UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Título:

MÓDULO PARA LA GESTIÓN INTEGRADA DE SERVIDORES DE TELEFONÍA IP BASADO EN POLÍTICAS EMPLEANDO WBEM.

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

.

Autor(es): Anais Gutiérrez Cuza

Saymel Hernández González

Tutora: MSc. Mónica Peña Casanova

Co-tutor: Ing. Duany Baró Menéndez

La Habana, 2015.

Año 57 de la Revolución

"Ver después no vale, lo que vale es ver antes y estar preparado."

José Martí

DECLARACIÓN DE AUTORÍA

Nosotros, Anais Gutiérrez Cuza y Saymel Hernández González, declaramos ser los únicos autores de este trabajo y autorizamos a la Facultad 1 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente a los____ días del mes de ____ del año ____.

Anais Gutiérrez Cuza

_____

Firma de Autor

Saymel Hernández González

_____

Firma de Autor

Mónica Peña Casanova

_____

Firma de Tutor

Duany Baró Menéndez

_____

Firma de Tutor

AGRADECIMIENTOS:

A mi mamita linda por haberme dado la oportunidad de nacer, por cuidarme siempre con tanto afán y tanto amor, por mimarme sin pedirlo, por ser el mejor ejemplo a seguir, por tener siempre esa enorme paciencia para entenderme y ayudarme, por siempre estar a mi lado apoyándome, guiando mis acciones, educándome, enseñándome a enfrentar la vida con la frente en alto y mente positiva, por haberme transmitido esos genes de constante perseverancia, por a pesar de querer protegerme haberme dejado cometer mis propios errores y recordarme siempre que cada uno de ellos son experiencias acumuladas. Simplemente gracias por ser la mejor madre y amiga que pueda existir en este mundo.

A mi papá por ser mi continuo ejemplo de sabiduría, responsabilidad, honradez y sencillez, por haber cuidado de mí todos estos años junto a mi mami, por inculcarme siempre la importancia de estudiar constantemente. Gracias por enseñarme como ser una mejor persona.

A Mayi y a mi Neito, mis padres, por ambos quererme incondicionalmente como una hija y más quererme educarme como tal.

A mi abuela Melba y mi tía Mary por siempre estar al pendiente de todo lo que ocurre en mi vida y llenarme siempre de mucho amor y cuidado.

A mi hermana por haber contribuido en mi formación profesional y haberme ayudado tanto en la elaboración de esta tesis.

A mi familia por apoyarme siempre.

A mis amigos: mi Nana, mi Kuki, mi Mimí, Lilí, Memy, Grettuchi, Raci, Rafi y Alexis por siempre estar ahí aun estando lejos de mí, por haberme apoyado siempre sin pedirlo en los momentos de mi vida que más los he necesitado, por conocer mis sentimientos, mis pensamientos con tan solo mirarme a los ojos, por ser mis compinches de guerra y fiestas. Por apoyarme siempre en todas mis decisiones aun no estando de acuerdo, por ser personas totalmente diferentes pero con un corazón enorme en el cual me han dejado ocupar un lugarcito bien especial.

A Saymel mi compañero de tesis por regalarme su amistad estos cinco años.

A Eric por haberme dado todo su apoyo durante cuatro años en la universidad al igual que a su familia por haberme permitido ser parte de ella y siempre brindarme tanto cariño.

A mis compañeros de aula por siempre haberme brindado su ayuda incondicional.

A los muchachos del grupo de trabajo de Duany que nos ayudaron incontables veces en todo lo relacionado con la programación de la aplicación.

A todos los profesores que a lo largo de mi carrera han contribuido a mi formación profesional y personal, sobre todo a los que hoy se han convertido en grades amigos.

A mis tutores por haberme ayudado a hacer posible la materialización de esta tesis.

A mi padrino Julio y a mi abuela Mimi por ayudarme, por siempre darme fuerza para seguir y triunfar.

Anais

A mis padres por estar a mi lado en todo momento y que sin pestañear me han dado todo.

A mi compañera de tesis Anais, por haberme soportado estos 5 años sin dudar.

A mis amigos Ana, Rafael, Lorena y Danay por hacerme cuestionar todo y enseñarme que el mundo es algo más que un grano de arena.

A mis tutores, gracias por su preciado tiempo.

Saymel

DEDICATORIA:

A la luz de mis ojos, mi razón de ser, mi madre.

Anais

A mi familia por su apoyo y amor incondicional.

A todos mis amigos, compañeros y profesores que han contribuido a mi formación profesional.

Saymel

RESUMEN

La telefonía IP se ha convertido en un servicio transformador de las telecomunicaciones que ha sido impulsado por el crecimiento de las redes de banda ancha y las reducciones de costos que ella implica como resultado de un uso más eficiente de la infraestructura (soporta más canales en el mismo ancho de banda). Entre estos factores se encuentran los principales motivos por lo cual es necesario efectuar una correcta gestión de los servidores que albergan este servicio, ya que de ello depende el hecho de brindar un servicio con calidad. La gestión integrada basada en políticas se presenta como una opción viable dado que permitirá el intercambio de información de gestión en entornos heterogéneos, a través de un modelo común de información vinculando las necesidades del negocio con la infraestructura de la organización.

El Centro de Desarrollo de Tecnologías de Telecomunicaciones (CDTT) de la Empresa de Tecnología de la Información para la Defensa (XETID), que se encuentra en la Universidad de las Ciencia Informáticas de Cuba, tiene desplegado en dicha entidad y en otras organizaciones del país una solución de telefonía IP. La gestión que se realiza sobre esta infraestructura presenta entre sus principales problemas la incompatibilidad entre datos de gestión, protocolos, aplicaciones y hardware, la realización de las tareas de operación de forma manual y la existencia de diversidad, posible duplicidad e inconsistencia de la información manejada. Este trabajo propone integrar al Sistema Integral de Gestión de Redes, el cual se centra en darle solución a los problemas de gestión de red existentes en la XETID, un módulo capaz de monitorizar y controlar los servidores de telefonía IP llevando a cabo una Gestión Empresarial Basada en Web (WBEM) y una gestión basada en políticas.

Palabras Claves: Telefonía IP, Gestión Integrada, Gestión Basada en Políticas.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

El continuo avance de las Tecnologías de la Información y las Comunicaciones (TIC) ha generado disímiles oportunidades de negocio, logrando obtener beneficios satisfactorios para las empresas en estos últimos años. El crecimiento de las redes TCP/IP, el desarrollo de técnicas avanzadas de digitalización de voz, los mecanismos de control y priorización de tráfico, los protocolos de transmisión en tiempo real, así como el estudio y desarrollo de nuevos estándares que posibilitan el mejoramiento de la calidad de servicio en las redes han propiciado la creación de un entorno donde es posible brindar el servicio de telefonía sobre IP (IP del inglés *Internet Protocol*).

La telefonía IP tiene entre sus principales ventajas el bajo costo de las llamadas telefónicas, dado que con el mismo ancho de banda, una red VoIP (acrónimo de "*Voice Over Internet Protocol*") puede transportar un mayor número de veces el número de canales transportados por la Red Telefónica Pública Conmutada, PSTN (*Public Switched Telephone Network*) con una calidad similar (1). Además brinda nuevas facilidades para los servicios tradicionales, la unificación de mensajería, y la fusión de las infraestructuras para voz y datos. Permite integrar servicios que están segmentados y que se facturan de manera independiente en las redes PSTN por temas tecnológicos entre los que se encuentran: el redireccionamiento de llamada, la trasmisión de video llamadas, la integración de buzón de voz, el fax, la autenticación, el control de acceso y la seguridad.

En una infraestructura de telefonía IP entre los elementos que la componen se destacan los servidores, *routers*, *switchs*, teléfonos IP[1], *softphone*[2] y adaptadores ATA (ATA del inglés *Analog Telephony Adapter*). Debido a la heterogeneidad de dispositivos existentes en una red de telefonía IP y a los requerimientos especiales que deben tener en cuenta los operadores en cuanto a señalización, consumo de ancho de banda, almacenamiento para los mensajes de voz, retardo, entre otros, es importante implementar acciones de control sobre la infraestructura que soporta la telefonía IP.

Actualmente existen en el mundo diversas soluciones que le permiten a las empresas poder implementar y gestionar de manera satisfactoria la tecnología VoIP y aprovechar al máximo sus ventajas, sin que se

[1] Dispositivo que encapsulan la señal de voz en paquetes IP y pueden implementar los servicios de la telefonía IP.
[2] Funcionalidades de un teléfono implementadas por software.

produzca un impacto negativo en el rendimiento de los datos en la red convergente. Según la Unión International de las Telecomunicaciones (UIT) además de los servicios de VoIP sobre línea fija, la VoIP móvil está emergiendo especialmente en los países en desarrollo en los que la rápida expansión de las redes móviles ha aumentado la disponibilidad de servicios.

Según el reporte sobre Sociedad de la Información 2014 de la UIT, Cuba está en el lugar 125 de 166 países en el Índice de Desarrollo de las TIC, para efectuar este ranking se analizaron 11 indicadores agrupados en tres subíndices principales: acceso, uso y capacidades de las TIC, se encuentra además, entre los países menos conectados con un 18% en telefonía celular incluyendo los TFA (Teléfonos Fijos Alternativos) y es uno de los 4 países de América que no cuenta con una red de banda ancha inalámbrica (2). Para revertir esta situación, se celebró en febrero del 2015 el Primer Taller Nacional de Informatización y Ciberseguridad; en los ejes estratégicos de las bases y prioridades para el perfeccionamiento de la sociedad en Cuba, se plantea: “impulsar la integración de los servicios de telefonía, televisión y datos utilizando las redes fijas y móviles, conformar una red nacional de centros de datos, garantizar la expansión constante de la infraestructura de telecomunicaciones que permita generalizar el uso de internet, impulsar el despliegue de la banda ancha en el país, incrementar la comercialización de equipamiento informático con precios asequibles e implementar programas de renovación tecnológica que favorezcan la reducción de la obsolescencia” (3). Una alternativa para realizar un despliegue a bajo costo de los servicios de voz, es el empleo de la VoIP, algunas instituciones cubanas ya han dado pasos en este sentido.

En la Universidad de las Ciencias Informáticas de Cuba, radica una entidad la cual se denomina Empresa de Tecnología de la Información para la Defensa (XETID) donde existe un centro de desarrollo que se encarga de implementar soluciones de comunicaciones unificadas, denominado Centro de Desarrollo de Tecnologías de Telecomunicaciones (CDTT). Dicho centro tiene desplegado en la propia entidad y en otras organizaciones del país una solución de telefonía IP llamada PlaTel, la cual proporciona un conjunto de servicios de telefonía IP (correo de voz, transferencia de llamadas, salones de conferencia, entre otros). Sin embargo la manera en que hoy se efectúa la gestión de dicha infraestructura tiene los siguientes problemas:

- Incompatibilidad entre datos de gestión, protocolos, aplicaciones y hardware.

- Diversidad, posible duplicidad e inconsistencia de la información.

- Las tareas de operación sobre la infraestructura que soporta la telefonía IP se realizan de manera manual.

- No se hace una adecuada planificación, supervisión y control de los servidores de telefonía IP.

-No existe un mecanismo para ejecutar de manera automática las políticas que tiene establecida la organización con respecto a este servicio, sobre la infraestructura que lo soporta y que considere las tecnologías para la gestión integrada.

A partir del análisis de lo antes expuesto se plantea el siguiente **problema a resolver**: ¿Cómo mejorar la gestión de servidores de telefonía IP de XETID de manera que se logre el control automatizado de los elementos monitorizados?

Para dar solución al problema planteado se define como **objeto de estudio** el proceso de gestión de redes y servicios telemáticos.

El **objetivo general** será implementar un módulo para la gestión de servidores de telefonía IP de XETID basado en políticas, de manera que se logre el control automatizado de los elementos monitorizados.

Con el propósito de complementar el objetivo general se han establecido los siguientes **objetivos específicos**:

- Elaborar el estado del arte y marco teórico mediante el estudio y análisis de la gestión de los servidores que soportan la voz sobre IP.
- Realizar el diseño y la modelación del módulo para la gestión de servidores de telefonía IP.
- Implementar el módulo de gestión de servidores de telefonía IP.
- Validar el módulo desarrollado.

La presente investigación se enfoca en el **campo de acción** de la gestión de servidores de telefonía IP.

Idea a defender

Un módulo para la gestión de servidores de telefonía IP basado en políticas permitirá que se logre el control automatizado de los elementos monitorizados.

Para guiar la realización del trabajo se establecen un conjunto de **tareas de investigación:**

1. Análisis de los estándares de gestión integrada de redes y su evolución.
2. Análisis de los principales conceptos asociados a la telefonía IP.
3. Análisis de las tecnologías y estándares del Grupo de Trabajo de Gestión Distribuida.
4. Comparación entre los diferentes sistemas de gestión de servidores de telefonía IP.
5. Análisis de la arquitectura de gestión de redes basada en políticas (PBNM) propuesta por el grupo de trabajo de *Internet Engineering Task Force* (IETF) y arquitectura PBMN utilizando WBEM (*Web Based Enterprise Management*).
6. Análisis y diseño del módulo para la gestión integrada de servidores de telefonía IP basado en políticas utilizando WBEM.
7. Implementación del módulo para la gestión integrada de servidores de telefonía IP basado en políticas utilizando WBEM.
8. Validación del módulo en la infraestructura de servidores XETID.

Durante el proceso de investigación, se emplearon los métodos de investigación siguientes:

Métodos teóricos

- Histórico – Lógico: Con el objetivo de identificar teóricamente cómo han evolucionado en el tiempo los Sistemas de Gestión de Red, en especial la Gestión de servidores de telefonía IP; así como las herramientas y tecnologías utilizadas en el monitoreo de los recursos de hardware de servidores de telefonía IP.
- Hipotético – Deductivo: Puesto en práctica en la formulación de la idea a defender y su verificación a partir del análisis de los resultados obtenidos.
- Analítico – Sintético: Empleado para el análisis de los elementos esenciales referentes a las teorías, documentos y literatura en general relacionada con la gestión de servidores de telefonía IP y las herramientas de monitoreo de hardware de servidores.

- Modelación: Empleado en la representación en forma de diagramas, de las características del sistema, relaciones entre objetos y las actividades que intervienen en los procesos gestión de servidores de telefonía IP.
- Análisis dinámico: Se aplica en la ejecución iterativa de las funcionalidades del módulo de configuración desarrollado para la medición de su funcionamiento, permitiendo el constante perfeccionamiento del mismo.
- Análisis estático: Se practica a través de la ejecución e inspección minuciosa de la configuración de servidores o determinada parte de este, para la detección de posibles errores y su posterior corrección.

Método empírico

- Observación: Permite la adquisición de información relevante, de manera visual sobre cómo se gestiona el sistema al cual se integrara el módulo y cómo se realiza una gestión de servidores de telefonía IP.

Justificación de la investigación

Los sistemas de gestión integral en la actualidad son un eslabón fundamental para lograr el éxito de cualquier organización ya que permite asociar los intereses del negocio a las tecnologías existentes en la organización.

Las entidades que cuentan con soluciones VoIP necesitan lograr esta gestión integral ya que por lo general las soluciones de comunicaciones se ven ajenas a los procesos medulares de la organización. Por tal motivo surge el presente trabajo de diploma con el objetivo de integrar las políticas asociadas al negocio a la infraestructura de VoIP que existe en la organización. Además mediante el modelo común de información se vinculan los conceptos del negocio con los tecnológicos obteniéndose como aporte un módulo de gestión de servidores IP con las características antes mencionadas.

Estructura del documento

El presente trabajo está estructurado en tres capítulos, los cuales se describen a continuación:

Capítulo 1: Fundamentación teórica

En este capítulo se presentan los conceptos fundamentales relacionados con la telefonía IP y la gestión de servidores de telefonía IP, las principales características de WBEM y de la arquitectura PBNM. Se efectúa un análisis de los modelos de gestión integrada. Se realiza un estudio de las propiedades más importantes de un grupo de aplicaciones de monitorización de servidores definidas por el cliente teniendo en cuenta las capacidades que presenta su infraestructura. Se lleva a cabo un estudio de las principales herramientas y tecnologías que se emplearán en el desarrollo del módulo para la gestión de servidores de telefonía IP basado en PBNM utilizando WBEM.

Capítulo 2: Características del módulo

En este capítulo se realiza la descripción de la propuesta de solución. Además se elabora el modelo de dominio, se definen y especifican los requisitos funcionales y no funcionales del sistema, así como el diseño de la arquitectura propuesta y los prototipos de interfaz de usuario. Además según los requisitos funcionales definidos y la arquitectura propuesta se elaborará el diseño del módulo para poder efectuar su posterior implementación.

Capítulo 3: Implementación y prueba

En este capítulo se mostrarán los diagramas de componentes y despliegue de la solución, se codifica dicha solución diseñada y se realiza su validación a través de diferentes tipos de pruebas.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

Los servicios de telecomunicaciones se encuentran actualmente en constante cambio y desarrollo. Ellos juegan un importante rol en la mayoría de las organizaciones, dado que una gran parte de las actividades que se realizan en las mismas dependen del adecuado funcionamiento de estos servicios. Dentro de los servicios de telecomunicaciones, se destaca la proliferación que ha ocurrido en los últimos años en el uso de la telefonía IP (1). Debido a la heterogeneidad de las infraestructuras que soportan este servicio, resulta imperioso lograr realizar una correcta gestión de servidores de telefonía IP porque la comunicación en muchas instituciones y entre ellas, depende del servicio que estos servidores hospedan. La correcta gestión de la telefonía IP propicia una disminución de costos y una mejora en la eficiencia, disponibilidad y rendimiento de la red. Además asegura un incremento de la productividad de operación de dicha red, siendo capaz con el diseño, implementación y operación de procedimientos y el uso de herramientas integradas lograr en la gestión de la red un comportamiento autónomo.

Para la implementación de un módulo que realice una gestión integrada basada en políticas de servidores de telefonía IP es indispensable explicar dos conceptos fundamentales: Telefonía IP y la Gestión de redes, por lo que parte de este capítulo está dedicado a exponer dichos conceptos, haciendo énfasis en los principales elementos que tributan al éxito de esta investigación. En el presente capítulo también se realiza un análisis de un conjunto de modelos de gestión integrada, un estudio y una evaluación de algunas de las herramientas de monitorización más utilizadas actualmente. Se exponen las características esenciales de las herramientas, las tecnologías y la metodología de desarrollo de software que se utilizarán para la implementación de la solución.

1.2 Fundamentos teóricos asociados a la telefonía IP y a la gestión de redes

1.2.1 Telefonía IP

La evolución de la telefonía en su comienzo fue un fenómeno paulatino, hecho que marca la diferencia con respecto a los últimos años donde su desarrollo ha sido ferviente y acelerado. Su punto de inicio fue la telefonía analógica o tradicional, como también se le conoce, la cual está basada en un sistema denominado

conmutación de circuitos. Los circuitos son los encargados de transmitir la señal eléctrica de la voz de manera analógica llevando adicionalmente la señalización necesaria para establecer, mantener y terminar una llamada (4). Luego, producto del creciente uso y ubicuidad de Internet nacen una serie de servicios entre los que se destacan la telefonía IP.

La **VoIP** es el acrónimo de "*Voice Over Internet Protocol*", es un estándar de la UIT, creado en 1996 (5), que se puede definir como una solución tecnológica que sirve para transmitir paquetes de voz sobre una red de datos basada en el estándar IP (4).

La **Telefonía IP** (llamada también **Telefonía sobre IP**) emplea la tecnología de VoIP para brindar todos los servicios de telefonía sobre la red de Internet. A diferencia de la Red Telefónica Publica Conmutada que utiliza conmutación de circuitos, la telefonía IP envía múltiples conversaciones a través del mismo enlace codificadas en paquetes y en flujos independientes. Cuando se produce un silencio en una conversación, los paquetes de datos de otras conversaciones pueden ser transmitidos por la red, lo que implica un uso más eficiente de la misma (5). El uso de tecnología asociada a la telefonía IP propicia una disminución de los costos, tanto de comunicación como del mantenimiento de las infraestructuras de telefonía. Además posee elevadas posibilidades de escalabilidad, portabilidad e integración con las redes de área local y los recursos de las empresas e Internet (6).

Como toda solución tecnológica, la telefonía IP no queda exenta de tener desventajas en cuanto a su uso, pero las mismas se suplen, tanto en importancia como en número, con cada una de sus ventajas. Los **principales beneficios que se obtienen al aplicar VoIP** son:

- **Reducción de costos**: Uno de los factores esenciales que propicia que los costos de la telefonía IP sean realmente bajos es el uso que se hace de Internet como medio de transporte para las comunicaciones. El pago de llamadas queda reducido al pago de la conexión a Internet de la que se disponga. Los costos asociados a la creación, desarrollo y mantenimiento de la infraestructura telefónica de la empresa disminuyen de manera considerable, una de las causas es el uso de teléfonos implementados por software en lugar de hardware, que son más caros (6).
- **Escalabilidad**: Debido a la posibilidad de instalar este servicio de forma distribuida (aunque también puede ser instalado de forma centralizada), haciéndolo más flexible y dispuesto a asumir mejoras

tecnológicas, dicho servicio llega a tener gran escalabilidad, tanto si se habla de servidores de telefonía como del número de usuarios soportados por el servicio (6).

- **Integración con redes locales y recursos existentes e Internet**: La integración de VoIP con cualquier red local de una empresa u organización es sencilla debido a que la misma trabaja sobre redes TCP/IP. Esto permite obtener beneficios considerables porque al disponer de conexiones de red existentes se eliminan las redes telefónicas tradicionales, de ahí que no resulte necesario extender la red telefónica de la empresa (6).
- **Mayor eficiencia**: La telefonía IP tiene como una de sus ventajas esenciales que al mismo tiempo que se tiene una conversación se puede utilizar el canal de comunicación para otros objetivos, obteniendo como resultado hacer un mejor uso del ancho de banda disponible. Esto se ratifica al aplicar compresión o eliminación de redundancia en la transmisión de voz (6). El desarrollo de diferentes tipos de códec para VoIP ha permitido que la voz se codifique en paquetes de datos cada vez de menor tamaño. Esto trae como consecuencia que las comunicaciones de voz sobre IP requieran un consumo cada vez menor del ancho de banda, junto al avance de distintas tecnologías de banda ancha este tipo de comunicaciones se hacen muy populares (5).
- **Características y prestaciones diferenciadoras**: La telefonía IP comparada con la telefonía tradicional queda en una posición sumamente ventajosa debido a que la misma brinda una elevada cantidad de servicios con alta calidad y fiabilidad. Los buzones de voz, las salas de conferencias (sin límite de participantes), distribución y gestión automática de llamadas, contestador automático, interacción con la red de telefonía tradicional, reconocimiento de llamadas, creación de números virtuales, transmisión de diferentes tipos de datos (video, imágenes, entre otros, debido a que trabaja en redes de conmutación de paquetes), uso de fax virtuales (mejorando la calidad, a bajo coste, sin máquina fax), respuesta interactiva de voz (IVR), siendo estos algunos de los servicios más destacados (6).
- **Flexibilidad**: Las soluciones VoIP en general ofrecen una gran flexibilidad tanto en su configuración como en su uso, sobre todo si hablamos de soluciones de software libre. Además, la red sobre la que trabaja VoIP es completamente flexible, no es necesario disponer de un escenario o topología de red determinado (6).

Con el fin de lograr una adecuada **calidad de servicio de la VoIP** (QoS) es necesario tener en cuenta diferentes factores, entre los que se destacan:

- ***Jitter***: Es la variación en el tiempo de la llegada de los paquetes causada por la congestión en la red, pérdida de sincronización o por las diferentes rutas seguidas por los paquetes para llegar al destino. Este es un problema frecuente en enlaces lentos o congestionados. Se recomienda que el *jitter* entre el punto inicial y final de la comunicación debiera ser inferior a 100 ms dado que si es mayor uno de los efectos que provoca es el desfase entre la imagen y el sonido percibido. Si el valor es menor a 100 ms el *jitter* puede ser compensado de manera apropiada. Posibles soluciones a este problema pueden ser: el uso de colas con prioridad, reservas de ancho de banda o enlaces con mayor velocidad (7).
- **Latencia**: Es el tiempo que demora en llegar un paquete a su destino, a la latencia también se le llama retardo. Este es un problema de las redes de telecomunicaciones, en ocasiones producto a las largas distancias que debe recorrer la información, un ejemplo de ello son los enlaces vía satelitales. Se recomienda que los valores de la latencia entre el punto inicial y final de la comunicación debieran ser inferiores a 150 ms. El oído humano es capaz de detectar latencias de unos 250 ms y 200 ms en el caso de personas bastante sensibles. Si se supera ese umbral la comunicación se vuelve molesta. Reservar un ancho de banda de origen a destino o señalizar los paquetes con valores de ToS (*Type of Service*) para intentar que los equipos sepan que se trata de tráfico en tiempo real y lo traten con mayor prioridad, puede ser una de las variantes para solucionar este problema (7).
- **Pérdida de paquetes**: Las comunicaciones en tiempo real están basadas en el protocolo UDP (*User Datagram Protocol)*. Este es un protocolo no orientado a conexión por lo que si se produce una pérdida de paquetes no se reenvían. La pérdida de paquetes puede ocurrir por descartes de paquetes que no llegan a tiempo al receptor. Cuando ocurren pérdidas de paquetes aislados el problema no es de gran magnitud, dado que la voz es muy predictiva o redundante y se puede recomponer de una manera bastante óptima; siendo todo lo contrario cuando ocurre la pérdida de paquetes en ráfagas. La pérdida de paquetes máxima admitida para que no se degrade la comunicación deber ser inferior al 1%. Sin embargo este hecho va sujeto al tipo de códec que se utiliza ya que cuanto mayor sea la compresión más perjudicial será el efecto de la pérdida de

paquete. Una posible solución a este problema es no transmitir los silencios. Producto a que las conversaciones constan de varios momentos de silencio resulta factible solo transmitir cuando haya información audible liberando así los enlaces y evitando fenómenos de congestión (7). Aunque también pudieran considerarse como soluciones emplear técnicas de corrección de errores, reducción de variaciones de retardo, enmascaramiento de los paquetes de voz perdidos y si se hace necesario el sobredimensionamiento de la red (solución más típica actualmente en VoIP) (8).

A continuación en la Figura 1 se muestra un esquema con los **componentes principales de la red de telefonía IP**:

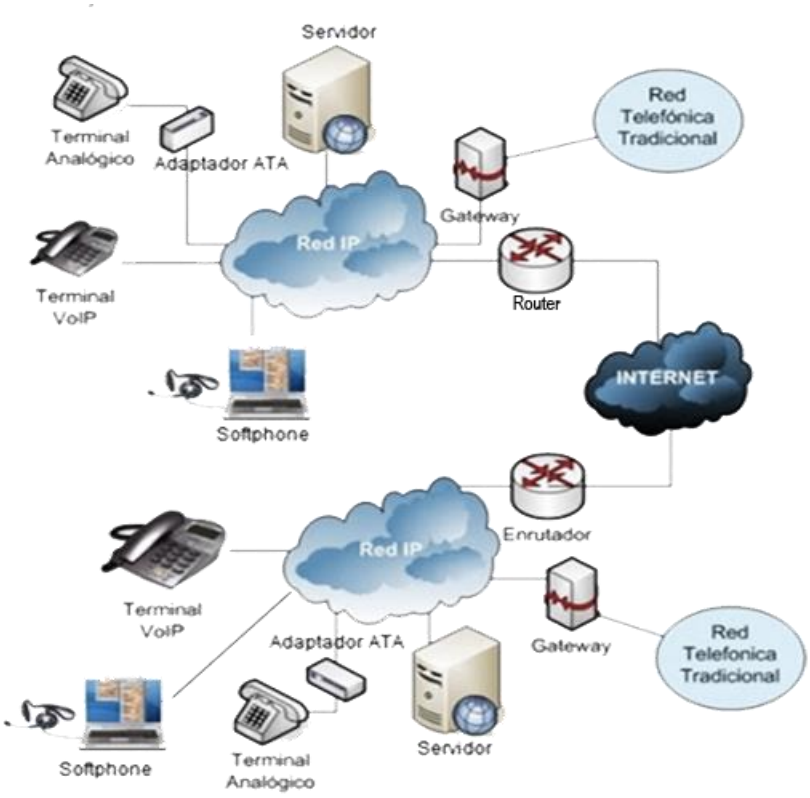


Figura 1. Componentes principales de la red de telefonía IP (Elaboración propia).

