PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

FACULTAD DE INGENIERÍA

INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

IMPLEMENTACIÓN DE UN AMBIENTE DE MONITOREO DE LA DISPONIBILIDAD Y RENDIMIENTO DE LOS COMPONENTES DE LA INFRAESTRUCTURA TECNOLÓGICA CON NOTIFICACIONES AUTOMÁTICAS DE EVENTOS E INCIDENTES. CASO DE ESTUDIO SECURE SOFT.

DIEGO ESTEBAN DEL SALTO MALDONADO

TRABAJO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO DE SISTEMAS Y COMPUTACIÓN

Quito, diciembre 2019

Índice de Contenidos

Tabla de Figuras 6

1.Introducción 1

1.1 Justificación 1

1.1.1 La importancia de los datos 1

1.2 Planteamiento del Problema 2

1.3 Objetivos 3

1.3.1 Objetivo General 3

1.3.2 Objetivos Específicos 3

1.4 Alcance 4

1.5 Limitaciones 4

Capítulo 2: Marco Teórico 6

2.1 Fundamentos Teóricos 6

2.1.1 SNMP 6

2.1.3 Servidor de Correo 8

2.1.4 Bot 11

2.1.5 A.P.I. 13

2.1.6 Linux 14

2.1.7 Open Source 16

2.2 Herramientas 17

2.2.1 Zabbix 17

2.2.2 Zimbra 21

2.2.3 Telegram Bot API 22

2.2.4 Python 24

2.3 Automatización de Procesos.....31

2.3.1 Notificación de Eventos.....31

2.3.2 Escalamiento de Incidentes 31

Capítulo 3: Especificación de Requerimientos 33

3.1 Introducción 33

3.2 Análisis de Requerimientos 34

3.2.1 Requisitos de Hardware/Requisitos Físicos 34

3.2.2 Requisitos propios de la organización 35

3.3.3 Enfoque del proyecto.....36

3.3 Requisitos Técnicos 38

3.3.1 Zabbix 38

3.3.2 Zimbra 40

3.3.3 Bot de Telegram.....41

Capítulo 4: Implementación.....43

4.1 Metodología elegida y aclaraciones 43

4.2 Aplicación de Cascada con Zabbix. 45

4.2.1 Análisis y Diseño 45

4.2.2 Implementación: Instalación y Configuración de Zabbix 45

4.2.3 Implementación: Instalación de Zabbix Server 47

4.2.4 Pruebas: Prueba de Monitoreo de Equipo 53

4.3 Aplicación de Cascada con Zimbra 55

4.3.1 Análisis y Diseño 55

4.3.2 Implementación: Instalación y Configuración de Zimbra 55

4.3.3 Pruebas: Prueba de Envío de Correo electrónico. 60

4.4 Aplicación de Cascada con Telegram 61

4.4.1 Análisis 61

4.4.2 Diseño: Diagrama de Paquetes 61

4.4.3 Implementación: Código Fuente 62

4.4 Pruebas: Envío de mensajes de pruebas a cuenta personal de Telegram 64

4.5 Aplicación General de Cascada 66

4.5.1 Análisis 66

4.5.2 Diseño: Diagrama de Red 67

4.5.3 Integración: De los entregables de las 3 cascadas anteriores 67

4.5.4. Pruebas: Envío de notificación de Zabbix vía correo y Telegram 71

Capítulo 5: Conclusiones y Recomendaciones 73

5.1 Conclusiones 73

5.2 Recomendaciones 74

Bibliografía 75

Tabla de Figuras

Figura 1. Proceso de entrega de email. Fuente: Polluks, eigene Arbeit, ArgoUML 9

Figura 2. Trabajando con el Response Code. Fuente: Python for Beginners (2013). 29

