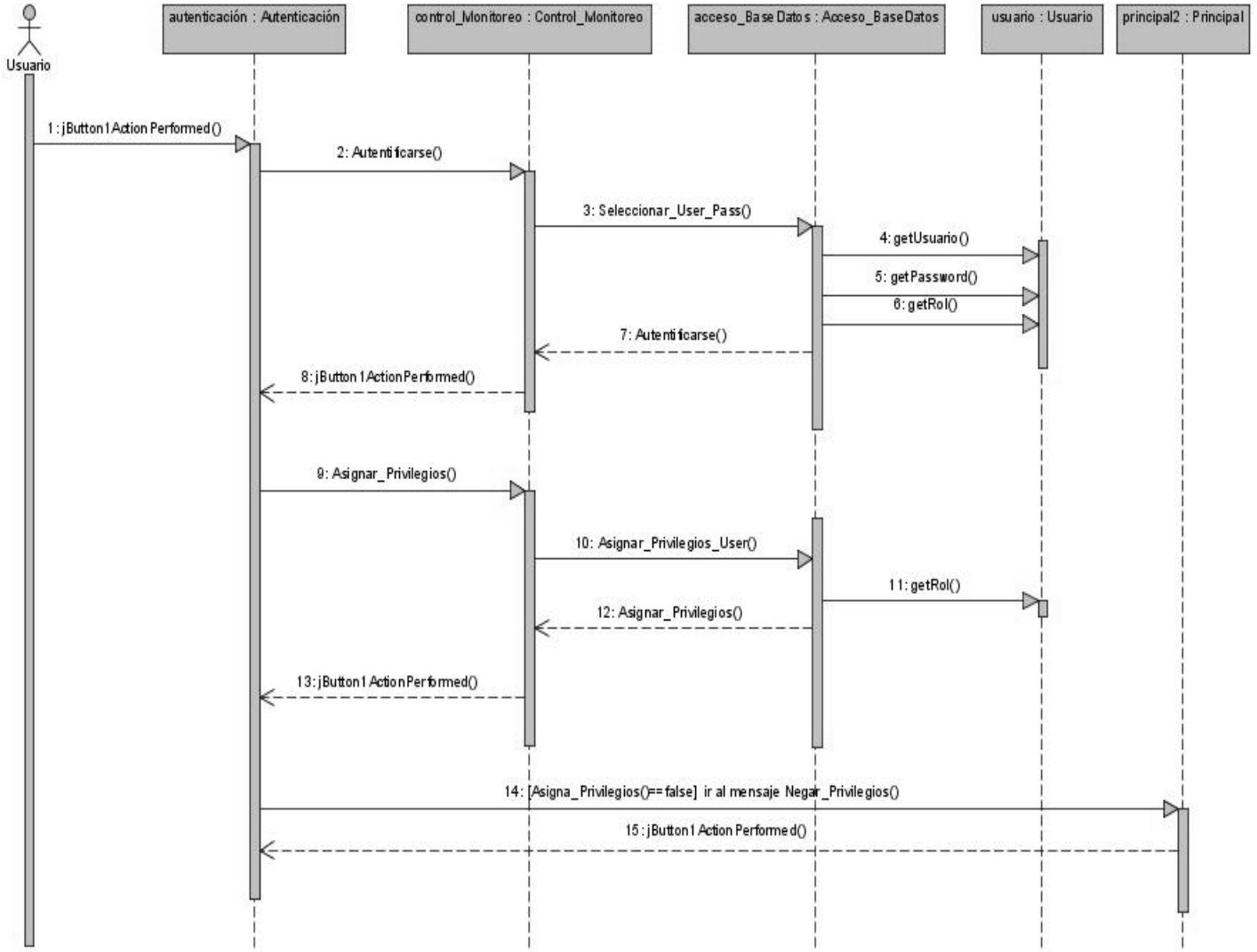


Fig. 11 Diagrama de Secuencia: Autenticar Usuario.

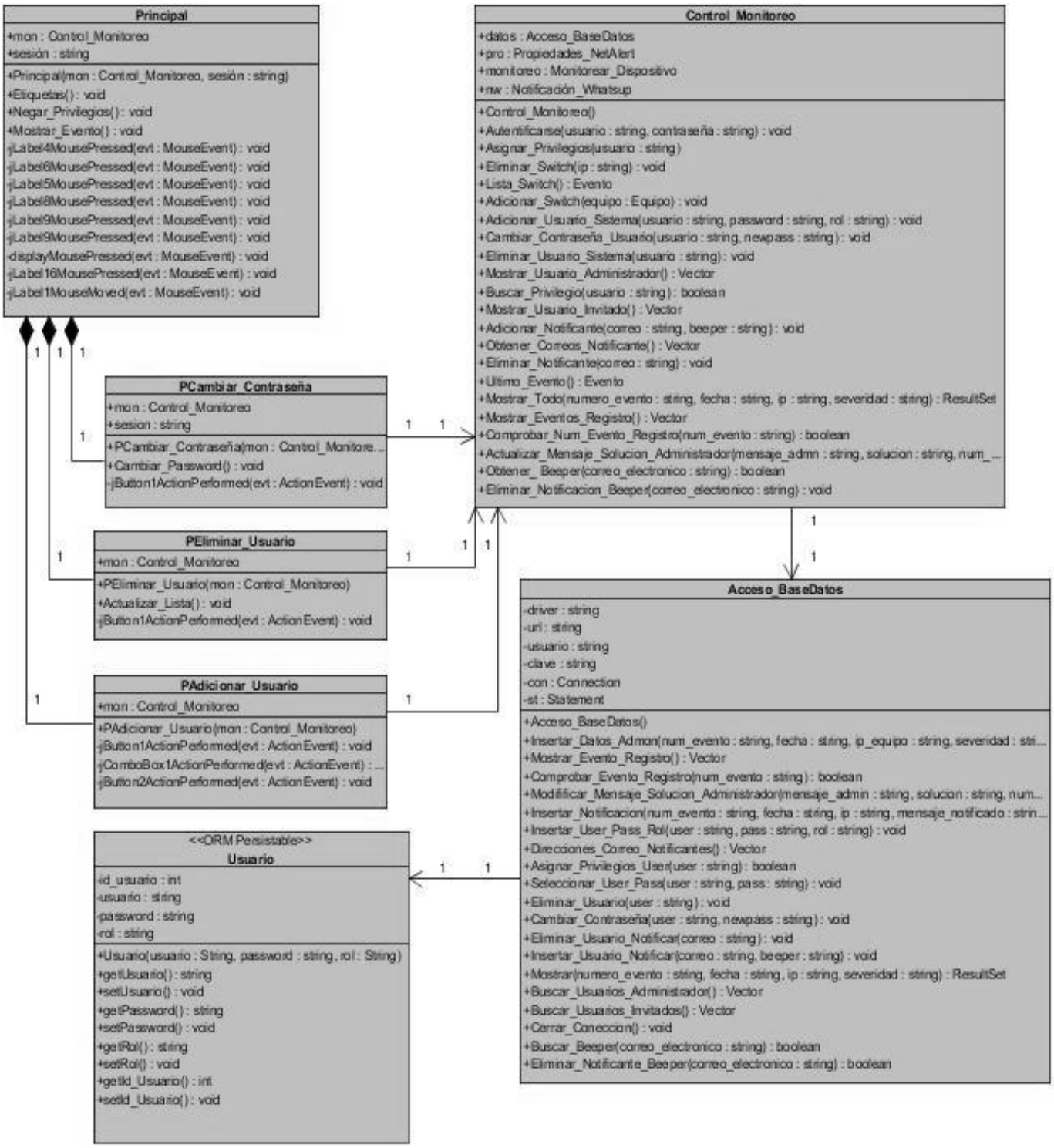


Fig. 12 Diagrama de Clases del Diseño: Gestionar Usuario.

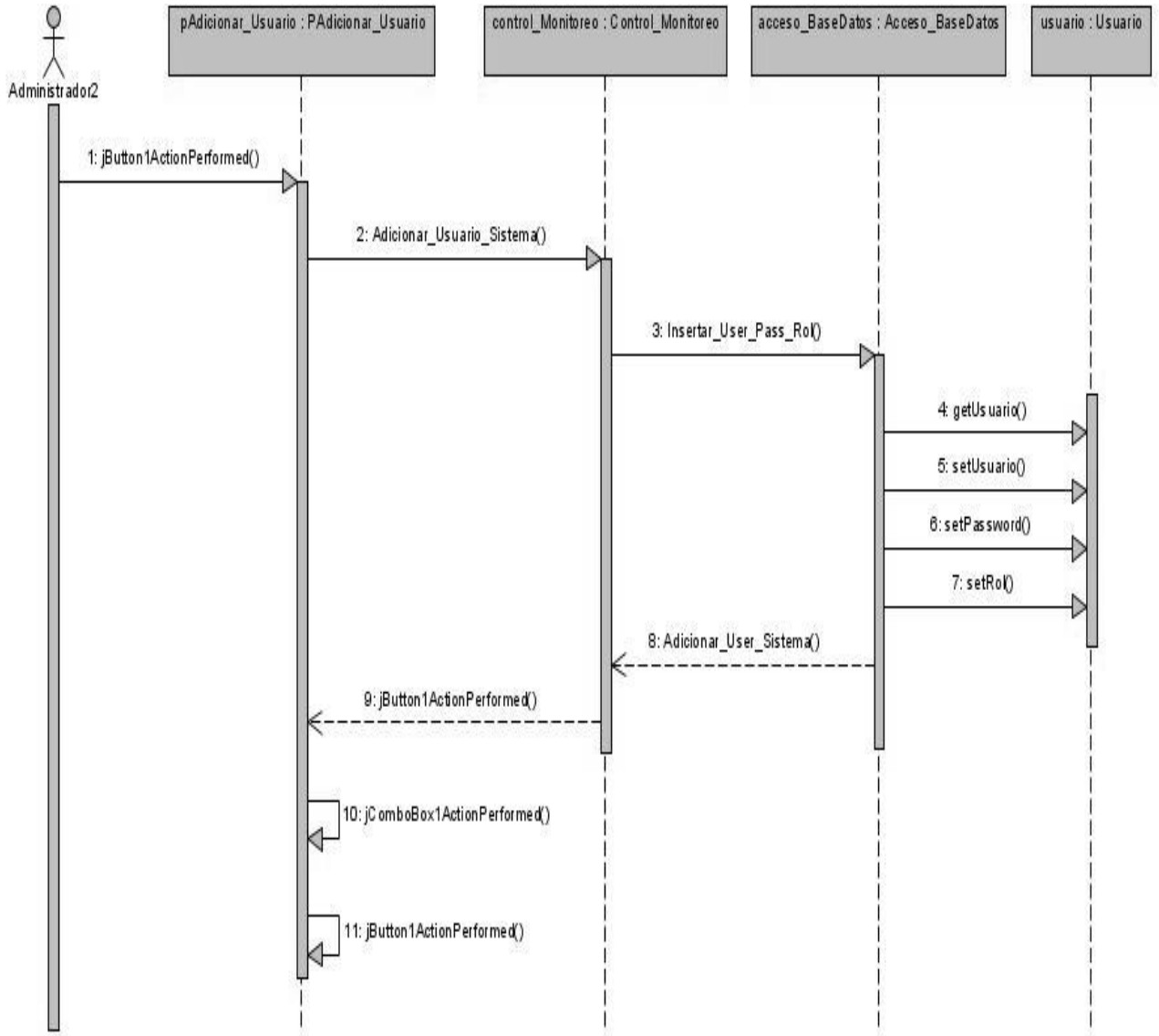


Fig. 13 Diagrama de Secuencia: Adicionar Usuario.

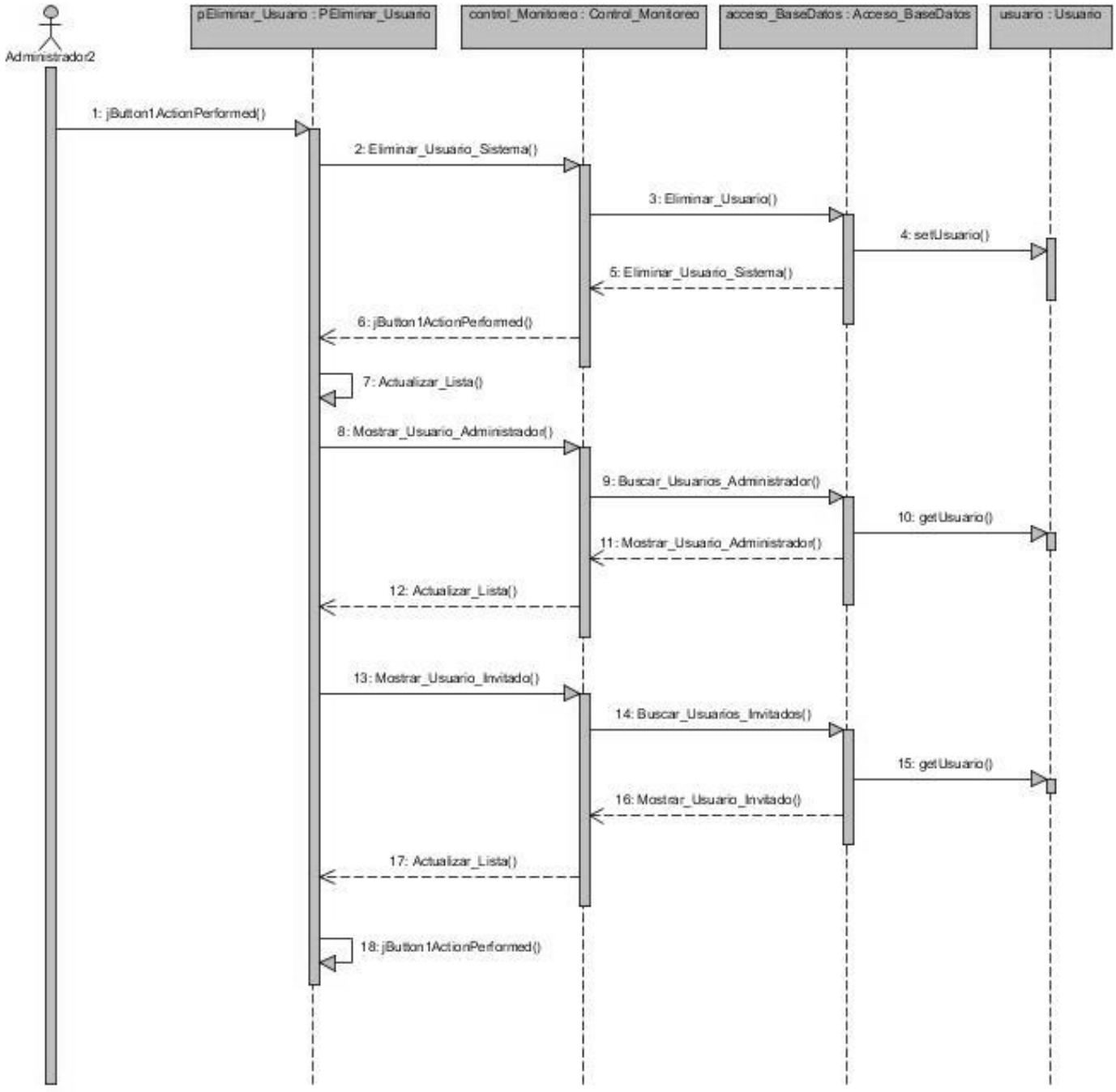


Fig. 14 Diagrama de Secuencia: Eliminar Usuario.