- **Terminales**: Los terminales que se observan en la Figura 1 son los teléfonos analógicos y VoIP, llamados también terminales y los *softphone* (Funcionalidades de un teléfono implementadas por software). Estos son los dispositivos que utilizan los usuarios para comunicarse, implementados tanto

en hardware como en software realizan las funciones de los teléfonos tradicionales (9). Aunque VoIP implica la transmisión digital de voz en paquetes, el propio teléfono puede ser analógico, como los que usamos habitualmente, o digital, es decir, un teléfono IP. La voz puede ser digitalizada tanto antes de empaquetar como a medida que se empaqueta (proceso en el que porciones del flujo de bits son agrupadas en paquetes IP), produciéndose siempre el proceso de codificación en el terminal emisor, y la decodificación en el terminal que recibe la llamada. Este proceso de codificar y decodificar se realiza utilizando un conjunto de algoritmos llamados Códec (7).

- ***Gateways***: Los *gateways* se encargan de conectar las redes VoIP con las redes de telefonía tradicional de forma transparente (9).
- **Servidores**: Los servidores tienen como función principal manejar las operaciones de base de datos en tiempo real. Entre las operaciones que manejan los servidores se tienen la recolección, el enrutamiento, la administración y control del servicio, el registro de los usuarios, entre otras (8).
- **Adaptador analógico IP (ATA)**: Es un adaptador de teléfonos analógicos que se encarga de transformar la señal analógica a digital, este adaptador permite conectar teléfonos comunes a la computadora o a la red para utilizarlos con VoIP (8).
- **Red IP**: Provee la conectividad entre los terminales, la misma puede ser una red IP privada, una Intranet o el propio Internet (5).
- **Red Telefónica Tradicional**: La Red Telefónica Pública Conmutada, PSTN (*Public Switched Telephone Network*) o Red Telefónica Tradicional es una red con conmutación de circuitos tradicionales optimizada para comunicaciones de voz en tiempo real. (10)
- **Routers**: Este dispositivo es el encargado de guiar la transmisión de los paquetes de datos a través de la red.

Protocolos de VoIP

La transmisión de voz sobre IP engloba gran cantidad de protocolos. Estos protocolos se pueden agrupar en tres grupos (4):

- **Protocolos de señalización:** En este grupo se agrupan los protocolos encargados de realizar las tareas de establecimiento de sesión, control del progreso de la llamada, entre otras (4). Estos protocolos surgen debido a las necesidades de calidad de servicio que hacen que sea necesaria una

gestión de recursos que asegure la optimización de la capacidad de transporte de la voz de extremo a extremo (8). Dentro de estos protocolos se encuentran:

- **Protocolo H.323:** Es un estándar creado por la UIT que define los protocolos, componentes y procedimientos que proveen sesiones de comunicación multimedia para las comunicaciones de audio en tiempo real, vídeo y datos en las redes IP. Con este protocolo no se garantiza calidad de servicio. Es independiente de la topología de la red y admite pasarelas, permitiendo usar más de un canal de cada tipo (voz, video, datos) al mismo tiempo (8).
- **Protocolo SIP** (*Session Initiation Protocol*): Es un protocolo de señalización a nivel de aplicaciones para el establecimiento y gestión de sesiones con múltiples participantes. Define el proceso de llamadas telefónicas, video-conferencias y otras conexiones multimedia sobre Internet. El propósito de SIP es la comunicación entre dispositivos multimedia, y este se logra gracias a los diferentes estándares existentes que son compatibles con SIP. Para el transporte de datos se utiliza el protocolo RTP/RTCP. Para negociar las capacidades de los participantes tales como direcciones IP, medio a utilizar, tipo de codificación, entre otras se usa el SDP (*Session Description Protocol*), el cual permite describir el contenido multimedia de la sesión. SIP es un protocolo que funciona tanto sobre UDP como TCP (5).

- **Protocolo de transporte de voz:** En este grupo se encuentran los protocolos encargados de realizar la transmisión de audio y video en tiempo real los cuales están definidos en la norma RFC 1889, entre ellos se destacan el RTP y RTCP (8).
  - **Protocolo RTP** (*Real-time Transport Protocol*): Es un protocolo a nivel de servicio utilizado para la transmisión de audio o video en tiempo real sobre Internet, permitiendo la sincronización de datos multimedia desde diferentes aplicaciones (8).
  - **Protocolo RTCP** (*Real Time Control Protocol*): Es un protocolo de control en tiempo real encargado de regular el intercambio de mensajes de control entre los participantes en una sesión multimedia. La información que este protocolo maneja hace referencia a la calidad de servicio proporcionada por RTP mediante varios parámetros como pueden ser: la latencia, la tasa de paquetes recibidos, paquetes perdidos, *jitter*, entre otros (8).

- **Protocolos de arquitectura TCP/IP:** En este grupo se encuentran los protocolos básicos de las redes IP que forman la base sobre la cual se añaden los protocolos de transporte de voz y que se utilizan para realizar el transporte de datos (8). Entre ellos se encuentran:
    - **Protocolo IP** (*Internet Protocol*): Es un protocolo no orientado a la conexión que trabaja a nivel de red donde la información se envía en paquetes llamados paquetes IP. Este protocolo ofrece un servicio de datagramas no fiable llamado "mejor esfuerzo" (*best effort*), posibilitando el envío de la información lo mejor posible pero sin garantizar que los paquetes lleguen a su destino (4).
    - **Protocolo TCP** (*Transmission Control Protocol*): Es un protocolo de transporte que se transmite sobre IP. Este protocolo se encarga de controlar que los datos transmitidos se encuentren libres de errores y que sean recibidos por las aplicaciones en el mismo orden en que fueron enviados, en caso de perderse datos en la transmisión introduce mecanismos para que estos datos sean reenviados (4).
    - **Protocolo UDP** (*User Datagram Protocol*): Es un protocolo de transporte que se transmite sobre IP. Este protocolo es considerado más rápido que TCP porque reduce la cantidad de información extra en los paquetes por lo que facilita la transmisión de información en tiempo real como la voz. UDP tiene como desventaja que no comprueba la ocurrencia de errores ni el orden en que tienen que ser entregados los paquetes (4).

Para lograr un correcto funcionamiento de los elementos que componen una infraestructura de telefonía IP mencionados con anterioridad es necesario que exista interoperabilidad entre ellos, esto se puede lograr llevando a cabo una gestión integrada de los componentes de la red de telefonía IP. A continuación se exponen un conjunto de conceptos referentes al tema de esta investigación.

1.2.2 La Gestión de redes

La gestión de redes es el conjunto de capacidades que permiten el intercambio y procesamiento de información de gestión a fin de ayudar a las administraciones a realizar sus actividades con eficiencia. Permite la planificación, organización, supervisión y control de los elementos que forman una red para garantizar un nivel de servicio de acuerdo a un costo. Su objetivo es mejorar la disponibilidad y rendimiento de las redes e incrementar su efectividad, lográndose una mayor productividad en la institución y un aumento

de la satisfacción de los usuarios (11). El monitoreo y control son las formas de actuación de la gestión de redes, en la monitorización es donde se engloban todas las operaciones de obtención de datos acerca del estado y comportamiento de los recursos de una red y en el control, teniendo en cuenta el procesamiento de los datos obtenidos en la monitorización se realiza una configuración a la red gestionada (12) (13) (14).

Los sistemas de gestión han experimentado una continua evolución consecuentemente con el desarrollo de las infraestructuras que gestionan. Las infraestructuras actuales se caracterizan por presentar una gran heterogeneidad. Entre los problemas que presentaba la gestión heterogénea se encontraban: La suma de las soluciones de gestión propietaria, que proporcionaba cada fabricante de tecnología, caracterizadas por la gran variedad de Interfaces a manejar por parte de los operadores, el hecho de que las herramientas de gestión estaban orientadas a componentes o elementos de la red, la no existencia de una base de datos común, la imposibilidad de obtener una única perspectiva o visión de toda la red y la dependencia tecnológica que todo esto suponía para las organizaciones; por lo que a partir de los años 90 comienza a desarrollarse la gestión integrada como una respuesta de los organismos internacionales de normalización a estos problemas de la gestión heterogénea. Esta nueva forma de gestión está presente hasta la actualidad y permite la interconexión, de una manera abierta, de los elementos de la red y las aplicaciones de gestión. Para que exista una verdadera gestión integrada hay dos factores fundamentales a tener en cuenta que son: la normalización de las comunicaciones, que tiene como objetivo poder intercambiar información entre los distintos componentes del sistema de gestión y el segundo factor a tener en cuenta es la normalización de la información que tiene como finalidad obtener una definición sintácticamente uniforme de todos los elementos gestionados independientemente del fabricante y esto se logra gracias a un modelo común de información (12) (13).

Para llevar a cabo la forma de gestión de red antes mencionada se crearon varios modelos que realizaban su implementación. Entre dichos modelos se encuentran: el de gestión de red para Internet, el de gestión de red OSI (*Open Systems Interconnection*), el de gestión para aplicaciones distribuidas y el modelo de gestión empresarial basado en web (WBEM). Estos modelos normalizaron el protocolo de comunicaciones, la estructura de la información de gestión y los conjuntos de definiciones de dicha información de gestión. A continuación se realiza un resumen de las principales características de estos modelos.

- **Modelo de gestión de red para Internet**

El modelo de gestión de red para Internet está basado en el **protocolo** SNMP (*Simple Network Management Protocol*). Este protocolo es considerado un estándar y fue definido por el IETF. SNMP permite el intercambio de información de administración entre dispositivos de red dándoles a los administradores la posibilidad de supervisar el funcionamiento de la red, buscar y resolver sus problemas, y planear su crecimiento (15).

Una red administrada a través de SNMP consta de tres componentes claves:

- **Dispositivo Administrado:** Es un dispositivo o elemento de red, como también se le suele nombrar, que contiene un agente SNMP y reside en la red gestionada. Estos elementos pueden ser *routers*, servidores, *switchs*, computadoras, entre otros.

- **Agente:** Es un módulo de un sistema de administración de red que reside en un dispositivo administrado. Un agente posee un conocimiento local de información de administración (memoria libre, número de paquetes IP recibidos, rutas, etcétera), la cual es traducida a un formato compatible con SNMP y organizada en jerarquías.

- **Sistema Administrador de Red** (NMS): Ejecuta aplicaciones que supervisan y controlan a los dispositivos administrados. Los NMS's proporcionan el volumen de recursos de procesamiento y memoria requeridos para la administración de la red.

En cuanto a **estructura de información** de gestión este modelo definió SMI (*Structure of Management Information*) la cual define una estructura muy simple. SMI emplea el concepto de *Object-Types* (objetos) los cuales son simples variables escalares sobre las que solo se definen operaciones de lectura y escritura por lo que las funciones de gestión únicamente van a permitir alterar o inspeccionar el valor de las variables que se definan y además contienen el valor de determinadas características de funcionamiento de los recursos.

En lo que refiere a **definición de la información** el modelo de gestión de red para Internet definió un conjunto de información estándar conocido como la MIB (*Management Information Base*). La MIB es una colección de objetos gestionados que representan una visión abstracta de la configuración o del estado del

elemento de red administrado (16). Esta información se encuentra organizada jerárquicamente. Existen varios tipos de MIBs:

1. Estándares: MIB-I (RFC 1156), MIB-II (RFC 1213 (16))
2. Experimentales
3. Privadas

- **Modelo de gestión de red OSI**

El modelo de gestión de red OSI se utiliza sobre todo para la gestión de equipos y servicios en redes de telecomunicaciones a través de TMN (*Telecommunication Management Network*). Este protocolo se caracteriza por contener definido en sí otros modelos de gestión de sistemas OSI como son: modelo funcional, modelo de organización, modelo de comunicación y el modelo de información (16).

Este modelo emplea como **protocolo** CMIP, el cual se considera un protocolo potente capaz de balancear la carga de la gestión entre el gestor y los agentes que están a cargo del control y la vigilancia de la red. Se encuentra situado en el nivel de aplicación del modelo OSI. Es un protocolo orientado a conexión lo cual hace que tenga mayor fiabilidad pero a cambio introduce una sobrecarga en las comunicaciones de gestión. Su uso es válido para entornos donde se pueden implementar agentes con un alto grado de complejidad. Este protocolo proporciona el servicio CMIS (*Common Management Information Service*) el cual contempla las características de la comunicación gestor-agente en los servicios que ofrece (servicio de notificación, de operación y de asociación de gestión) y además proporciona un conjunto de métodos (Alcance, Filtrado y Sincronización) que brindan la posibilidad de invocar servicios de operación sobre un conjunto de objetos gestionados de forma simultánea (16).

En lo que a **estructura de la información** se refiere OSI definió GDMO (*Guidelines for the Definition of Managed Objects*). Esta es una estructura orientada a objetos con gran capacidad expresiva. Utiliza una serie de conceptos que le atribuyen gran potencia a esta estructura entre los que se encuentran: Encapsulamiento y Clase de objetos (Especialización, Herencia, Árbol de herencia, Herencia múltiple entre otros que son introducidos por el concepto de clase de objetos). Especifica la notación que se tiene que utilizar para la definición de las clases de objetos gestionados, también llamada GDMO, la cual se basa en

el concepto de plantilla o macro. Además de la sintaxis y la notación, GDMO incluye ciertos consejos para ayudar a la construcción de MIB OSI entre los que están: tener claro los requisitos a satisfacer en la definición de los objetos gestionados, adecuar la complejidad de la definición de una clase de objetos gestionados a la de los recursos reales a gestionar y reutilizar la máxima cantidad posible de definiciones ya existentes, entre otros (16).

En cuanto a **definición de la información** de gestión este modelo hace uso de las MIB OSI. La MIB OSI es la recopilación de las definiciones de objetos gestionados de un conjunto de recursos gestionados. Incluye las definiciones de las clases, atributos, acciones, comportamientos, notificaciones, paquetes condicionales, ligaduras de nombrado y tipos ASN.1 (*Abstract Syntax Notation One*) que estén asociados a dichas clases. Sus componentes se describen con la sintaxis de GDMO. Todos los elementos definidos en una MIB OSI deben tener asignado un Identificador de Objeto. Existen en la actualidad gran variedad de MIBs definidas y normalizadas (16).

Estos modelos de gestión presentan como principal problema la incompatibilidad entre ambos, dado que cada uno utiliza su propio protocolo, su propia estructura de la información y sus propias definiciones de información. Por tanto se propone analizar el modelo de gestión empresarial basado en la web el cual brinda la posibilidad de integrar los modelos anteriormente estudiados a través de un modelo común de información (16).

- **Modelo de gestión empresarial basado en web**

Uno de los estándares, que en la actualidad ha tenido mayor impacto para lograr una gestión integrada es la iniciativa de Gestión Empresarial Basada en la Web o WBEM (*Web Based Enterprise Management*) del Grupo de Trabajo de Gestión Distribuida o DMTF (*Distributed Management Task Force*). WBEM es un conjunto de tecnologías de gestión y estándares de Internet que unifican la gestión en ambientes de computación distribuida, facilitando el intercambio de datos a través de diferentes tecnologías y plataformas (17).

Estándares WBEM

1. Modelo de Información Común (CIM, *Common Information Model*)

Es un modelo de referencia para información de gestión jerárquica, orientado a objeto, extensible por el usuario y que puede ser representado con el Lenguaje de Modelado Unificado (UML), mantiene una definición común de información de gestión para describir entornos de cómputo, redes y almacenamiento. Las definiciones comunes de CIM permiten que proveedores diversos intercambien la información de gestión enriquecida semánticamente a través de la red. Facilita que los datos de gestión se entiendan de manera común en diferentes aplicaciones de gestión.

CIM está compuesto por esquemas que incluyen modelos, clases, propiedades y métodos e infraestructura o especificaciones para la integración con otros modelos de gestión.

Dentro del esquema CIM se incluyen diferentes modelos cada uno con una finalidad específica, como son: los **modelos de núcleo,** que proporcionan el vocabulario básico para describir sistemas gestionados y que son aplicables a todas las áreas de gestión, los **modelos comunes** a las áreas de gestión especiales pero que son independientes de una u otra tecnología y un **modelo de extensiones** que representa extensiones específicas de la tecnología del esquema común. Además proporciona un formato para representar los objetos gestionados MOF (*Managed Object Format*) que es independiente de la plataforma y el lenguaje de programación (18).

2. XMLCIM

El DMTF ha desarrollado un DTD (*Document Type Definition*) para CIM, el cual define cómo los elementos CIM pueden ser representados por elementos XML (*eXtensible Markup Language)*. La razón para representar CIM en XML, es el hecho de que XML se ha convertido en el principal formato para representar datos sobre Internet y el objetivo de WBEM es usar estándares basados en Web. A pesar de no ser la única manera en que se pueden representar los datos con CIM, sigue siendo la más extendida (17).

3. Operaciones CIM sobre HTTP (*Hyper Text Transport Protocol*)

Es una especificación de cómo intercambiar información CIM sobre el protocolo HTTP. Toda la información CIM que es intercambiada entre clientes y servidores se realiza a través de mensajes CIM. Estos mensajes son independientes del protocolo, por tanto pueden ser enviados sobre cualquiera de ellos. Sin embargo,

HTTP fue seleccionado como el protocolo para la transportar mensajes CIM debido a su amplio uso en Internet. (17)

Arquitectura WBEM

La arquitectura del estándar WBEM consta de los siguientes elementos: el CIMOM (CIM *Object Manager*), los proveedores y los clientes.

- CIMOM: Es la parte central de WBEM. Tiene un repositorio donde almacena todos los esquemas CIM. Es el responsable de manejar la interacción entre las aplicaciones de gestión y los proveedores de objetos. Cuando una aplicación hace una petición, el CIMOM se comunica con el repositorio (información estática), o con el proveedor apropiado (información dinámica), dependiendo del tipo de información requerida (19).
- Proveedores: Pueden ser vistos como una interfaz entre el recurso gestionado y el CIMOM. Los datos proporcionados por un proveedor son llamados datos dinámicos. Cuando el CIMOM requiere datos dinámicos, el proveedor los obtiene desde el recurso gestionado y los retorna al CIMOM.
  Usualmente los proveedores residen en la misma computadora que el CIMOM, y a diferencia de la comunicación entre clientes y CIMOM, la comunicación entre proveedores y CIMOM no está estandarizada (19).
  El proyecto SBLIM (*Standards Based Linux Instrumentation for Manageability*) de IBM, desarrolló dos APIs escritas en C, para lograr la comunicación entre proveedores y cualquier CIMOM. Ellas son: NPI (*Native Provider Interface*) y CMPI (*Common Manageability Programming Interface*). NPI ya no está siendo desarrollada, mientras que CMPI, su sucesora, está en proceso de estandarización por la iniciativa *WBEM source*. La iniciativa WBEM *source* es una organización encargada de coordinar implementaciones WBEM de fuente abierta, con el fin de lograr interoperabilidad y portabilidad entre ellas (19).
- Cliente: Puede ser visto como una interfaz entre el administrador y el CIMOM. Las operaciones CIM sobre HTTP son el estándar de comunicación entre el cliente y el CIMOM. Sin embargo, la mayoría de las implementaciones también soportan otros mecanismos para comunicación, como RMI (*Remote Method Invocation*) para Java, DCOM (*Distributed Component Object Mode*l) para Microsoft, e IPC (*Inter Process Communication*) para implementaciones Unix. Pero únicamente el

uso de operaciones CIM sobre HTTP garantiza compatibilidad entre cualquier cliente y cualquier CIMOM (19).

Luego de haber realizado el análisis de los anteriores modelos de gestión integrada, se decide elegir WBEM teniendo en cuenta que permite unificar la gestión en ambientes de computación distribuida, facilitando el intercambio de datos a través de diferentes tecnologías y plataformas. Sustentado todo lo anterior en las siguientes características:

- Integra los modelos analizados anteriormente.
- Permite el desarrollo de herramientas y tecnologías que reduzcan la complejidad y los costos de la gestión asociados a la instalación y actualización del hardware y el software de computadoras, su configuración y mantenimiento.
- Se basa en un modelo común de información para la gestión.

En la actualidad, con el desarrollo del paradigma de proveer y consumir los servicios en “la nube”, se trabaja en el desarrollo de una nueva forma de gestión, que hasta el momento solamente se ha implementado, de manera parcial en supercomputadoras, tal es el caso de la Gestión Autonómica. Las facilidades más importantes que brindará esta forma de gestión son: la auto-configuración (es la capacidad que debe tener un sistema de adaptarse inmediatamente a los cambios en su entorno, y con la mínima intervención humana posible), auto-recuperación (es la funcionalidad que permite detectar operaciones incorrectas e iniciar a continuación las acciones correctivas necesarias sin afectar el correcto funcionamiento de las aplicaciones del sistema), auto-protección (es la característica que define cuál es el momento oportuno para proporcionar la información exacta a los usuarios correctos) y auto-optimización (es la funcionalidad que brinda la opción de maximizar de forma efectiva la reserva de recursos y su utilización para satisfacer la demanda del usuario final con la mínima intervención humana posible), una de las vías que prevé la comunidad científica asociada a la gestión de redes para llegar a la gestión autonómica es la gestión basada en políticas. (12) (20)

1.2.2.1 Gestión de red basada en políticas.

Dado que ya se esclareció qué es WBEM y cómo utilizarán sus características en función de lograr una gestión Integrada; se definen seguidamente dos conceptos que responden a una vía de solución para lograr un comportamiento autónomo en una red: Política y Gestión basada en políticas.

Política

Una política es la combinación de reglas y de servicios, donde las reglas definen los criterios para el acceso y el uso de los recursos de la red. Cada regla de política abarca un sistema de condiciones y un sistema correspondiente de acciones. La condición define cuando la regla de la política es aplicable. Una vez que una regla de política es activada, unas o más acciones contenidas por esa regla pueden ser ejecutadas (21).

El código de instrucción (acción) puede residir en cualquier combinación de sistemas de gestión de redes, routers, conmutadores, servidores, bases de datos, aplicaciones y sistemas de almacenamiento. Aunque actualmente las políticas suelen implementarse en routers y conmutadores, es posible hacer que las políticas residan en módulos de software cargables en los nodos de hardware, en programas dependientes de las plataformas de gestión de red o en servidores dedicados específicamente a esta tarea. Estos elementos se encargan de recibir las peticiones de tráfico solicitadas por los conmutadores u otros nodos, recabar información de políticas de bases de datos y directorios y en consecuencia, configurar los dispositivos de red de acuerdo con las peticiones recibidas y los derechos establecidos en la especificación de nivel de servicio acordada entre el usuario y el proveedor de servicio Internet (22).

Gestión de red basada en política

La gestión de red basada en políticas (PBNM) permite controlar y coordinar, de manera dinámica los elementos de red, tomando decisiones de forma automática a través de reglas, peticiones de usuarios o de servicios (23) (24) (25).

El grupo de trabajo de *Internet Engineering Task Force* (IETF) brinda una arquitectura para la gestión de red basada en políticas que está compuesta por:

- Una herramienta de creación de políticas, la cual ayuda a los administradores de red a crear políticas (26).
- Un repositorio de políticas (MPBs), basado en LDAP (*Lightweight Director Access Protocol)*, su objetivo es almacenar las políticas (26).

- Un punto de ejecución de políticas (*Policy Enforcement Point,* PEP), su principal función es recibir las políticas en forma de acciones de configuración específica además es el responsable de establecerlas en el dispositivo. Esta entidad puede contar de forma opcional con un punto local de decisión de políticas (*Local Policy Decisión Point*, LPDP), el cual mantiene la operación en caso de que se vea interrumpida la conexión con el PDP (*Policy Decisión Point*) (26).
- Un punto de decisión de políticas (PDP), sus tareas fundamentales son consultar las políticas en el repositorio, traducirlas al formato específico del dispositivo y generar las decisiones acordes con las peticiones de los PEPs (26).

Para la comunicación entre el PDP y el PEP el IETF propone el uso del protocolo COPS (*Common Open Policy Service*), cuando se opera bajo el paradigma cliente-servidor y se requiere alto nivel de seguridad, o el protocolo SOAP (*Simple Object Access Protocol*) cuando se intercambia información en un ambiente distribuido y descentralizado (27). SOAP permite establecer un protocolo estándar de invocación de servicios remotos, basados en protocolos estándares de Internet: HTTP para la transmisión y para la codificación de datos XML. Las políticas almacenadas en los MPB se deben basar en algún modelo de información, la IETF propone el modelo PCIM (*Policy Core Information Model*) (28).
En la Figura 2 se muestra la arquitectura general propuesta por la IETF.

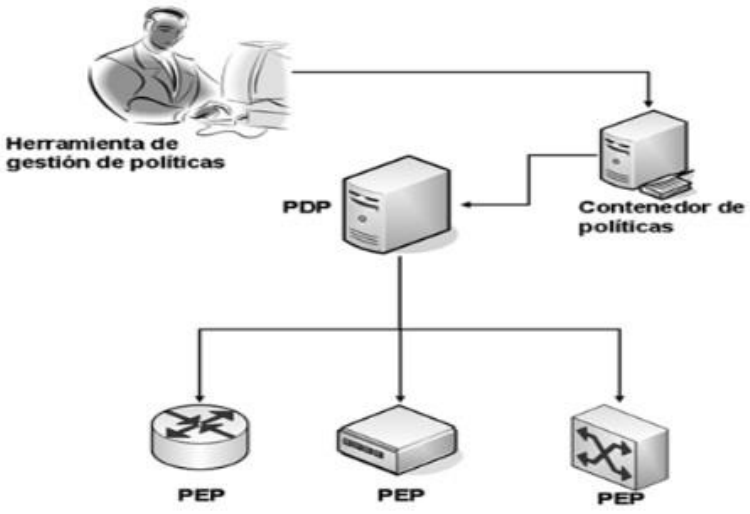


Figura 2. Elementos principales de un modelo de gestión de red basado en políticas (29).

1.3 Análisis de herramientas de gestión de servidores

Para efectuar una correcta gestión de red se hace necesario realizar los dos modos de actuación de la gestión, monitorización y control. A continuación se muestra un resumen con las principales características de un conjunto de herramientas dedicadas a la gestión de servidores, definidas por el cliente teniendo en cuenta las capacidades que presenta su infraestructura, con el objetivo de seleccionar una e integrarla a la solución.

1.3.1 Cacti

Cacti es una herramienta de monitorización de desempeño y utilización de los recursos, especializada en graficado en red. Está diseñada para aprovechar el poder de almacenamiento y la funcionalidad de graficar que posee RRDtool (*Round Robin Database tool*). Está desarrollada en PHP (acrónimo recursivo de *PHP: Hypertext Preprocessor*), soporta el protocolo SNMP (*Simple Network Management Protocol*), es multiplataforma y se encuentra bajo la licencia GPL (*General Public License*) (30).

Esta herramienta provee una interfaz de usuario fácil de usar, contiene plantillas para reutilizar definiciones de gráficos, datos y fuentes de dispositivos. Cuenta con una funcionalidad para garantizar su seguridad, que permite administrar los usuarios localmente o vía LDAP, además puede asignar niveles detallados de autorización basados en usuarios o grupos. Es capaz de recopilar la utilización del canal en las interfaces de sus equipos, así como los registros de errores. También puede medir capacidad de disco duro, carga del CPU (*Central Process Unit*) entre otros. Reaccionando a ciertas condiciones puede enviar alarmas, basándose en umbrales. Existen muchísimos *plugins* que permiten extender la capacidad de Cacti (31).

Cacti tiene como desventajas que la configuración de las interfaces suele ser trabajosa y la creación de *plugins* no es un proceso trivial por lo que hacer actualizaciones de la misma puede tornarse complejo. Sin embargo cuenta con una comunidad grande y activa en torno a los foros de Cacti que proporciona *scripts*, plantillas y consejos sobre creación de estos últimos (30) (32).