Figura 3. Obtener el contenido. Fuente: Python for Beginners (2013) 29

Figura 4. Esquema general de la implementación de Cascada en este proyecto. 44

Figura 5. Instalación MariaDB. 45

Figura 6. Instalación de Apache 45

Figura 7. Dependencias PHP.....46

Figura 8. Iniciación de Apache y MariaDB en el arranque 46

Figura 9. Iniciación de servicios de Apache y MariaDB 46

Figura 10. Configuración inicial de MariaDB 47

Figura 11. Repositorio de Zabbix para CentOS.....47

Figura 12. Instalación de Zabbix 48

Figura 13. Instalación del frontend de Zabbix 48

Figura 14. Creación de la base de datos usando MariaDB 48

Figura 15. Creación de usuario con acceso a la base creada.....49

Figura 16. Importación del esquema de la base de datos.....49

Figura 17. Accediendo al archivo de configuración de Zabbix.....49

Figura 18. Añadiendo la contraseña del usuario de la base de datos creada. 49

Figura 19. Habiltando el servicio de Zabbix en el arranque. 49

Figura 20. Iniciación del servicio de Zabbix. 49

Figura 21. Cambio de zona horaria en PHP 50

Figura 22. Configuración de SELinux 50

Figura 23. Configuración de Zabbix vía web – Primera pantalla. 51
Figura 24. Configuración de Zabbix vía web - Revisión de prerrequisitos 51
Figura 25. Configuración de Zabbix vía web - Conexión con la Base de Datos 52
Figura 26. Configuración de Zabbix vía web - Finalización del proceso de instalación 53
Figura 27. Configuración de un host en Zabbix. 54
Figura 28. Monitoreo con Zabbix (1) 54
Figura 29. Monitoreo con Zabbix (2) 54
Figura 30. Instalación de Ubuntu Server 55
Figura 31. Configuración de DNS en servidor externo. 56
Figura 32. Instalación de dependencias 56
Figura 33. Configuración IP estática 57
Figura 34. Reiniciando el servicio de red en Ubuntu. 57
Figura 35. Removiendo Apparmor 57
Figura 36. Instalación servidor DNS 57
Figura 37. Configuración servidor DNS reverso 58
Figura 38. Validación de resolución de dominios de manera interna. 58
Figura 39. Validación de resolución de dominios de manera externa. 59
Figura 40. Descompresión de archivos de instalación. 59
Figura 41. Instalación de la herramienta desde el directorio descomprimido. 59
Figura 42. Validación de la instalación. 60
Figura 43. Comprobación del envío del correo electrónico. 60
Figura 44. Comprobación del recibimiento del correo electrónico. 61
Figura 45. Diagrama de paquetes de la implementación 61

Figura 46. Importación de librerías y módulos necesarios 62
Figura 47. Función para enviar el mensaje 62
Figura 48. Función para lidiar con posibles errores al enviar el mensaje. 63
Figura 49. Función main del script. 63
Figura 50. Archivo de configuración. 64
Figura 51. Creación de bot. 64
Figura 52. Pruebas de script 65
Figura 53. Archivo de log 65
Figura 54. Comprobación del envío del mensaje 65
Figura 55. Instalación biblioteca Pycco. 65
Figura 56. Generación de archivos de configuración en base a los scripts realizados. 66
Figura 57. Diagrama de red del sistema 67
Figura 58. Copiando el script desarrollado en el directorio de alertas de Zabbix. 67
Figura 59. Configuración de medio de notificación por Telegram 68
Figura 60. Configuración de Telegram para envío de notificaciones 68
Figura 61. Configuración de medio de notificación por correo; configuración de medio de notificación por correo. 69
Figura 62. Configuración de correo para envío de notificaciones; nivel de severidad de eventos definido. 69
Figura 63. Condiciones por cumplir para disparar una alerta en Zabbix. 70
Figura 64. Prototipo del mensaje a enviar. 70
Figura 65. Detalle de las operaciones a realizar. 71
Figura 66. Permisos de firewall para envío de alertas al bot de Telegram. 71

Figura 67. Notificación de evento en Zabbix 71
Figura 68. Envío de notificación por correo electrónico. 72
Figura 69. Envío de Notificación por Telegram 72