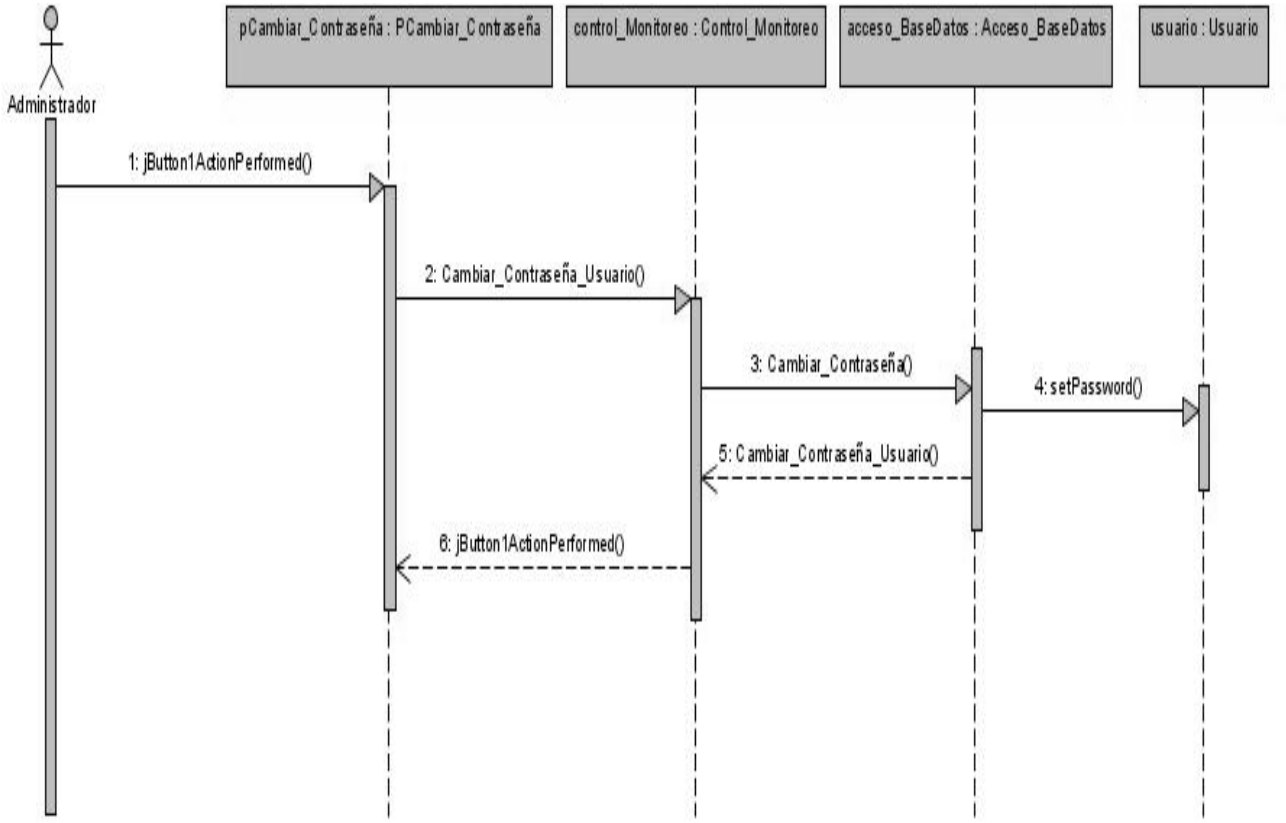


Fig. 15 Diagrama de Secuencia: Cambiar Contraseña.

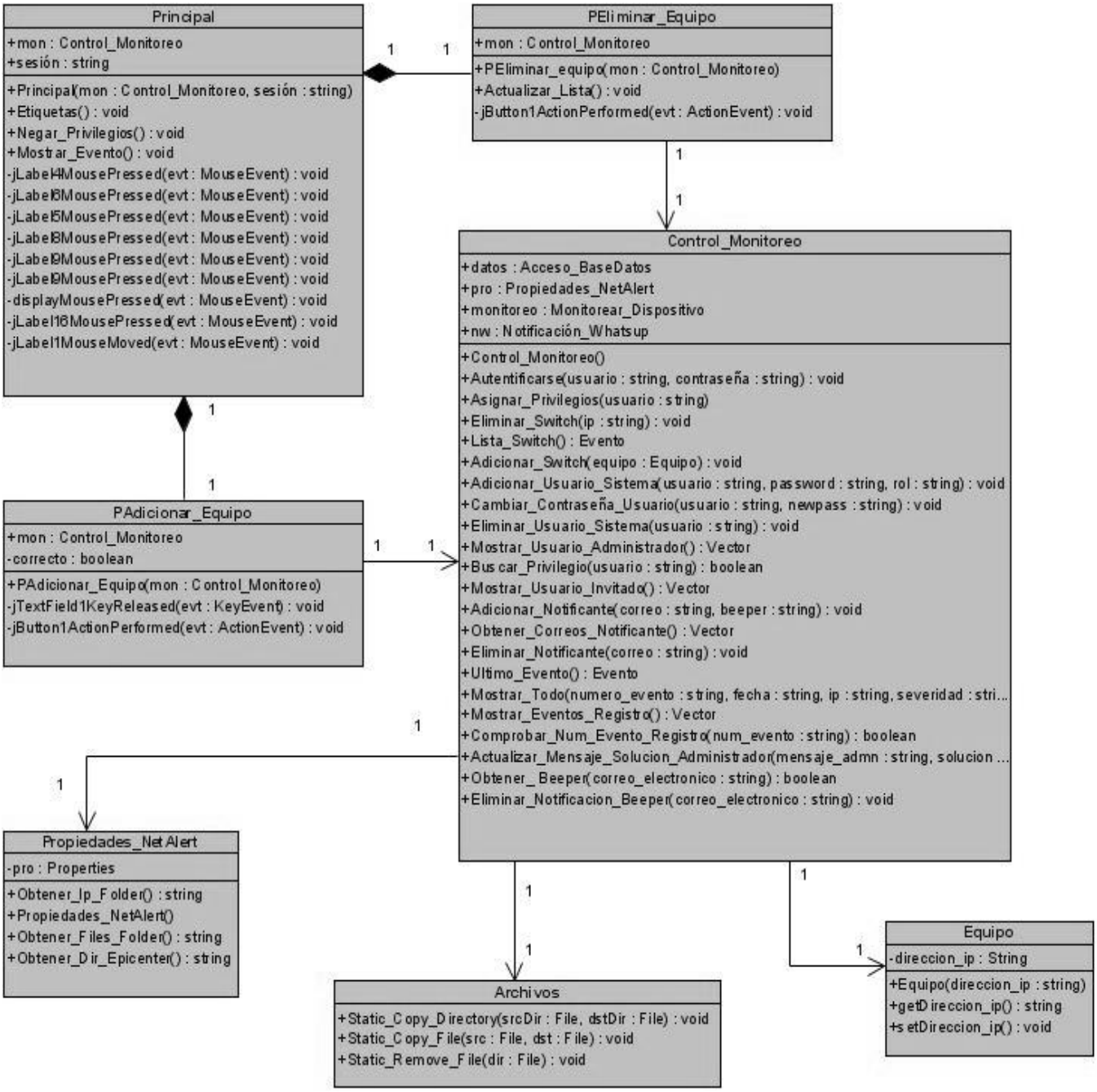


Fig. 16 Diagrama de Clases del Diseño: Gestionar Equipo.

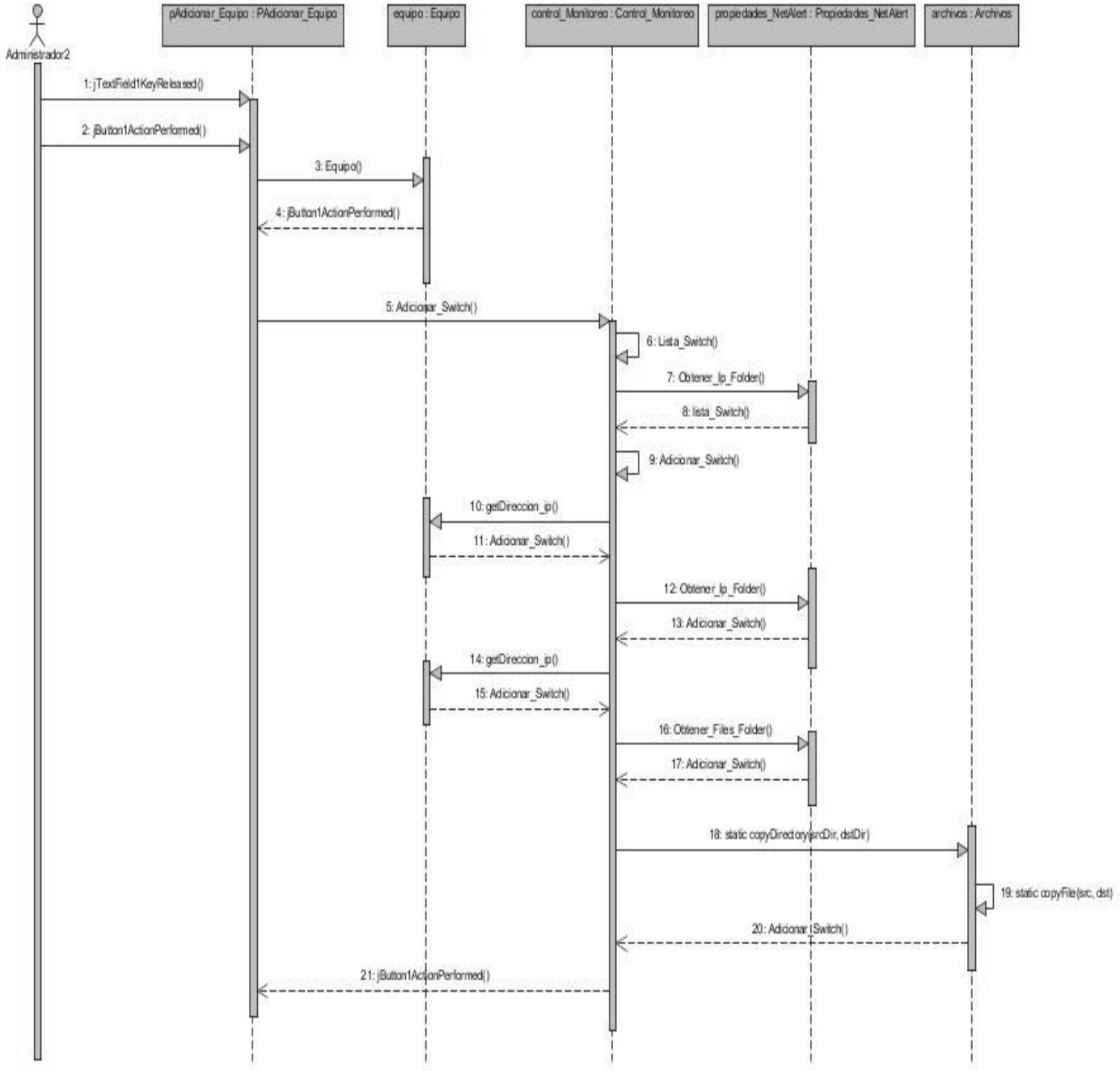


Fig. 17 Diagrama de Secuencia: Adicionar Equipo.

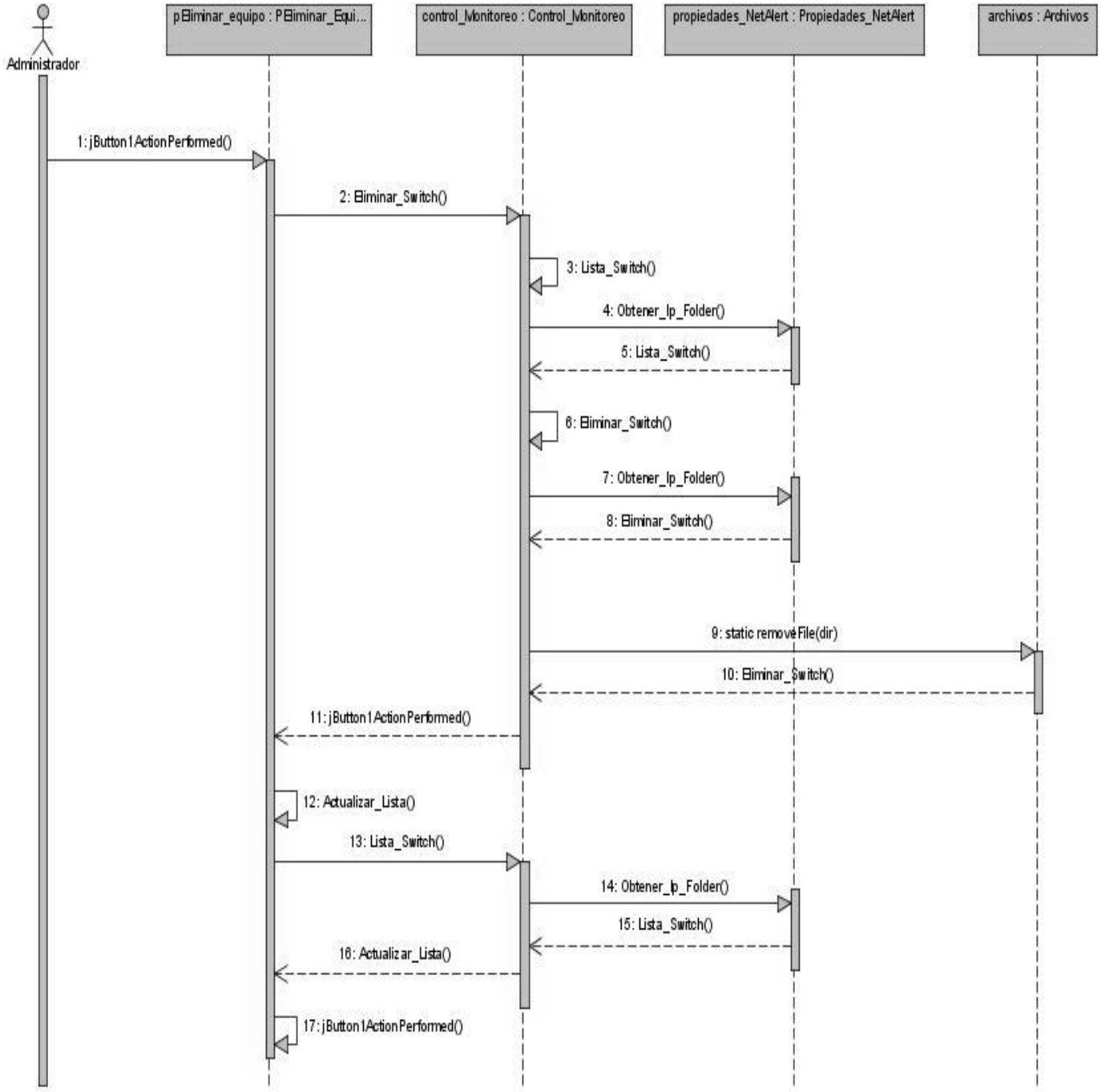


Fig. 18 Diagrama de Secuencia: Eliminar Equipo.

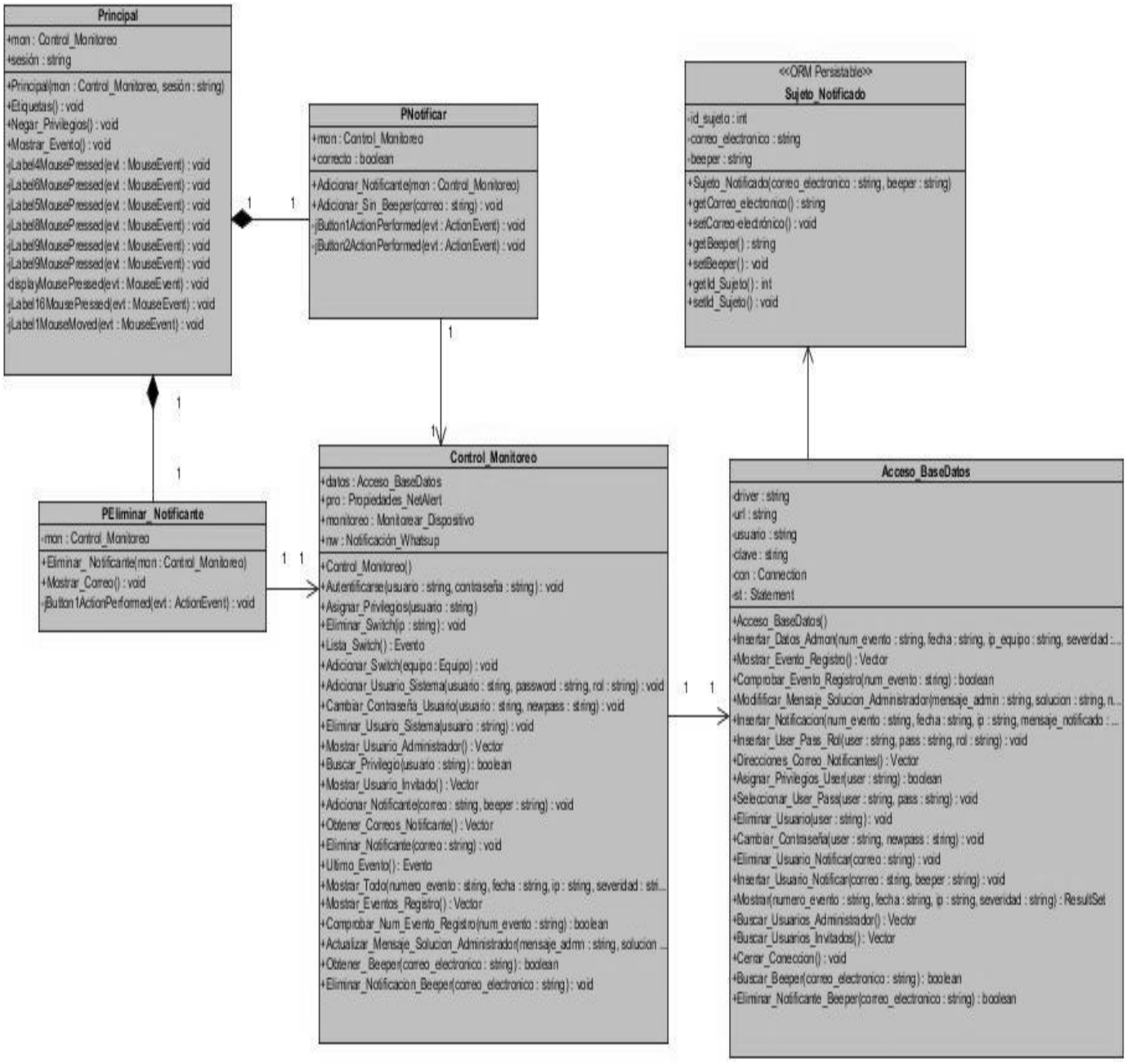


Fig. 19 Diagrama de Clases del Diseño: Gestionar Notificante.