1.3.2 Munin

Munin es una aplicación de software libre y código abierto que se encarga de monitorizar el uso de los recursos de cada máquina tales como: disco duro, red, uso de CPU, memoria RAM (*Random Access*

Memory), aunque también es capaz de realizar la monitorización de procesos de Apache, Squid, consultas de MySQL entre otros. Se diseñó en el lenguaje de programación Perl, su servidor corre sobre Linux, el agente puede correr sobre Linux y Windows (con algunas limitaciones), soporta el protocolo SNMP y se encuentra bajo la licencia GPL (33).

Esta herramienta brinda la posibilidad de configurar umbrales de alerta para los estado de advertencia y crítico, además a través de la recolección de datos, puede mostrar tendencias que pueden ayudar a predecir cuellos de botella. También utiliza RRDtool para presentar resultados en gráficos a través de una interfaz web. Tiene una arquitectura gestor/agente en la que el maestro se conecta a todos los nodos para interrogarlos, recopilar información y en caso de ser necesario actualizar los gráficos en la interfaz web, esto sucede a intervalos regulares (normalmente cada 5 minutos). Su énfasis está en las capacidades *plug and play*. Existen más 500 *plugins* disponibles actualmente en su repositorio oficial, escritos en diferentes lenguajes como: Bash, Perl, Python, Ruby, PHP, Shell, entre otros (33).

Munin presenta como inconveniente que recibe la información del servidor central sin autenticación y en texto plano, por lo tanto no es factible su uso si existe información sensible o confidencial. Su interfaz web sólo es para visualizar los resultados, no proporciona un control sobre la aplicación y por tanto toda configuración debe realizarse de forma manual por línea de comandos. No permite utilizar otro gestor de base de datos que no sea RRDtool. No dispone de una versión comercial o Enterprise ni de ningún tipo de soporte profesional (33).

1.3.3 Zenoss

Zenoss es un producto de supervisión de infraestructuras TIC. Es capaz de gestionar la configuración, salud, rendimiento de dispositivos, servidores y aplicaciones. Es una herramienta de código abierto programada en Python, que se encuentra bajo la licencia GPL. Esta herramienta combina programación propia con tecnologías de código abierto como son: Zope, Python, Net-SNMP, RRDtool, MySQL, Twisted PB, entre otras, dando como resultado un *software* más completo (34).

Ofrece monitorización de dispositivos y servicios en la red (SNMP, HTTP, POP3 (*Post Office Protocol*), etc.), además puede llegar a detectar automáticamente nuevos recursos y cambios en su configuración. Zenoss utiliza una tecnología sin agentes (SNMP, SSH (*Secure SHell*), Telnet y WMI (*Windows Management*

Instrumentation)) y se inicia con una Base de datos de la Gestión de Configuración (CMBD), en la que se almacenan detalles relevantes de cada elemento y con la cual se puede llevar a cabo la administración de fallos, notificaciones, alertas, tareas para contrarrestar los fallos e informes. Posee una comunidad que dispone de un repositorio de *plugins* llamado *ZenPacks*, con los cuales los miembros de dicha comunidad pueden extender las funcionalidades de Zenoss. Además tiene como ventaja que el *software* soporta el formato de *plugins* de Nagios. Ofrece una interfaz o consola web con la visualización y control total de la aplicación que permite el manejo del estado y situación de la infraestructura. Puede ser personalizada por los usuarios y se puede integrar con *Google Maps* para georreferenciar la visualización de problemas (34).

Sin embargo esta herramienta al no utilizar tecnología con agentes en los clientes, requiere una configuración previa en su instalación del protocolo SNMP en cada uno de las máquinas a monitorizar siguiendo un procedimiento distinto según la versión o tipo de sistema operativo. Necesita la instalación de paquetes adicionales *ZenPacks* para monitorización de servicios básicos como HTTP o FTP (*File Transfer Protocol*), información del sistema operativo y algunos recursos *hardware*, por ejemplo para obtener información de la CPU. La instalación de la herramienta en el equipo *host* es rápida, pero no cada configuración del cliente desde la consola web, debido a que es necesario configurarle una plantilla personalizada para que monitorice lo que se requiera de dicho dispositivo. Zenoss para funcionar en sistemas operativos de *Microsoft Windows* necesita de la aplicación externa *VMplayer* (34).

1.3.4 Zabbix

Zabbix es una herramienta de código abierto diseñada para la monitorización avanzada de los parámetros de la red, su salud e integridad. Se encuentra distribuido bajo la licencia GPL. Es una aplicación multiplataforma programada en PHP y C, que ofrece un control centralizado (35).

Esta herramienta permite la monitorización de recursos de hardware: procesos de carga, actividad en la red, actividad en disco, parámetros del sistema operativo y la disponibilidad y capacidad de respuesta de los servicios estándar, como SMTP (*Simple Mail Transfer Protocol*) o HTTP. Incluye soporte para monitorizar de forma remota o sin agentes a través de: SNMP, ICMP (*Internet Control Message Protocol*), TCP (*Transport Control Protocol*), IPMI (*Intelligent Platform Management Interface*), SSH y Telnet. Además cuenta con una API mediante la cual se puede mantener una comunicación con la herramienta brindando la posibilidad de administrar y configurar la información que ella gestiona. Es capaz de ejecutar algunas

acciones de control a través de elementos llamados *triggers* (disparadores). Utiliza un mecanismo flexible de notificación que permite que los usuarios configuren las alarmas basadas en correo electrónico o mensajes (incluido XMPP, Protocolo Extensible de Mensajería y Comunicación de Presencia, llamado *Jabber*), tanto cuando ocurre un problema como cuando se resuelve. Zabbix tiene grandes funcionalidades de visualización incluidas las vistas definidas por el usuario, *zoom*, y la cartografía. Para almacenar los datos de seguimiento puede soportar una base de datos en MySQL, PostgreSQL, Oracle o SQLite. Además es capaz de realizar la autodetección de dispositivos y servicios monitorizados. Esta herramienta posee gran cantidad de información sobre su instalación, uso y soporte. Cuenta con una amplia comunidad que desarrolla continuamente *plugins* y aplicaciones que facilitan el uso y administración de esta herramienta (36).

A pesar de todas las ventajas que posee Zabbix, un aspecto que se debiera mejorar es el tema del almacenamiento en una base de datos. Si el número de parámetros monitorizados crece y su intervalo de actualización es muy reducido, es posible que, de no haber ajustado convenientemente el rendimiento de la base de datos, nos encontremos con un cuello de botella que resultará en muchos datos de monitorización encolados (35) (36).

1.3.5 Nagios

Nagios es un sistema *Open Source* de monitorización de equipos y de servicios de red. Este sistema está escrito en C y publicado bajo la licencia GPL. Fue diseñado originalmente para ejecutarse en sistemas Linux, pero en la mayoría de los UNIX funciona sin problemas (37).

Esta herramienta proporciona supervisión de los servicios de red (SMTP, POP3, HTTP, NNTP (*Network News Transport Protocol*), ICMP, SNMP, FTP, SSH) y recursos de *host* (carga del procesador, uso de disco, los registros del sistema, uso de memoria, entre otros) en varios sistemas operativos, incluso en sistemas Microsoft Windows con el *plugin* NRPE_NT o también por medio del protocolo SNMP. Permite monitorizar de forma remota mediante túneles SSL (*Secure Sockets Layer*) cifrados o SSH. Tiene la capacidad de definir una topología o jerarquía de red que permita distinguir entre servicios caídos o inaccesibles. Cuando ocurren problemas en servicios o *hosts*, y cuando son resueltos se realizan notificaciones a los contactos (a través de correo electrónico, *Jabber*, SMS, o cualquier método definido por el usuario junto con su correspondiente complemento). El diseño de *plugins* es simple y además pueden ser escritos en varios lenguajes (Bash,

C++, Perl, Ruby, Python, PHP, C#, Java, entre otros). Nagios está formado por 2 módulos diferenciados: el núcleo, llamado Nagios Core, que contiene los componentes fundamentales del *software*, y los *plugins*, donde cada *plugin* monitoriza una serie de recursos o de servicios. Provee una interfaz web para visualizar el estado actual de la red con la posibilidad de generar informes y gráficas, haciendo uso también de la herramienta *RRDtool* (37).

A pesar de sus grandes ventajas y de ser un *software* popularmente conocido, una de sus principales desventajas la constituye, el hecho de que los usuarios que no están familiarizados con la herramienta deben aprender el funcionamiento de un sistema complejo que no dispone de una herramienta intuitiva de configuración. Además cualquier modificación en la configuración requiere un reinicio completo del sistema, ya que por ejemplo, no es capaz de autodescubrir nodos nuevos que se incluyan al mismo. Otro de los inconvenientes de esta herramienta es que posee una interfaz web que solo sirve para visualizar los acontecimientos, mientras que cualquier cambio debe realizarse manualmente desde el servidor de Nagios. Es necesario destacar que dispone de una consola de eventos muy débil, que no permite configurar acciones automáticas ante nuevos eventos en el sistema. Otra crítica importante a Nagios es que a pesar de ser un software gratuito, supone un gasto importante para las empresas ya que necesita personal calificado dedicado a su configuración y mantenimiento (31) (37).

1.3.6 Evaluación de las herramientas

Con el objetivo de desarrollar la presente investigación se evalúan las principales herramientas de monitoreo y control de servidores existentes en la actualidad, teniendo en cuenta un conjunto de indicadores los cuales están asociados a los requerimientos que se deben satisfacer. Según lo anteriormente planteado los indicadores determinantes para la selección de las herramientas más adecuadas son:

- **Soberanía Tecnológica**: Este indicador hace referencia a la capacidad de poseer el poder absoluto sobre determinada tecnología, contemplándose esto como la posibilidad de conocer, modificar y distribuir la tecnología. La evaluación de este parámetro se realiza sobre la base de la existencia o no en las herramientas de este indicador, identificándose con las siglas **ST**.
- **Facilidad de Extensión:** Este indicador hace referencia a la flexibilidad que posee la herramienta de extenderse con la adición de nuevas funcionalidades. Entre los principales elementos que se tienen en cuenta están la posibilidad de crearle *plugins*, la facilidad de implementación y la

documentación existente para realizar esta tarea. Este indicador será evaluado teniendo en cuenta la presencia de las características antes mencionadas en la herramienta a analizar, contemplándose en **Bajo**, **Medio** o **Alto**. Para identificarlo se utilizará la sigla **FE**.

- **API**: Este indicador muestra la capacidad que tiene la herramienta de brindar servicios para realizar una conexión a ella a través de un API (*Application Programming Interface*) de programación. La evaluación de este parámetro se realiza sobre la base de la existencia o no en la herramienta de este indicador, identificándose con las siglas **AR**.
- **Usabilidad**: Este indicador reflejará cuán intuitiva es para el usuario la herramienta analizada contemplándose principalmente la facilidad de uso. La evaluación de este parámetro se realiza sobre la base de la existencia o no en la herramienta de este indicador, identificándose con la sigla **U.**
- **Licencia**: En este indicador se mostrará el tipo de licencia bajo la cual fue creada la herramienta analizada. Para identificarlo se utilizará la sigla **L**.
- **Gestión**: Este indicador hace referencia a la capacidad que posee la herramienta analizada por si sola (sin la utilización de *plugins*) de llevar a cabo los dos modos de actuación de la gestión de red (monitorización y control). Este indicador será evaluado teniendo en cuenta la presencia de las características antes mencionadas en la herramienta a analizar, contemplándose en **Bajo**, **Medio** o **Alto**. Para identificarlo se utilizará la sigla **G**.
- **Tecnología de gestión**: Este indicador mostrará el protocolo que utiliza la herramienta analizada para llevar a cabo las funciones de gestión. Para identificarlo se utilizará la sigla **TE**.

Herramientas	**ST**	**FE**	**API**	**U**	**L**	**G**	**TE**
Cacti	Si	Baja	No	Si	GPL	Baja	SNMP
Munin	Si	Alta	No	Si	GPL	Baja	SNMP
Zenoss	Si	Alta	No	Si	GPL	Media	SNMP
Nagios	Si	Media	No	No	GPL	Baja	SNMP
Zabbix	Si	Alta	Si	Si	GPL	Media+	SNMP

Tabla 1. Evaluación de herramientas de monitorización y control de servidores.

Concluida la evaluación se decide elegir Zabbix, entre las principales características que justifican esta elección se encuentran el hecho de que de las herramientas analizadas según la bibliografía estudiada Zenoss y Zabbix son las únicas que realizan la monitorización y el control parcial de servidores, y de ambas, Zabbix, lleva a cabo estas acciones sin la adición de *plugins*. Además esta herramienta proporciona al usuario una API desde la cual es posible configurar totalmente el funcionamiento de la misma, facilitando así la administración de toda la información que en ella se maneja, e igualmente se lleva a cabo sin la adición de plugins.

Otro de los aspectos más valorados y que motiva, en parte elegir Zabbix, es el hecho de estar basado en C lo que posibilitaría que el equipo de desarrollo no tenga que aprender un nuevo lenguaje de programación como ocurriría en el caso de emplear Nagios, Cacti o Munin.

También se destaca de Zabbix la posibilidad de crear gráficos con los que apoyar la presentación de los datos. En muchos casos el comportamiento del sistema se hace mucho más comprensible a través de representaciones gráficas que solamente por medio de datos numéricos. Para cada parámetro de monitorización que tenga un valor numérico, Zabbix puede crear automáticamente gráficos en los que se observa la evolución del parámetro en el tiempo. Junto a esos gráficos por defecto, el usuario puede crear sus propios gráficos personalizados para mostrar los valores de monitorización que estime oportunos (38). Otros elementos que sirven como basamento de esta elección son la alta facilidad de extensión que brinda la herramienta, la gran cantidad de plugins con los que ya cuenta y la posibilidad de encontrar mucha información en los libros y foros dedicados a dicha herramienta.

1.4 Proceso de desarrollo de software

Un proceso de desarrollo de software se define como un conjunto estructurado de actividades y resultados asociados, requeridos para desarrollar un software (39). Tiene como propósito la producción eficaz y eficiente de un producto de software que reúna los requisitos deseados por el cliente (40). A pesar de que no existe un proceso de software universal que sea efectivo para todos los contextos de proyectos de desarrollo, hay un conjunto de actividades que son genéricas, presentes en todos ellos:

- Especificación del software: Se debe definir la funcionalidad y restricciones operacionales que debe cumplir el software.

- Diseño e Implementación: Se diseña y construye el software de acuerdo a la especificación.
- Validación: El software debe validarse, para asegurar que cumpla con lo que quiere el cliente.
- Evolución: El software debe evolucionar para adaptarse a las necesidades del cliente.

La Unidad de Compatibilización Integración y Desarrollo de Software para la Defensa (UCID) del Ministerio de las Fuerzas Armadas Revolucionarias (MINFAR) de Cuba creó un proceso de desarrollo de software llamado Prodesoft, el cual se utiliza en la empresa XETID. Debido a que este módulo será integrado con una solución que se implementa bajo las pautas de Prodesoft, se elige mantener su uso con el objetivo de seguir una misma línea de desarrollo y lograr una adecuada integración entre ambas soluciones. A continuación se exponen las características esenciales de dicho proceso de desarrollo de software.

Prodesoft

Prodesoft es un documento concebido por la UCID que tiene como objetivo servir de guía en el proceso de construcción y desarrollo de software. En él se realiza una descripción detallada de las distintas actividades que se llevan a cabo durante dicho proceso y que forman parte del ciclo de vida de un proyecto. Prodesoft está basado en un desarrollo iterativo e incremental y en un desarrollo basado en componentes (41).

El desarrollo iterativo e incremental está enfocado en descomponer las distintas fases del ciclo de vida del proyecto en iteraciones. Todas las iteraciones están formadas por cada una las fases del ciclo de vida de un proyecto (Modelación de procesos del negocio, Definición de requisitos, Diseño de la arquitectura, Implementación y Pruebas). En el modelo de desarrollo iterativo e incremental se entrega una serie de lanzamientos, llamados incrementos, que proporcionan en forma progresiva más funcionalidades para los clientes a medida que se entregan cada uno de dichos incrementos (42). Existen dos factores que fundamentan lo que se implementará en una iteración:

-La implementación de un grupo de funcionalidades que juntas extienden la utilidad del producto desarrollado hasta el momento.

-La implementación de los componentes más complejos o riesgos más importantes (41).

El punto de partida de las iteraciones sucesivas son los artefactos desarrollados en la última iteración tal como quedaron al final de la misma. En un proceso iterativo e incremental en cada nueva versión se reducen

los riesgos más significativos para el éxito del proyecto. Por lo general las iteraciones están asociadas al tiempo y a los componentes de alto nivel que forman la arquitectura, mientras que los incrementos manejan la estrategia en la que se obtienen los resultados (41).

Un desarrollo basado en componentes es un modelo que se centra en la creación de componentes que pueden ser implementados y luego optimizados continuamente, trayendo como beneficio elevar la calidad de la aplicación con el paso del tiempo. Con este modelo se obtiene un mayor nivel de reutilización de software, inclusive cuando el mismo se haya diseñado con otra finalidad. Una de sus características principales es que no es necesario tener un producto terminado, es decir el software con todos los componentes acoplados, para realizarle pruebas, esta acción se puede efectuar sobre los componentes por separados. En caso de existir poca dependencia entre los componentes, sin afectar otras partes del sistema, se pueden desarrollar o actualizar componentes, según como sea necesario (41).

Ciclo de vida

El ciclo de vida está compuesto por 5 fases: inicio, modelación, construcción, explotación experimental y despliegue, Figura 3. Cada fase terminará en un hito con el objetivo fundamental de evaluar y decidir el paso a la siguiente fase de desarrollo.

Figura 3. Fases del ciclo de vida del proyecto (41).

En la fase de Inicio se establece una visión preliminar de la problemática a resolver y se definen los recursos relevantes para la ejecución del proyecto. Se describen los objetivos y el alcance del proyecto, las entidades involucradas, se realiza el cronograma general de actividades, se establece la estrategia a seguir para realizar la modelación del negocio y la captura de requisitos y de ser necesario se estiman los recursos materiales que deberán ser adquiridos.

En la fase de Modelación se identifican los procesos de negocio fundamentales y se aceptan los requisitos funcionales, obteniéndose la línea base de la arquitectura y una estrategia de construcción de la aplicación. El hito fundamental de esta fase es la liberación de la arquitectura de sistema, datos y despliegue.

En la fase de Construcción se definen los requisitos restantes y se culmina el desarrollo del sistema sobre una arquitectura estable. Todas las características, componentes y requisitos deben ser integrados, implementados y probados en su totalidad, obteniendo una versión liberada del producto.

En la fase de Explotación Experimental es donde se eliminan los errores que surgen durante las pruebas y se convierte la versión liberada del producto en una solución estable.

En la fase de Despliegue se instala y configura el sistema para un ambiente de producción real, se capacita al personal que usará la aplicación y se continúa dando soporte durante la explotación del sistema, culminando de ser preciso con transferencias tecnológicas (41).

1.5 Herramientas y tecnologías para la solución del problema

Se asumen las mismas herramientas y tecnologías del sistema al cual se integrará el módulo, con el objetivo de seguir la misma línea de desarrollo contribuyendo a la integración entre las soluciones.

1.5.1 Lenguajes de modelado

El lenguaje de modelado constituye un eslabón fundamental en los procesos de diseño y construcción de un software. Es un conjunto estandarizado de notaciones que incluye símbolos y las distintas formas de organizarlos, estructurarlos y disponerlos lógicamente.

Lenguaje de Modelado Unificado (*Unified Modeling Language*, UML) v2.0

UML es una serie de normas y estándares gráficos respecto a cómo se deben representar los esquemas relativos al software. Es un lenguaje basado en una notación gráfica la cual permite especificar, construir, visualizar y documentar los objetos de un sistema programado. Proporciona además una organización en el proceso de diseño de forma tal que analistas, clientes, desarrolladores y otras personas involucradas en el desarrollo del sistema lo comprendan (43).

1.5.2 Herramientas de Modelado

En la actualidad los ingenieros de *software* para desarrollar y mantener un *software* se apoyan esencialmente en las herramientas CASE, el cual es un acrónimo para *Computer-Aided Software Engineering*. Una herramienta CASE facilita el ciclo de desarrollo de software auxiliando el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores, entre otros. Por lo que con la realización de todas estas actividades se aumenta la productividad en el desarrollo de *software* reduciendo el coste de las mismas en términos de tiempo y de dinero (44).

Visual Paradigm v8.0

Visual Paradigm v8.0 es una herramienta CASE multiplataforma que utiliza UML. Proporciona un conjunto de ayudas para el desarrollo de programas informáticos que parten desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y de la documentación, siendo válido destacar que la misma puede estar en varios formatos. Bajo el paradigma orientado a objetos permite realizar el modelado visual del software. Esta herramienta ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo de software a través de la representación de diagramas (45).

1.5.3 Lenguajes de Programación

Un lenguaje de programación es un lenguaje que puede ser utilizado para controlar el comportamiento de una máquina, particularmente una computadora. Consiste en un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones (46).

Python v2.7

Python es un lenguaje de programación dinámico y orientado a objetos usado para el desarrollo de software. Es un lenguaje fuertemente tipado, su sintaxis es simple, clara y sencilla ofreciendo así un código muy legible. Entre sus características principales vale destacar que el mismo brinda gran soporte e integración con otros lenguajes y herramientas, es multiplataforma y además provee una extensa cantidad de librerías. Otro beneficio que ofrece este lenguaje es el hecho de estar distribuido bajo la licencia *Open Source* OSI que lo hace libre para ser usado inclusive en el desarrollo de productos comerciales.

Python es un lenguaje interpretado, lo cual puede ahorrar mucho tiempo durante el desarrollo ya que no es necesario compilar ni enlazar. El intérprete de Python y la extensa biblioteca estándar están a libre disposición en forma binaria y de código fuente para las principales plataformas desde el sitio web de Python donde puede ser distribuido libremente. Además este intérprete puede extenderse fácilmente con nuevas funcionalidades y tipos de datos implementados en C o C++ (u otros lenguajes accesibles desde C). Python permite escribir programas compactos y legibles ya que los tipos de datos de alto nivel permiten expresar operaciones complejas en una sola instrucción, la agrupación de instrucciones se hace por sangría en vez de llaves de apertura y cierre, y no es necesario declarar variables ni argumentos (47).

1.5.4 Sistemas gestores de base de datos (SGBD)

Un sistema gestor de base de datos es un software que permite a los usuarios procesar, describir, administrar y recuperar los datos almacenados en una base de datos. Estos sistemas brindan un conjunto de programas, procedimientos y lenguajes, interrelacionados unos con otros con el objetivo de que cada usuario pueda efectuar sus tareas con los datos; además de proporcionar la integridad, confidencialidad y seguridad de los mismos y en esta última actividad es que reside el éxito rotundo de un SGBD. Dichos sistemas deben permitir (48):

1- Definir una base de datos. Permitirle a los diseñadores de bases de datos: especificar tipos de datos, crear estructuras adecuadas para integrar apropiadamente los mismos y definir restricciones sobre ellos.

2- Manipular la base de datos. Permitir a través de consultas y actualizaciones modificar y utilizar adecuadamente los datos de las bases de datos adecuadamente. Esta función se realiza mediante el lenguaje de modificación de datos.
3- Controlar la base de datos. Permite que los datos permanezca seguros además de brindar confidencialidad sobre los mismos, es decir solo puedan acceder a los datos los usuarios autorizados en el momento requerido (48).

PostgreSQL v9.4

Es un gestor libre de base de datos relacional orientado a objetos, licenciado bajo la licencia BSD (*Berkeley Software Distribution*) y es multiplataforma. Este gestor soporta distintos tipos de datos, la replicación de base de datos asíncrona, cuenta con un sistema denominado MVCC *(Multi-Version Concurrency Control*) que permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Además brinda la posibilidad de realizar copias de seguridad en caliente (*Online/hot backups*), la cual consiste en realizar una copia de las bases de datos, aunque se estén utilizando y procesando peticiones. Su sintaxis SQL es estándar, fácil de aprender y cuenta con una completa documentación. Cuenta con una API flexible para programar en C/ C++, Java, Python, PHP entre otros. Posee un buen sistema de seguridad y una gran capacidad de almacenamiento. Todas estas son las características más notables de PostgreSQL que hacen del mismo uno de los gestores de base de datos más potente y robusto del mercado (49).

1.5.5 Entorno de desarrollo integrado (IDE)

Un entorno de desarrollo integrado, traducido del inglés, *Integrated Development Environment* (IDE), es una aplicación de *software*, que proporciona servicios integrales para facilitarle al programador el desarrollo de un *software*. Normalmente un IDE consiste de un editor de código fuente, herramientas de construcción automáticas y un depurador. Una de sus principales ventajas es el auto-completado inteligente de código.

PyCharm v4.0.2

PyCharm es un IDE o entorno de desarrollo integrado multiplataforma utilizado para desarrollar en el lenguaje de programación Python. Proporciona análisis de código, depuración gráfica, integración con VCS

(*Version Control System*) / DVCS (*Distributed Version Control System*), soporte para el desarrollo web con Django y puede servir como intérprete para otros lenguajes que pudieran ser usados en el mismo proyecto como HTML, CSS (*Cascade Style Sheet*) y JavaScript, entre otras bondades. PyCharm es desarrollado por la empresa JetBrains y debido a la naturaleza de sus licencias tiene dos versiones, la comunidad que es gratuita y orientada a la educación y al desarrollo puro en Python y la *Professional*, que incluye más características como el soporte a desarrollo web con varios precios. (50)

PgAdmin v1.18.1

PgAdmin es una herramienta de código abierto para la administración de bases de datos PostgreSQL y derivados. Se diseñó para responder a las necesidades de la mayoría de los usuarios, desde escribir simples consultas SQL hasta desarrollar bases de datos complejas. Además su interfaz gráfica soporta todas las características de PostgreSQL y hace simple la administración. Está disponible en más de una docena de lenguajes y para varios sistemas operativos, incluyendo Microsoft Windows, Linux, FreeBSD, Mac OSX y Solaris (51).

1.5.6 Marco de trabajo

Según Jerome Lafosse un marco de trabajo o *framework* es: un conjunto de bibliotecas, herramientas y normas a seguir que ayudan a desarrollar aplicaciones. Un *framework* está compuesto por varios componentes que interactúan los unos con los otros. Las aplicaciones pueden escribirse de manera más eficaz si se utiliza un *framework* adaptado al proyecto. En proyectos de desarrollo a gran escala y de diseño en equipo, son muy útiles, incluso imprescindibles. Permiten la reutilización de código, la estandarización del desarrollo y la utilización del ciclo de desarrollo (52).

Django v1.7

Django es un marco de trabajo de código abierto escrito en Python que permite construir aplicaciones web más rápido y con menos código, con el objetivo de desarrollar aplicaciones web empresariales. Fue inicialmente desarrollado para gestionar aplicaciones web de páginas orientadas a noticias de World Online, más tarde se liberó bajo licencia BSD. Respeta el paradigma conocido como Modelo-Plantilla-Vista. Su meta fundamental es la creación de sitios web complejos y para ello pone especial énfasis en la reutilización,

conectividad, extensibilidad de los componentes y además se diseñó para que las tareas comunes de desarrollo web fueran rápidas y simples (53).

ExtJS v4.1.2

ExtJS 4 es un marco de trabajo multiplataforma desarrollado en JavaScript. Unas de las principales características de este *framework* es que cuenta con gran cantidad de componentes pre-construidos agilizando el trabajo de los programadores además de poseer una arquitectura para construir aplicaciones que funcionan tanto en navegadores antiguos como en los últimos dispositivos táctiles. Entre sus principales ventajas se encuentran que ExtJS brinda la facilidad de funcionar en cualquier navegador sin tener que realizar configuraciones específicas para dicho navegador y que en esta herramienta el diseño está completamente separado de las funcionalidades (54).

1.5.7 Servidor para aplicaciones web

Nginx v1.4.6

Nginx es un servidor web y proxy inverso de alto rendimiento. Incluye servicios de correo electrónico con acceso al *Internet Message Access Protocol* (IMAP) y al servidor POP. Es un *software* libre de código abierto que se encuentra bajo una licencia estilo BSD. Nginx es conocido por su estabilidad, configuración simple, y bajo consumo de recursos. (55).