1.Introducción

1.1 Justificación

Gracias al constante crecimiento de las funciones en una organización apoyada en el uso de tecnología y, por lo tanto, el igual incremento de componentes, la idea de unificar el monitoreo de todos estos módulos se hace necesario. Las compañías de la actualidad se pueden definir como grandes puntos de encuentro entre la tecnología y el talento humano, en donde, todo debe confluir armónicamente si se esperan resultados positivos que permitan la continuación del negocio. Es debido a este frágil equilibro que se hace necesario el control de lo que sucede en cada punto funcional de la organización. Centralizar los datos de disponibilidad y rendimiento de cada uno de los componentes de una empresa se realiza con el fin de recolectar y almacenar dicha información para un posterior análisis de los mismos. Este análisis se fundamenta en la mejora continua que se propone como filosofía primordial dando paso a operativos correctivos tales como ampliación de recursos, corrección de defectos, mantenimiento de equipos tecnológicos, así como la correcta prevención de eventos importantes para evitar problemas futuros.

El presente trabajo se enfoca puramente en la sección tecnológica de este común incidente empresarial, dejando de lado enfoques que traten de manejar el problema desde la parte humana.

1.1.1 La importancia de los datos

Además de la importancia de crear un ambiente seguro en el que la organización sienta que está en control de lo que sucede con cada componente tecnológico y pueda sacarle ventaja, también se debe mencionar el hecho de la importancia que los datos tienen en la actualidad. En la competitividad en la que se encuentran sumergidas muchas compañías de estos días, muchas veces la correcta gestión de sus datos es lo que hace la diferencia con el resto. Si bien es cierto muchas

organizaciones se detienen a analizar y proteger los datos de ventas o aquellos que estén estrechamente relacionados con el negocio, también es menester de los empleados y empleadores el preocuparse de los datos internos de cómo están gestionando el negocio (tales como los datos recolectados por un servidor centralizado de funcionamiento y rendimiento de los distintos equipos tecnológicos pertenecientes a la organización).

1.2 Planteamiento del Problema

Para ejemplificar y solucionar el problema descrito brevemente en la sección anterior, se plantea el trabajo enfocado en la organización "Secure Soft" cuyo fin comercial es el de brindar todo tipo de soluciones de seguridad, tales como:

- Seguridad perimetral.
- Seguridad web.
- Seguridad de correo.
- Seguridad de bases de datos.
- Autenticación segura de doble factor.
- Administración de ancho de banda
- Optimización de enlaces.
- Soluciones anti-malware.
- Controlador de entrega de aplicaciones.
- Entre otros.

Debido al extenso abanico de servicios con los que dispone Secure Soft, la variedad y cantidad de equipos en la infraestructura tecnológica de la organización, tanto interna como externa, está en constante crecimiento. Esta continua ampliación de componentes permite crear el

ambiente propicio para la recolección de información pertinente al correcto funcionamiento, disponibilidad y métricas de rendimiento de cada uno de los mecanismos involucrados, por medio del uso de un servidor centralizado que sea capaz de almacenar los datos, que posteriormente serán analizados con el fin de que los operativos correspondientes sean desplegados y se notifique de cualquier actualización en los equipos, ya sean reportes de estado o posible malfuncionamiento.

1.3 Objetivos

1.3.1 Objetivo General

Implementar un ambiente de monitoreo de disponibilidad y rendimiento de los componentes de la infraestructura tecnológica de la organización Secure Soft, que sea capaz de enviar notificaciones automáticas de eventos e incidentes que se generen en los equipos de manera que exista un control total por parte de los operarios encargados.