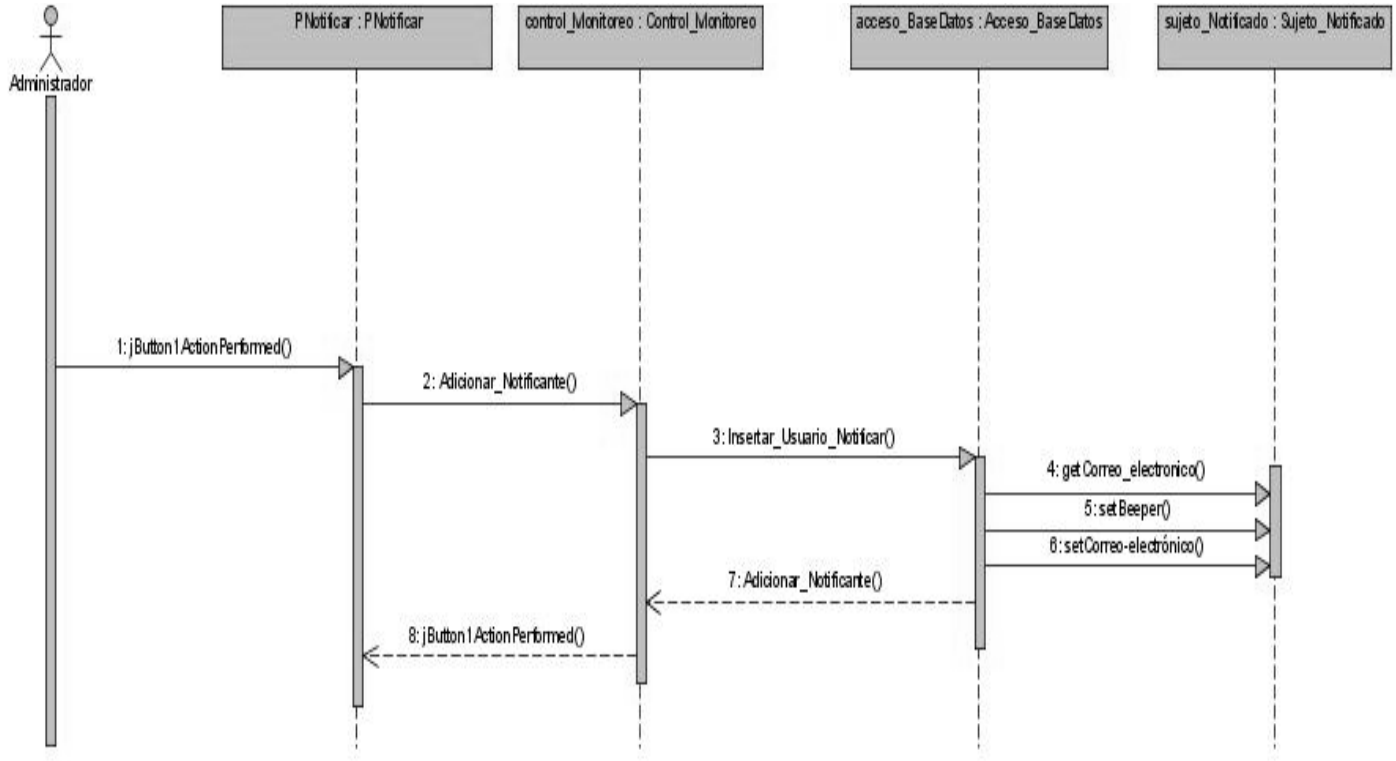


Fig. 20 Diagrama de Secuencia: Adicionar Notificante.

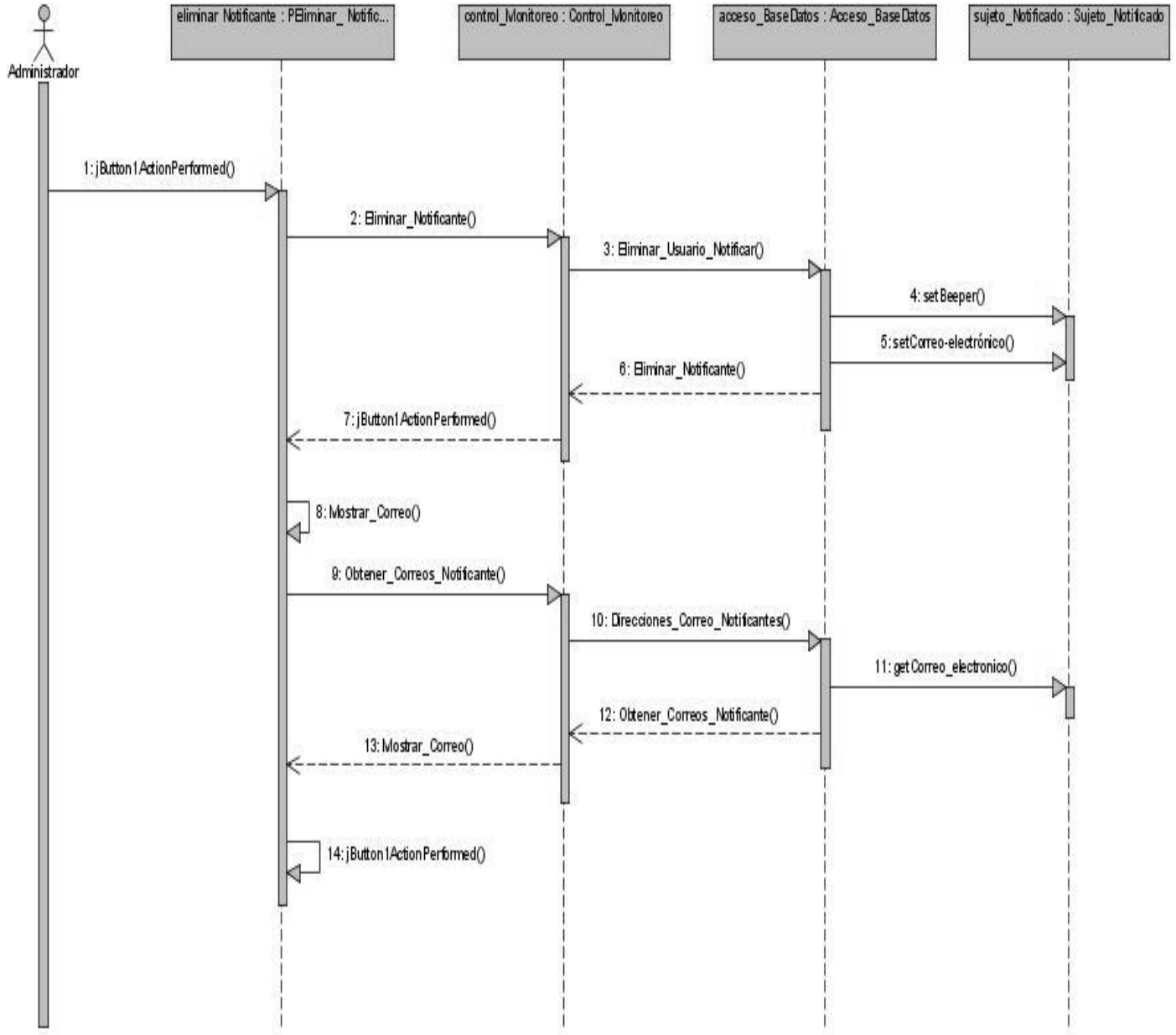


Fig. 21 Diagrama de Secuencia: Eliminar Notificante.

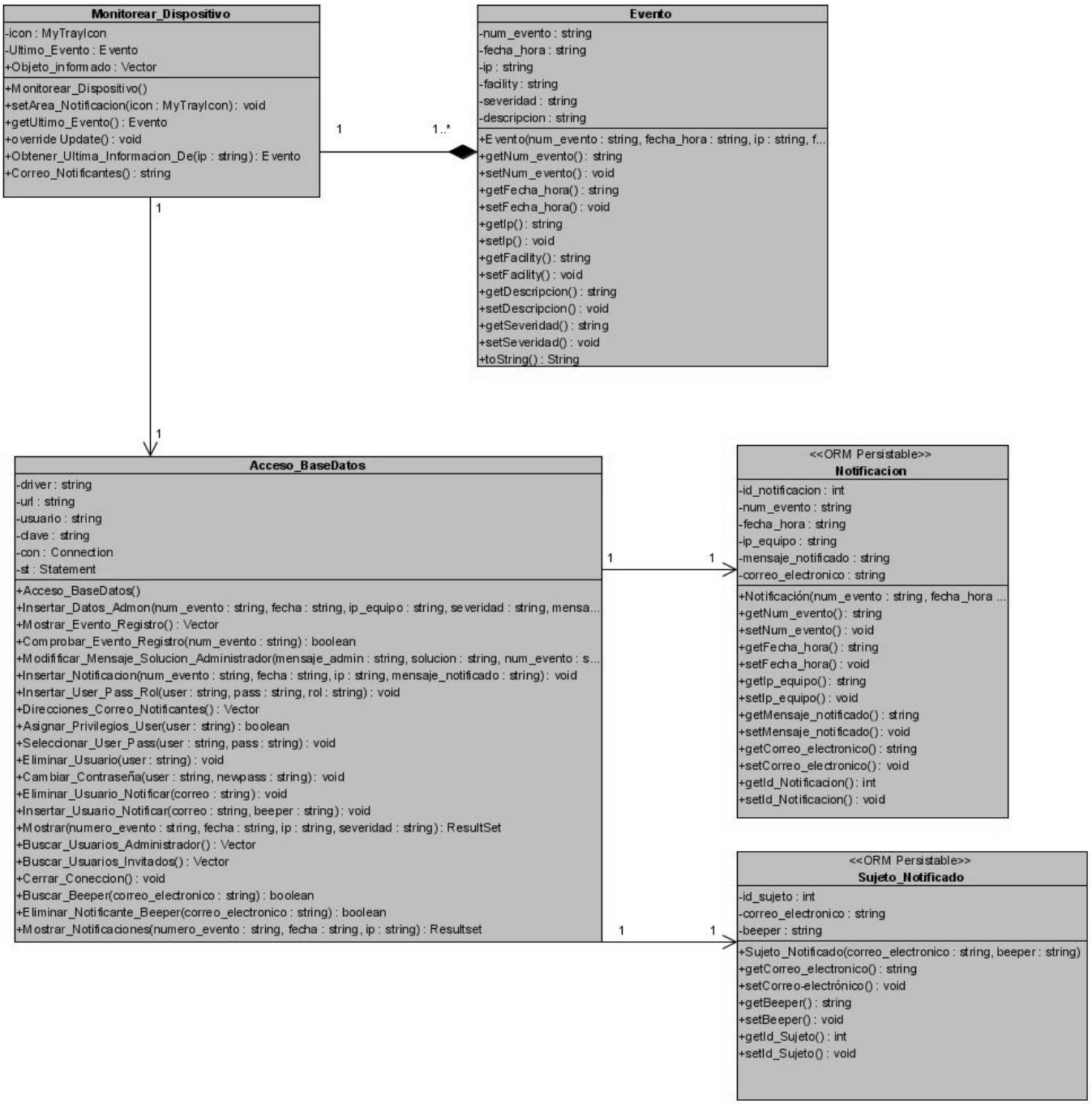


Fig. 22 Diagrama de Clases del Diseño: Registrar Notificación.

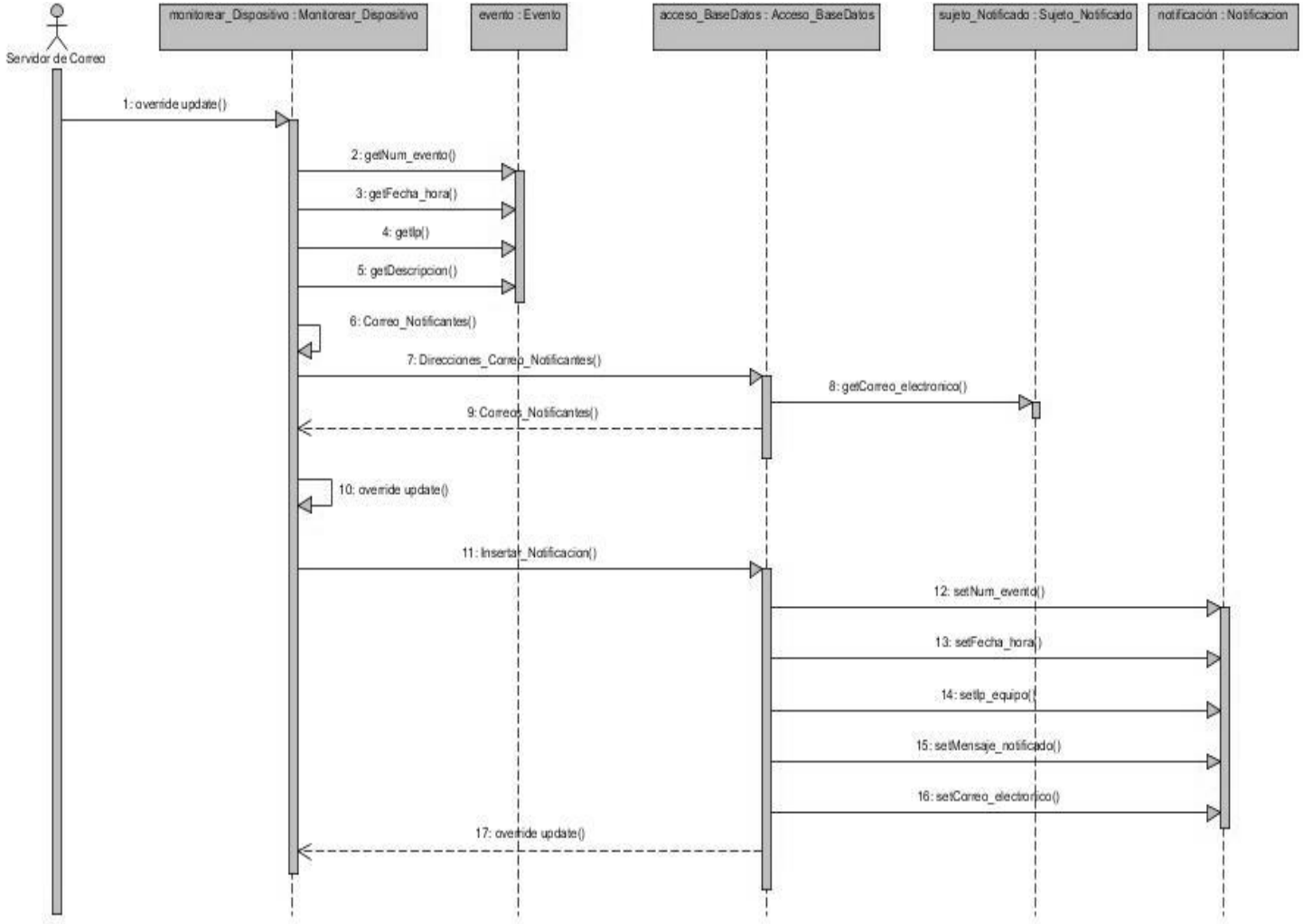


Fig. 23 Diagrama de Secuencia: Registrar Notificación.