1.6 Conclusiones parciales

Durante el análisis de los preceptos teóricos abordados en el capítulo, se pudo constatar que a pesar de haber numerosas herramientas para la monitorización de servidores de telefonía IP no existe una que automatice el control total de los mismos. Se hace necesario destacar también que todas esas herramientas hacen uso del protocolo SNMP, que a pesar de ser un protocolo robusto, para brindar un correcto servicio de telefonía de IP no es conveniente su utilización. Además a partir del análisis de las principales características de los modelos de gestión integrada de red, se elige utilizar WBEM por lo que la propuesta de solución se basa en CIM. Con el empleo de CIM como modelo común de información se logra llevar a cabo una gestión integrada de los servidores de telefonía IP y con la realización de una gestión basada en políticas se dan los primeros pasos hacia una gestión autonómica.

Manteniendo el proceso de desarrollo de software, la línea tecnológica y el uso de las herramientas utilizadas en el desarrollo del Sistema Integral de Gestión de Redes del CDTT, se logra mayor integración y acoplamiento del módulo en dicho sistema. Manteniendo además un soporte multiplataforma basado en software libre.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

2.1 Introducción

En este capítulo se abordan las principales características de la solución propuesta. Se obtienen un conjunto de artefactos correspondientes a la modelación de los procesos del negocio como el mapa de procesos del negocio y el modelo conceptual. Se realiza la captura de los requisitos funcionales y de los no funcionales con la especificación de los mismos y se describe la arquitectura propuesta del módulo. También se obtiene otro conjunto de artefactos que serán de gran valor para la fase de construcción como son: el diagrama de clases del diseño, el modelo de datos e igualmente se especifica la utilización de un conjunto de patrones dentro del diseño del módulo. Logrando de esta forma alcanzar un mayor entendimiento del funcionamiento del módulo para su posterior implementación.

2.2 Propuesta de solución

El módulo Gestión de Servidores de telefonía IP integrado al Sistema Integral de Gestión de Redes del CDTT, el cual incluye la solución de telefonía IP PlaTel, será capaz de llevar a cabo una gestión integrada de los servidores de telefonía IP. En este módulo se monitoriza el por ciento de usabilidad de la RAM, la CPU y del disco duro; el tráfico en las tarjetas de red y los principales parámetros que permiten medir la calidad de servicio de un servidor de telefonía: la pérdida de paquetes, el *jitter*, la latencia, el estado de las extensiones (usuarios registrados y no registrados), y la cantidad de llamadas concurrentes y fallidas. Para completar el proceso de gestión se efectúa un control sobre cada uno de estos parámetros a través de políticas, que serán establecidas por el administrador de la red. Esta acción tiene como objetivo lograr que los elementos y servicios que hospedan los servidores de telefonía IP, tenga un comportamiento autónomo. Además se generan reportes en el cual se muestra el comportamiento de los parámetros monitorizados durante un tiempo determinado (el usuario tiene la posibilidad de seleccionar el intervalo de tiempo). Con la finalidad de llevar a cabo una gestión integrada de la red se hace uso de CIM como modelo común de información, manteniendo así la interoperabilidad entre los recursos de la misma.

2.3 Modelo conceptual y sus conceptos

El modelo conceptual refleja los principales conceptos asociados al módulo a desarrollar, su objetivo principal es brindar una mayor comprensión de su entorno. A continuación en la Figura 4 se presenta el modelo conceptual de la solución.

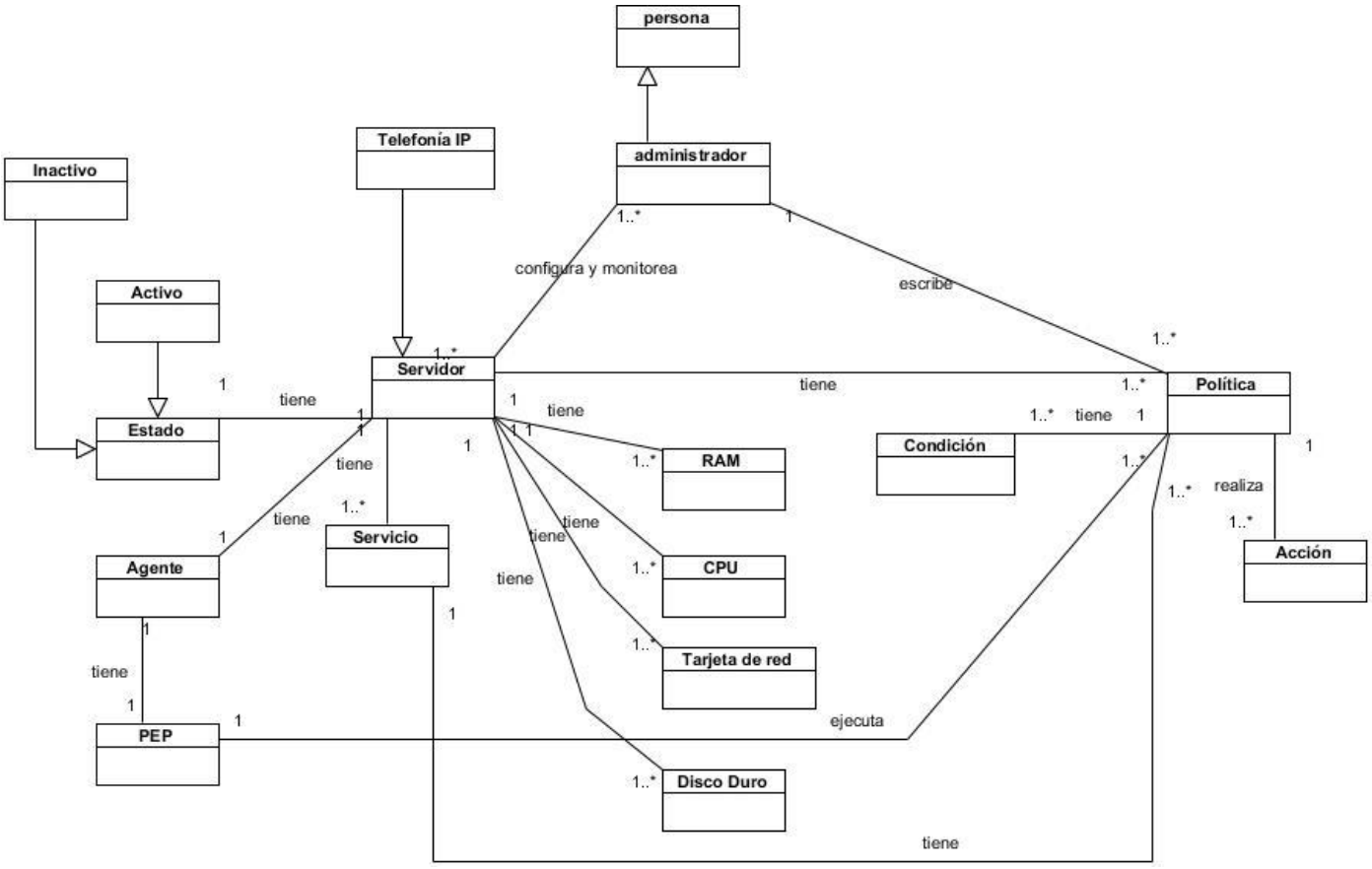


Figura 4. Modelo conceptual.

Glosario de conceptos del modelo conceptual:

- **Persona**: Visión más abstracta del usuario que interactúa con el sistema.
- **Administrador**: Es el encargado de configurar, monitorizar y controlar a través de políticas los servidores de telefonía IP.

- **Servidor**: Ordenador donde se encuentran todos los servicios de la telefonía IP.
- **Estado**: Situación en la que se encuentra el servidor en un instante de tiempo determinado.
- **Activo**: Tipo de estado en que puede encontrarse el servidor; este estado implica que el servidor se encuentra en ejecución.
- **Inactivo**: Tipo de estado en el que puede encontrarse el servidor; este estado implica que el servidor no se encuentra en ejecución.
- **Agente**: Es la aplicación que se encarga de extraer los datos del servidor y enviarlos al punto de ejecución de políticas.
- **PEP** (Punto de ejecución de políticas): Nodo encargado de recibir las políticas y de establecerlas en el dispositivo.
- **Servicio**: Medio que responde a las necesidades de un cliente.
- **RAM**: Memoria de trabajo del servidor para el sistema operativo, los programas y la mayor parte del *software*.
- **CPU**: La Unidad Central de Procesamiento del servidor que interpreta las instrucciones de un programa informático.
- **Tarjeta de red**: Es el periférico que actúa de interfaz de conexión entre dispositivos de una red.
- **Disco Duro**: Dispositivo de almacenamiento el cual sirve para almacenar información.
- **Política**: Reglas o acciones que se ejecutarán sobre el servidor.
- **Condición**: Son las condiciones definidas por el usuario que definen el estado en que se encuentran una o varias características de un servidor.

- **Acción**: Operación o conjunto de operaciones a ejecutar al cumplirse cada una las condiciones definidas.

2.4 Proceso de negocio

Un Proceso de negocio es un conjunto de actividades que son realizadas en coordinación, en entornos organizacionales y técnicos (56). Por lo que también se puede traducir como una secuencia específica de actividades de trabajo a través del tiempo y del espacio, con un inicio, un final y entradas y salidas claramente definidas (57).

A continuación se muestra el diagrama de proceso de negocio de Supervisión de servidores en la Figura 5. Este proceso se inicia con la monitorización del comportamiento de los servidores a través de una herramienta (software) dedicada a ejecutar este proceso, a registrar dicho comportamiento de los servidores y a generar alarmas en caso existir algún problema. Luego de realizar esta acción paralelamente se realizan dos actividades: la herramienta de monitorización continúa realizando su trabajo y el administrador ejecuta una acción que le dé solución al problema ocurrido, informando posteriormente sobre la solución efectuada.

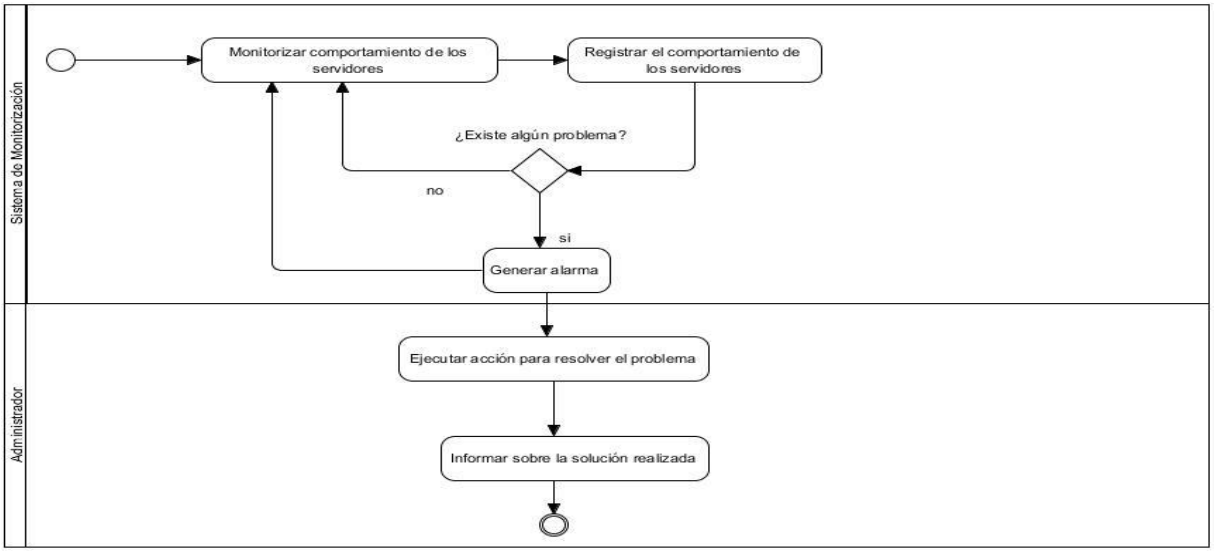


Figura 5. Diagrama de proceso de negocio de Supervisión de servidores.

2.5 Requisitos funcionales

Los requisitos funcionales son un conjunto de requerimientos los cuales especifican cada una de las funcionalidades que el sistema debe ser capaz de realizar. A continuación se enumeran los requisitos funcionales del sistema implementado. La descripción detallada de los mismos se encuentra en sus especificaciones. Anexo 1.

- **RF1:** Configurar servidores de Telefonía IP.
    - RF1.1: Crear servidor de telefonía IP.
    - RF1.2: Modificar servidor de telefonía IP.
    - RF1.3: Eliminar servidor de telefonía IP.
- **RF2:** Configurar las políticas del servidor de telefonía IP.
    - RF2.1: Crear políticas de rendimiento.
    - RF2.2: Modificar políticas de rendimiento.
    - RF2.3: Eliminar políticas de rendimiento.
- **RF3:** Mostar la información de la monitorización del rendimiento de servidores de telefonía IP.
    - RF3.1: Mostar el por ciento de uso de la **RAM** del servidor de telefonía IP.
    - RF3.2: Mostar el por ciento de uso de la **CPU** del servidor de telefonía IP.
    - RF3.3: Mostar el tráfico de la **tarjeta de red** del servidor de telefonía IP.
    - RF3.4: Mostar el por ciento de uso del **disco duro** del servidor de telefonía IP.
    - RF3.5: Mostrar el comportamiento del ***jitter*** en la red VoIP.
    - RF3.6: Mostrar el comportamiento de la **pérdida de paquete** en la red VoIP.

- RF3.7: Mostrar el comportamiento de la **latencia** en la red VoIP.
- RF3.8: Mostrar el **estado de las extensiones** (usuarios registrados y no registrados) del servidor de telefonía IP.
- RF3.9: Mostrar la cantidad de **llamadas concurrentes** en la red VoIP.
- RF3.10: Mostrar la cantidad de **llamadas fallidas** en la red VoIP.

- **RF4:** Generar alertas.
- **RF5:** Generar reportes.

2.6 Requisitos no funcionales del sistema

Los requisitos no funcionales son propiedades o cualidades que el *software* debe tener. A continuación se describen los requisitos no funcionales del sistema desarrollado.

Seguridad:

- El acceso al módulo debe estar restringido por el uso de claves asignadas a cada uno de los usuarios.
- El sistema de autenticación recogerá datos necesarios de los usuarios y tendrá control de identidades de los mismos.
- El sistema debe proveer algún mecanismo seguro de encriptación y transferencia de datos.

Software:

- El software tiene que ser desarrollado con tecnologías libres.

Usabilidad

- Las vistas del módulo deben ser intuitivas, indicar en cada momento la acción que se está realizando.

- Para el despliegue del sistema se contará con un servidor de base de datos PostgreSQL 9.4 o superior, con un sistema operativo Ubuntu en su versión 14 o superior, Nginx 1.4.6 y *framework* Django versión 1.7

- Para el uso del sistema se requiere una PC cliente con cualquier navegador web, aunque se recomienda utilizar Morzilla Firefox en su versión 13 o superior y los siguientes sistemas operativos: Windows en su versión 7 o superior, o Ubuntu en su versión 14 o superior.

- El servidor tendrá los siguientes componentes de hardware: Microprocesador de 2 núcleos, 4GB de memoria RAM, 250 GB disco duro y una fuente de 800 W como mínimo.

- Para la ejecución del módulo se requiere que la PC cliente tenga los siguientes componentes de *hardware*: Pentium 4 o superior, 512 MB RAM.

Confiabilidad

- El tratamiento de las excepciones permitirá guardar información acerca del lugar dónde se produjo el error y de los parámetros de configuración del sistema que lo provocaron. Cuando ocurre una excepción el sistema mostrará un mensaje explicativo del error ocurrido y permanecerá en el mismo estado sin realizar ninguna otra operación.

- Se debe proporcionar un sistema de seguridad basado en roles que permita asignar permisos a los usuarios teniendo en cuenta las funcionalidades y acciones a las que tiene permiso.

Interfaz

- La comunicación entre el cliente y el servidor de aplicaciones se lleva a través del protocolo HTTP.

- El sistema se integrará con el directorio activo de la institución cliente mediante el protocolo LDAP.

- Existirá un componente interno para la validación correcta de las entrada/salida de datos de cada una de las interfaces de la aplicación.

Validación de la información

- El sistema debe validar automáticamente la información contenida en los formularios de ingreso. En el proceso de validación de la información, se deben tener en cuenta aspectos tales como obligatoriedad de campos, longitud de caracteres permitida por campo, manejo de tipos de datos, entre otros.

2.7 Prototipos de interfaz de usuario

Un prototipo de interfaz de usuario es una muestra de cómo se vería la interfaz de usuario de la aplicación, la cual es realizada en una herramienta CASE. Seguidamente se muestra la interfaz de usuarios principal del módulo, Figura 6. Los restantes prototipos se encuentran en el **Anexo 2. Prototipos de interfaz de usuario**.

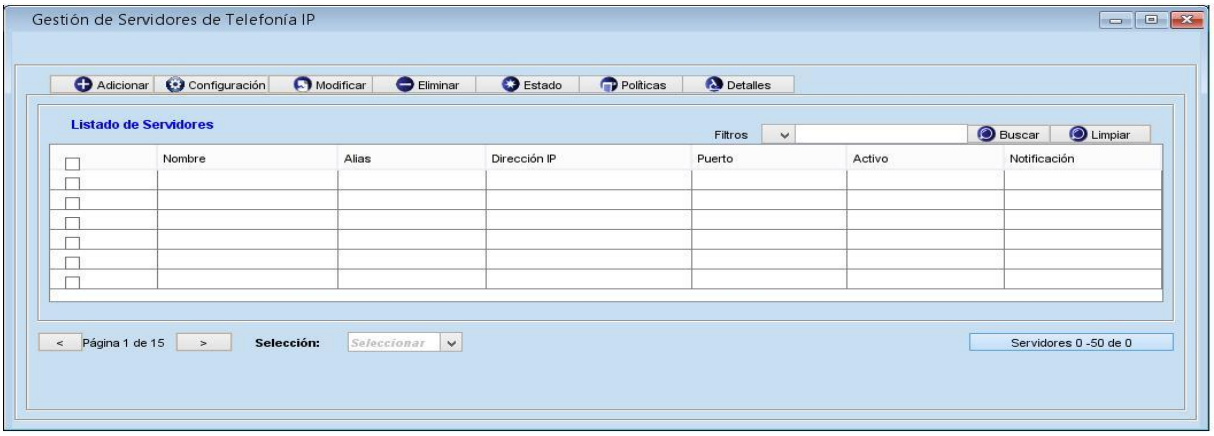


Figura 6. Prototipo de la Interfaz principal Gestión de Servidores de Telefonía IP.

2.8 Diseño de la arquitectura propuesta

Una arquitectura de software es la organización de un sistema en términos de sus componentes de software, incluyendo los subsistemas, las relaciones e interacciones entre ellos, y los principios que guían el diseño de ese sistema de software. Establece de forma coherente los patrones y abstracciones para que los analistas y desarrolladores trabajen en una línea común hacia la implementación del sistema de información. Una arquitectura sigue un patrón o un conjunto de patrones que proporcionan una estructura lógica para lograr la funcionalidad requerida por el cliente.

El Sistema Integral de Gestión de Red está basado en el patrón arquitectónico n-capas, como se muestra en la Figura 7. Este patrón sigue un estilo de programación en la cual el objetivo principal es separar la lógica de negocios de la lógica de diseño (58).

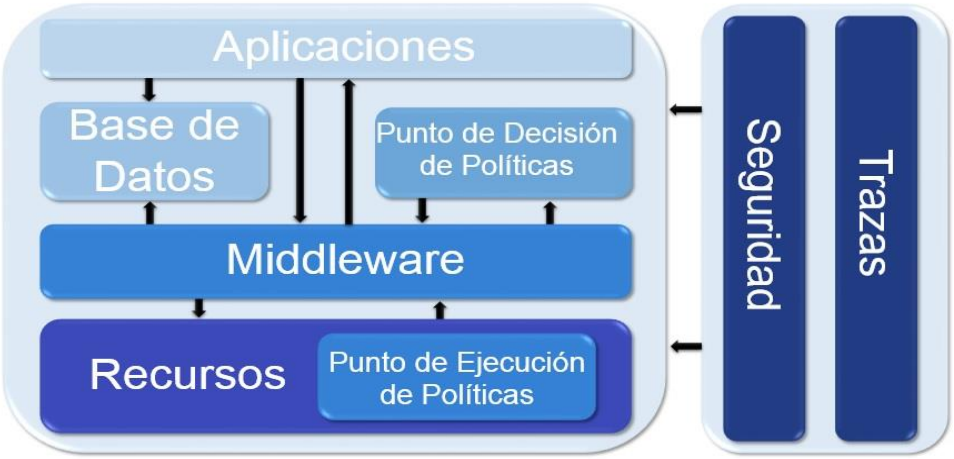


Figura 7. Arquitectura del Sistema Integral de Gestión de Red (Elaboración propia).

- **Capa de Recursos**: En esta capa se encuentran todos los dispositivos que serán monitorizados y controlados con agentes Zabbix, mediante los cuales el cliente puede acceder a la red, ejemplo de ellos son: *switchs*, servidores, *routers*, teléfonos entre otros. Además en esta capa se localiza el Punto de Ejecución de Políticas (PEP), el cual tiene la función de recibir las políticas en forma de acciones de configuración y de establecerlas en el dispositivo.

- **Capa Middleware**: Es la responsable de permitir una traducción entre los objetos CIM-XML y cada uno de los dispositivos existentes, garantizando la interoperabilidad entre los sistemas existentes y nuevos que puedan aparecer, posibilitando integrar diferentes modelos de información y hacer la traducción al modelo CIM. Además permite la traducción entre los diferentes modelos de gestión de red y el modelo común de información.

- **Capa de Decisión**: Esta capa es la encargada de traducir las operaciones ejecutadas por el servidor Zabbix a un leguaje que pueda ser interpretado por los agentes de gestión que operan en los

elementos de la red y además se encarga de decidir sobre cuál de esos elementos se deben ejecutar dichas operaciones.

- **Capa de Base de Datos**: Esta capa consta con los elementos y servicios necesarios para proveer a las demás capas los servicios para acceder a los datos. Dicha capa permite realizar la conexión al servidor de base de datos y a los repositorios de información.

- **Capa de Seguridad**: Esta capa se establece de forma horizontal, dado que es un principio de la programación segura contar con medidas de seguridad para cada capa de un sistema. En esta capa residen los elementos necesarios de autenticidad, autenticación y autorización que brindan seguridad al sistema y garantizan la confidencialidad, integridad y disponibilidad de los datos, siendo estos el activo más preciado de una organización.

- **Capa de Trazas**: La función de esta capa es monitorizar y registrar constantemente cada una de las acciones que el usuario realiza. Se establece horizontalmente actuando sobre cada capa de la arquitectura.

- **Capa de aplicaciones**: Esta capa es la responsable de la presentación visual del sistema, lo cual permite que el usuario acceda e interactúe con los diferentes módulos que lo componen. Entre los módulos podemos encontrar: Gestión de servidores y SLA.

En esta última capa se encuentra el módulo para la gestión de servidores de telefonía IP, el cual presenta una arquitectura MTV (*Model-Template-View*) dado que el mismo se implementa con el *framework* Django. Esta arquitectura se descompone de la siguiente forma:

- ***Model*** (Modelo): El modelo establece qué datos son almacenados. En él se define cómo acceder a los datos, cómo validarlos, cuál es su comportamiento, y las relaciones entre ellos. Se encarga también de enviar los datos que solicite la Vista.

- ***Template*** (Plantilla): Es una página HTML (*Hyper Text Markup Language*) con algunas etiquetas extras propias del *framework*, que también crea contenido XML, CSS, JavaScript, entre otros. Su función es recibir los datos de la vista y organizarlos posteriormente para su presentación al navegador.

- ***View*** (Vista): Es la encargada de recibir la solicitud realizada por el navegador e interactuar con el modelo con el fin de obtener los datos deseados. Su propósito es determinar qué datos serán visualizados.

Para una mayor compresión en la Figura 8 se muestra el funcionamiento de esta arquitectura:

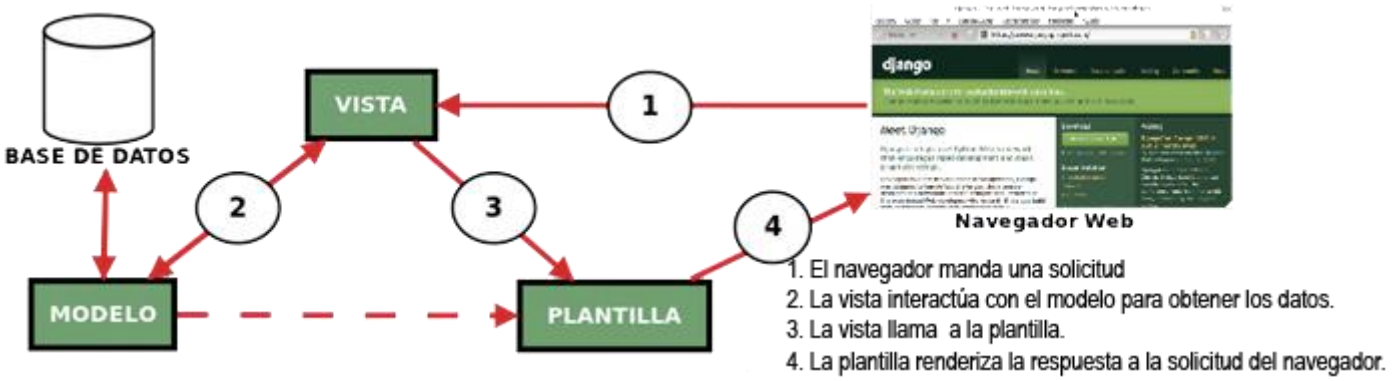


Figura 8. Funcionamiento del MTV de Django (59).

El módulo que se implementará se encuentra estructurado por paquetes. Esta estructura se compone por los siguientes módulos: Gestionar servidores, Seguridad, Contrato SLA, Auditoría y *Framework*. De ellos solo se implementa Gestionar servidores, los restantes ya se encuentran implementados por lo que solo se utilizan.

- **Gestionar servidores**: Es el encargado de monitorizar y controlar a través de políticas los servidores de telefonía IP.

- **Módulo Seguridad**: Es el módulo que se encarga de gestionar los roles de cada usuario según los privilegios y accesibilidad que tendrán al sistema. Con esto se garantiza que solo puedan acceder a la información necesaria en el momento que lo requiera.

- **Módulo SLA**: Su objetivo es gestionar la creación de contratos y la solicitud de servicios y productos.

- **Módulo de trazas**: Permite registrar cada una de las acciones que son realizadas por la aplicación web almacenando usuario, url, fecha y hora.

- **Módulo de *Framework***: Es donde se hospeda el *framework* ExtJS con el cual se diseña cada una de las interfaces del sistema.

Por la descripción realizada anteriormente, cada módulo del sistema hace uso del *framework* ExtJS, el cual se utiliza para la implementación de las vistas del sistema. Dicho *framework* implementa una arquitectura MVC (Modelo-Vista-Controlador), la cual separa el modelado del dominio, la presentación y las acciones basadas en datos ingresados por el usuario en tres clases diferentes: Modelo, Vista y Controlador (60).

- **Modelo**: Administra el comportamiento y los datos del dominio de aplicación, responde a requerimientos de información sobre su estado y responde a instrucciones de cambiar el estado.

- **Vista**: Se encargan de la visualización de la información contenida en el modelo.

- **Controlador**: Gestiona las entradas del usuario y realiza según la acción ejecutada por el usuario peticiones al modelo o a las vistas.

En la siguiente Figura 9 se ilustra cómo funciona esta arquitectura.