1.3.2 Objetivos Específicos

- Asignar un dominio a la IP pública de la empresa y configurar el servidor DNS asociado.
- Implementar y configurar un servidor de correo electrónico para el envío de notificaciones.
- Implementar y configurar un bot para el envío de notificaciones por medio del servicio de mensajería Telegram. Según Freitas (2017), un bot puede definirse como una pieza de software que aprovecha la Inteligencia Artificial para realizar tareas específicas de un ser humano, es decir, automatizar ciertas labores y de ahí su nombre, relacionado con el ya conocido término de “robot”.
- Implementar y configurar un servidor de monitoreo.

1.4 Alcance

Para llevar a cabo el proceso de monitoreo se plantea el uso de software open source que se caracteriza por especializarse en pequeñas tareas, enfocando su efectividad en la tarea de completar dichas labores. Posteriormente se planea realizar una unificación general mediante la comunicación a nivel de servicios.

A continuación, se describen algunos de los productos de software que se utilizarán para completar distintas tareas:

- Para el servidor de monitoreo se va a usar la plataforma Zabbix.
- Para el servidor de correo electrónico se hará uso de la plataforma Zimbra.
- La implementación del bot se la realizará por medio de las APIs de Telegram y Zabbix para poder enviar un mensaje de notificación a un usuario. Una API

1.5 Limitaciones

Debido al planteamiento del presente estudio, así como el detalle de su alcance, existen dos grandes grupos de limitaciones que se derivan y que han de ser tomadas en cuenta para posteriores análisis.

El primer grupo de limitaciones es aquel que viene acompañado naturalmente por la selección del software con el que se trabajará. Si bien es cierto que, previo a la elección de determinado producto de software existe una reflexión que implica el tener en cuenta: funcionalidades disponibles, precio, accesibilidad, disponibilidad, etcétera, todo programa computarizado cuenta con limitaciones. En este caso, la primera limitación es la dada por el API de Telegram que, como bien se detalla en su manual online (Telegram), existen limitantes en la construcción de bots, como por ejemplo, la manera en que éstos envían o reciben mensajes, la

manera en que se manejan los archivos multimedia, la frecuencia con la que los bots pueden enviar mensajes sin obtener errores provenientes de la misma infraestructura de la aplicación, entre algunos otros. Asimismo, en la documentación provista por Zabbix se detalla su funcionamiento y, aunque no se especifican límites per sé, se pueden deducir por medio del análisis de la configuración y levantamiento de las funcionalidades relacionadas con esta API.

Por otro lado, se encuentran las limitaciones relacionadas con la organización misma, esto es, por ejemplo, la elección de determinado sistema operativo para montar herramientas de monitoreo debido a la infraestructura preestablecida de la organización, lo que se detalla posteriormente en el apartado de Especificación de Requerimientos, así como en la Implementación. Cuando se realiza un trabajo con un caso de estudio, las primeras limitantes con las que se debe lidiar son, precisamente, las derivadas del entorno de trabajo en donde se aplicarán los distintos procesos del estudio; siempre existe la posibilidad de tratar este tema con un enfoque hacia el cambio, promoviendo la participación de nuevas herramientas de software, pero dadas las complejidades de migración y relacionados, se prefiere adaptarse a las condiciones de la organización.

Capítulo 2: Marco Teórico

2.1 Fundamentos Teóricos

2.1.1 SNMP

2.1.1.1 Introducción

El SNMP (Simple Network Management Protocol) es un estándar que se encuentra en la mayoría de los computadores de la actualidad, debido a que se basa en la recopilación de dispositivos administrados en redes IP y que se orienta a cambiar la información del comportamiento de dichos dispositivos. Forma parte del conjunto de protocolos TCP/IP y permite, entre otras cosas, el manejo del rendimiento de red, encontrar y resolver problemas relacionados con la red y la generación de una planificación general para el crecimiento de dicha red. En la actualidad existen dos versiones de SNMP, pero usualmente se emplea la Versión 2 debido a la incorporación de distintas mejoras como el trabajo en conjunto de protocolos adicionales.

2.1.1.2 Componentes Básicos

Una red administrada SNMP tiene tres elementos principales: los dispositivos administrados, los agentes y los NMS(Network-management systems).