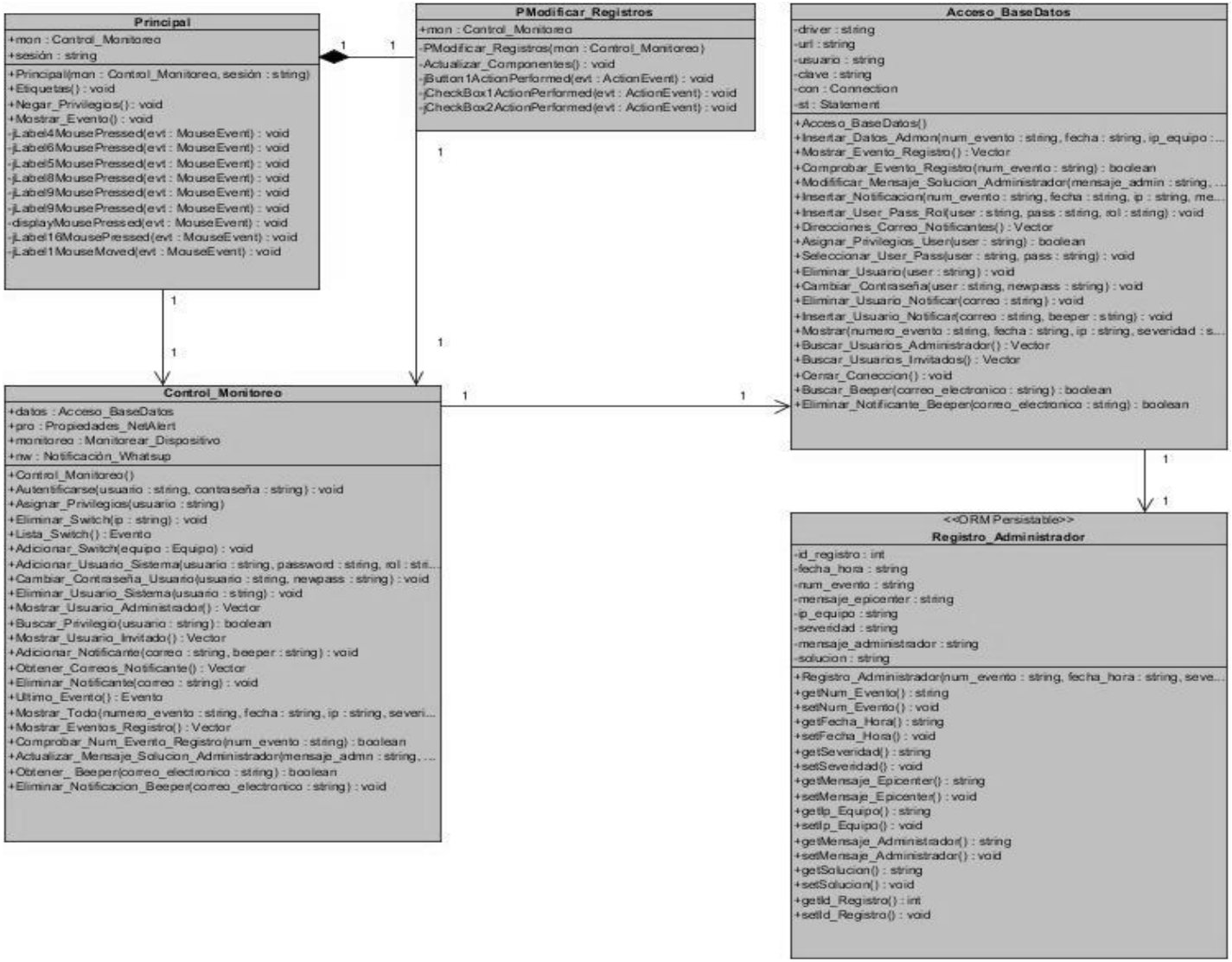


Fig. 24 Diagrama de Clases del Diseño: Actualizar Registro.

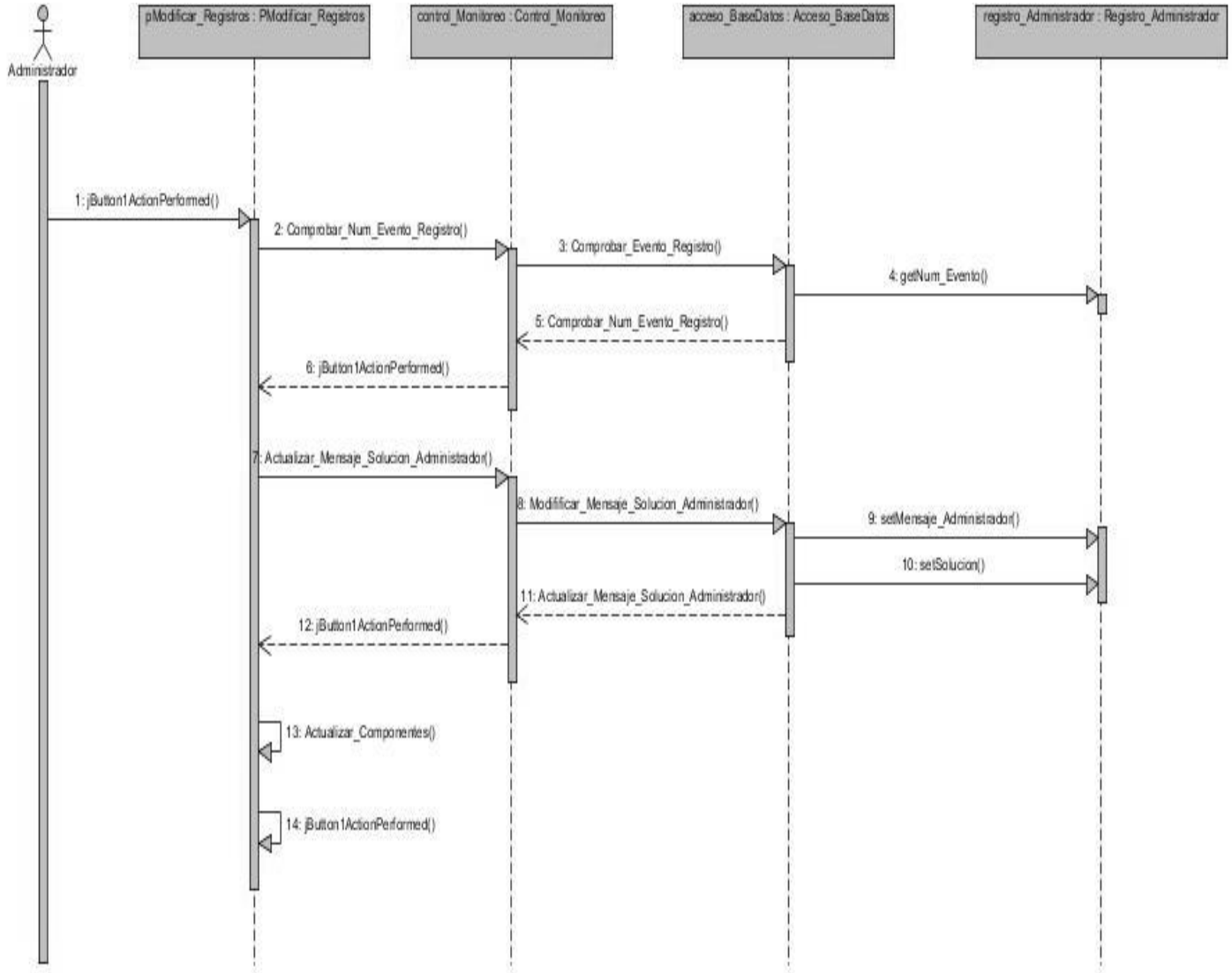


Fig. 25 Diagrama de Secuencia: Actualizar Registro.

Fig. 26 Diagrama de Clases del Diseño: Mostrar Registro.

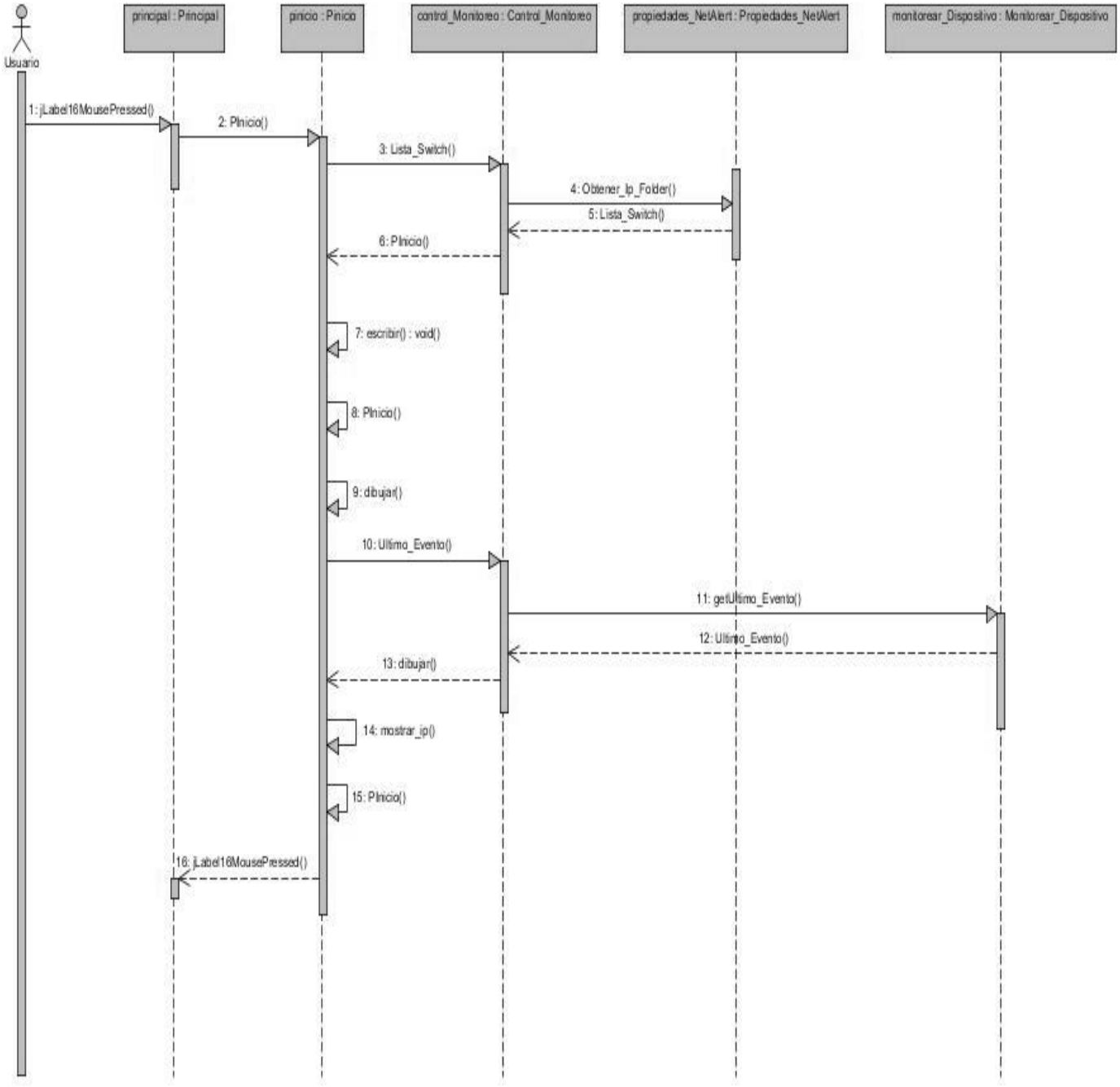


Fig. 27 Diagrama de Secuencia: Mostrar Inicio.

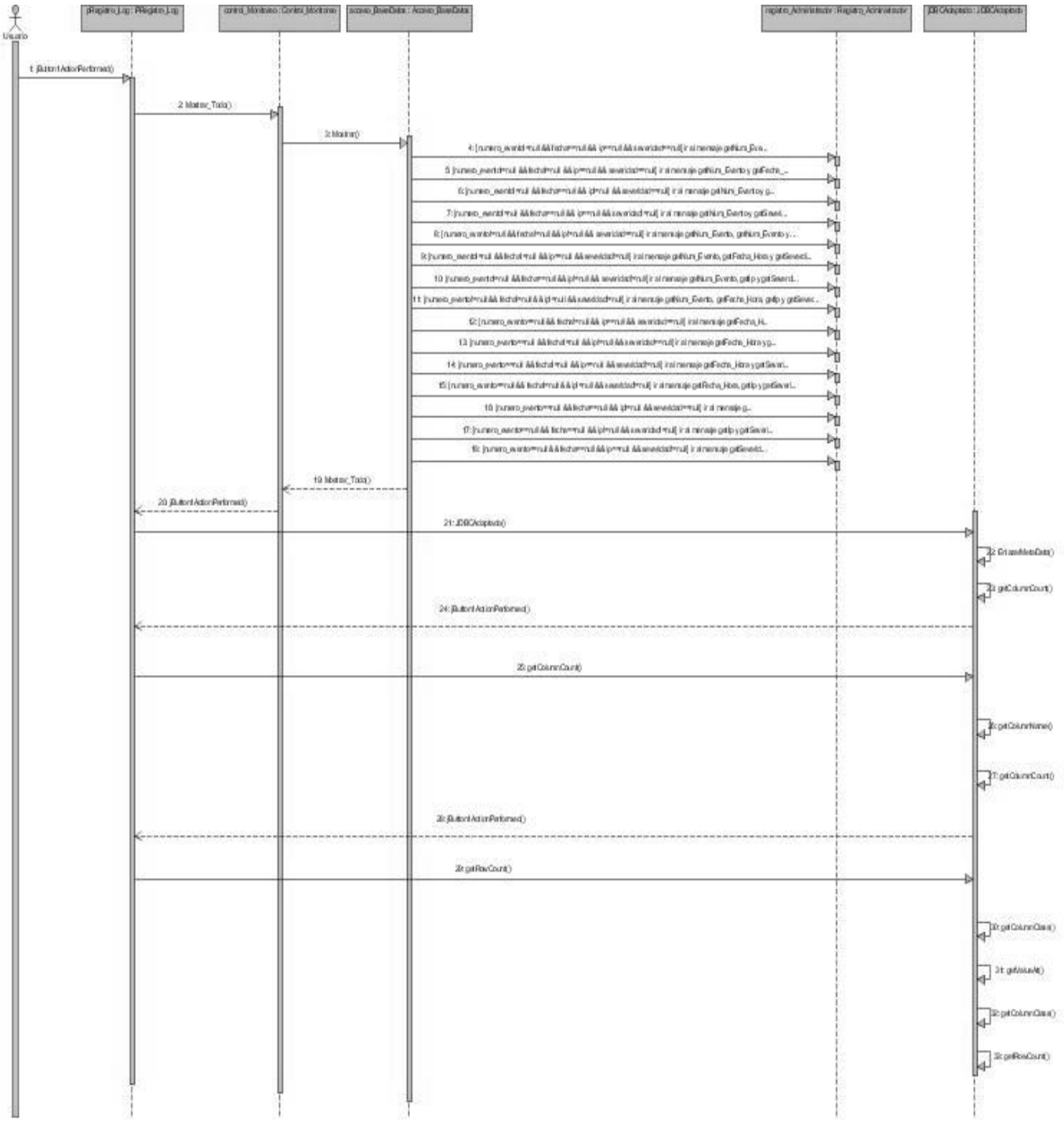


Fig. 28 Diagrama de Secuencia: Mostrar Registros Log

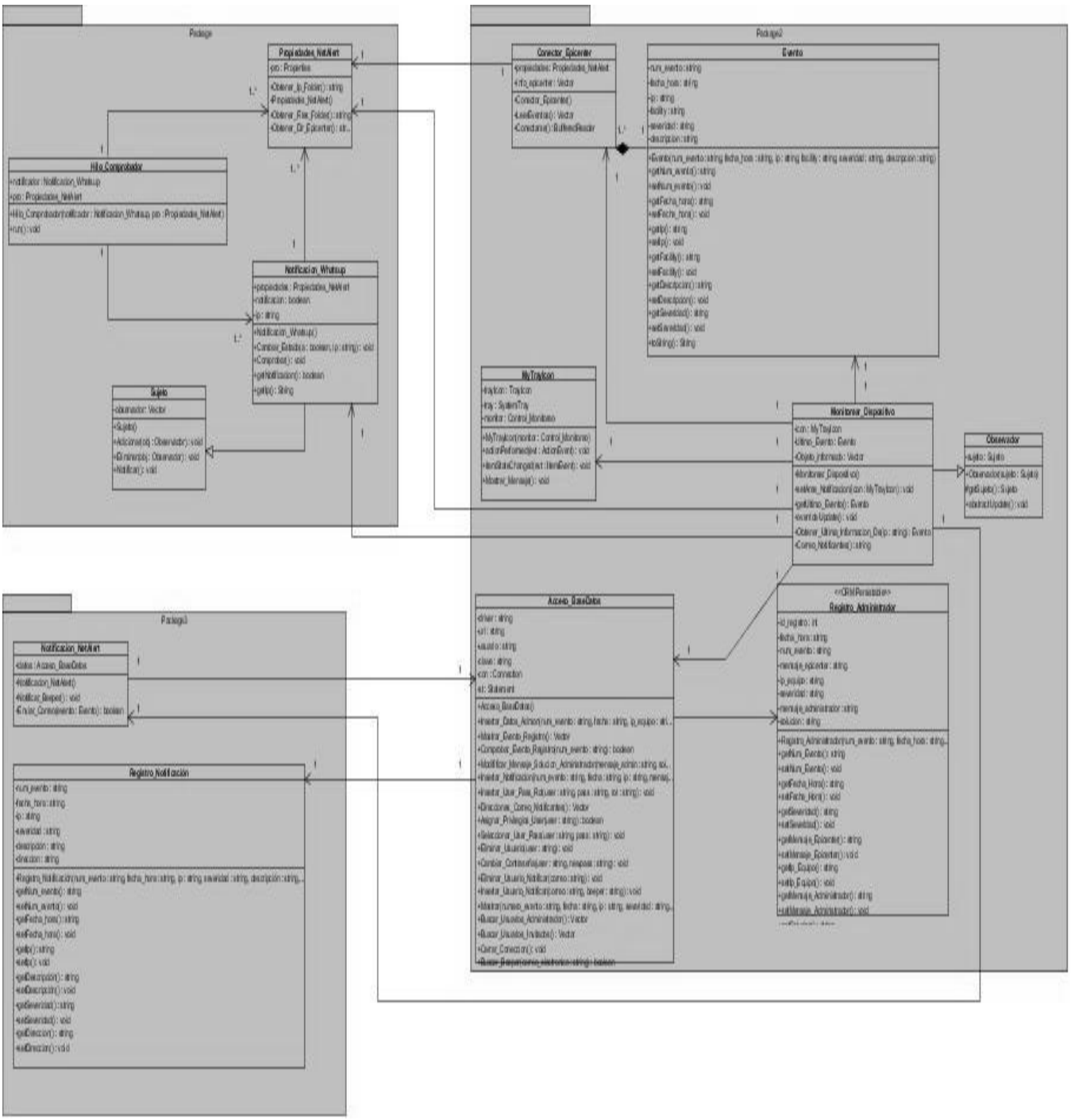


Fig. 29 Diagrama de Clases del Diseño: Verificar Estado.