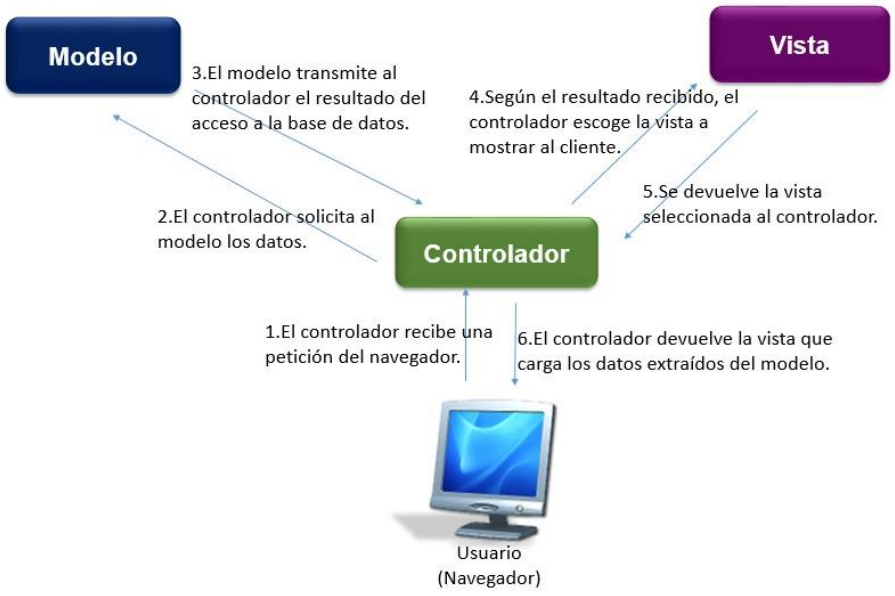


Figura 9. Funcionamiento del MVC en ExtJS (Elaboración propia).

2.9 Diagramas de clase del diseño

Un diagrama de clases de diseño (DCD) representa las especificaciones de las clases e interfaces software en una aplicación. Entre la información general encontramos clases asociaciones y atributos, interfaces, con sus operaciones y constantes, métodos, información acerca del tipo de los atributos, navegabilidad y dependencia (43). En la Figura 10 se muestra el diagrama de clase del diseño del escenario Gestionar servidor de telefonía IP.

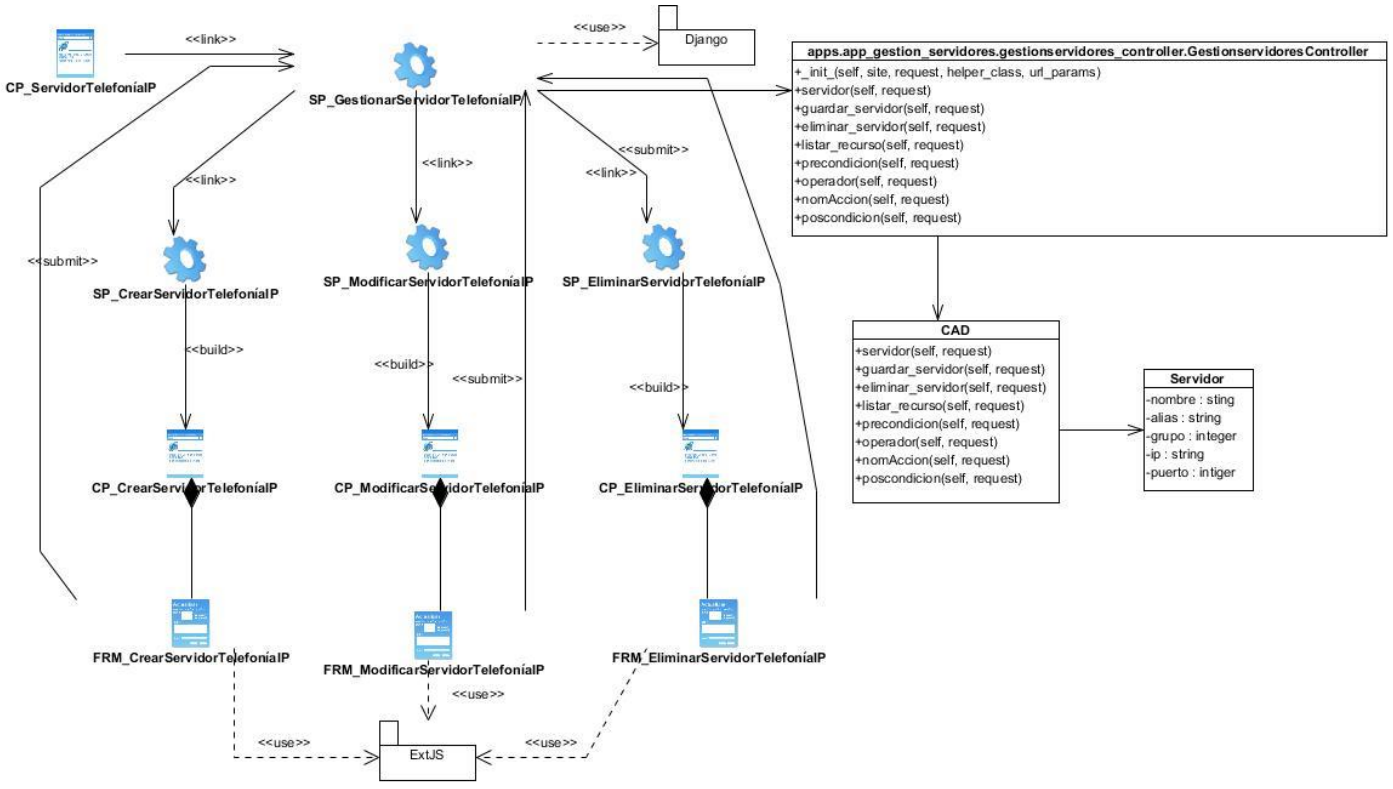


Figura 10. DCD Gestionar Servidor de Telefonía IP.

2.10 Diseño de la base de datos

Una base de datos es un conjunto de datos almacenados sin redundancias innecesarias en un soporte informático y accesible simultáneamente por distintos usuarios y aplicaciones. Los datos deben estar estructurados y almacenados de forma totalmente independiente de las aplicaciones que la utilizan (61). En

la Figura 11 se muestra el diagrama Entidad-Relación del módulo Gestión de servidores de telefonía IP, donde dicho módulo hace uso de la base de datos de Zabbix. En el **Anexo 3. Descripción del Diagrama Entidad-Relación** se explican cada una de las tablas que conforman el diagrama.

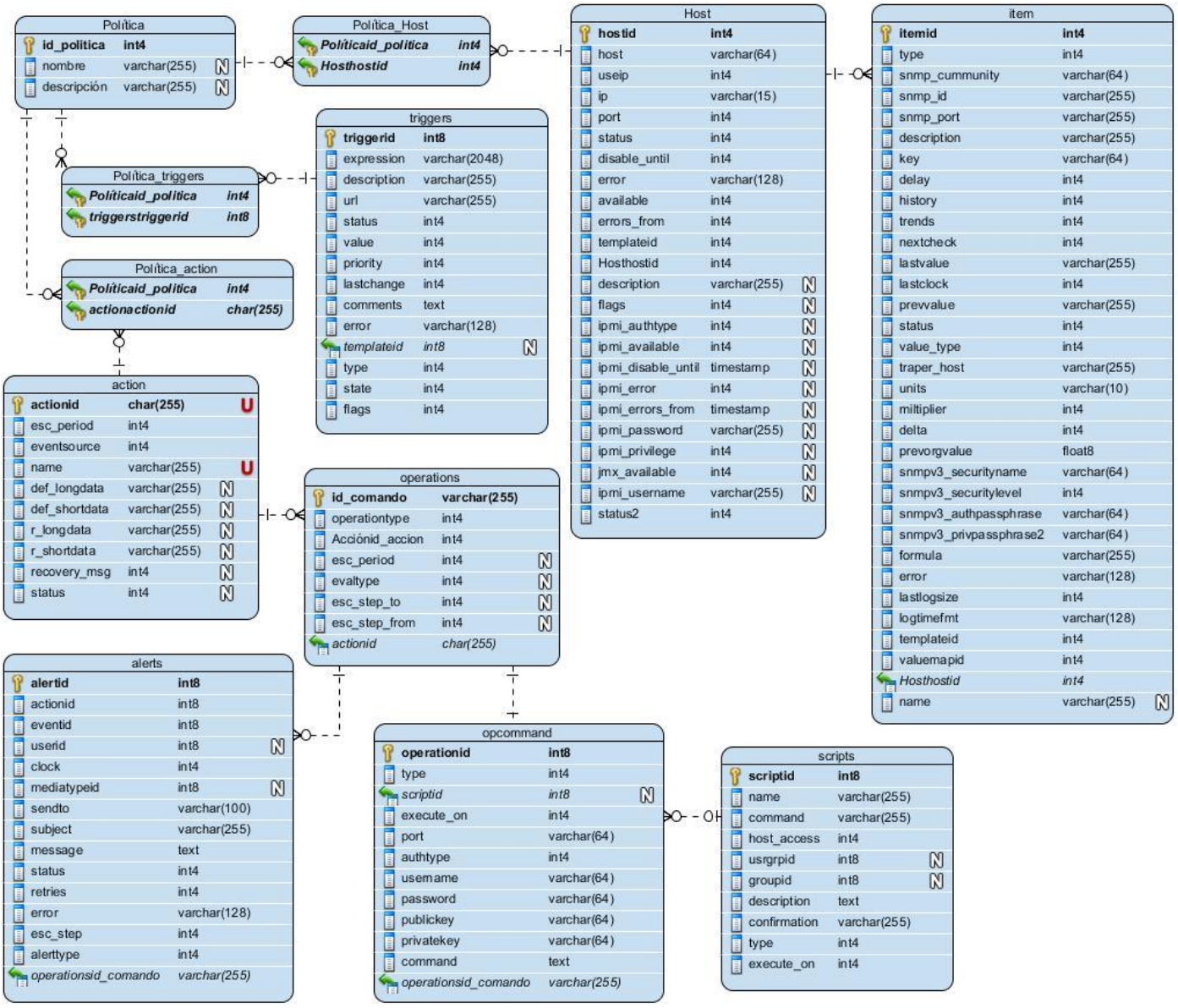


Figura 11. Diagrama Entidad-Relación.

2.11 Patrones de diseño

Los patrones de diseño describen una estructura de diseño que resuelve un problema de diseño particular dentro de un contexto específico y en medio de fuerzas que pueden tener impacto en la manera en que se aplica y se utiliza el patrón. La finalidad de cada patrón de diseño es proporcionar una descripción que le permita al diseñador determinar si el patrón es aplicable al trabajo actual, si se puede reutilizar y si puede servir como guía para desarrollar un patrón similar, pero diferente en cuanto a la funcionalidad o estructura (42).

Patrones GRASP

Los Patrones Generales de Software para Asignar Responsabilidades GRASP (*General Responsibility Assignment Software Pattern* por sus siglas en inglés), representan los principios básicos de la asignación de responsabilidades a objetos, expresados en forma de patrones (62).

Para el diseño de la solución se aplicaron los patrones Experto, Creador, Controlador, Alta Cohesión y Bajo Acoplamiento por lo que a continuación se realiza una descripción de los patrones mencionados:

- **Experto:** Se asignan responsabilidades al experto en información el cual es representado por una clase que cuenta con la información requerida para cumplir ciertas tareas. Un ejemplo evidente de este patrón son las clases librerías del módulo, las cuales constan con la información necesaria para cumplir con las responsabilidades sobre los elementos del negocio como puede ser la librería pyzabbix la cual tiene la responsabilidad de gestionar la información manejada y hacer cumplir las acciones en el mismo Zabbix (43).
- **Creador**: Se refiere a la asignación de la responsabilidad de crear objetos donde el propósito principal es encontrar un creador que se conecte con el objeto producido en algún evento. Se evidencia a la hora de crear políticas, teniendo en cuenta que las mismas contienen condiciones y acciones, o sea la política debe crear condiciones y acciones (43).
- **Controlador**: Asigna la responsabilidad del manejo de alertas de los eventos de un sistema a una clase. El patrón controlador se refleja en las clases controladoras pertenecientes al módulo, que son las clases que se encargan de obtener datos y enviarlos a las librerías y las vistas. Al delegar a un

controlador la responsabilidad de la operación de un sistema entre las clases del dominio favorece la reutilización de la lógica para manejar los procesos afines del negocio en aplicaciones futuras (43).

- **Bajo Acoplamiento:** Asigna responsabilidades de manera que un elemento (clase, subsistemas, sistemas, entre otros) no dependa demasiado, según el contexto, de otros elementos. El Bajo Acoplamiento soporta el diseño de clases que son más independientes, lo que reduce el impacto del cambio (43) . Esto se pone en evidencia en la clase **Política** donde, aun cuando no se conocen los recursos de los servidores, es posible crear una política aplicable a todos.
- **Alta Cohesión**: Asigna responsabilidades a los elementos (clases, subsistemas, entre otros) de tal manera que permanezca alta la medida de la fuerza con la que se relacionan y el grado de focalización de dichas responsabilidades (43). En el sistema se evidencia en la clase **GestionServidores** la cual es encargada solamente de lo referente a la información de los servidores y no las demás funcionalidades del sistema.

Patrones de diseño GoF

Los patrones GoF (*Gang of Four*) se clasifican en 3 categorías basadas en su propósito: creacionales, estructurales y de comportamiento, donde los patrones creacionales abstraen el proceso de creación de instancias ocultando los detalles de cómo los objetos son creados o inicializados, mientras que los patrones estructurales se ocupan de cómo las clases y objetos se combinan para formar grandes estructuras proporcionando nuevas funcionalidades. Por otra parte, los patrones de comportamiento están relacionados con los algoritmos y la asignación de responsabilidades entre los objetos. Son utilizados para organizar, manejar y combinar comportamientos (63).

Para el diseño del módulo se aplican los patrones GoF siguientes, con el objetivo de codificar y hacer reutilizables un conjunto de principios a fin de diseñar aplicaciones de alta calidad.

- **Instancia única** (*Singleton*): Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia (64). Este patrón se refleja en las clases controladoras que son instancias únicas. Los patrones de comportamiento estudian las relaciones de llamadas entre los diferentes objetos, normalmente ligados con la dimensión temporal. En el módulo se evidencia en la realización de la conexión a la base de datos garantizando una única conexión.

- **Mediador** (*Mediator*): Define un objeto que coordine la comunicación entre objetos de distintas clases, pero que funcionan como un conjunto (64). Se refleja en las librerías, que funcionan como mediadoras entre las clases controladoras y las modelos o acceso a datos.
- **Observador** (*Observer*): Define una dependencia de uno-a-muchos entre objetos, de tal forma que cuando un objeto cambie de estado se notifique y actualicen automáticamente todos los objetos que dependen de él (64). Se refleja en la clase *extend* que es el objeto *extend* de las clases controladoras, cuya función es cargar los elementos del marco de trabajo, dígase librerías, modelos y se encarga de actualizar la controladora instanciada.

2.12 Conclusiones parciales

En este capítulo se expusieron las características esenciales de la propuesta de solución desarrollada, con el fin de esclarecer las funcionalidades del módulo y los beneficios que aporta su implementación. Se definió el flujo de las actividades del proceso de negocio y se realizó el modelado conceptual del mismo, logrando obtener de esta manera los requisitos funcionales y no funcionales que debe cumplir el producto. Además se describió en capas lógicas la arquitectura sobre la cual se realiza la implementación.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

3.1 Introducción

En el presente capitulo, a partir de los resultados obtenidos en el diseño, se llega a la fase de construcción donde finalmente se materializa la solución y se verifica que funciona correctamente de acuerdo con cada uno de los requisitos establecidos en el capítulo anterior. Se muestran los diagramas de componentes y de despliegue, donde se observan las dependencias lógicas entre los elementos de software y los nodos necesarios para el despliegue del sistema respectivamente. Para validar el correcto funcionamiento de la solución se realizan pruebas de caja blanca y caja negra con las diferentes estrategias que define Pressman.

3.2 Diagrama de componentes

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Los componentes representan todos los tipos de elementos software que entran en el desarrollo de aplicaciones informáticas. En la Figura 12 se describe el diagrama de componentes del módulo Gestión de servidores de telefonía IP.

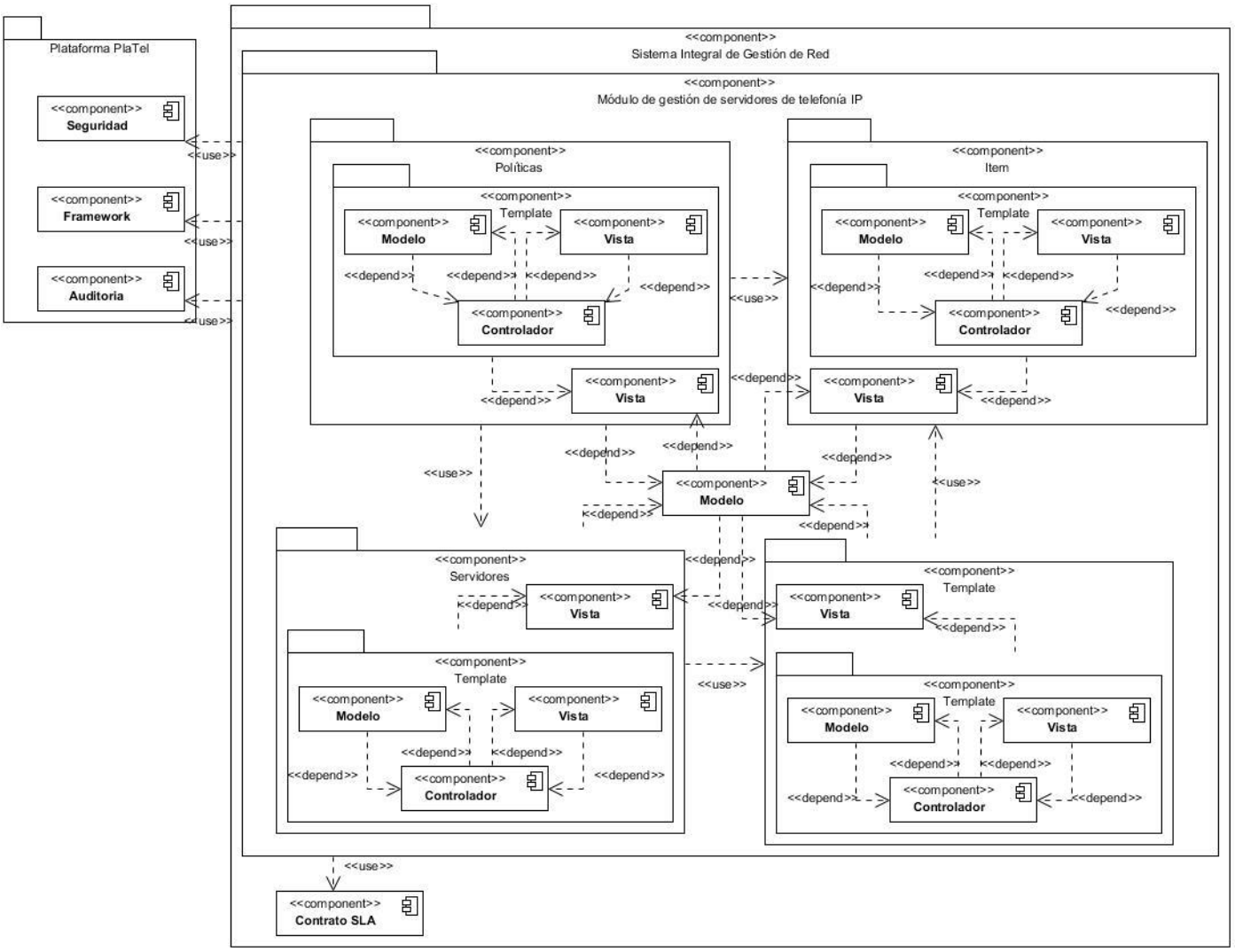


Figura 12. Diagrama de Componentes del módulo Gestión de servidores de telefonía IP.

3.3 Diagrama de despliegue

Un diagrama de despliegue modela la arquitectura en tiempo de ejecución de un sistema, que se utiliza para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes. En la Figura 13 se muestra el diagrama de despliegue del módulo Gestión de servidores de telefonía IP que está compuesto por una computadora, la cual tiene como requisito indispensable que debe contar con un navegador web Firefox versión 4 o superior, para que a través del protocolo HTTP se realice el intercambio

de información con el servidor de aplicaciones comunicándose este con el servidor Zabbix el cual almacena en su base de datos la información recibida por los agentes mediante el protocolo TCP/IP.

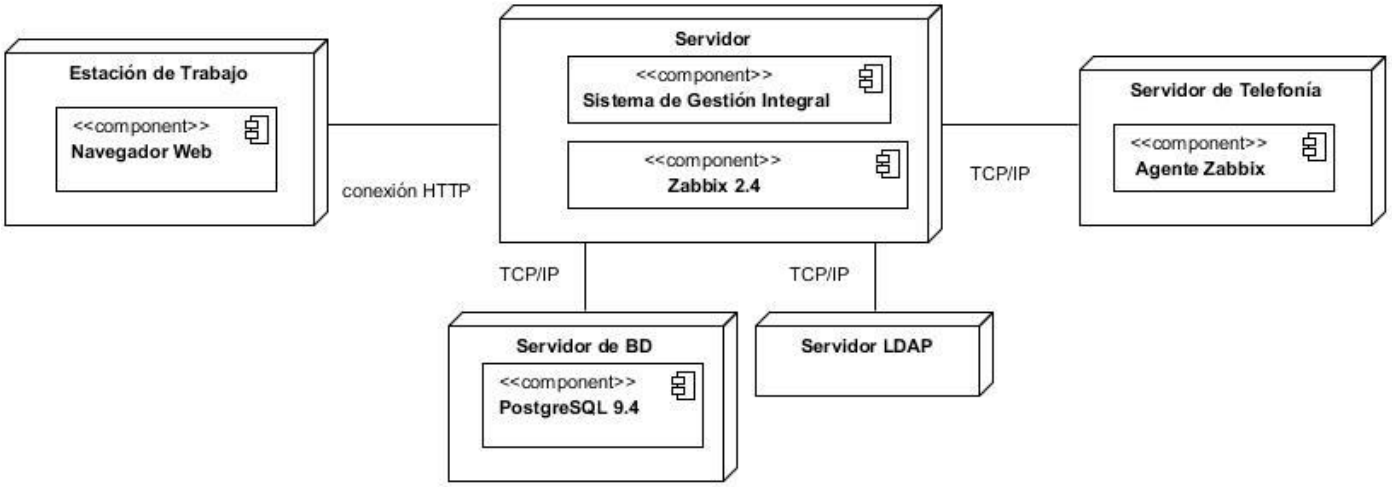


Figura 13. Diagrama de Despliegue del módulo Gestión de servidores de telefonía IP.

3.4 Prototipo de interfaz de usuario funcional

Los prototipos de interfaz de usuario funcional son las interfaces operativas que le permiten al usuario interactuar en tiempo real con la aplicación. A continuación en la Figura 14 se muestra el prototipo de interfaz de usuario funcional del Listado de Políticas. Los restantes se encuentran en el **Anexo 4. Prototipos de interfaz de usuario funcional**.

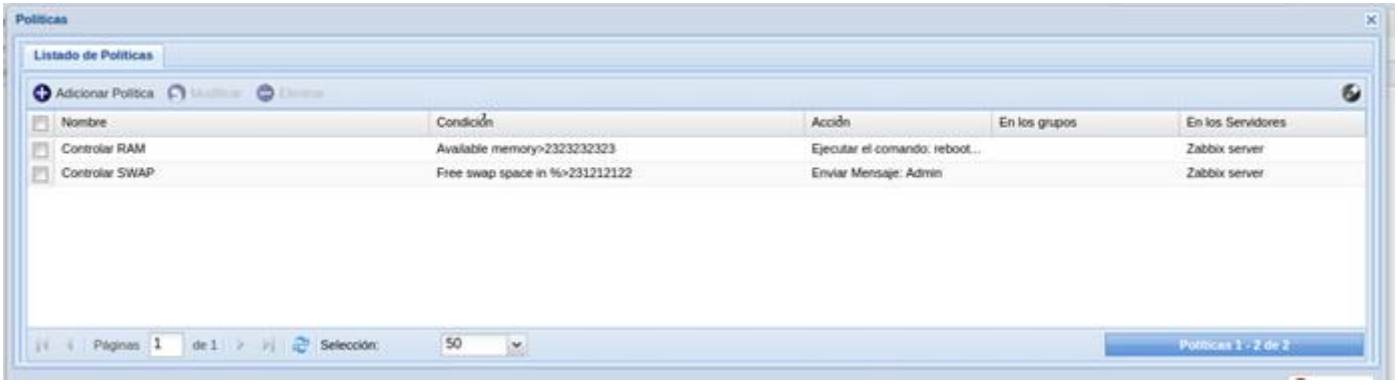


Figura 14. Prototipo de interfaz de usuario funcional Listado de Políticas.

3.5 Implementación

En la etapa de implementación, partiendo del análisis y diseño de la solución, se procede a desarrollar el correspondiente programa que solucione el problema mediante el uso de determinadas herramientas y tecnología.

3.5.1 Estándares de codificación

El proceso de implementación está guiado por una serie estándares de codificación que se encuentran estrechamente vinculados con el estilo, lenguaje y método de programación, con el fin de obtener un producto fácil de comprender y mantener. La implementación de la propuesta de solución se llevó a cabo bajo el estilo de codificación del lenguaje Python, que a continuación se describe.

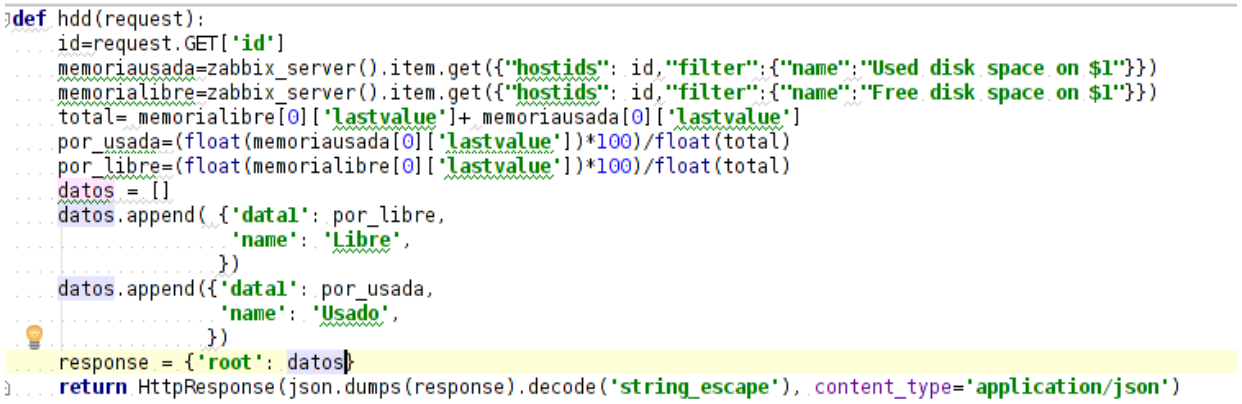
- **Indentación**: Se utilizan 4 espacios por cada nivel de indentación. Para el código realmente antiguo que no se desee estropear, se puede continuar usando indentaciones de 8 espacios. Las líneas de continuación deben alinearse verticalmente con el carácter que se ha utilizado (paréntesis, llaves, corchetes) o haciendo uso de la *hanging indent* (aplicar tabulaciones en todas las líneas con excepción de la primera). Al utilizar este último método es necesario tener en cuenta que no deben haber argumentos en la primera línea y debe utilizarse más indentación para distinguirse claramente como una línea de continuación (65). A continuación en la Figura 15 se muestra cómo se evidencia en el código.

```
Ext.define('grid_templates' , {
 extend: 'Ext.grid.Panel',
 region: 'center',
 dockedItems: [{xtype: 'toolbar',
                items: [{text: 'Crear Plantilla',
                         icon: '/static/framework/img/icons/adicionar.png',
                         itemId: 'AdicionarTemplate',
                         tooltip: 'Craer una  Plantilla nueva',
                         disabled: false
                         //...
                        },
                        '-',
                        {itemId: 'Eliminar',
                         text: 'Eliminar',
                         tooltip: 'Eliminar Plantilla',
                         icon: '/static/framework/img/icons/eliminar.png',
                         disabled: true}
                        //...
                       ]
               }],
```

Figura 15. Indentación.

- **Espacios en blanco en expresiones y sentencias**: Se evita usar espacios en blanco en las siguientes situaciones:
    - Inmediatamente después de entrar en un paréntesis o antes de salir de un paréntesis, corchete o llave.
    - Inmediatamente antes de una coma, punto y coma, o dos puntos.
    - Inmediatamente antes de abrir un paréntesis para una lista de argumentos de una llamada a una función.
    - Más de un espacio alrededor de un operador de asignación (u otro operador) para alinearlo con otro (65).