- Dispositivo administrado: es un nodo de red que contiene un agente SNMP y reside en un administrador de red. Los dispositivos administrados recopilan y almacenan información de administración y establecen la disponibilidad de dicha información para los NMS utilizando el protocolo SNMP. Estos dispositivos, a veces llamados elementos de red, pueden ser enrutadores, servidores de acceso, conmutadores, puentes, concentradores o hosts de computadoras/impresoras.

- Agente: un agente es un módulo de software de administración de red que reside en un dispositivo administrado. Un agente tiene conocimiento local de la información de gestión (administración) y traduce esa información en un formulario compatible con el protocolo SNMP.
- NMS: ejecuta aplicaciones que monitorean y controlan los dispositivos administrados; proporcionan la mayor parte de los recursos de procesamiento y memora necesarios para la gestión de la red. Se debe contar con al menos un NMS en la red.

2.1.1.3 Comandos Básicos

Los dispositivos administrados se monitorean y controlan mediante cuatro comandos básicos: lectura, escritura, captura y operaciones transversales.

- Lectura: es utilizado por un NMS para monitorear dispositivos administrados. El NMS examina diferentes variables que mantienen estos equipos.
- Escritura: es usado por un NMS para el control de los dispositivos administrados, cambiando los valores de las variables administradas dentro de estos equipos.
- Captura: se utiliza por parte de los dispositivos administrados para informar eventos de forma asíncrona al NMS.
- Operaciones transversales: usadas por el NMS para determinar qué variables admite un dispositivo administrado y para recopilar información secuencialmente en tablas variables (como una tabla de enrutamiento).

2.1.1.4 Base de Información de Gestión de SNMP

Es una recopilación de información que se organiza jerárquicamente y se accede a estas bases mediante el protocolo SNMP (u otro protocolo de gestión similar). Son compuestos por objetos gestionados y se identifican mediante identificadores de objeto. Un objeto administrado (a veces llamado un objeto MIB, un objeto o un MIB) es uno de cualquier número de características específicas de un dispositivo administrado. Los objetos administrados están compuestos por una o más instancias de objetos, que son esencialmente variables.

2.1.3 Servidor de Correo

Dentro del sistema de correo electrónico de Internet, un agente de transferencia de mensajes o un agente de transferencia de correo (MTA) o retransmisor de correo es un software que transfiere mensajes de correo electrónico de una computadora a otra usando SMTP. Los términos servidor de correo, intercambiador de correo y servidor MX también se usan en algunos contextos.

Los mensajes intercambiados a través de las redes se pasan entre los servidores de correo, incluidos los archivos de datos adjuntos (como imágenes, multimedia o documentos). Estos servidores también suelen mantener buzones de correo electrónico. El acceso a este correo electrónico por parte de los usuarios finales generalmente es a través de correo web o un cliente de correo electrónico.

Un agente de transferencia de mensajes recibe correo de otro MTA, un agente de envío de correo (MSA) o un agente de usuario de correo (MUA). Los detalles de transmisión se especifican mediante el Protocolo simple de transferencia de correo (SMTP). Cuando un buzón de destinatario de un mensaje no está alojado localmente, el mensaje se retransmite, es decir, se reenvía a otro MTA. Cada vez que un MTA recibe un mensaje de correo electrónico, agrega un campo de

encabezado de seguimiento recibido en la parte superior del encabezado del mensaje, creando así un registro secuencial de los MTA que manejan el mensaje. El proceso de elegir un MTA objetivo para el próximo salto también se describe en SMTP, pero generalmente se puede anular configurando el software MTA con rutas específicas.