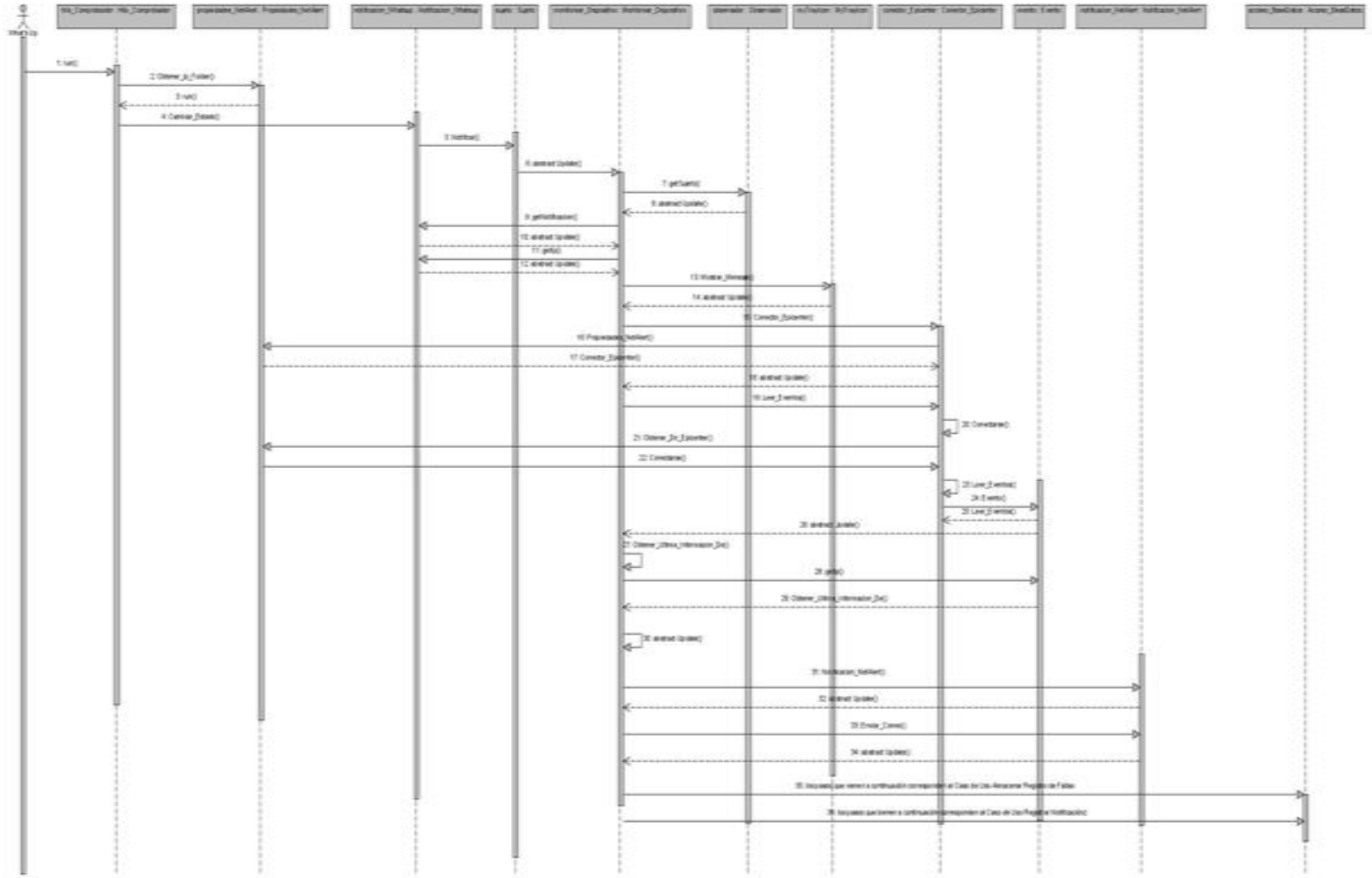
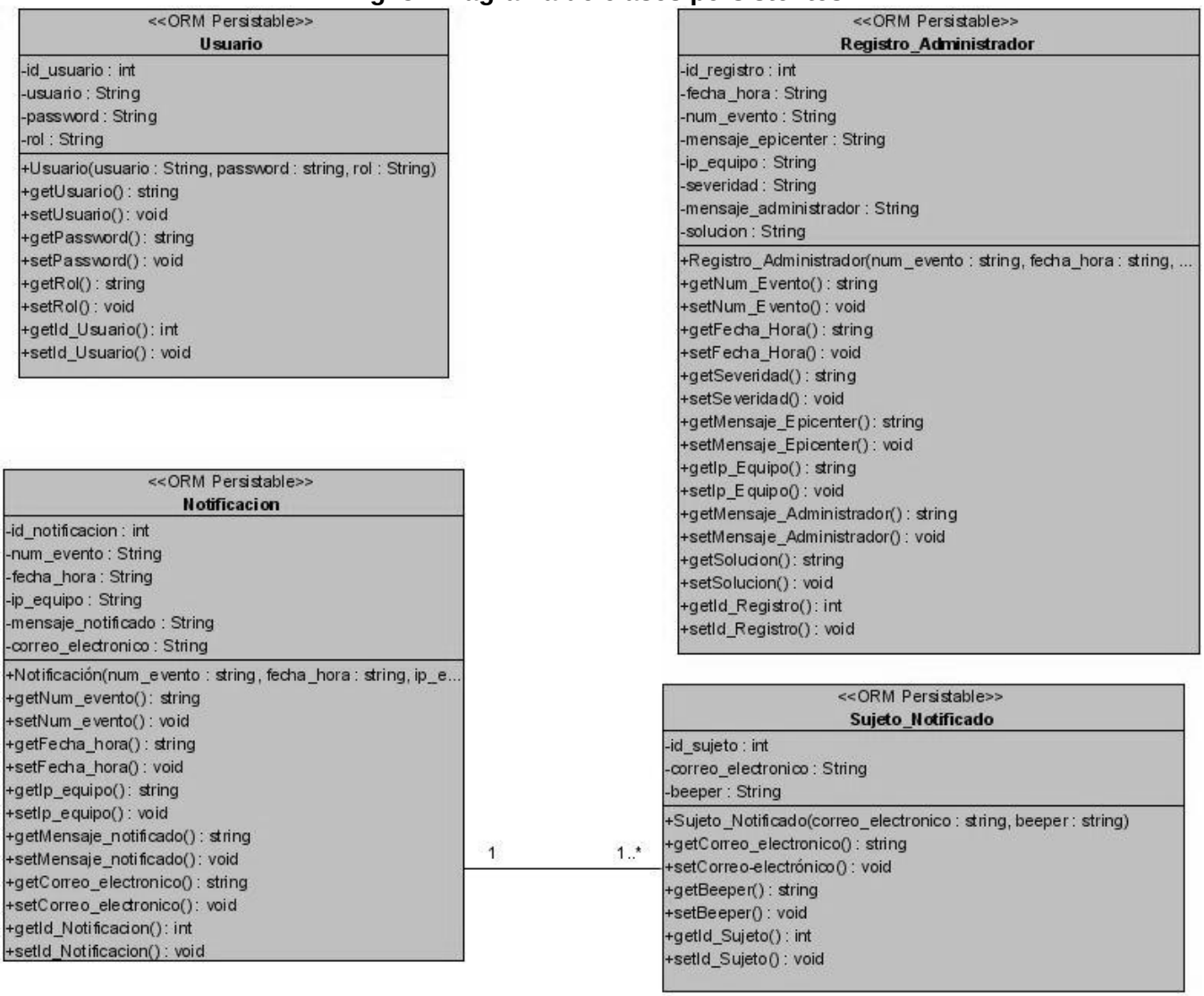


Fig. 30 Diagrama de Secuencia: Verificar Estado.

3.4 Diagrama de clases persistentes.

Una clase persistente es una clase entidad que posee la capacidad de mantener su valor en el espacio y en el tiempo. En el diagrama de clases persistentes están presentes dichas clases y las relaciones entre ellas.

Fig. 31 Diagrama de clases persistentes.



3.5 Modelo de datos.

Un modelo de datos es aquel que describe de una forma abstracta la forma en que se representan los datos, sea en una empresa, una institución, un hospital o en un sistema de gestión de base de datos. Básicamente consiste en una descripción de algo conocido como contenedor de datos, algo en donde se almacenan los datos de información y permite realizar diferentes acciones desde hacer reportes de estos datos, adicionar, eliminar o actualizar la información deseada.

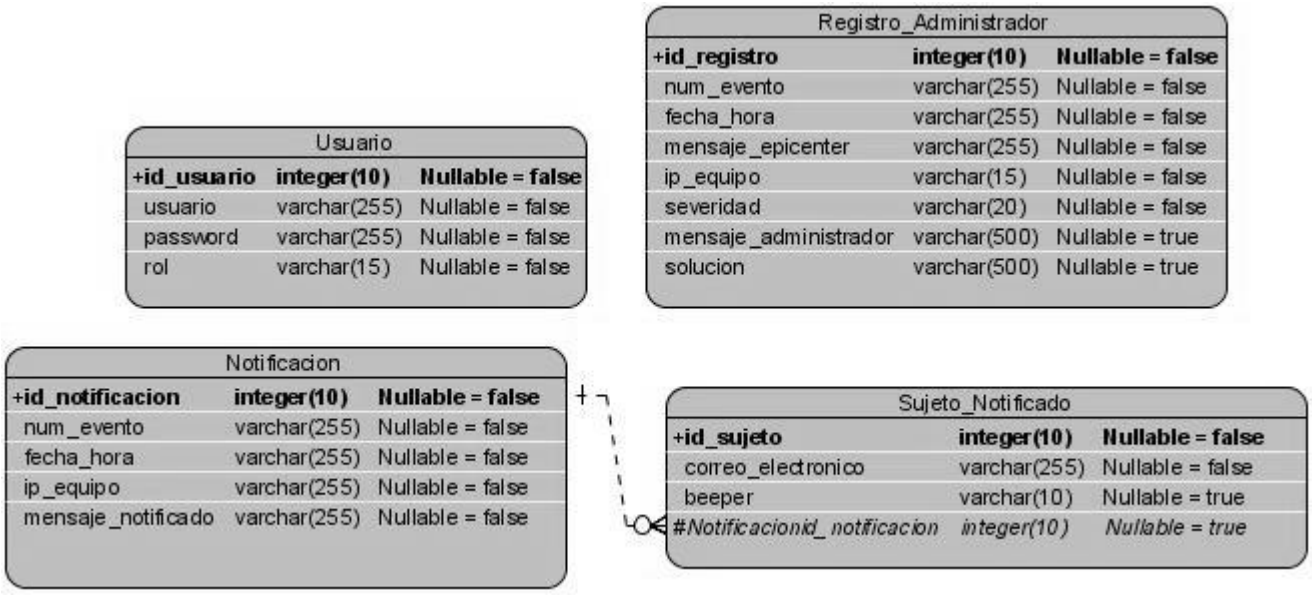


Fig. 32 Modelo físico.

3.5 Diagrama de Despliegue.

El modelo de despliegue se utiliza para capturar los elementos de configuración del procesamiento y las conexiones entre ellos. Es un modelo de objeto que describe la distribución física del sistema y 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.

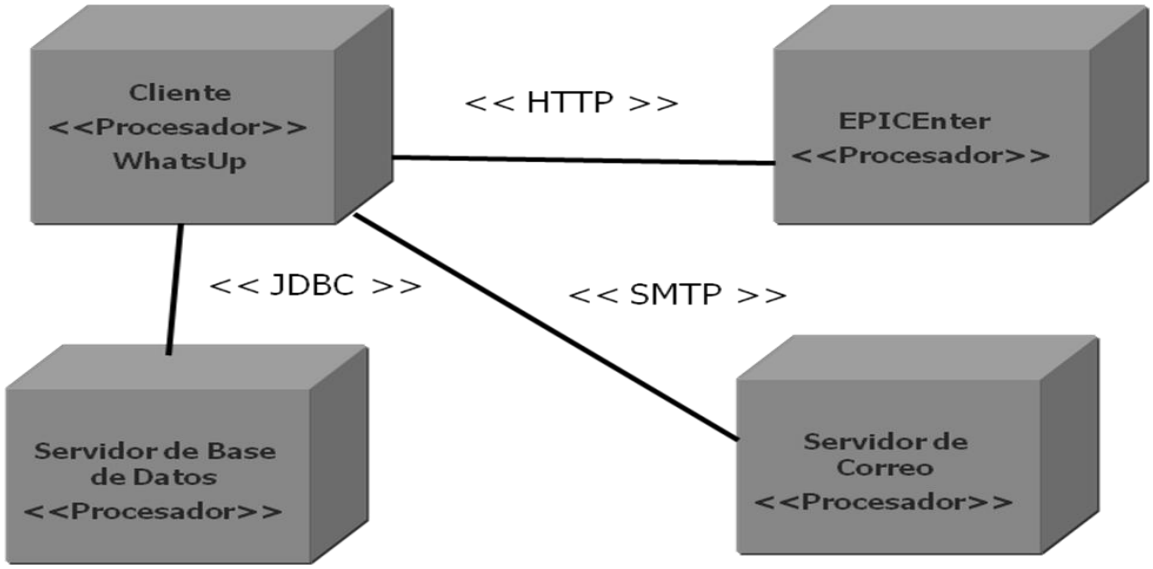


Fig. 33 Diagrama de despliegue.

3.6 Conclusiones.

En este capítulo se trataron los principios de diseño que se siguieron para el proceso de desarrollo del sistema. Se describieron los diferentes diagramas de clases de diseño para cada caso de uso del sistema, así como los diagramas de secuencia correspondientes, explicando detalladamente su funcionamiento. De la misma manera se mencionaron los distintos patrones de diseño y la arquitectura utilizada para la realización del sistema.

Capítulo 4: Implementación y Pruebas.

4.1 Introducción.

En el presente capítulo se desarrolla el modelo de implementación. Se muestran los diagramas de componentes, donde se detallan en componentes las diferentes clases y objetos utilizados en la implementación, para que se tenga una idea del funcionamiento de las mismas.
Además se muestran ejemplos de código fuente de algunas clases que emplean el uso de los patrones de arquitectura y diseño. Por último se realizan las pruebas de caja negra.

4.2 Diagramas de Componentes.

"Un diagrama de componentes representa las dependencias entre componentes software, incluyendo componentes de código fuente, componentes del código binario, y componentes ejecutables. Un módulo de software se puede representar como componente. Algunos componentes existen en tiempo de compilación, algunos en tiempo de enlace y algunos en tiempo de ejecución, otros en varias de éstas" (8).
Los diagramas de componentes fueron realizados por casos de uso, con el objetivo de facilitar la comprensión de éstos.

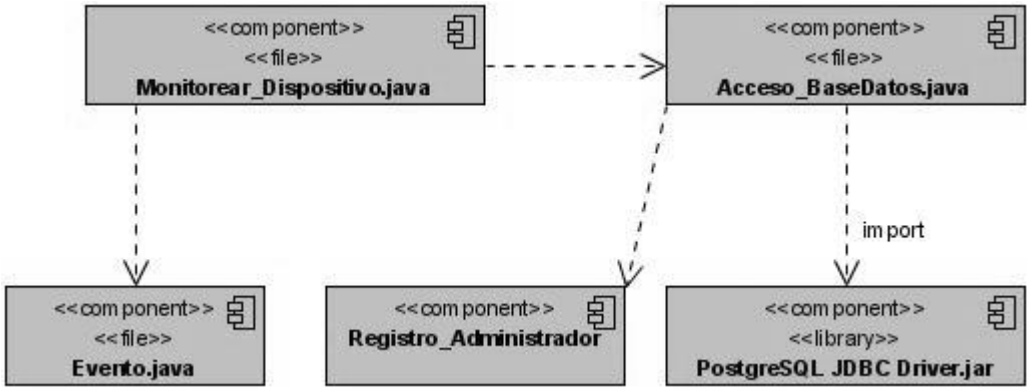


Fig. 34 Diagrama de Componentes del CU: Almacenar Registro de Fallas.