    A continuación en la Figura 16 se muestra cómo se evidencia en el código.



```
def hdd(request):
    id=request.GET['id']
    memoriausada=zabbix_server().item.get({"hostids": id,"filter":{"name":"Used disk space on $1"}})
    memorialibre=zabbix_server().item.get({"hostids": id,"filter":{"name":"Free disk space on $1"}})
    total= memorialibre[0]['lastvalue']+ memoriausada[0]['lastvalue']
    por_usada=(float(memoriausada[0]['lastvalue'])*100)/float(total)
    por_libre=(float(memorialibre[0]['lastvalue'])*100)/float(total)
    datos = []
    datos.append( {'datal': por_libre,
                   'name': 'Libre',
                  })
    datos.append({'datal': por_usada,
                  'name': 'Usado',
                 })
    response = {'root': datos}
    return HttpResponse(json.dumps(response).decode('string_escape'), content_type='application/json')
```

Figura 16. Espacios en blanco en expresiones y sentencias.

- **Comentarios**: Los comentarios deberían ser frases completas. Si un comentario es una frase o sentencia, la primera palabra debería estar en mayúsculas, a menos que sea un identificador que comience con una letra en minúsculas. Si un comentario es corto, se puede omitir el punto al final (65). A continuación en la Figura 17 se muestra cómo se evidencia en el código.

```
#------Instanciando API-----------------------------------------------------------
    zapi = ZabbixAPI(server = server, path="", log_level=6)
    zapi.login(username, password)
    return zapi
    hosts = zapi.host.get({ "sortfield": "name"})
# -----Aqui se obtienen los datos del Zabbix-----------------------------------------
def listar_servidores(request):
    hosts = zabbix_server().host.get({ "sortfield": "name"})
    return HttpResponse(json.dumps(hosts).decode('string_escape'), content_type='application/json')
```

Figura 17. Comentarios.

- **Estilos de nombrados**:
    - *UpperCamelCase*: Todas las palabras se escriben juntas comenzando cada una con letra inicial mayúscula.
    - *lowercase_separated_by_underscores*: Las palabras se escriben en minúsculas separadas por guiones bajos (65).

A continuación en la Figura 18 se muestra cómo se evidencia en el código

```
def eliminar_servidor(request):

def listar_servidores(request):
```

Figura 18. Estilos de nombrados.

3.6 Pruebas

Las pruebas son el conjunto de actividades dentro del desarrollo de un software que permiten erradicar los errores aún no descubiertos, dando como resultado una solución con calidad. La planeación y diseño de las pruebas puede comenzar a partir de que el modelo de análisis esté completo, no es necesario esperar a que se genere código. Se aconseja primero se realicen pruebas a cada componente individualmente, luego a grupos integrados de componentes y por último al sistema completo (42).

3.6.1 Métodos de prueba

A continuación se describen los métodos o enfoques que se tienen en cuenta para la realización de las pruebas:

Prueba de caja blanca:

La prueba de caja blanca, en ocasiones llamada prueba de caja de cristal, es un método de diseño que usa la estructura de control descrita como parte del diseño al nivel de componentes para derivar los casos de pruebas. Al emplear los métodos de prueba de caja blanca, el ingeniero de software podrá derivar casos de prueba que garanticen que todas las rutas independientes dentro de un módulo se han ejercitado por lo menos una vez, que ejerciten los lados verdaderos y falsos de todas las decisiones lógicas, que ejecuten todos los bucles en sus límites y dentro de sus límites operacionales y que además ejerciten estructuras de datos internos para asegurar su validez. Existen ciertas técnicas para el diseño de este tipo de pruebas que permiten decidir qué sentencias o caminos se deben examinar con los casos de prueba, entre las que se encuentran las siguientes (42):

- **Prueba de la ruta básica:** El método de ruta básica permite que el diseñador de casos de pruebas obtenga una medida de complejidad lógica de un diseño procedimental y que use esta medida como guía para definir un conjunto básico de rutas de ejecución. Los casos de pruebas derivados para ejercitar el conjunto básico deben garantizar que se ejecuta cada instrucción del programa por lo menos una vez durante la prueba (42).
- **Prueba de condición:** Es un método de diseño de caso de prueba que ejercita las condiciones lógicas contenidas en un módulo del programa (42).
- **Prueba de flujo de datos:** Se seleccionan rutas de prueba en un programa de acuerdo con las ubicaciones de las definiciones y los usos de las variables del programa. El enfoque de prueba de flujo de datos se ilustra suponiendo que a cada instrucción de un programa se le asigna un número de instrucción, y que ninguna función modifica sus parámetros o variables globales (42).
- **Prueba de bucles:** Es una técnica que se centra exclusivamente en la validez de la construcción de bucles (42).

Los casos de pruebas de caja blanca del módulo Gestión de servidores de telefonía IP son generados por la técnica de prueba de la ruta básica.

Prueba de Caja Negra:

Las pruebas de caja negra, también denominadas, pruebas de comportamiento, son el tipo de prueba que se concentran en los requisitos funcionales y se aplican a la interfaz del software. Este tipo de prueba examina algún aspecto funcional de un sistema que tiene poca relación con la estructura lógica interna del software. Es decir, permiten al ingeniero de software derivar conjuntos de condiciones de entrada que ejercitarán por completo todos los requisitos funcionales de un programa. Las pruebas de caja negra tratan de encontrar errores en las siguientes categorías: funciones incorrectas o faltantes, errores de interfaz, errores en estructura de datos o en acceso a base de datos externas, errores de comportamiento o desempeño y errores de inicialización y término (42).

Para desarrollar la prueba de caja negra existen varias técnicas, entre ellas se encuentran:

- **Partición equivalente:** Es un método de prueba de caja negra que divide el dominio de entrada de un programa en clases de datos a partir de las cuales pueden derivarse casos de prueba. La partición equivalente se esfuerza por definir un caso de prueba que descubra ciertas clases de errores, reduciendo así el número total de casos de prueba que deben desarrollarse. Una clase de equivalencia representa un conjunto de estados válidos o no válidos para condiciones de entrada. Típicamente, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición lógica (42).
- **Análisis de valores al límite**: Es una técnica de diseño de casos de pruebas que completa la partición equivalente. En lugar de seleccionar cualquier elemento de una clase de equivalencia lleva a la selección de casos de prueba en los extremos de la clase. Esta técnica en lugar de centrarse solamente en las condiciones de entrada, obtiene casos de prueba también para el campo de salida.
- **Prueba funcional**: Es una técnica de diseño de casos de prueba centrada en verificar que se cumple cada uno de los requisitos funcionales del software.

Los casos de pruebas de caja negra del módulo Gestión de servidores de telefonía IP se definieron a través de pruebas funcionales comprobando que dicho módulo responde correctamente a cada uno de los requisitos funcionales especificados.

3.6.2 Estrategias de Prueba

Una estrategia de prueba del software integra los métodos de diseño de caso de pruebas del software en una serie de pasos bien planeados. La estrategia de prueba debe incorporar la planeación de pruebas, el diseño de casos de pruebas, la ejecución de pruebas y la recolección y evaluación de los datos resultantes (42).

- **Prueba de unidad**: Se concentra en el esfuerzo de verificación de la unidad más pequeña del diseño del software: componente o módulo de software. Esta prueba asegura que cada componente de manera individual funciona correctamente como unidad. Tomando como guía la descripción del diseño al nivel de componentes, se prueban importantes caminos de control para descubrir errores dentro de los límites del módulo. Las pruebas de unidad se concentran en la lógica del procesamiento interno y en las estructuras de datos dentro de los límites de un componente (42).
- **Prueba de regresión**: Son la repetición de las pruebas realizadas anteriormente con la finalidad de comprobar que no existan nuevos errores producto de los cambios realizados.

Las pruebas unitarias fueron realizadas al código a medida que se implementaron las funcionalidades del módulo Gestión de servidores de telefonía IP con el objetivo de lograr un correcto funcionamiento del mismo. Posterior a cada una de estas pruebas se ejecutaron pruebas de regresión a dicho módulo con la finalidad de verificar que los cambios realizados no hubieran generado nuevos errores y de ser así corregirlos.

3.7 Diseño de los casos de prueba

Un diseño de caso de prueba no es más que un conjunto de entradas, condiciones de ejecución y resultados esperados desarrollados para determinar si el requisito de una aplicación está parcial o completamente satisfecho. Su propósito es especificar una forma de probar el sistema que incluya las entradas, los resultados esperados y las condiciones bajo las que ha de probarse (41). A continuación se describen las distintas estrategias que se siguieron para elaborar los casos de prueba.

3.7.1 Diseño de los casos de prueba de caja blanca

El método de prueba de caja blanca se le aplicó a varios métodos de la clase **views**, haciendo uso de la técnica del camino básico; las respuestas esperadas al ejecutar las funcionalidades son visualizadas a través del complemento de Mozilla Firefox (Firebug). A continuación se muestra el diseño de caso de prueba para el método **guardar_servidor(request)**, los restantes casos de prueba se pueden encontrar en el **Anexo 5. Diseño de los casos de prueba de caja blanca**.

Clase Views

- Prueba unitaria de la clase: **views**
- Casos de prueba del método: **guardar_servidor(request)**
- Variables a considerar en el caso de prueba**: request**
- Clase de Equivalencia para la variable: **request**

Clase de Equivalencia	Clase de Equivalencia	Clasificación de las Clases de Equivalencia
1	Verificar que los datos del servidor se guarden correctamente.	Válida.

*Tabla 2. Clase de Equivalencia para la variable **request.***

Caso de Prueba:

1. **guardar_servidor(request)**

Caso de Prueba	Objetivo de la Prueba	Datos de Entrada	Salida Esperada	Salida Obtenida
1	Verificar que se pueda guardar los datos del contrato.	**request** pasado por parámetro.	Datos guardados correctamente.	Datos guardados correctamente.

Tabla 3. Diseño de Caso de Prueba para el método Guardar Servidor.

3.7.2 Diseño de los casos de prueba de caja negra

Los casos de pruebas de caja negra se diseñaron siguiendo la técnica de las pruebas de funcionalidad. Dichas pruebas de funcionalidad se le realizaron a varios requisitos funcionales del módulo Gestión de servidores de telefonía IP con el objetivo de verificar que dicho módulo brinda la respuesta correcta ante las diferentes entradas que admite. A continuación se muestra el diseño de caso de prueba para la funcionalidad Gestionar servidores. Los restantes diseños de casos de prueba se encuentran en el **Anexo 6. Diseño de los casos de prueba de caja negra**.

.

Escenario	Descripción	Nombre del Servidor	Nombre Visible	Descripción del Servidor	Seleccionar Grupo de Servidores	Seleccionar Plantilla	Agregar Interfaces	Respuesta del Sistema
Adicionar datos del Servidor.	Se introducen los datos del servidor.	V	V	V	V	V	V	Se adiciona el servidor a la lista de servidores mostrada en la interfaz principal.
		Letras	Letras	Letras	Seleccionar al menos uno.	Puede o no estar seleccionada alguna plantilla.	Adicionar dirección ip #1<255. #2<255. #3<255. #4<254	
		NA	V	V	V	V	V	El módulo no permite que en el campo entren esos datos.
		Caracteres extraños.	Letras	Letras	Seleccionar al menos uno.	Puede o no estar seleccionada alguna plantilla.	Adicionar dirección ip #1<255. #2<255. #3<255. #4<254	
		V	NA	V	V	V	V	El módulo no permite que se active el botón de guardar.
		Letras	Vacío	Letras	Seleccionar al menos uno.	Puede o no estar seleccionada alguna plantilla.	Adicionar dirección ip #1<255. #2<255. #3<255.	

							#4<254	
		NA	V	V	V	V	V	
		Vacío	Letras	Letras	Seleccionar al menos uno.	Puede o no estar seleccionada alguna plantilla.	Adicionar dirección ip #1<255. #2<255. #3<255. #4<254	
		NA	NA	V	NA	V	NA	
		Vacío	Vacío	Letras	Sin seleccionar.	Puede o no estar seleccionada alguna plantilla.	Vacío	
		V	V	V	NA	V	V	
		Letras	Letras	Letras	Sin seleccionar	Puede o no estar seleccionada alguna plantilla	Adicionar dirección ip #1<255. #2<255. #3<255. #4<254	
		V	V	V	V	V	NA	
		Letras	Letras	Letras	Seleccionar al menos uno.	Puede o no estar seleccionada alguna plantilla.	Vacío	
		V	V	V	V	V	NA	El módulo no permite

		Letras	Letras	Letras	Seleccionar al menos uno.	Puede o no estar seleccionada alguna plantilla.	Letras	que en el campo entren esos datos.

Tabla 4. Diseño de Caso de Prueba para la funcionalidad Gestionar Servidores.

3.8 Resultado de las pruebas realizadas al módulo.

La realización de estas pruebas permitió detectar algunas no conformidades la cuales fueron erradicadas inmediatamente. A continuación se muestra la Tabla 5 con el análisis de las no conformidades por iteración:

Iteración	**Cantidad de no conformidades**		**Relacionadas con**
1era	Pruebas Unitarias	Pruebas de Regresión	Errores en las funcionalidades del código.
	57	41	
2da	Pruebas Unitarias	Pruebas de Regresión	Errores en las funcionalidades del código.
	20	11	
3ra	Pruebas Unitarias	Pruebas de Regresión	Errores de interfaz y de validación.
	5	1	
4ta	Pruebas Unitarias	Pruebas de Regresión	Error de validación.
	1	0	

Tabla 5. Análisis de las no conformidades por iteración.

3.9 Conclusiones parciales

En este capítulo a través de los diagramas de despliegue y componentes se logró conocer la distribución física en términos de componentes y nodos. Se logró implementar la propuesta de solución obteniendo así un módulo capaz de realizar una gestión integrada de servidores de telefonía IP basado en políticas. Además se describieron los estándares de codificación utilizados con el fin de que los programadores de

los restantes módulos del Sistema Integral de Gestión de Redes, los conozcan y puedan guiarse por dichos estándares manteniendo así un producto legible y fácil de comprender. Aplicando pruebas de caja negra y caja blanca con las diferentes estrategias que define Pressman se obtuvo como resultado un correcto funcionamiento del sistema, libre de errores. Estas pruebas además permitieron comprobar los requisitos funcionales definidos para el módulo a partir de los diseños de casos de pruebas, evaluándose la calidad.

CONCLUSIONES GENERALES

Culminando el desarrollo de la presente solución, teniendo en cuenta los objetivos trazados en la investigación se obtienen resultados favorables los cuales pueden concluirse de la siguiente forma:

- Con el estudio de WBEM y de PBNM se demostró la necesidad de llevar a cabo una gestión integrada donde se vinculen las políticas asociadas al negocio con la infraestructura existente en la organización.
- Luego de haber estudiado los principales sistemas capaces de realizar la monitorización y control de servidores de la telefonía IP, se demostró a través de una evaluación que Zabbix era el sistema indicado para integrar a la solución con el cual se puede llevar a cabo una gestión integrada de servidores de telefonía IP.
- La realización del análisis y diseño de la solución posibilitó esclarecer las funcionalidades y los requisitos que debe cumplir el módulo, sirviendo además como punto de partida para la implementación de dicho módulo.
- La implementación de las funcionalidades sobre la base de la arquitectura del Sistema Integral de Gestión de Redes, sustentada en componentes y mediante el empleo de patrones adecuados, potenció la reutilización del código y la reducción del tiempo de desarrollo.
- Las pruebas unitarias y de regresión permitieron encontrar y corregir los errores no detectados durante la implementación, permitiendo comprobar las especificaciones requeridas y la validación del módulo implementado.

De esta manera se da solución a la situación problemática, con un módulo capaz de monitorizar y controlar a través de políticas basándose en una gestión integrada los servidores de telefonía IP del CDTT y que además se integra con los restantes módulos del Sistema Integral de Gestión de Redes.

RECOMENDACIONES

Después de haber logrado los objetivos trazados al inicio del presente trabajo de diploma, como parte de la culminación de dicho trabajo se recomienda:

- Extender las funcionalidades del módulo para que el mismo sea capaz de monitorizar y controlar a través de políticas mediante una gestión integrada todos los elementos (hardware y software) que componen una infraestructura de VoIP.
- Añadir la capacidad en el módulo de adaptarse a cualquier sistema de monitorización y control existente.
- Continuar la implementación de los restantes módulos del Sistema Integral de Gestión de Red, guiándose por el proceso de desarrollo de software definido por UCID, manteniendo la línea tecnológica y el uso de las herramientas empleadas hasta el momento.

REFERENCIAS BIBLIOGRÁFICAS:

1. Unión International de las Telecomunicaciones (UIT), Oficina de Desarrollo de las Telecomunicaciones (BDT). *Measuring the Information Society Report.* Ginebra : s.n., 2014. 978-92-61-14661-0.

2. Gómez López, Julio y Gil Montoya, Francisco. *VoIP y Asterisk Redescubriendo la telefonía.* Almería : Alfaomega Ra-Ma, 2008.

3. Cubadebate. [En línea] 19 de Febreo de 2015. [Citado el: 20 de Marzo de 2015.] http://www.cubadebate.cu/wp-content/uploads/2015/02/Resumen-de-la-propuesta-de-Bases-y-Prioridades-de-Informatizaci%C3%B3n.pdf.

4. Landívar, Edgar. *Comunicaciones Unificadas con Elastix.* 2009. Vol. 1.

5. Mayo Murillo, Diana Lorena . *Sistemas de Voz sobre IP en una Red de Infraestructura Mesh para Gestión de Emergencias.* Universidad Politénica de Valencia. Valencia : s.n., 2013.

6. Fernándes, Eugenio Eduardo Villar. *Virtualización de servidores de telefonía IP en GNU/Linux.* Almería : s.n., 2010.

7. Moya Ferrer, Jaime. Adquisición de Conocimientos Básicos. *Análisis de Herramientas de Gestión de VoIP.*

8. Baró Menéndez , Duany y Vicet Wilson, Madelayne. *ArPlaTel: Arquitectura para una Plataforma Telefónica Basada en Voz sobre IP.* Universidad de las Ciencias Informáticas . 2009.

9. Gutiérrez Gil, Roberto. *Seguridad en VoIP: Ataques, Amenazas y Riesgos.* Universidad de Valencia. 2008.

10. Microsoft. Microsoft. *Developer Network.* [En línea] [Citado el: 13 de Enero de 2015.] https://msdn.microsoft.com/es-es/library.

11. López de Vergara, J. E. *Especificación de Modelos de Información de Gestión de Red Integrada Mediante el Uso de Ontologías y Técnicas de Representación del Conocimiento.* Ingeniería de Sistemas Telemáticos, Universidad Politécnica de Madrid. Madrid : s.n., 2003.

12. *Telecommunications Network Management into the 21st century.* autores, Varios. s.l. : IEEE PRESS, 1994.

13. *Worlwide Intelligent Systems: Approaches to Telecommunications and Network Management.* autores, Varios. s.l. : IOS Press, 1995.

14. *Data Communications and Networking.* Forouzan, Behrouz A. New York : Published by McGraw-Hill, 2013.

15. *Analysis of Telecommunication Management Technologies.* Laghari, Khalil ur rehman, Yahia, Imen Grida ben y Crespi, Noel. 2, Noviembre de 2009, International Journal of Computer Science & Information Technology (IJCSIT), Vol. 1.

16. *Maestría de Informática Avanzada curso de Gestión de Red.* Peña Casanova, Mónica. La Habana : s.n., 2014.

17. "Web-Based Enterprise Management (WBEM) ". *DMTF Standards.* [En línea] DMTF. [Citado el: 15 de Enero de 2015.] http://www.dmtf.org/standards/wbem.

18. "Common Information Model (CIM) ". *DMTF Standards.* [En línea] DMTF. http://www.dmtf.org/standards/standard_cim.php..

19. *Aplicación de gestión para estaciones de usuarios basada en el estándar WBEM y su aplicación a la Red de Datos de la Universidad del Cauca.* Agredo Méndez, Guefry, Maya Ortíz, Natalia y Maya Ortíz, Eva Juliana. Cauca : s.n., 2003.

20. *Gestión Autonómica.* Ramos Encinosa, Alejandro y Alavés García , Dalia. 1, 2012, Revista Telem@tica, Vol. 10.

21. *The representation of policies as System Objects.* Moffet, J. y Sloman, M. 2 & 3, s.l. : SIGOIS Bulletin, 1991, Vol. 12, págs. 171-184.

22. *Lenguajes de libre distribución para la gestión de redes basada en políticas.* Reyes, Angélica y Barba, Antoni. Cataluña : s.n., 2005.

23. Alpers, B. y Plansky, H. Domain and Policy Based Management: Concepts and Implementation Architecture. *IEEE/IFIP Workshop on Distributed Systems Operations and Management.* Toulouse : s.n., 1994.

24. *Policy Driven Management for Distributed Systems .* Sloman, M. [ed.] OM. 4, Malek : Plenum Publishing Corporation, 1995, Journal of Network and Systems Management, Vol. 2, págs. 333-360.

25. Verma, D. C. *"Policy-Based Networking. Architecture and Algorithms".* s.l. : New Riders Publishing, 2001.

26. IETF. The IETF Policy Framework Group . [En línea] IETF. [Citado el: 20 de Febrero de 2015.] http://www.ietf.org/html.charters/policy-charter.html.

27. Durham, D., y otros. *"The COPS (Common Open Policy Protocol)" .* s.l. : IETF, 2000.

28. Moore, B. *Policy Core Information Model.* 2001.

29. *Un acercamiento a la getión de redes basada en políticas.* Santiesteban Pérez, Irina Ivis y Pérez Reyes, Carlos Miguel. La Habana : s.n., 2013, RCCI. 2227-1899.

30. Cacti. Cacti. [En línea] [Citado el: 10 de Enero de 2015.] http://www.cacti.net.

31. Caballero Cruz, Luis. *Sistema de monitorización: Pandora FMS.* Lenguajes y Sistemas Informáticos, Universidad de Sevilla. Sevilla : s.n., 2012.

32. Cacti. Cacti Users. [En línea] [Citado el: 20 de Enero de 2015.] http://cactiusers.org.

33. Munin. Munin. [En línea] [Citado el: 21 de Enero de 2015.] http://munin-monitoring.org.

34. Zenoss. Zenoss. [En línea] [Citado el: 20 de Enero de 2015.] http://www.zenoss.com.

35. Zabbix. Zabbix. [En línea] [Citado el: 20 de Enero de 2015.] http://www.zabbix.com/es.

36. —. Zabbix. [En línea] [Citado el: 20 de Enero de 2015.] http://www.zabbix.org.

37. Nagios. Nagios. [En línea] [Citado el: 21 de 2015 de Enero.] http://www.nagios.org.

38. González Pérez, Emilio Manuel. *Sistema de Monitorización de la Infraestructura CCTV en la UC3M con Zabbix.* Universidad Carlos III de Madrid. Madrid : s.n., 2010.

39. Méndez, Gonzalo. *Proceso Software y Ciclo de Vida.* Ingeniería de Software e Inteligencia Artificial, Universidad Complutense . Madrid : s.n., 2009.

40. Jacaboson, I., Booch, G y Rumbaugh, J. *El Proceso Unificado de Desarrollo de Software.* s.l. : Addison Wesley, 2000.

41. Unidad de Compatibilización Integración y Desarrollo De Software para la Defensa. *Proceso de Desarrollo y Gestión de Proyectos de Software (Versión 1.5).* La Habana : s.n., 2012.

42. Pressman, Roger S. *Ingeniería del Software: Un Enfoque Práctico.* Sexta. Madrid : McGraw-Hill, 2005. 9701054733.

43. Larman, Craig. *UML y patrones: una introducción al análisis y diseño orientado a objetos y al proceso unificado.* s.l. : Pearson Educación, 2003. 8420534382/ 9788420534381.

44. *Herramientas CASE. ¿Cómo incorporarlas con éxito en nuestra organización?* González López, Pascual, González López, Ana Amelia y Gallud Lázaro, José Antonio. 10, Albacete : Revista de la Facultad de Educación de Albacete, 1995, págs. 195-208. 0214-4824.

45. Visual Paradigm International. Visual Paradigm. [En línea] [Citado el: 4 de Febrero de 2015.] http://www.visual-paradigm.com.

46. Louden, Kenneth C. *Lenguajes de programación: principios y práctica.* s.l. : Thomson, 2004. 9706862846, 9789706862846.

47. Van Rossum, Guido . *El tutorial de Python.* s.l. : Fred L. Draked, 2009.

48. Sánchez Asenjo, Jorge. *Gestión de Bases de Datos Usando Oracle SQL y PL/SQL.* Villamuriel de Cerrato : s.n., 2013.

49. PostgreSQL Global Development Group. PostgreSQL. [En línea] [Citado el: 5 de Febrero de 2015.] http://www.postgresql.org.es.

50. Jetbrains. [En línea] https://www.jetbrains.com/pycharm/.

51. pgAdmin Development-Team. pgAdmin. [En línea] [Citado el: 5 de Febrero de 2015.] http://www.pgadmin.org/.

52. Lafosse, Jarome. *El framework de desarrollo de aplicaciones Java EE.* Barcelona : ENI, 2010. 9782746055421.

53. Django. [En línea] http://www.djangoproject.com.

54. Sencha. [En línea] http://www.sencha.com/products/extjs/.

55. Nginx Community. [En línea] http://wiki.nginx.org/Main.

56. Weske, M. *Business Process Management.Concepts, Languages,Architectures.* Berlin : Springer, 2007. 978-3-540-73521-2.

57. Sparks, Geoffrey. *El Modelo de Proceso de Negocio, Una Introducción al UML.* Australia : Sparx Systems.

58. Moquillaza Henríquez, Santiago Domingo, Vega Huerta, Hugo y Guerra Grados, Luis. *Programación en N capas.* Universidad Nacional Mayor de San Marcos. San Marcos : s.n., 2010. 1816-3823.

59. Infante Montero, Sergio. *Curso Django El framework para detallistas con deadlines.* Eugenia Tobar. s.l. : Maestros del web, 2012.

60. Reynoso , Carlos y Kicillof, Nicolás. *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* Universidad de Buenos Aires. Buenos Aires : s.n., 2004.

61. Cobo, Angel. *Diseño y programación de bases de datos.* s.l. : Visión Libros. 8499831478, 9788499831473.

62. Saavedra, Jose. [En línea] [Citado el: 12 de Marzo de 2015.] http://jorgesaavedra.wordpress.com/category/patrones-grasp/.

63. Desarrollo.net. [En línea] [Citado el: 5 de Marzo de 2015.] http://desarrollo.net.

64. Patrones de diseño. [En línea] [Citado el: 5 de Marzo de 2015.] http://patronesdediseno.net16.net/.

65. Van Rossum, Guido, Warsaw, Barry y Coghlan , Nick . *PEP 0008 -- Style Guide for Python Code.* 2001.

66. Anías Calderon, Caridad, Villariño Bolaño, Luis A. y Preciado Velasco, Jorge E. *Gestión de Red Basada en Políticas.* 2005.
67. The IBM Autonomic Computing Initiative. [En línea] http://www.ibm.com/autonomic/.
68. WTCS. [En línea] http://www.wtcs.org/snmp4tpc/snmp.htm.
69. *Cuestión 19-2/1: Implementación de los servicios de telecomunicaciones IP en los países en desarrollo UIT-D.* Unión Internacional de Telecomunicaciones, Oficina de Desarrollo de las Telecomunicaciones. Ginebra : s.n., 2014.

BIBLIOGRAFÍA

- **Gómez López, Julio y Gil Montoya, Francisco.** "*VoIP y Asterisk Redescubriendo la telefonía"*, 2008.
- **UIT, BDT.** "*Measuring the Information Society Report"* , 2014.
- **Landívar, Edgar. "***Comunicaciones Unificadas con Elastix",* 2009**.**
- **Laghari, Khalil ur rehman, Yahia, Imen Grida ben y Crespi, Noel. "***Analysis of Telecommunication Management Technologies"* , IJCSIT, 2009.
- **Ramos Encinosa, Alejandro y Alavés García , Dalia.** "*Gestión Autonómica"*, 2012.
- **Verma, D. C.** *"Policy-Based Networking. Architecture and Algorithms"*, 2001.
- **Caballero Cruz, Luis.** "*Sistema de monitorización: Pandora FMS"*, 2012.
- **Unidad de Compatibilización Integración y Desarrollo De Software para la Defensa.** "*Proceso de Desarrollo y Gestión de Proyectos de Software",* 2012.
- **Pressman, Roger S.** "*Ingeniería del Software: Un Enfoque Práctico"*, 2005.
- **Larman, Craig.** "*UML y patrones: una introducción al análisis y diseño orientado a objetos y al proceso unificado"*, 2003.
- **Anías Calderon, Caridad, Villariño Bolaño, Luis A. y Preciado Velasco, Jorge E.** "*Gestión de Red Basada en Políticas",* 2005.
- **UIT, BDT.** "*Cuestión 19-2/1: Implementación de los servicios de telecomunicaciones IP en los países en desarrollo",* 2014.