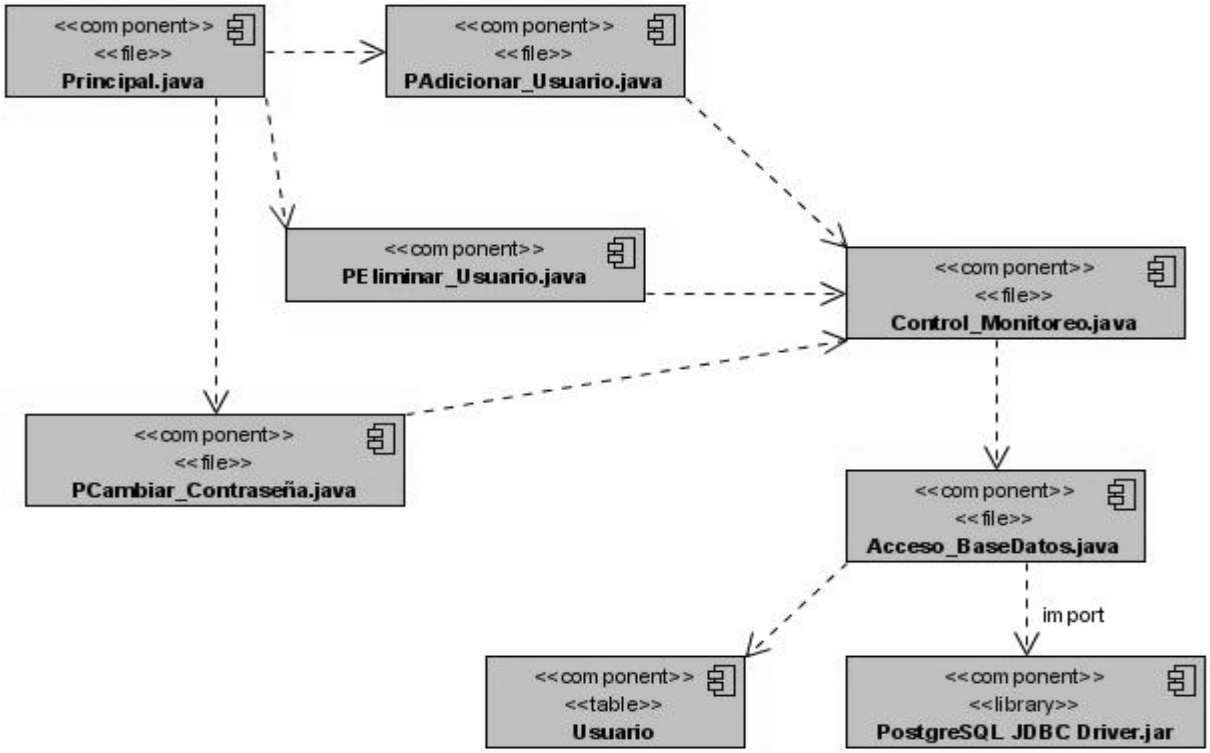


Fig. 35 Diagrama de Componentes del CU: Gestionar Usuario.

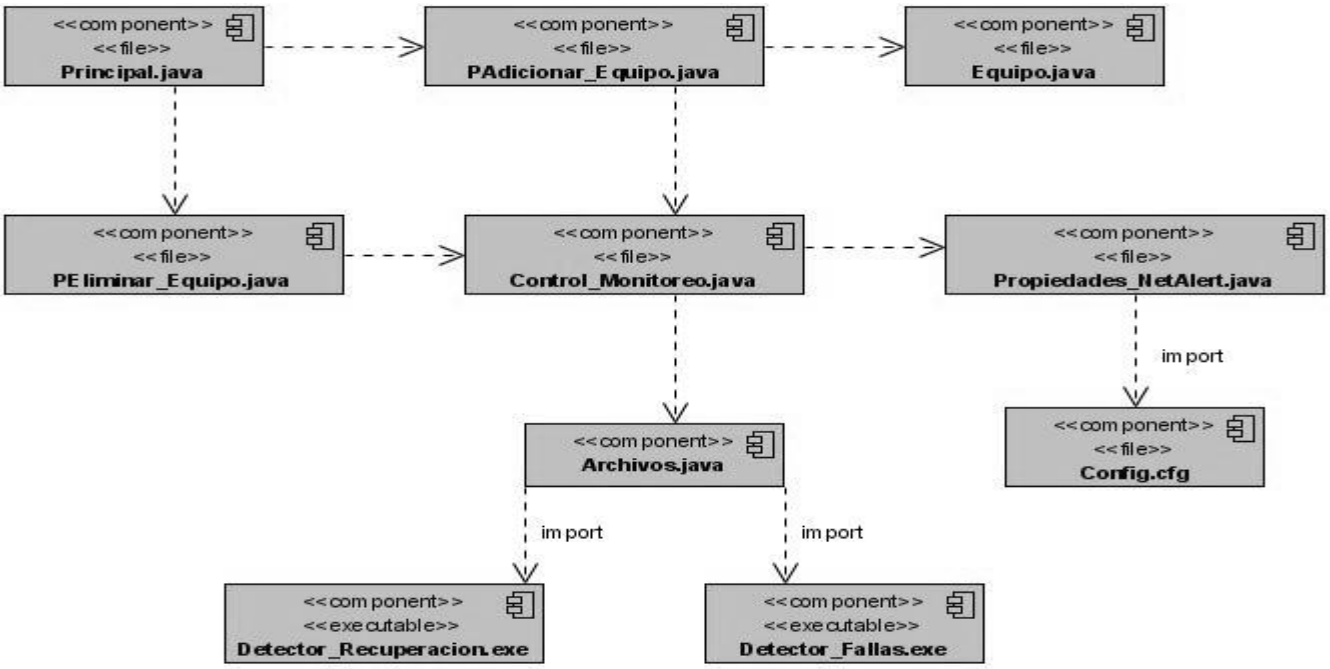


Fig. 36 Diagrama de Componentes del CU: Gestionar Equipo.

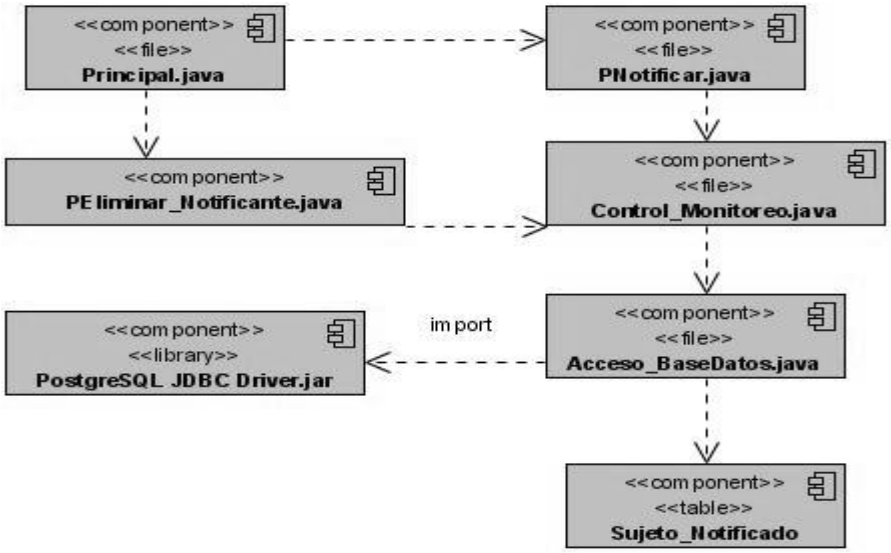


Fig. 37 Diagrama de Componentes del CU: Gestionar Notificante.

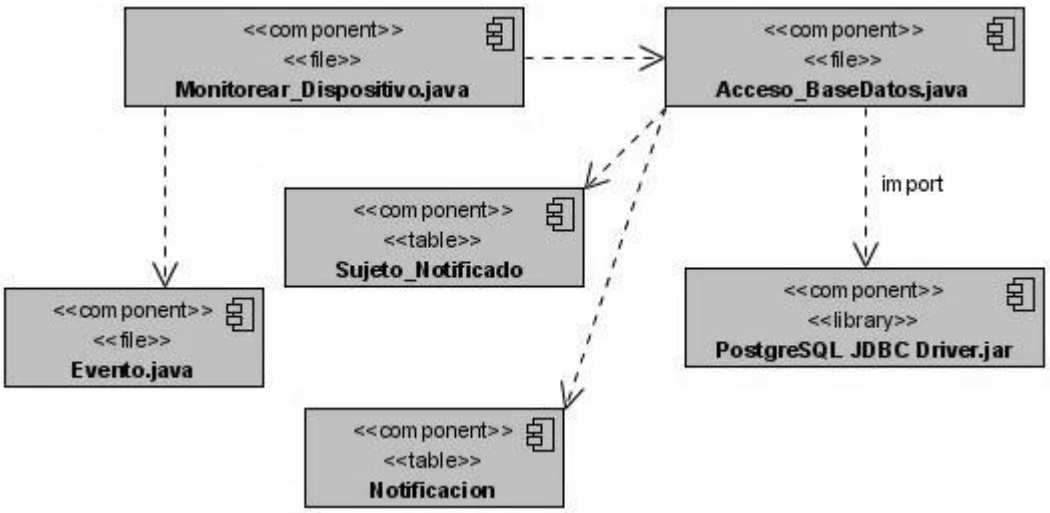


Fig. 38 Diagrama de Componentes del CU: Gestionar Notificante.

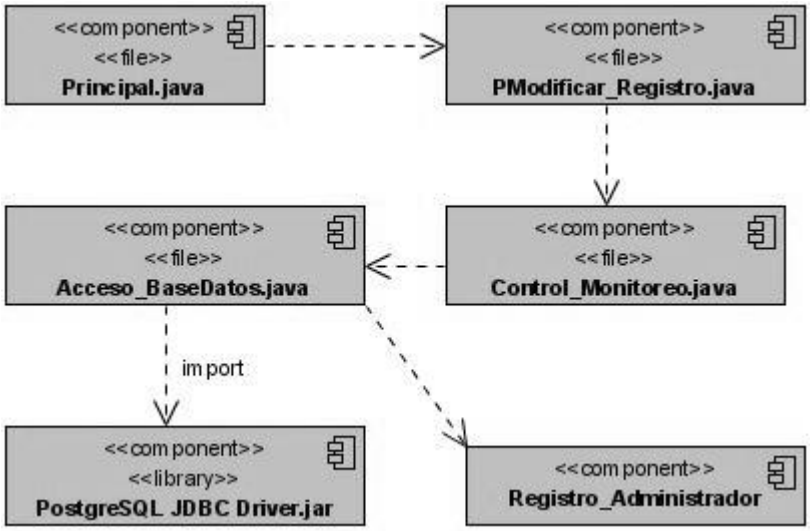


Fig. 39 Diagrama de Componentes del CU: Actualizar Registro

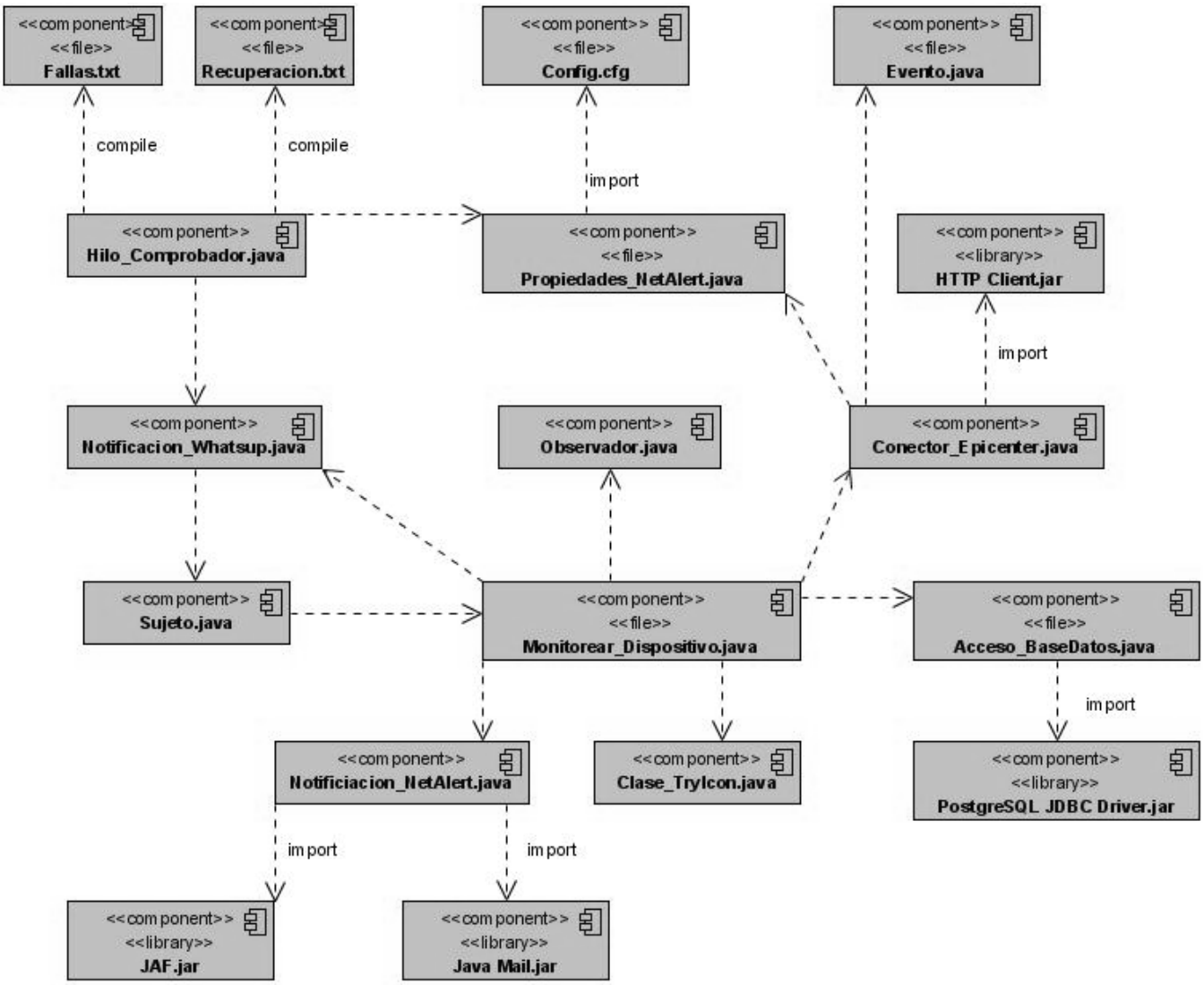


Fig. 40 Diagrama de Componentes del CU: Verificar Estado.

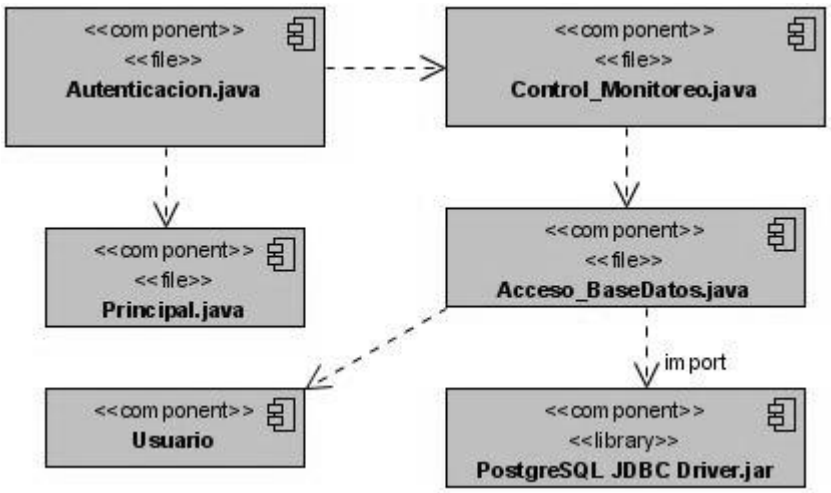


Fig. 41 Diagrama de Componentes del CU: **Autenticar Usuario.**

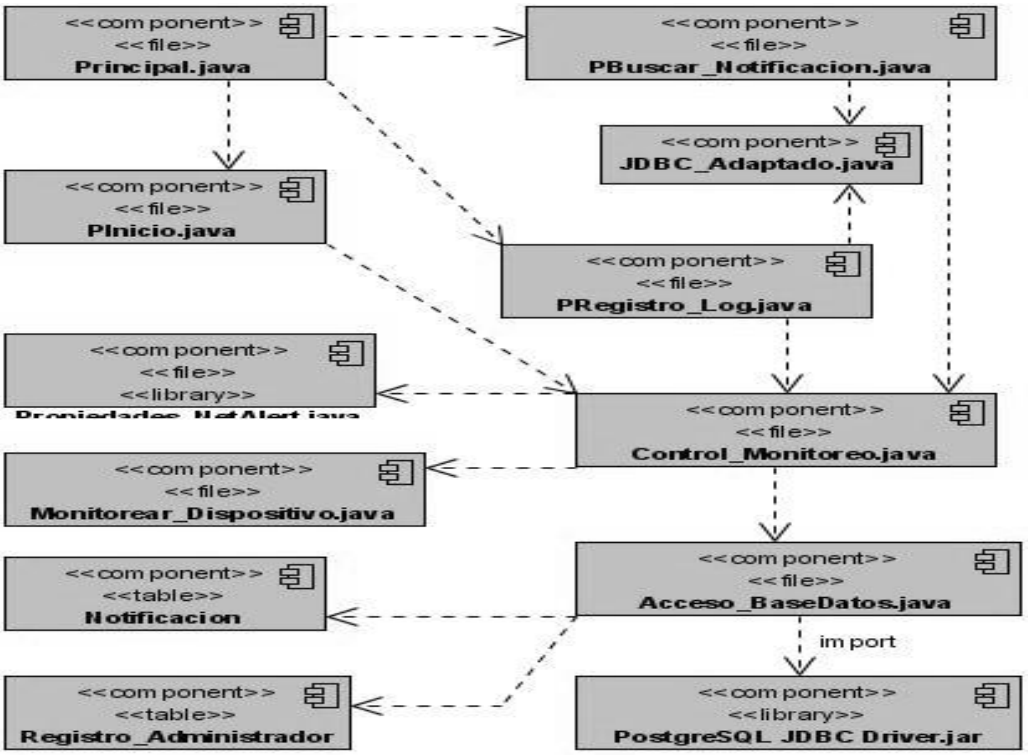


Fig. 42 Diagrama de Componentes del CU: **Autenticar Usuario.**

4.3 Código Fuente de Clases que representan el uso del Patrón Observador.

Clase Sujeto

```
public abstract class Sujeto {

   private Vector<Observador> observador;


   public Sujeto()
   {
      observador = new Vector<Observador>();
   }

   public void Adicionar(Observador obj)
   {
      observador.add(obj);


   }

   public void Eliminar(Observador obj)
         {
                  observador.remove(obj);
         }

   public void Notificar()
   {
      for(int i=0;i<observador.size();i++)
         observador.get(i).update();
   }
```

Clase Observador

```
public abstract class Observador {

    private Sujeto sujeto;


    protected Sujeto getSujeto()
        {
            return this.sujeto;
        }

        public Observador(Sujeto sujeto)
        {
            this.sujeto = sujeto;
        }

        public abstract void update();
}
```

Clase que hereda de Sujeto

Esta clase es la que se encarga dada la ocurrencia de un evento de notificar el cambio de estado y así la clase observadora ejecute sus acciones.

```
public class Notificacion_Whatsup extends Sujeto{

   Propiedades_NetAlert propiedades;
   private boolean notificacion;
   private String ip;



    public Notificacion_Whatsup()
    {
        super();
        try {
            propiedades = new Propiedades_NetAlert();


        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }



    public void cambiarEstado(boolean a, String ip)
```

```
    {
        this.notificacion = a;
        this.ip = ip;
        Notificar();
    }

    public void Comprobar()
    {
      Hilo_Comprobador hilo = new Hilo_Comprobador(this, propiedades);
      hilo.start();

    }

    public boolean getNotificacion() {
        return notificacion;
    }

    public String getIp() {
        return ip;
    }

}
```

Clase que hereda de Observador.

Esta clase es la que se encarga de heredar de Observador y dada la notificación de la clase que hereda de sujeto realiza las acciones correspondientes.

```
public class Monitorear_Dispositivo extends Observador {

    private Clase_TryIcon icon = null;
   private Evento ultimoEvento = null;
    Acceso_BaseDatos dato = new Acceso_BaseDatos();


    Vector<Evento> objeto_informado = new Vector<Evento>();

    public Monitorear_Dispositivo(Notificacion_Whatsup n) {
        super(n);
    }

    public void setArea_Notificacion(Clase_TryIcon icon)
         {
       this.icon = icon;
    }
```

```
public Evento getUltimo_Evento()
       {
      return ultimoEvento;
   }

   @Override
   public void update() {
      Notificacion_Whatsup not = (Notificacion_Whatsup) getSujeto();

      boolean notificacion = not.getNotificacion();
      String ip = not.getIp();
      if (notificacion == true) {
         try {
             String mensage = "Se ha detectado un error en "+ip;
            if(icon!=null)
             icon.MostrarMensaje("Equipo con falla",mensage, MessageType.ERROR);
            else
              System.out.println(mensage);


            Conector_Epicenter lector = new Conector_Epicenter();
            objeto_informado = lector.LeerEventos();
           Evento ultimo_evento = Obtener_Ultima_Informacion_De(ip);
            System.out.println(ultimo_evento);


            Notificacion_NetAlert notificador = new Notificacion_NetAlert();

            dato.Insertar_Datos_Admon(ultimo_evento.getNum_evento(),
ultimo_evento.getFecha_hora(), ultimo_evento.getIp(),
ultimo_evento.getSeveridad(),ultimo_evento.getDescripcion(),"Mensaje del Admón.", "Solución del
admon");

            this.ultimoEvento = ultimo_evento;


            //Las acciones a realizar si falla la red

             dato.Insertar_Notificacion(ultimo_evento.getNum_evento(), ultimo_evento.getFecha_hora(),
ultimo_evento.getIp(),ultimo_evento.getDescripcion(),CorreoNotificantes());

            notificador.EnviarCorreo(ultimo_evento);

         } catch (HttpException ex) {
            System.out.println("Error de conexión");

         } catch (IOException ex) {
           System.out.println("Error de conexión IO");
```

```
            }
            catch (Exception ex) {
                System.out.println("Error de conexión");
            }

        }
        else
          {
            this.ultimoEvento=null;
            String mensage = "El equipo "+ip+" se ha recuperado";
              if(icon!=null)
               icon.MostrarMensaje("Equipo recuperado",mensage, MessageType.INFO);
              else
                System.out.println(mensage);
            }
    }

    public String CorreoNotificantes()
    {
         String correo = "";
              for(int i=0; i< dato.Correos_Notificantes().size();i++)
              {
                  correo += dato.Correos_Notificantes().elementAt(i)+ "    ";
              }
         return correo;
    }

    public Evento Obtener_Ultima_Informacion_De(String ip) {
        for (int i = 0; i < objeto_informado.size(); i++) {
            if (objeto_informado.get(i).getIp().equals(ip)) {
                return objeto_informado.get(i);
            }

        }
        return null;
    }
}
```

4.4 Pruebas.

La prueba es el proceso de ejecutar un sistema bajo unas condiciones o requerimientos especificados con la intención de descubrir errores. Las pruebas tienen como objetivo detectar las áreas propensas a errores. Esto ayuda a la prevención de errores en un sistema e incrementan el valor del producto, al adaptarlo a las necesidades del usuario.

4.4.1 Pruebas de caja negra.

Se refiere a las pruebas que se llevan a cabo sobre la interfaz del software, con el objetivo de demostrar que las funciones del sistema son operativas. Estas pruebas se centran fundamentalmente en los requisitos funcionales. Se realiza una muestra por casos de usos del funcionamiento del sistema, para tener medida de la calidad y eficiencia con que cuenta.

Tabla 12: Caso de prueba: CU Verificar Estado.

Nombre del caso de uso Verificar Estado		
Entrada	**Resultados**	**Condiciones**
Se provoca un error en un dispositivo.	La aplicación muestra un mensaje reflejando el estado del equipo y envía un correo notificando sobre la falla.	Que el sistema contenga al equipo dentro de sus dispositivos a gestionar.

Tabla 13: Caso de prueba: CU Almacenar Registro de Fallas.

Nombre del caso de uso Almacenar Registro de Fallas		
Entrada	**Resultados**	**Condiciones**
Se provoca un error en un dispositivo.	La aplicación almacena el registro de log correspondiente al error.	Debe existir conexión con el EPICenter.

Tabla 14: Caso de prueba: CU Registrar Notificación.

Nombre del caso de uso Registrar Notificación		
Entrada	**Resultados**	**Condiciones**
Se provoca un error en un dispositivo.	La aplicación almacena la notificación enviada por correo.	Debe existir conexión con el EPICenter.

Tabla 15: Caso de prueba: CU Gestionar Equipo.

Nombre del caso de uso Gestionar Equipo		
Nombre del caso de prueba: Adicionar Equipo		
Entrada	**Resultados**	**Condiciones**
1. Se adiciona al sistema una dirección IP correcta. 2. Se adiciona al sistema una dirección IP incorrecta.	La aplicación adiciona el equipo a la lista de dispositivos a gestionar y muestra un mensaje de: “El quipo ha sido adicionado correctamente”. La aplicación muestra un mensaje especificando que la dirección del equipo no es correcta.	
Nombre del caso de prueba: Eliminar Equipo		
El usuario selecciona un dispositivo para eliminar.	La aplicación elimina el equipo de sus lista de dispositivos a gestionar y muestra un mensaje donde se refleja que el dispositivo ha sido eliminado correctamente.	

Tabla 16: Caso de prueba: CU Gestionar Usuario.

Nombre del caso de uso Gestionar Usuario		
Nombre del caso de prueba: Adicionar Usuario		
Entrada	**Resultados**	**Condiciones**
1. Se adiciona al sistema todos los datos del usuario. 2. El administrador no llena todos los campos	La aplicación adiciona los datos insertados y muestra un mensaje donde se refleja que el usuario ha sido adicionado correctamente. La aplicación muestra un	

correspondientes al usuario.	mensaje donde especifica los campos que quedan por llenar.	
Nombre del caso de prueba: Eliminar Usuario		
1. Se selecciona un usuario para eliminar. 2. Se selecciona el usuario "Administrador" para eliminar.	La aplicación elimina al usuario de su base de datos y muestra un mensaje donde se refleja que el usuario ha sido eliminado correctamente. La aplicación muestra un mensaje especificando que por seguridad el usuario seleccionado no puede ser eliminado.	
Nombre del caso de prueba: Cambiar Contraseña		
Se selecciona un usuario, se introduce la nueva contraseña y se confirma la misma.	La aplicación actualiza la contraseña del usuario seleccionado y muestra un mensaje reflejando que la acción se ha realizado satisfactoriamente.	

Tabla 17: Caso de prueba: CU Autenticar Usuario.

Nombre del caso de uso Autenticar Usuario		
Entrada	**Resultados**	**Condiciones**
1. Se introduce un usuario y su contraseña. 2. Se introduce un usuario incorrecto.	La aplicación valida que los mismos son correctos e inicializa la forma principal. La aplicación muestra un mensaje especificando que el usuario no es correcto.	

Tabla 18: Caso de prueba: CU Actualizar Registro.

Nombre del caso de uso Actualizar Registro		
Entrada	**Resultados**	**Condiciones**
Se selecciona un número de evento y se introduce los datos correspondientes con la descripción y la solución del problema.	La aplicación actualiza el registro de log con los datos introducidos.	

Tabla 19: Caso de prueba: CU Mostrar Registro.

Nombre del caso de uso Mostrar Registro		
Entrada	**Resultados**	**Condiciones**
Se introduce tipo de severidad.	La aplicación muestra todos los registros de log correspondientes al tipo de severidad seleccionada.	

Tabla 20: Caso de prueba: CU Gestionar Notificante.

Nombre del caso de uso Gestionar Notificante		
Nombre del caso de prueba: Adicionar Notificante		
Entrada	**Resultados**	**Condiciones**
1. Se adiciona una dirección de correo y un número de beeper.	La aplicación adiciona los datos introducidos y muestra un mensaje especificando que se informará al usuario de correo y al número de beeper señalado.	
2. Se adiciona una dirección de correo sin número de beeper.	La aplicación adiciona el dato introducido y muestra un mensaje especificando que solo se informará al usuario de correo señalado.	

3. Se introduce una dirección de correo no valida.	La aplicación muestra un mensaje especificando que debe introducir una dirección de correo válida.	
Nombre del caso de prueba: Eliminar Equipo		
1. El usuario selecciona una dirección de correo, luego especifica que desea eliminar el mismo de la lista de notificantes. 2. El usuario selecciona una dirección de correo, luego especifica que desea eliminar el mismo de las notificaciones por beeper. 3. El usuario selecciona una dirección de correo a la cual no se le notifica por beeper y luego especifica que desea eliminarla de la lista de notificantes por beeper.	La aplicación elimina la dirección seleccionada de la lista de notificantes y muestra un mensaje especificando dicha acción. La aplicación elimina dicho usuario de la lista de notificaciones por beeper y muestra un mensaje especificando que ya no se notificará por esa vía al usuario seleccionado. La aplicación muestra un mensaje donde se especifica que al usuario seleccionado no se le notifica por beeper.	

4.5 Conclusiones.

En el presentado capítulo se mostró y comprobó el funcionamiento específico del sistema desarrollado, mediante las pruebas de caja negra. Los diagramas de componentes mostraron la estructura de alto nivel del modelo de implementación. Además se presentaron las clases que se implementaron rigiéndose por las orientaciones del patrón de diseño Observador.

Conclusiones Generales.

Al terminar la etapa de elaboración de este trabajo se pudo afirmar que se realizó la captura de requisitos del sistema propuesto obteniéndose los requisitos funcionales y no funcionales del mismo.

Se desarrolló su diseño e implementación, obteniendo como resultado una versión funcional del producto, con el cual se logra un mejor funcionamiento en el proceso de gestionar las fallas ocurridas en la red telemática de la UCI, pues se integraron las funcionalidades de gestión, monitoreo y notificación de dispositivos de red.

Se realizaron las pruebas de caja negra lo que permitió validar las capacidades operativas del sistema desarrollado.

Es importante destacar que el sistema fue abalado por el departamento de redes de la universidad. Se presentó en diferentes eventos como VI Jornada Científica Estudiantil obteniendo relevante a nivel de facultad y participando a nivel de universidad, además el trabajo fue expuesto en el XVI Fórum de Ciencia y Técnica reconocido como destacado por la Vicerrectoría de Tecnología.

.

Recomendaciones.

- Adicionar funcionalidades al sistema que permitan automatizar las acciones antes las fallas, haciendo uso de la base de conocimientos desarrollada.

Bibliografía

1. **EQUIHUA, LEONEL SORIANO.** *La Importancia de la Gestión de Redes.* s.l. : Universidad de Colima.
2. **Magedanz, T.Saydam and T.** Networks and Network Management into Service and Service Management. *Journal of Networks and Systems Management,.* diciembre de 1996.
3. IntroduccionGestionRedes. *Facultad de informatica de Coruña.* [En línea] [Citado el: 10 de diciembre de 2007.] www.fic.udc.es/files/asignaturas/56XR/files/IntroduccionGestionRedes-2008.pdf -.
4. CienporCien. [En línea] 6 de Septiembre de 2007. [Citado el: 23 de enero de 2008.] www.CienporCien.com\software\sistemas\Ipswitch Inc_ anuncia la disponibilidad de WhatsUp Gold v11.htm.
5. **Cayuqueo, Sergio Daniel.** Monitoreo y análisis de Red con Nagios. [En línea] [Citado el: 14 de diciembre de 2007.] www.cayu.com.ar.
6. bregai. [En línea] 14 de Abril de 2002. [Citado el: 28 de Septiembre de 2007.] http://bregai.com/bregaipro/redes.htm.
7. **Sánchez, Carlos.** [En línea] Universidad de Coruña, 28 de septiembre de 2004. [Citado el: 6 de marzo de 2008.] http://oness.sourceforge.net/proyecto/html/ch05.html. 4.
8. **Ivar Jacobson, Grady Booch, James Rumbaugh.** *El proceso unificado de desarrollo de software*. s.l. : Pearson Education, 2000.
9. **Sukowsky, John.** *Java 2 J2SE 1.4.* . Madrid : EDICIONES ANAYA MULTIMEDIA, 2003.
10. **Marañón, Gonzalo Álvarez.** Caracteristicas del lenguaje java. [En línea] [Citado el: 23 de noviembre de 2007.] http://www.iec.csic.es/CRIPTONOMICON/java/quesjava.html.
11. **Introducción al Lenguaje PERL.** [En linea] [Citado el: 25 de mayo de 2008.] http://infodisc.es/abcd/tutoriales/program.html
12. **Sabater, Jaume.** [En línea] 18 de 12 de 2006. [Citado el: 4 de febrero de 2008.] http://bulma.net/body.phtml?nIdNoticia=2361.
13. **R Kasman, Bass Clements.** *Software Architecture in Practice.* s.l. : Addison-Wesley, 1998.

Referencias bibliográficas

1. **EQUIHUA, LEONEL SORIANO.** *La Importancia de la Gestión de Redes.* s.l. : Universidad de Colima.

2. **Magedanz, T.Saydam and T.** Networks and Network Management into Service and Service Management. *Journal of Networks and Systems Management,.* diciembre de 1996.