GLOSARIO DE TÉRMINOS

Banda ancha: Es un tipo de tecnología que ofrece conexiones de baja latencia, alta capacidad para transportar grandes cantidades información en muy poco tiempo y donde el servicio Internet está sujeto a actualizaciones instantáneas en tiempo real.

COPS: El Protocolo de Servicio Público de Políticas Comunes (COPS, *Common Open Policy Service Protocol*) es un protocolo estándar propuesto para el intercambio de información sobre políticas de control sobre la calidad de servicio de una red.

DMTF: El Grupo de Trabajo de Gestión Distribuida (DMTF, *Distributed Management Task Force*) es una organización enfocada al desarrollo, la adopción y la unificación de estándares de administración e iniciativas para entornos de escritorio, empresariales y de Internet.

IETF: El Grupo de Trabajo de Ingeniería de Internet (IETF, *Internet Engineering Task Force*) es una organización internacional abierta a cualquier persona interesada, esta organización contribuye a la ingeniería y evolución de las tecnologías de Internet siguiendo la misión de hacer que Internet funcione mejor mediante la producción de documentos técnicos de alta calidad, que influyan en la forma en que las personas diseñen, usen y manejen Internet.

IVR: La Respuesta Interactiva de Voz (IVR, *Interactive Voice Response*) es un sistema automatizado de respuesta interactiva, orientado a entregar y/o capturar información automatizada a través del teléfono.

RFC: Las Peticiones de Comentarios (RFC, *Request for Comments*) son un conjunto de normas que describen el funcionamiento interno de Internet, servicios de la red, protocolos y sus aplicaciones.

UIT: La UIT (Unión Internacional de Telecomunicaciones) es el organismo de la Organización de las Naciones Unidas (ONU) especializado en las tecnologías de la información y la comunicación dedicado a elaborar normas técnicas que garantizan la interconexión continua de las redes y las tecnologías.

ANEXOS

Anexo 1. Descripción de los requisitos funcionales

Tabla 6. Descripción del requisito funcional Crear servidor de telefonía IP.

	Conceptos	**Atributos**
Conceptos tratados	Servidor	
	Precondiciones	**Pre-requisitos**
Precondiciones	1. El usuario tiene que estar previamente autenticado en el sistema.	
Descripción	1. Se selecciona la opción **Adicionar.** 2. Se muestra la interfaz **Adicionar datos del servidor.** 3. Se introduce el **Nombre**, el **Alias**, la **Dirección IP** y el **Puerto** del servidor que se desea adicionar. 4. Se introduce el **Usuario** y la **Contraseña** para la conexión del servidor. 5. Se selecciona la opción **Cancelar** o **Guardar.** 5.1 Si se selecciona la opción **Guardar** se mostrará una notificación con el siguiente mensaje: "*El servidor ha sido adicionado correctamente*." 5.2 Si se selecciona la opción **Cancelar** desaparecerá esta ventana y se mostrará la interfaz principal nombrada **Gestionar Servidores.**	
Validaciones	1. Todos los campos son obligatorios. En caso de que se deje alguno en blanco se mostrará una notificación de error con el siguiente mensaje: "*Verifique que haya llenado todos los campos.*"	

	2. En caso que exista algún error en la entrada de los datos, el sistema mostrará el siguiente mensaje de información: "*Por favor verifique nuevamente que hay campo(s) con valor(es) incorrecto(s).*"
Post-condiciones	1. Se muestre en la interfaz de inicio el nuevo servidor insertado con todos sus datos correctamente.
Post-requisitos	1. Modificar datos del servidor. 2. Eliminar servidor del sistema.

Tabla 7. Descripción del requisito funcional Modificar servidor de telefonía IP.

	Conceptos	**Atributos**
Conceptos tratados	Servidor	
Precondiciones	**Precondiciones**	**Pre-requisitos**
	1. El usuario tiene que estar previamente autenticado en el sistema. 2. El usuario tiene que haber seleccionado previamente el servidor al cual le modificará los datos.	
Descripción	1. Se selecciona la opción **Modificar.** 2. Se muestra la interfaz **Modificar datos del servidor** mostrando los datos del servidor seleccionado. 3. Se borran los datos que se desean modificar y se escriben los datos nuevos. 4. Se selecciona la opción **Cancelar** o **Guardar.** 4.1 Si se selecciona la opción **Guardar** se mostrará una notificación con el siguiente mensaje: "*Los datos del servidor ha sido modificado correctamente*."	

	4.2 Si se selecciona la opción **Cancelar** desaparecerá esta ventana y se mostrará la interfaz principal nombrada **Gestionar Servidores.**
Validaciones	1. Todos los campos son obligatorios. En caso de que se deje alguno en blanco se mostrará una notificación de error con el siguiente mensaje: "*Verifique que haya llenado todos los campos.*" 2. En caso que exista algún error en la entrada de los datos, el sistema mostrará el siguiente mensaje de información: "*Por favor verifique nuevamente que hay campo(s) con valor(es) incorrecto(s).*"
Post-condiciones	1. Se muestre en la interfaz de inicio el nuevo servidor con los datos modificados.
Post-requisitos	1. Modificar datos del servidor. 2. Eliminar servidor del sistema.

Tabla 8. Descripción del requisito funcional Eliminar servidor de telefonía IP.

	Conceptos	**Atributos**
Conceptos tratados	Servidor	
	Precondiciones	**Pre-requisitos**
Precondiciones	1. El usuario tiene que estar previamente autenticado en el sistema. 2. El usuario tiene que haber seleccionado previamente el servidor que desea eliminar.	
Descripción	1. Se selecciona la opción **Eliminar**. 2. Se muestra la interfaz de confirmación de la acción.	

	3. Se selecciona la opción **Cancelar** o **Aceptar.** 3.1 Si se selecciona la opción **Aceptar** se mostrará una notificación con el siguiente mensaje: "*El servidor ha sido eliminado*." 3.2 Si se selecciona la opción **Cancelar** desaparecerá esta ventana y se mostrará la interfaz principal nombrada **Gestionar Servidores.**
Validaciones	1. El servidor debe existir en el sistema.
Post-condiciones	1. Se muestre en la interfaz de inicio los servidores existentes sin el que fue eliminado.
Post-requisitos	1. Adicionar datos de servidor. 2. Modificar datos de servidor.

Tabla 9. Descripción del requisito funcional Crear política de servidor de telefonía IP.

	Conceptos	**Atributos**
Conceptos tratados	Servidor, Política, Condición, Acción	
	Precondiciones	**Pre-requisitos**
Precondiciones	1. El usuario tiene que estar previamente autenticado en el sistema.	
Descripción	1. Se selecciona la opción **Adicionar Política.** 2. Se muestra la interfaz **Adicionar datos de Política.** 3. Se introduce el **Nombre** y la **Descripción** con que se va identificar la política. 4. Se selecciona el grupo de servidores o el/los servidores específicos donde se va ejecutar la política.	

	5. Se conforma la o las condiciones de la política a través de la selección de los recursos a controlar, el operador a medir y el valor de la cantidad a comparar con el valor inmediato del recurso. 6. Luego se escoge la operación o las operaciones a realizar una vez cumplida la condición que pueden ser Enviar correo o Ejecutar comando. 6.1 Si se selecciona la opción **Enviar correo** se mostrará una interfaz que debe proporcionar la información necesaria para escoger el **Grupo de usuarios** o los **usuarios** específicos a los cuales se le notificará una vez escogido el tipo de notificación que puede ser **Correo, Jabber o SMS**. 6.2 Si se selecciona la opción **Ejecutar comando** se mostrará la interfaz requerida para la entrada del **comando** a ejecutar además de destino en el cual va a ser ejecutados **Servidor** o **Agente**. 7. Se selecciona la opción **Cancelar** o **Guardar.** 7.1 Si se selecciona la opción **Guardar** se mostrará una notificación con el siguiente mensaje: "*La política ha sido adicionado correctamente*." 7.2 Si se selecciona la opción **Cancelar** desaparecerá esta ventana y se mostrará la interfaz principal nombrada **Gestionar Servidores.**
Validaciones	1. Todos los campos son obligatorios. En caso de que se deje alguno en blanco se mostrará una notificación de error con el siguiente mensaje: "*Verifique que haya llenado todos los campos.*" 2. En caso que exista algún error en la entrada de los datos, el sistema mostrará el siguiente mensaje de información: "*Por favor verifique nuevamente que hay campo(s) con valor(es) incorrecto(s).*"
Post-condiciones	1. Se muestra en la interfaz de inicio con la nueva política insertada con todos sus datos correctamente.
Post-requisitos	1. Modificar datos de política.

	2. Eliminar política del sistema.

Tabla 10. Descripción del requisito funcional Modificar política de servidor de telefonía IP.

Conceptos tratados	**Conceptos**	**Atributos**
	Servidor, Política, Condición, Acción	
Precondiciones	**Precondiciones**	**Pre-requisitos**
	1. El usuario tiene que estar previamente autenticado en el sistema. 2. El usuario tiene que haber seleccionado previamente la política al cual se le modificará los datos.	
Descripción	4. Se selecciona la opción **Modificar política de servidor.** 2. Se muestra la interfaz **Modificar política** mostrando los datos de la política seleccionada. 3. Se borran los datos que se desean modificar y se escriben los datos nuevos. 4. Se selecciona la opción **Cancelar** o **Guardar.** 4.1 Si se selecciona la opción **Guardar** se mostrará una notificación con el siguiente mensaje: "*Los datos de la política han sido modificados correctamente*." 4.2 Si se selecciona la opción **Cancelar** desaparecerá esta ventana y se mostrará la interfaz principal nombrada **Gestionar Servidores.**	
Validaciones	1. Todos los campos son obligatorios. En caso de que se deje alguno en blanco se mostrará una notificación de error con el	

	siguiente mensaje: "*Verifique que haya llenado todos los campos.*" 2. En caso que exista algún error en la entrada de los datos, el sistema mostrará el siguiente mensaje de información: "*Por favor verifique nuevamente que hay campo(s) con valor(es) incorrecto(s).*"
Post-condiciones	1. Se muestra en la interfaz de inicio la política modificada con sus datos nuevos.
Post-requisitos	1. Modificar datos de política. 2. Eliminar política del sistema.

Tabla 11. Descripción del requisito funcional Eliminar política de servidor de telefonía IP.

Conceptos tratados	**Conceptos**	**Atributos**
	Servidor, Política, Condición, Acción	
Precondiciones	**Precondiciones**	**Pre-requisitos**
	1. El usuario tiene que estar previamente autenticado en el sistema. 2. El usuario tiene que haber seleccionado previamente la política que desea eliminar.	
Descripción	1. Se selecciona la opción **Eliminar política.** 2. Se muestra la interfaz de confirmación de la acción. 3. Se selecciona la opción **Cancelar** o **Aceptar.** 3.1 Si se selecciona la opción **Aceptar** se mostrará una notificación con el siguiente mensaje: "*La política ha sido eliminada*."	

	3.2 Si se selecciona la opción **Cancelar** desaparecerá esta ventana y se mostrará la interfaz principal nombrada **Gestionar Servidores.**
Validaciones	1. La política debe existir en el sistema.
Post-condiciones	1. Se muestra en la interfaz de inicio los servidores existentes sin los datos de la política que fue eliminada.
Post-requisitos	1. Adicionar datos de política. 2. Modificar datos de política.

Tabla 12. Descripción del requisito funcional.

Conceptos tratados	**Conceptos**	**Atributos**
	Servidor	
Precondiciones	**Precondiciones**	**Pre-requisitos**
	1. El usuario tiene que estar previamente autenticado en el sistema. 2. El usuario tiene que haber seleccionado previamente el servidor que desea ver la información actual.	Adicionar Servidor
Descripción	1. Se selecciona la opción **Estado.** 2. Se muestra la interfaz de Mostrar Estado de Servidor donde se despliega la información acerca del porciento de uso de la **RAM,** el porciento de uso de la **CPU,** el tráfico de la **tarjeta de red,** el porciento de uso del **disco duro,** el comportamiento de la **perdida de paquete,** el comportamiento de la **latencia**, la cantidad de **llamadas concurrentes** y la cantidad de **llamadas fallidas**. 3. Se selecciona la opción **Cerrar** y se desaparecerá esta ventana y se mostrará la interfaz principal nombrada **Gestionar Servidores.**	
Validaciones	1. El servidor debe existir en el sistema.	

Post-condiciones	1. El sistema no debe sufrir ningún cambio.
Post-requisitos	1. Adicionar servidor. 2. Modificar Servidor 3. Eliminar Servidor.

Tabla 13. Descripción del requisito funcional Generar alertas.

Conceptos tratados	**Conceptos**	**Atributos**
	Servidor, Política, Condición, Alerta	
Precondiciones	**Precondiciones**	**Pre-requisitos**
	1. El usuario tiene que estar previamente autenticado en el sistema. 2. El sistema debe tener políticas para controlar, y dichas políticas deben ser válidas.	
Descripción	1. El sistema debe ser capaz de reconocer cuando se cumple alguna condición de cualquier política en determinado servidor y así lanzar la alerta al usuario en se instante de tiempo. 2. Se muestra la interfaz de alerta en la esquina superior izquierda de la ventana.	
Validaciones	1. La política debe existir en el sistema. 2. La condición de la política debe cumplirse.	
Post-condiciones	1. Se actualiza el registro de alertas.	
Post-requisitos		

Tabla 14. Descripción del requisito funcional Generar reporte.

Conceptos tratados	**Conceptos**	**Atributos**
	Reportes, Servidor	
Precondiciones	**Precondiciones**	**Pre-requisitos**
	1. El usuario tiene que estar previamente autenticado en el sistema. 2. EL usuario debe seleccionar previamente el servidor del cual desea conocer su información registrada en el tiempo.	
Descripción	1. Seleccionar Ver Registro. 2. Se muestra la interfaz de obtenida desde la vista de la interfaz Zabbix. 3. Se escoge el recurso y el intervalo de tiempo en el cual se visualizará la información de dicho recurso en el servidor determinado.	
Validaciones	1. El servidor debe existir en el sistema. 2. El servidor debe tener registro almacenado.	
Post-condiciones		
Post-requisitos		

Anexo 2. Prototipos de interfaz de usuario

Figura 19. PIU Autenticación.

Figura 20. PIU Adicionar Servidor.

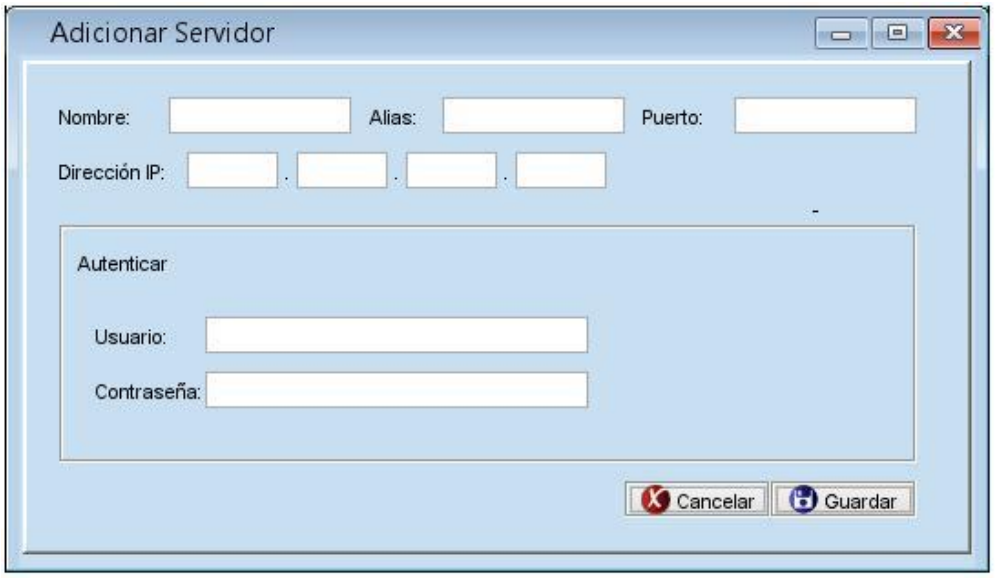


Figura 21. PIU Configurar Almacenamiento del Servidor.

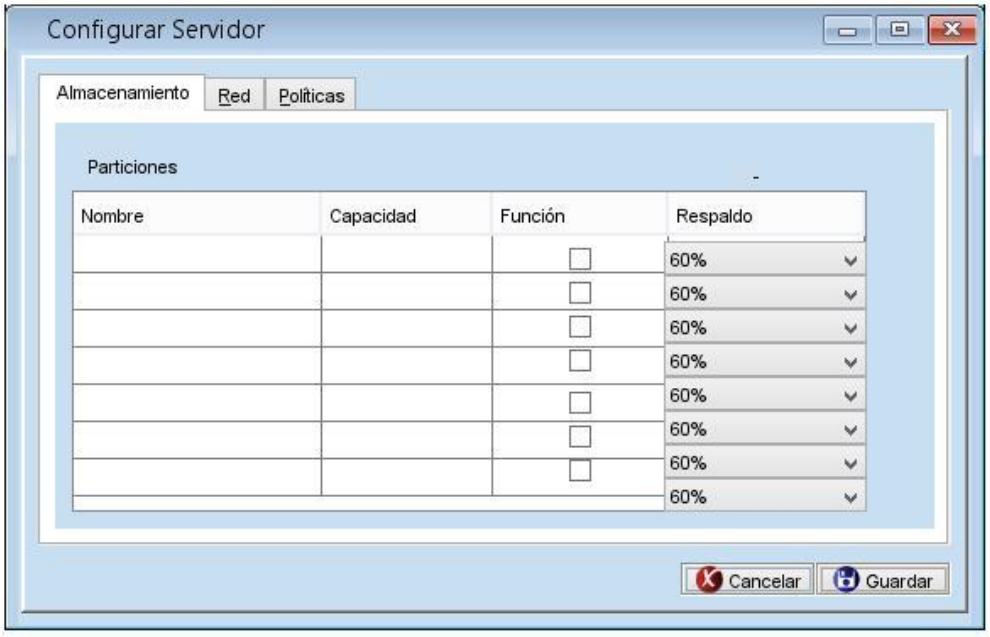


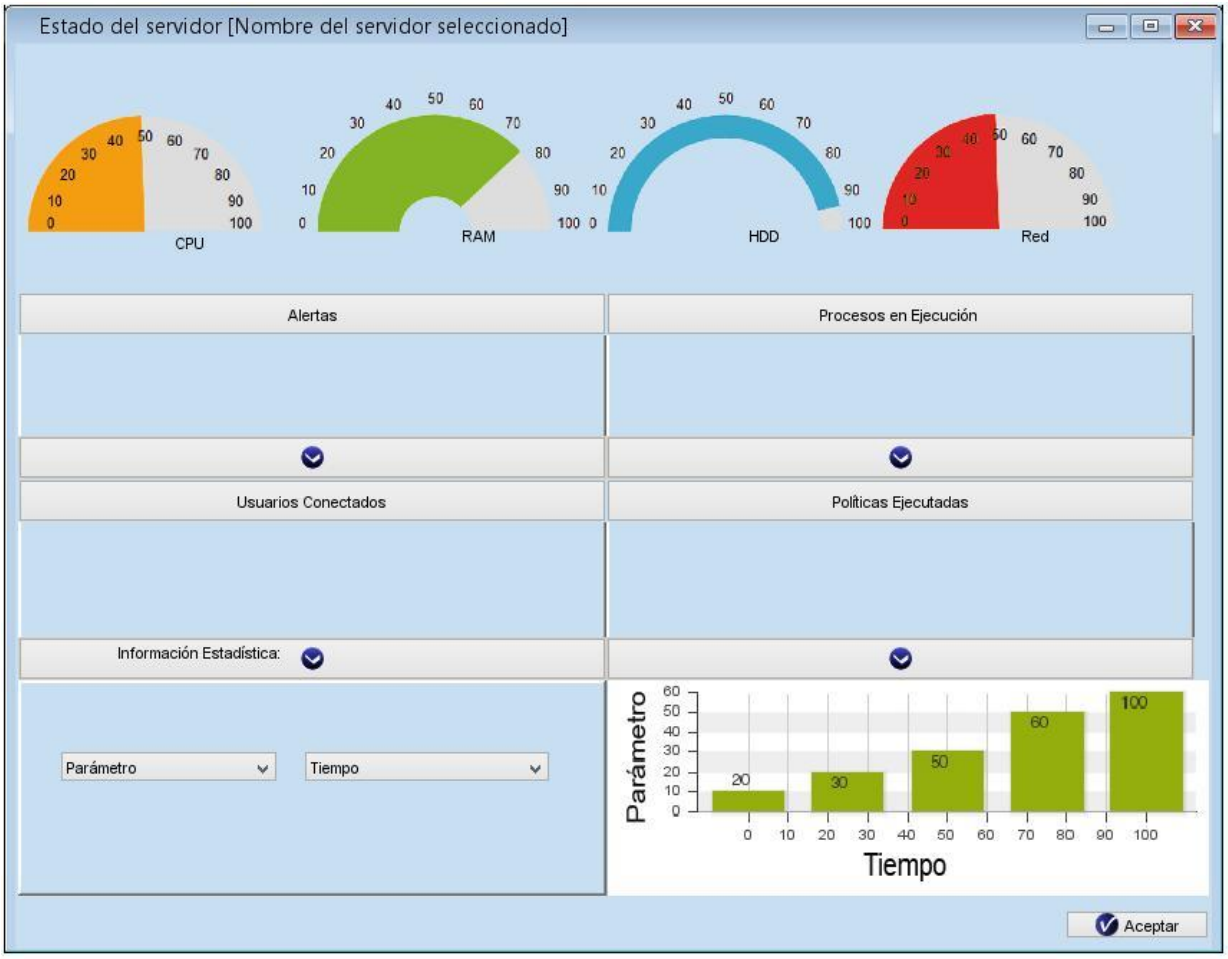
Figura 22. PIU Configurar Red del Servidor.

Figura 23. PIU Configurar Políticas del Servidor.

Figura 24. PIU Configurar Precondición de la Política.

Figura 25. PIU Configurar Poscondición de la Política.

Figura 26. PIU Monitorización del Estado del Servidor.



Anexo 3. Descripción del Diagrama Entidad-Relación

Tabla item: Guarda los datos de los diferentes recursos de los servidores.

Tabla item			
Atributos	**Tipo de Datos**	**Nulo**	**Definición**
Itemid (llave)	int4	no	Identificador del recurso.
delay	int4	no	Cantidad de tiempo en segundos para actualizar el valor del recurso.
hostid	int4	no	Identificación del servidor al cual pertenece el recurso.
key	varchar(255)	no	Llave del recurso del servidor.
name	varchar(255)	no	Nombre del Recurso.
type	int4	no	Utilizado para identificar el tipo de ítem como puede ser: 0 - Zabbix agent; 1 - SNMPv1 agent; 2 - Zabbix trapper; 3 - simple check; 4 - SNMPv2 agent; 5 - Zabbix internal; 6 - SNMPv3 agent; 7 - Zabbix agent (active); 8 - Zabbix aggregate; 9 - web item; 10 - external check; 11 - database monitor; 12 - IPMI agent; 13 - SSH agent; 14 - TELNET agent;

			15 - calculated; 16 - JMX agent; 17 - SNMP trap.
value_type	int4	no	Tipo de información que porta el recurso. Puede tener los siguientes valores: 0 - Numérico float; 1 -Caracter; 2 - log; 3 - Numérico (sin asignar); 4 - Texto.
data_type	integer(10)	si	Utilizado para indicar el tipo de dato del recurso. Ejemplo: 0 – *(Por defecto)* Decimal; 1 - octal; 2 - hexadecimal; 3 - boolean.
delay_flex	varchar(255)	si	Intervalos flexibles, cada serie de intervalo consiste en un período de tiempo para actualizar el valor del recurso.
delta	int4	si	Valor del ítem que será almacenado en el instante de tiempo.
description	varchar(255)		Descripción del recurso de servidor.
error	varchar(128)	si	Texto que indica si hay problemas con la actualización del valor del recurso.
flags	int4	si	Origen del recurso.
formula	varchar(255)	si	Operador para Multiplicar. Por defecto su valor es 1.
history	int4	si	Número de días para mantener la información histórica del recurso en el sistema.

inventory_link	int4	si	Identificador del inventario del servidor que es poblado por el recurso. Valor por defecto:90
ipmi_sensor	varchar(255)	si	Sensor IPMI usado solo por recursos IPMI.
lastclock	timestamp	si	Fecha de la última actualización del valor del recurso.
lastns	int4	si	Nanosegundo desde la última actualización del valor del recurso.
lastvalue	varchar(255)	si	Valor del recurso obtenido desde la última actualización.
logtimefmt	varchar(255)	si	Formato usado para almacenar la fecha en los log. Usado solo por los recursos de tipo log.
mtime	timestamp	si	Fecha de la última actualización del log de monitoreo. Usado solo por los recursos de tipo log.
multiplier	int4	si	Usar un Multiplicador personalizado.
params	varchar(255)	si	Cadena de parámetros adicionales dependiendo del tipo de recurso.
password	varchar(255)	si	Contraseña para autenticación.
port	varchar(255)	Si	Puerto monitorizado por el recurso. Usado solo para recursos SNMP.
prevorgvalue	Float8	Si	Valor obtenido en la penúltima actualización del valor del recurso.
snmpv3_authpassphrase	varchar(255)	si	Frase de autenticación para SNMP versión 3.

snmpv3_authprotocol	integer(10)	si	Protocolo de autenticación para SNMP versión 3 como ejemplo: 0 - *(default)* MD5; 1 - SHA.
snmpv3_contextname	varchar(255)	si	Nombre de contexto para SNMP versión 3.
snmpv3_privpassphrase	varchar(255)	Si	Frase privada para SNMP versión 3.
snmpv3_privprotocol	integer(10)	si	Protocolo de privacidad para SNMP versión 3.
snmpv3_securitylevel	int4	si	Nivel de seguridad para SNMP versión 3
snmpv3_securityname	varchar(255)	si	Nombre de seguridad para SNMP versión 3.
status	int4	Si	Estado del recurso.
templateid	int4	si	Identificador del recurso de plantilla.
trapper_hosts	varchar(255)	si	Identificador del host.
trends	int4	si	Número de días para mantener la información histórica de las trends.
units	varchar(10)	si	Valor.
valuemapid	int4	si	Identificador para graficar.

Tabla 15. DBD de la tabla items

Tabla host: Guarda los datos de los diferentes servidores.