3. IntroduccionGestionRedes. *Facultad de informatica de Coruña.* [En línea] [Citado el: 10 de diciembre de 2007.] www.fic.udc.es/files/asignaturas/56XR/files/IntroduccionGestionRedes-2008.pdf -.

4. CienporCien. [En línea] 6 de Septiembre de 2007. [Citado el: 23 de enero de 2008.] www.CienporCien.com\software\sistemas\Ipswitch Inc_ anuncia la disponibilidad de WhatsUp Gold v11.htm.

5. **Cayuqueo, Sergio Daniel.** Monitoreo y análisis de Red con Nagios. [En línea] [Citado el: 14 de diciembre de 2007.] www.cayu.com.ar.

6. bregai. [En línea] 14 de Abril de 2002. [Citado el: 28 de Septiembre de 2007.] http://bregai.com/bregaipro/redes.htm.

7. **Sánchez, Carlos.** [En línea] Universidad de Coruña, 28 de septiembre de 2004. [Citado el: 6 de marzo de 2008.] http://oness.sourceforge.net/proyecto/html/ch05.html. 4.

8. **Ivar Jacobson, Grady Booch, James Rumbaugh.** *El proceso unificado de desarrollo de software.* s.l. : Pearson Education, 2000.

9. **Sukowsky, John.** *Java 2 J2SE 1.4.* . Madrid : EDICIONES ANAYA MULTIMEDIA, 2003.

10. **Marañón, Gonzalo Álvarez.** Caracteristicas del lenguaje java. [En línea] [Citado el: 23 de noviembre de 2007.] http://www.iec.csic.es/CRIPTONOMICON/java/quesjava.html.

11. **Introducción al Lenguaje PERL.** [En linea] [Citado el: 25 de mayo de 2008.] http://infodisc.es/abcd/tutoriales/program.html

12. **Sabater, Jaume.** [En línea] 18 de 12 de 2006. [Citado el: 4 de febrero de 2008.] http://bulma.net/body.phtml?nIdNoticia=2361.

13. **R Kasman, Bass Clements.** *Software Architecture in Practice.* s.l. : Addison-Wesley, 1998.

ANEXOS

Anexo 1: Descripción de las clases.

Tabla 21: Descripción de la Clase: Control Monitoreo

Nombre: Control Monitoreo	
Tipo de clase : Controladora	
Atributo	**Tipo**
datos	Acceso_BaseDatos
pro	Propiedades UCI-Alert
monitoreo	Monitorear_Dispositivo
nw	Notificacion_Whatsup
Responsabilidades	
1.Nombre.	Autenticarse()
Descripción.	Encargado de verificar que el usuario que se quiere autenticar este registrado en el sistema.
2.Nombre	Asignar_Privilegios()
Descripción.	Encargado de verificar si el usuario autenticado es Invitado o Administrador, para los permisos dentro del sistema.
3. Nombre.	Adicionar_Switch()
Descripción.	Encargado de adicionar los dispositivos de red (switch capa 3) que se desean monitorear.
4.Nombre.	Eliminar_Switch()
Descripción.	Encargado de eliminar los dispositivos de red de la lista de equipos a monitorear.
5.Nombre.	Lista_Switch()
Descripción.	Encargado de contener en una lista todos los dispositivos de red activos en el sistema.
6.Nombre.	Adicionar_Usuario_Sistema()
Descripción.	Encargado de adicionar un nuevo usuario que opere en el sistema, con la contraseña correspondiente, que debe tener 5 o más caracteres, dicho usuario podría ser Administrador o Invitado.
7. Nombre.	Eliminar_Usuario_Sistema()

Descripción.	Encargado de eliminar determinado usuario del sistema.
9.Nombre	Cambiar_Password_Usuario()
Descripción	Encargado de cambiar la contraseña de determinado usuario, cumpliendo la medida de seguridad de presentar 5 o más caracteres.
10.Nombre	Mostrar_Usuario_Admin()
Descripción.	Encargado de mostrar todos los usuarios que son Administradores.
11. Nombre.	Mostrar_Usuario_Invitado()
Descripción.	Encargado de mostrar todos los usuarios que son Invitados
14. Nombre.	Buscar_Privilegios()
Descripción.	Encargado de saber que rol presenta el usuario que se ha autenticado, si es Administrador o Invitado.
13.Nombre	Adicionar_Notificante()
Descripción	Encargado de adicionar un usuario al cual se le quiere notificar ante la presencia de una falla en la red.
14.Nombre	Eliminar_Notificante()
Descripción.	Encargado de eliminar de la lista de notificantes aquel usuario que ya el sistema no debe enviarle ningún mensaje.
15.Nombre	Obtener_Correos_Notificante()
Descripción.	Encargado de de obtener todos las direcciones de correo electrónicos de los usuarios a los cuales se les notifica.
16.Nombre	Ultimo_Evento()
Descripción.	Encargado de contener la información del último evento ocurrido que ocasionó una falla en la red.
17. Nombre.	Mostrar_Todo()
Descripción.	Encargado de mostrar la información contenida en el último evento.
18. Nombre.	Eliminar_Notificacion_Beeper()
Descripción.	Encargado de eliminar la notificación por beeper a determinados usuario.
19. Nombre.	Comprobar_Num_Evento_Registro()
Descripción.	Encargado de comprobar si el número de evento entrado por el usuario existe dentro de la base de datos.
20.Nombre	Actualizar_Mensaje_Solucion_Administrador()

Descripción.	Encargado de actualizar o modificar la información referente al administrador de red con respecto a la causa y la posible solución de la falla ocurrida.
21.Nombre	Obtener Beeper()
Descripción	Encargado de obtener el numero de beeper de todos los usuarios a notificar

Tabla 22: Descripción de la Clase: Acceso_Basedatos.

Nombre: Acceso_Basedatos	
Tipo de clase Acceso a Datos	
Atributo	**Tipo**
driver	String
url	String
usuario	String
clave	String
con	Connection
st	Statement
Responsabilidades	
1.Nombre.	Insertar_Datos_Admon()
Descripción.	Encargado de insertar en la base de datos lo relacionado con la información generada por la falla ocurrida, así como la descripción y la posible solución brindada por el administrador de red.
2.Nombre	Mostrar_Evento_Registro()
Descripción.	Encargado de seleccionar los números de evento del Epicenter almacenados en la base de datos.
3. Nombre.	Comprobar_Evento_Registro()
Descripción.	Encargado de comprobar si el número de evento entrado por el usuario existe dentro de la base de datos.
4.Nombre.	Modifificar_Mensaje_Solucion_Administrador()
Descripción.	Encargado de actualizar o modificar la información referente al administrador de red con respecto a la causa y la posible solución de la falla ocurrida.
5.Nombre.	Insertar_Notificacion()
Descripción.	Encargado de insertar en la base de datos lo referente al mensaje notificado ante la presencia de fallas en la red.

6.Nombre.	Correos_Notificantes()
Descripción.	Encargado de obtener todas las direcciones de correo electrónico de los usuarios a los cuales se les notifica.
7. Nombre.	Insertar_User_Pass_Rol()
Descripción.	Encargado de insertar en la base datos un nuevo usuario que opere en el sistema, con la contraseña correspondiente que debe tener 5 o más caracteres, además el rol de dicho usuario: Administrador o Invitado.
9.Nombre.	Asignar_Privilegios()
Descripción.	Encargado de verificar en la base de datos si el usuario autenticado es Invitado o Administrador, para los permisos dentro del sistema.
10.Nombre.	Seleccionar_User_Pass()
Descripción.	Encargado de verificar en la base de datos si el usuario autenticado existe y si es correcta la contraseña.
11.Nombre.	Eliminar_Usuario_Sistema()
Descripción.	Encargado de eliminar de la base de datos determinado usuario del sistema.
12.Nombre.	Cambiar_Pass()
Descripción.	Encargado de cambiar en la base de datos la contraseña de determinado usuario.
13.Nombre.	Insertar_Usuario_Notificar()
Descripción.	Encargado de insertar en la base de datos los usuarios a los cuales se les desea notificar ante la presencia de fallas en la red.
14.Nombre.	Eliminar_Usuario_Notificar()
Descripción.	Encargado de eliminar en la base de datos los usuarios a los cuales ya no se les desea notificar ante la presencia de fallas en la red.
15.Nombre.	Mostrar()
Descripción.	Encargado de mostrar la información contenida en el último evento generado por la falla ocurrida.
16.Nombre.	Buscar_Usuarios_Admin()
Descripción.	Encargado de seleccionar en la base de datos los usuarios que son Administradores del sistema.
17.Nombre.	Buscar_Usuarios_Invitados()
Descripción.	Encargado de seleccionar en la base de datos los usuarios que son

	Invitados del sistema.
18.Nombre.	Buscar_Beeper()
Descripción.	Encargado de seleccionar en la base datos números de beeper de los usuarios a notificar que presentan este dispositivo.
19.Nombre.	Eliminar_Notificante_Beeper()
Descripción.	Encargado de eliminar en la base de datos el número de beeper de determinado usuario que ya no se le notificar por esa vía

Tabla 23: Descripción de la Clase: Clase_TrayIcon.

Nombre Clase_TrayIcon	
Tipo de clase: Controladora	
Atributo	**Tipo**
trayIcon	TrayIcon
tray	SystemTray
monitor	Control_Monitoreo
Responsabilidades	
1. Nombre.	actionPerformed()
Descripción.	Encargado de permitir ir desde el TrayIcon hasta la interfaz de autenticar.
2.Nombre	itemStateChanged()
Descripción.	Encargado de la creación del TrayIcon.
3. Nombre.	Mostrar_Mensaje()
Descripción.	Encargado de mostrar el mensaje cuando ocurre el cambio de estado de un dispositivo de red.

Tabla 24: Descripción de la Clase: Conector_Epicenter.

Nombre Conector_Epicenter	
Tipo de clase: Controladora	
Atributo	**Tipo**
propiedades	Propiedades_NetAlert
Info_epicenter	Vector <String>
Responsabilidades	
1.Nombre.	Conectarse()

Descripción.	Encargado de conectarse al EPICenter.
2.Nombre	LeerEventos()
Descripción.	Encargado de después de haberse conectado al EPICenter alcanzar la información referente a las fallas ocurridas.

Tabla 25: Descripción de la Clase: Equipo.

Nombre Equipo	
Tipo de clase: Controladora	
Atributo	**Tipo**
direccion_ip	String
Responsabilidades	
1. Nombre.	getDireccion_ip()
Descripción.	Encargado de permitir acceder a la dirección ip de determinado dispositivo de red.
2.Nombre	setDireccion_ip()
Descripción.	Encargado de permitir modificar dicha dirección ip.

Tabla 26: Descripción de la Clase: Evento

Nombre Evento	
Tipo de clase: Controladora	
Atributo	**Tipo**
num_evento	String
fecha_hora	String
ip	String
facility	String
severidad	String
descripcion	String
Responsabilidades	
1. Nombre.	getNum_evento()
Descripción.	Encargado de permitir acceder al num_evento.
2.Nombre	setNum_evento()
Descripción.	Encargado de permitir modificar al num_eventol.

3.Nombre	getFecha_hora()
Descripción.	Encargado de permitir acceder a la fecha_hora.
4.Nombre	setFecha_hora()
Descripción.	Encargado de permitir modificar a la fecha_hora.
5.Nombre	getIp()
Descripción.	Encargado de permitir acceder a la dirección ip.
6.Nombre	setIp()
Descripción.	Encargado de permitir modificar a la dirección ip.
7.Nombre	getFacility()
Descripción.	Encargado de permitir acceder al local.
8.Nombre	setFacility()
Descripción.	Encargado de permitir modificar al local.
9.Nombre	getSeveridad()
Descripción.	Encargado de permitir acceder a la severidad.
10.Nombre	setSeveridad()
Descripción.	Encargado de permitir modificar a la severidad.
11.Nombre	getDescripcion()
Descripción.	Encargado de permitir acceder a la descripción.
12.Nombre	setDescripcion()
Descripción.	Encargado de permitir modificar a la descripción.
13.Nombre	toString()
Descripción.	Encargado de mostrar la información referente al evento ocurrido de forma concatenada.

Tabla 27: Descripción de la Clase: Hilo_Controlador.

Nombre Hilo_Controlador	
Tipo de clase: Controladora	
Atributo	**Tipo**
notificador	Notificacion_Whatsup
pro	Propiedades_NetAlert
Responsabilidades	
1. Nombre.	run()

Descripción.	Encargado de verificar cuando ha ocurrido un cambio en el estado de algún dispositivo de red, de notificar dicha acción, de enviar el mensaje de dicho cambio al TrayIcon, el cual lo muestra mediante un pop-up. Este método es el que se encarga de estar verificando el aviso del Whats'up.

Tabla 28: Descripción de la Clase: Monitorear_Dispositivo

Nombre Monitorear_Dispositivo	
Tipo de clase: Controladora	
Atributo	**Tipo**
icon	Clase_TryIcon
ultimoEvento	Evento
objeto_informado	Vector<Evento>
Responsabilidades	
1. Nombre.	getUltimo_Evento()
Descripción.	Encargado de permitir acceder al último_evento.
2. Nombre.	update()
Descripción.	Encargado de esperar la notificación referente a si ha ocurrido el cambio de estado de algún dispositivo de red para de esa forma realizar las acciones correspondientes al sistema, ya sean enviar mensajes al correo electrónico, al beeper, insertar en la base de datos, etc.
3. Nombre.	CorreoNotificantes()
Descripción.	Encargado de en una variable de tipo string, concatenar todos los correos electrónicos de los usuarios a notificar, esto se utiliza para cuando se a insertar la notificación en la base de datos.
4. Nombre.	Obtener_Ultima_Informacion_De()
Descripción.	Encargado de dada la dirección ip del equipo que ha fallado conocer la información referente a ese evento.

Tabla 29: Descripción de la Clase: Notificacion_UCI-Alert

Nombre Notificacion_UCI-Alert	
Tipo de clase: Controladora	
Atributo	**Tipo**

datos	Acceso_BaseDatos
Responsabilidades	
1. Nombre.	EnviarCorreo()
Descripción.	Encargado de enviar mensajes a los correos electrónicos de los usuarios escogidos para notificarles.
2. Nombre.	NotificarBeeper()
Descripción.	Encargado de enviar mensajes al beeper de los usuarios escogidos para notificarles.

Tabla 30: Descripción de la Clase: Notificacion_Whatsup

Nombre Notificacion_Whatsup	
Tipo de clase: Controladora	
Atributo	**Tipo**
propiedades	Propiedades_NetAlert
notificacion	boolean
ip	String
Responsabilidades	
1. Nombre.	cambiarEstado()
Descripción.	Encargado de notificar cuando ha ocurrido algún cambio de estado en algún dispositivo de red.
2. Nombre.	Comprobar()
Descripción.	Encargado de iniciar la verificación acerca de si ha ocurrido algún cambio de estado
3. Nombre.	getNotificacion()
Descripción.	Encargado de permitir acceder a la notificación
3. Nombre.	getIp()
Descripción.	Encargado de permitir acceder a la dirección ip del dispositivo de red.

Fig. 43 UCI-Alert.

Glosario

Redes telemáticas: Es un conjunto de ordenadores conectados entre sí, estableciendo un instrumento integrado de medios y de aspectos lógicos soportados, con los cuales se puede establecer una comunicación bidireccional entre cada uno de los elementos integrados.

Switch: Es un dispositivo electrónico de interconexión de redes telemáticas.

Switch capa 3: Es un switch con características superiores, que trabaja con el 3er nivel de OSI, es equivalente al enrutado TCP/IP. En esta capa encontramos el protocolo IP y es la encargada del enrutamiento y de dirigir los paquetes IP de una red a otra.

Protocolo: Es un conjunto de reglas predefinidas con el propósito de estandarizar las actividades informáticas.

Dirección IP: Es un número que identifica de manera lógica y jerárquica a una interfaz de un dispositivo dentro de una red que utilice el protocolo IP.

Protocolo IP: Es un protocolo no orientado a conexión, usado tanto por el origen como por el destino para la comunicación de datos a través de una red de paquetes conmutados.

OSI: Modelo de referencia de Interconexión de Sistemas Abiertos.

TCP: Protocolo de Control de Transmisión.

Log: Es un registro de datos o información sobre quién, qué, cuándo, dónde y por qué un evento ocurre para un dispositivo en particular.

Evento: Es una ocurrencia que ha ocurrido y que ha sido registrada.

Notificación: Es el aviso de que ha ocurrido un determinado evento.

Software: Término genérico que designa al conjunto de programas que posibilitan realizar una tarea específica en un ordenador.

Patrones: Unidad de información nombrada, instructiva e intuitiva que captura la esencia de una familia exitosa de soluciones probadas a un problema recurrente dentro de un cierto contexto.

Requisitos: Capacidades o condiciones que se deben cumplir.

IDE: Entorno Integrado de Desarrollo.

Script: Es un guión o conjunto de instrucciones.