Tabla host			
Atributos	**Tipo de Datos**	**Nulo**	**Definición**
hostid (llave)	int4	No	Identificación del servidor.
host	varchar(255)	No	Nombre técnico del servidor o el nombre más identificativo para usar comúnmente.
available	int4	Si	Habilitado solo para agentes Zabbix.
description	varchar(255)	Si	Descripción del servidor.
disable_until	int4	Si	La próxima fecha de sondeo del agente Zabbix.
error	Varchar(128)	Si	Texto de error si el agente Zabbix está inhabilitado.
errors_from	int4	Si	Tiempo donde el agente Zabbix se vuelve inhabilitado.
flags	int4	Si	Orígenes del servidor.
ipmi_authtype	int4	Si	Algoritmo de autenticación IPMI.
ipmi_available	int4	Si	Indica si está habilitado el agente IPMI.
ipmi_disable_until	timestamp	Si	La próxima fecha de sondeo del agentes IPMI deshabilitadas.

ipmi_error	varchar(255)	Si	Mensaje de error si el Agente IPMI esta deshabilitado.
ipmi_errors_from	timestamp	Si	Tiempo donde el agente IPMI se vuelve inhabilitado.
ipmi_password	varchar(255)	Si	Contraseña para Agentes IPMI.
ipmi_privilege	int4	Si	Indica el nivel de privilegio para agentes IPMI.
ipmi_username	varchar(255)	Si	Nombre de usuario para agentes IPMI.
jmx_available	int4	Si	Indica si está habilitado el agente JMX.
jmx_disable_until	timestamp	Si	La próxima fecha de sondeo del agentes JMX deshabilitadas.
jmx_error	varchar(255)	Si	Mensaje de error si el Agente JMX esta deshabilitado.
name	varchar(255)	Si	Nombre visible del servidor.
proxy_hostid	varchar(255)	Si	Identificación del proxy que es usado para monitorizar el servidor.
snmp_available	int4	Si	Indica si está habilitado el agente SNMP.
snmp_disable_until	timestamp	Si	La próxima fecha de sondeo del agentes SNMP deshabilitados.
snmp_error	varchar(255)	Si	Mensaje de error si el Agente SNMP esta deshabilitado.

snmp_errors_from	timestamp	Si	Tiempo donde el agente SNMP se vuelve inhabilitado.
status	int4	Si	Estado y función del servidor.

Tabla 16. DBD de la Tabla host

Tabla trigger: Guarda los datos de las diferentes condiciones de políticas a ejecutar sobre el servidor.

Tabla trigger			
Atributos	**Tipo de Datos**	**Nulo**	**Definición**
triggerid (llave)	Int 8	No	Identificación de la condición.
description	varchar(255)	No	Nombre de la condición.
expression	varchar(255)	No	Expresión reducida de la condición.
comments	text	Si	Comentarios adicionales de la condición.
error	varchar(255)	Si	Mensaje de error si hay problemas cuando se actualiza el estado.
flags	Int 4	Si	Origen de la condición.
lastchange	Int 4	Si	Fecha de cuando la última vez que la condición cambio el estado.
priority	Int 4	Si	Gravedad de la condición.
state	Int 4	Si	Estado de la condición.

status	Int 4	Si	Informa si la condición está activa o inactiva.
templateid	Int 8	Si	Identificación de la plantilla.
type	Int 4	Si	Si la condición puede generar múltiples eventos de problema.
url	varchar(255)	Si	URL asociada con la condición.
value	integer(10)	Si	Indica si la condición está en buen estado o en estado de problema.

Tabla 17. DBD de la Tabla trigger

Tabla action: Guarda los datos de las diferentes acciones a realizar después de las condiciones de las políticas a ejecutar sobre el servidor.

Tabla action			
Atributos	**Tipo de Datos**	**Nulo**	**Definición**
actionid (llave)	char	No	Identificación de la acción.
esc_period	Int4	No	Duración de las operaciones por defecto. Debe ser más de 60 segundos.
eventsource	Int4	No	Tipo de eventos que la acción deberá manejar.
name	varchar(255)	No	Nombre de la acción.
def_longdata	varchar(255)	Si	Texto Mensaje de error.
def_shortdata	varchar(255)	Si	Sujeto del mensaje de error.

r_longdata	varchar(255)	Si	Texto de Mensaje de recuperación.
r_shortdata	varchar(255)	Si	Sujeto de Mensaje de recuperación.
recovery_msg	Int4	Si	Indica si el mensaje de recuperación está habilitado.
status	Int4	Si	Indica si la acción está activa o inactiva.

Tabla 18. DBD de la Tabla action

Tabla política: Guarda los datos de las políticas.

Tabla política			
Atributos	**Tipo de Datos**	**Nulo**	**Definición**
id_politica (llave)	Int4	No	Identificación de la de la política.
nombre	varchar(255)	No	Nombre identificativo que se le otorga a la política.
descripción	varchar(255)	Sí	Información adicional que se realiza en caso de especificar algo.

Tabla 19. DBD de la Tabla política

Anexo 4. Prototipos de interfaz de usuario funcional

Figura 27. PIUF Gestionar Servidores.

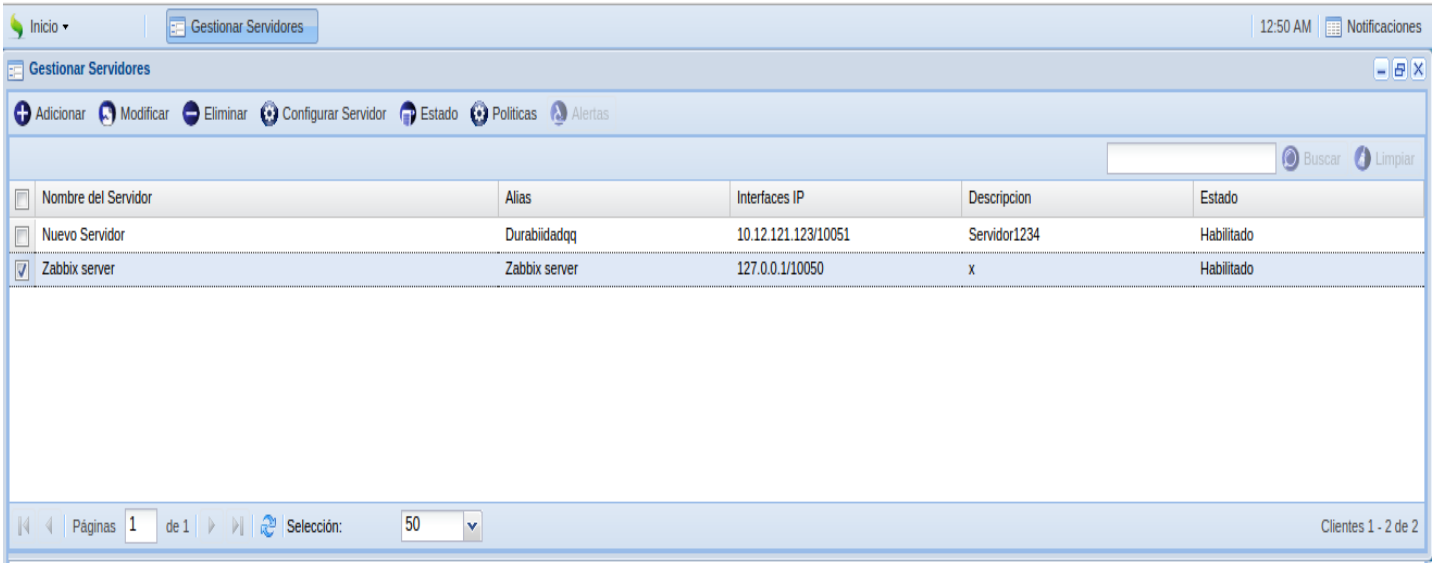


Figura 28. PIUF Adicionar datos de Políticas General.

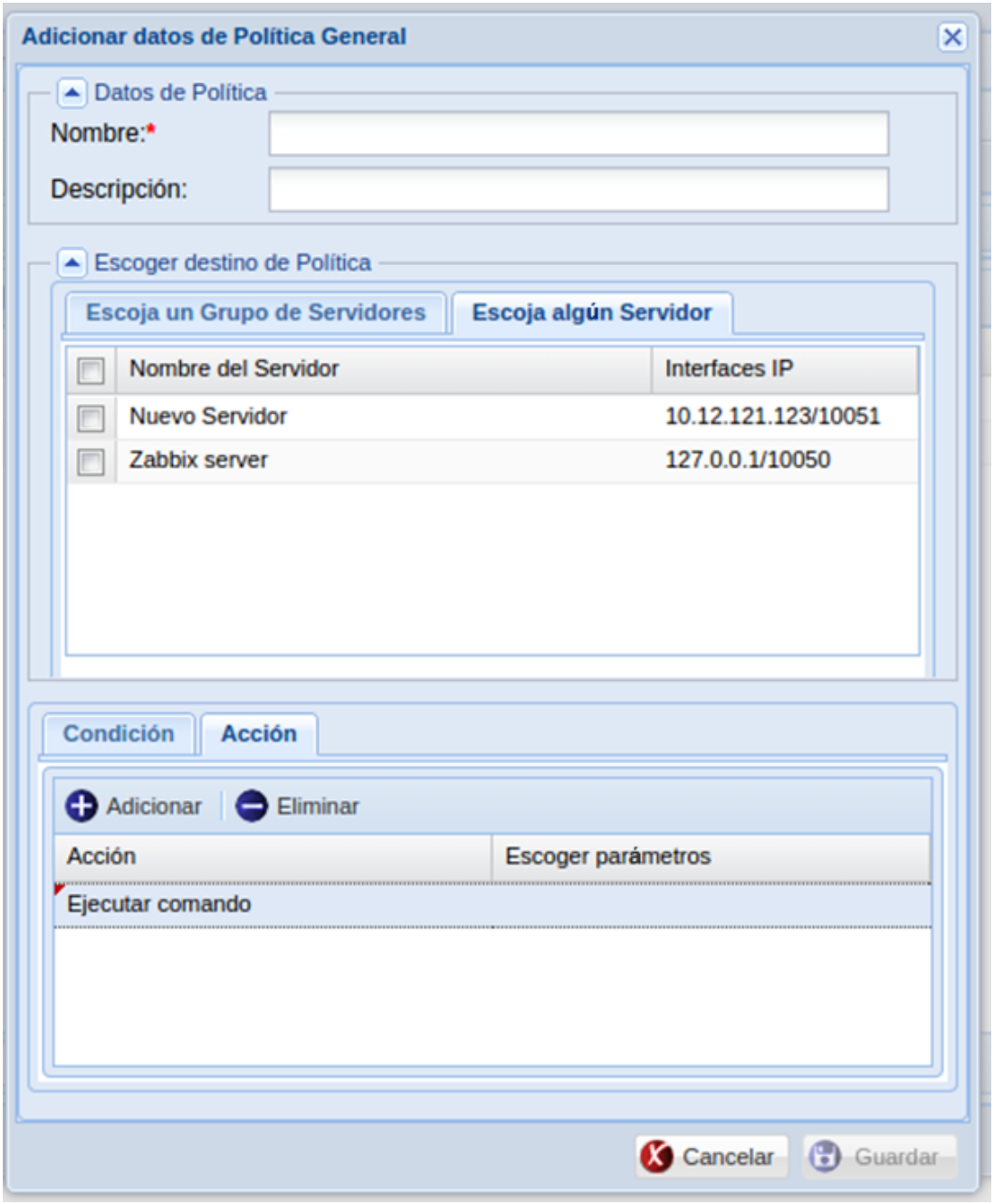


Figura 29. PIUF Configurar Servidor.

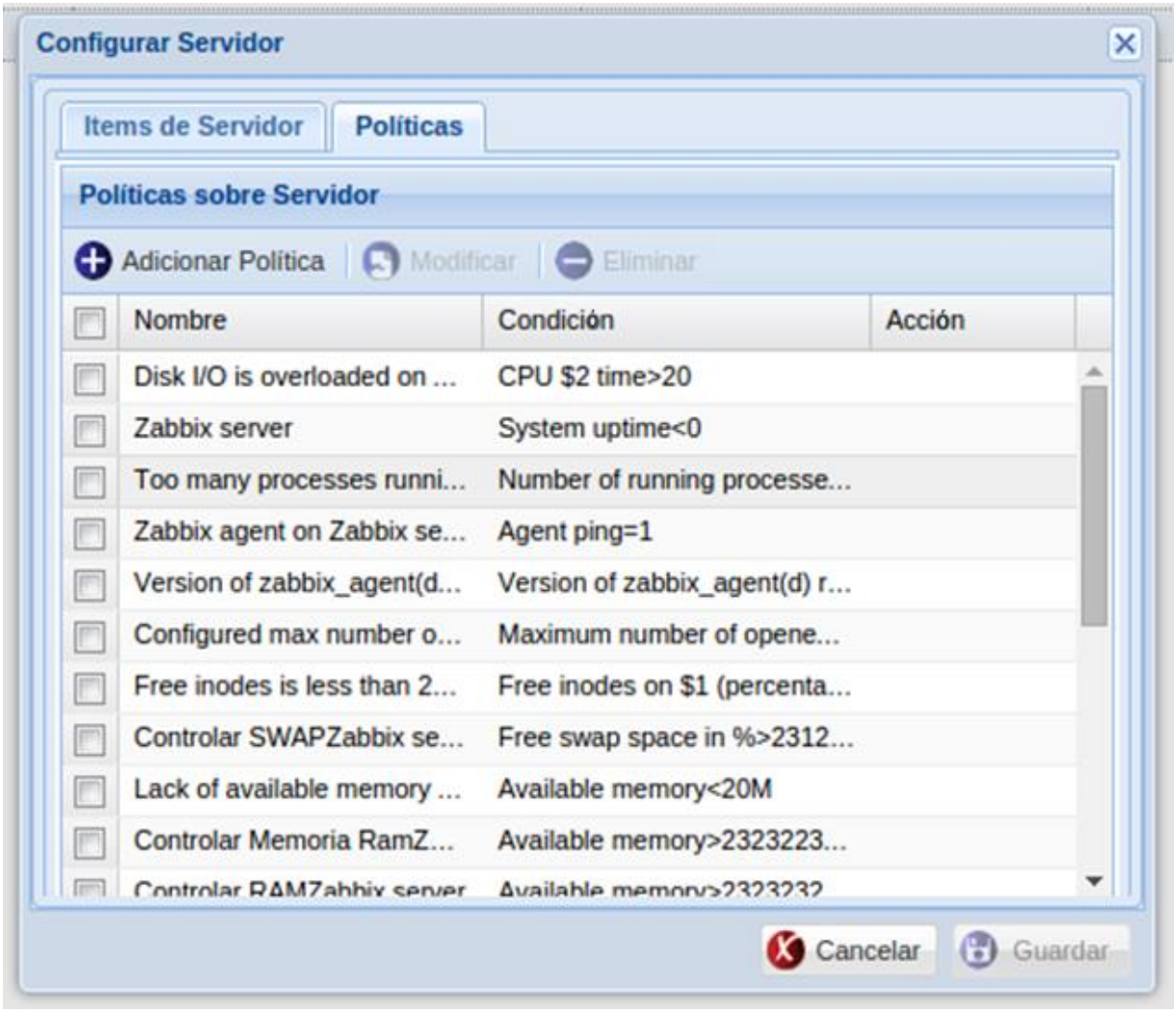


Figura 30. PIUF Enviar Mensaje.

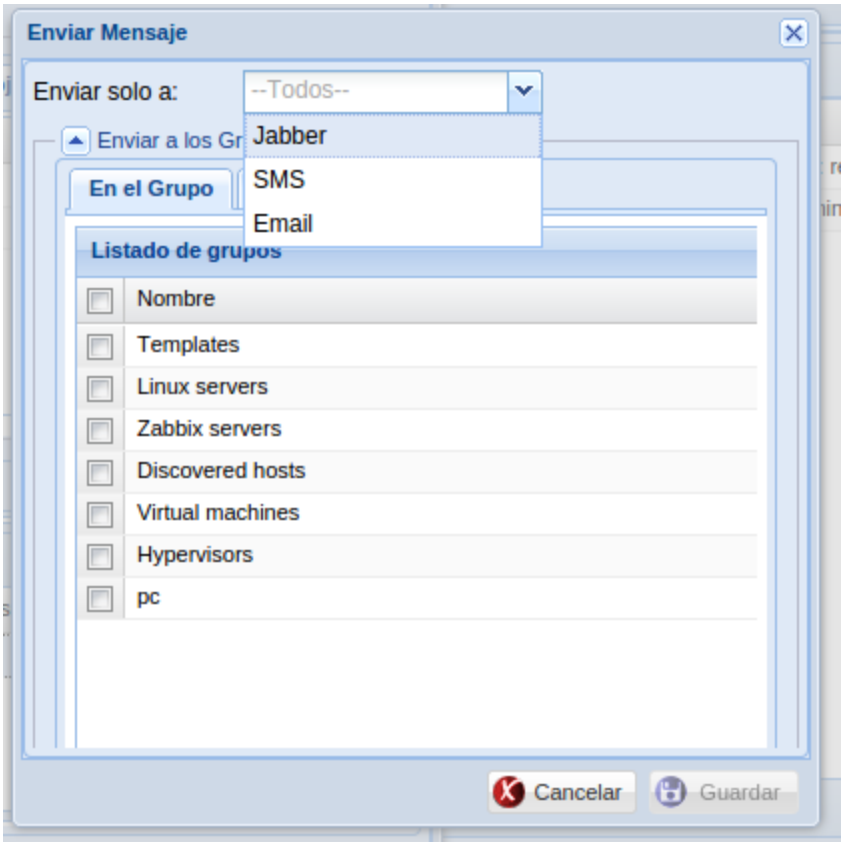
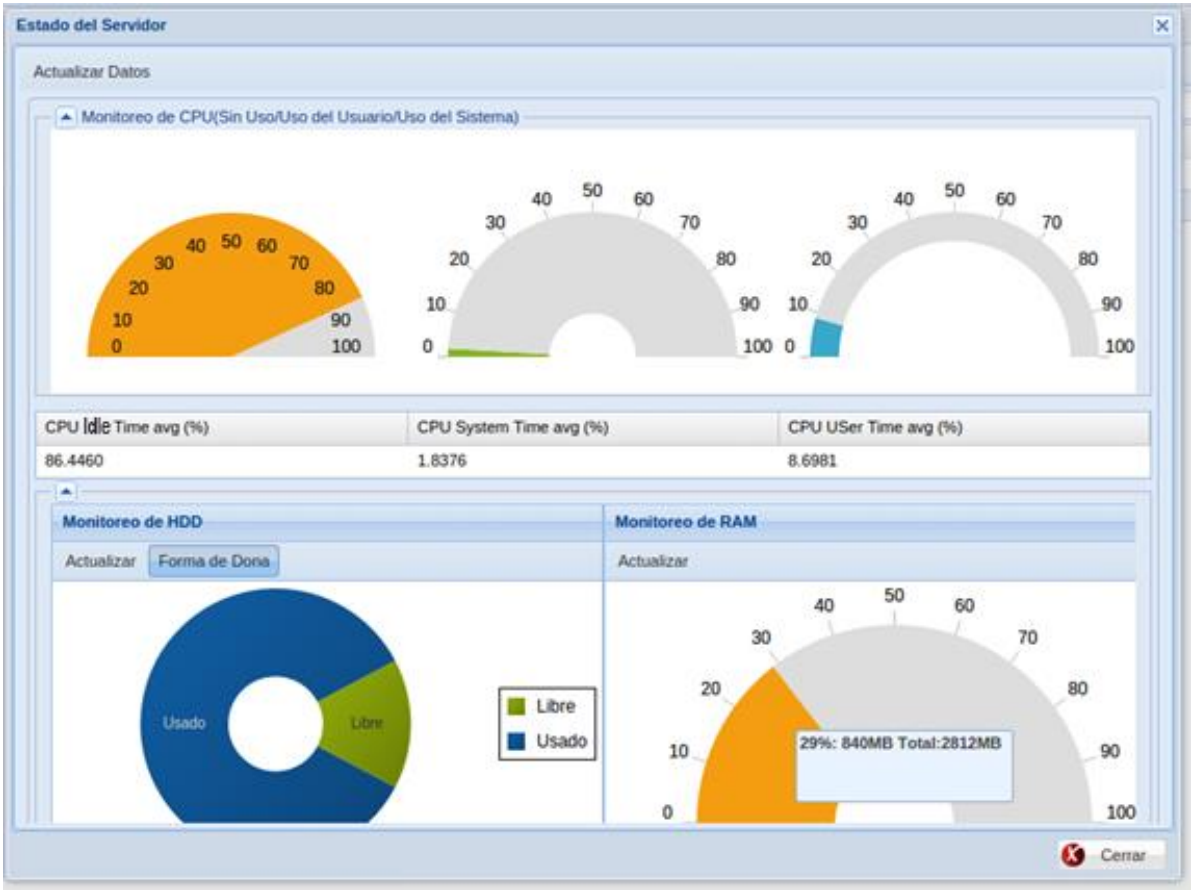


Figura 31. PIUF Estado del Servidor.



Anexo 5. Diseño de los casos de prueba de caja blanca

- Casos de prueba del método: **modificar_servidor(request)**
- Variables a considerar en el caso de prueba**: servidor**
- Clase de Equivalencia para la variable: **servidor**

Clase de Equivalencia	Clase de Equivalencia	Clasificación de las Clase de Equivalencia
1	Verificar que se modifique los datos del servidor.	Válida.

*Tabla 20. Clase de Equivalencia para la variable **servidor***

Caso de Prueba:

1- modificar _servidor(request)

Caso de Prueba	Objetivo de la Prueba	Datos de Entrada	Salida Esperada	Salida Obtenida
1	Verificar que pasado un id de un servidor y los datos a modificar se modifique la información del mismo en la base de datos.	request pasado por parámetro .	Modificación exitosa.	Modificación exitosa.

Tabla 21. Diseño de Caso de Prueba para el método Modificar Servidor

- Casos de prueba del método: **eliminar_servidor(request)**
- Variables a considerar en el caso de prueba**: id_servidor**
- Clase de Equivalencia para la variable: **id_servidor**

Clase de Equivalencia	Clase de Equivalencia	Clasificación de las Clase de Equivalencia
1	Verificar que se eliminen los datos del servidor.	Válida.

*Tabla 22. Clase de Equivalencia para la variable **id_servidor***

Caso de Prueba:

eliminar _servidor(request)

Caso de Prueba	Objetivo de la Prueba	Datos de Entrada	Salida Esperada	Salida Obtenida
1	Verificar que pasado el identificador de un servidor se elimine el mismo de la base de datos.	request pasado por parámetro.	Eliminación exitosa.	Eliminación exitosa.

Tabla 23. Diseño de Caso de Prueba para el método Eliminar Servidor

Anexo 6. Diseño de los casos de prueba de caja negra

Tabla 24. Diseño de Caso de Prueba para la funcionalidad Modificar servidores de Telefonía IP.

Escenario	Descripción	Nombre del Servidor	Nombre Visible	Descripción del Servidor	Seleccionar Grupo de Servidores	Seleccionar Plantilla	Agregar Interfaces	Respuesta del Sistema
Modificar datos del Servidor.	Se introducen los datos del servidor.	V	V	V	V	V	V	Se adiciona el servidor a la lista de servidores mostrada en la interfaz principal.
		Letras	Letras	Letras	Seleccionar al menos uno.	Puede o no estar seleccionada alguna plantilla.	Adicionar dirección ip #1<255. #2<255. #3<255. #4<254	
		V	NA	V	V	V	V	El módulo no permite que en el campo entren esos datos.
		Letras	Vacío	Letras	Seleccionar al menos uno.	Puede o no estar seleccionada alguna plantilla.	Adicionar dirección ip #1<255. #2<255. #3<255. #4<254	
		NA	V	V	V	V	V	El módulo no permite que se active el botón de guardar cambios.
		Vacío	Letras	Letras	Seleccionar al menos uno.	Puede o no estar seleccionada alguna plantilla.	Adicionar dirección ip #1<255. #2<255. #3<255. #4<254	
		NA	NA	V	NA	V	NA	
		Vacío	Vacío	Letras	Sin seleccionar.	Puede o no estar seleccionada alguna plantilla.	Vacío	
		V	V	V	NA	V	V	

		Letras	Letras	Letras	Sin seleccionar.	Puede o no estar seleccionada alguna plantilla.	Adicionar dirección ip #1<255. #2<255. #3<255. #4<254	
		V	V	V	V	V	NA	
		Letras	Letras	Letras	Seleccionar al menos uno.	Puede o no estar seleccionada alguna plantilla.	Vacío	
		V	V	V	V	V	NA	
		Letras	Letras	Letras	Seleccionar al menos uno.	Puede o no estar seleccionada alguna plantilla.	Letras	
		NA	NA	V	V	V	NA	El módulo no permite que en el campo entren esos datos.
		Caracteres extraños.	Caracteres extraños.	Letras	Seleccionar al menos uno.	Puede o no estar seleccionada alguna plantilla.	Letras	

Tabla 25. Diseño de Caso de Prueba para la funcionalidad Eliminar servidores de Telefonía IP.

Escenario	**Descripción**	***Checkbox***	**Respuesta del sistema**
Eliminar Servidor.	Se elimina un servidor seleccionado.	V	El módulo muestra un mensaje diciendo: ¿Está seguro que desea eliminar el servidor?
		Seleccionado	
		NA	El módulo no activará el botón de eliminar.
		Ninguno seleccionado	

Tabla 26. Diseño de Caso de Prueba para la funcionalidad Adicionar política.

Escenario	**Descripción**	**Nombre de política**	**Descripción de política**	**Seleccionar Grupo de Servidores**	**Seleccionar Servidor**	**Agregar Condición**	**Agregar Acción**	**Respuesta del Sistema**
Adicionar política.	Se introducen los datos de la política.	V	V	V	V	V	V	Se adiciona la política a la lista de política.
		Letras	Letras	Puede o no estar seleccionada algún grupo.	Puede o no estar seleccionada algún Servidor.	Seleccionar Recurso Operador Cantidad.	Seleccionar Acción Ejecutar comando Enviar mensaje.	
		NA	NA	V	V	V	**V**	El módulo no permite que en el campo entren esos datos.
		Caracteres extraños.	Caracteres extraños.	Puede o no estar seleccionada algún grupo.	Puede o no estar seleccionada alguna plantilla.	Seleccionar Recurso Operador Cantidad.	Seleccionar Acción Ejecutar comando Enviar mensaje.	
		NA	V	V	V	NA	NA	El módulo no permite que se active el botón de guardar.
		Vacío	Letras	Puede o no estar seleccionada algún grupo.	Puede o no estar seleccionada alguna plantilla.	Vacío	Vacío	
		NA	NA	NA	NA	NA	NA	
		Vacío	Vacío	Vacío	Vacío	Vacío	Vacío	
		V	V	V	V	NA	NA	
		Letras	Letras	Puede o no estar seleccionada algún grupo.	Puede o no estar seleccionada alguna plantilla.	Vacío	Vacío	

		V	V	NA	NA	V	V	
		Letras	Letras	Vacío	Vacío	Seleccionar Recurso Operador Cantidad.	Seleccionar Acción Ejecutar comando Enviar mensaje.	

Tabla 27. Diseño de Caso de Prueba para la funcionalidad Modificar política.

Escenario	**Descripción**	**Nombre de política**	**Descripción de política**	**Seleccionar Grupo de Servidores**	**Seleccionar Servidor**	**Agregar Condición**	**Agregar Acción**	**Respuesta del Sistema**
Modificar política.	Se introducen los datos de la política que se desea modificar.	V	V	V	V	V	V	Se adiciona la política a la lista de política.
		Letras	Letras	Puede o no estar seleccionado algún grupo.	Puede o no estar seleccionada algún Servidor.	Seleccionar Recurso Operador Cantidad.	Seleccionar Acción Ejecutar comando Enviar mensaje.	
		NA	NA	NA	NA	V	**V**	El módulo no permite que en el campo entren esos datos.
		Caracteres extraños.	Caracteres extraños.	Vacío	Vacío	Seleccionar Recurso Operador Cantidad.	Seleccionar Acción Ejecutar comando Enviar mensaje.	
		NA	V	V	V	NA	NA	El módulo no permite que se active el
		Vacío	Letras	Puede o no estar.	Puede o no estar seleccionada	Vacío	Vacío	

				seleccionado algún grupo.	alguna plantilla.			botón de guardar.
		NA	NA	NA	NA	NA	NA	
		Vacío	Vacío	Vacío	Vacío	Vacío	Vacío	
		V	V	V	V	NA	NA	
		Letras	Letras	Puede o no estar seleccionado algún grupo.	Puede o no estar seleccionada alguna plantilla.	Vacío	Vacío	
		V	V	NA	NA	V	V	
		Letras	Letras	Vacío	Vacío	Selecciona r Recurso Operador Cantidad.	Seleccionar Acción Ejecutar comando Enviar mensaje.	

Tabla 28. Diseño de Caso de Prueba para la funcionalidad Eliminar Política.

Escenario	**Descripción**	***Checkbox***	**Respuesta del sistema**
Eliminar Política.	Se elimina una política seleccionada.	V	El módulo muestra un mensaje diciendo: ¿Está seguro que desea eliminar la política?
		seleccionada	
		NA	El módulo no activará el botón de eliminar.
		Ninguna seleccionada	