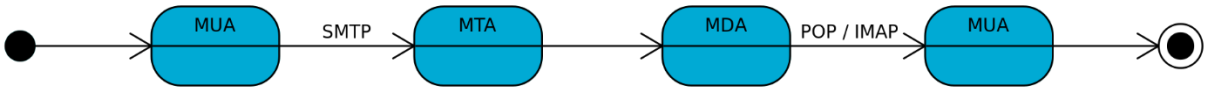


Figura 1. Proceso de entrega de email. Fuente: Polluks, eigene Arbeit, ArgoUML

Un MTA funciona en segundo plano, mientras que el usuario generalmente interactúa directamente con un agente de usuario de correo. Se puede distinguir el envío inicial como el primer paso a través de un MSA: el puerto 587 se usa para la comunicación entre un MUA y un MSA, mientras que el puerto 25 se usa para la comunicación entre MTA o de un MSA a un MTA; esta distinción es hecha primero en RFC 2476.

Para los destinatarios alojados localmente, la entrega final de correo electrónico a un buzón del destinatario es tarea de un agente de entrega de mensajes (MDA). Para este fin, el MTA transfiere el mensaje al componente de servicio de manejo de mensajes del agente de entrega de mensajes (MDA). Tras la entrega final, el campo Ruta de retorno se agrega al sobre para registrar la ruta de retorno.

2.1.3.1 Componentes

- POP3: En informática, el protocolo de oficina postal (POP) es un protocolo estándar de Internet de capa de aplicación utilizado por los clientes de correo electrónico para recuperar el correo electrónico de un servidor de correo. POP versión 3 (POP3) es la versión de uso común. El Protocolo de la Oficina de Correos proporciona acceso a través de una red de Protocolo de Internet (IP) para una aplicación cliente de usuario a un buzón (maildrop)

mantenido en un servidor de correo. El protocolo admite operaciones de descarga y eliminación de mensajes. Los clientes POP3 se conectan, recuperan todos los mensajes, los almacenan en la computadora del cliente y finalmente los eliminan del servidor. Este diseño de POP y sus procedimientos fue impulsado por la necesidad de que los usuarios tuvieran solo conexiones temporales a Internet, como el acceso por marcación, lo que les permite recuperar el correo electrónico cuando están conectados y, posteriormente, ver y manipular los mensajes recuperados cuando están desconectados. Los clientes POP3 también tienen una opción para dejar el correo en el servidor después de la descarga. Por el contrario, el Protocolo de acceso a mensajes de Internet (IMAP) se diseñó para dejar normalmente todos los mensajes en el servidor para permitir la administración con múltiples aplicaciones de cliente y para admitir modos de operación tanto conectados (en línea) como desconectados (fuera de línea).

- IMAP: En informática, el Protocolo de Acceso a Mensajes de Internet (IMAP) es un protocolo estándar de Internet utilizado por los clientes de correo electrónico para recuperar mensajes de correo electrónico de un servidor de correo a través de una conexión TCP / IP. IMAP se define por RFC 3501. IMAP fue diseñado con el objetivo de permitir la administración completa de un buzón de correo electrónico por parte de múltiples clientes de correo electrónico, por lo tanto, los clientes generalmente dejan mensajes en el servidor hasta que el usuario los elimine explícitamente. Un servidor IMAP normalmente escucha en el número de puerto 143. A IMAP sobre SSL (IMAPS) se le asigna el número de puerto 993. Prácticamente todos los clientes y servidores de correo electrónico modernos son compatibles con IMAP, que junto con el anterior POP3 (Protocolo de oficina de correos) son los dos protocolos estándar más frecuentes para la recuperación de correo electrónico.

Muchos proveedores de servicios de correo web como Gmail, Outlook.com y Yahoo! Mail también proporciona soporte para IMAP y POP3.

- SMTP: El Protocolo Simple de Transferencia de Correo (SMTP) es un protocolo de comunicación para la transmisión de correo electrónico. Como estándar de Internet, SMTP se definió por primera vez en 1982 por RFC 821, y se actualizó en 2008 por RFC 5321 a las adiciones SMTP extendidas, que es la variedad de protocolo de uso generalizado en la actualidad. Los servidores de correo y otros agentes de transferencia de mensajes usan SMTP para enviar y recibir mensajes de correo. Sistemas patentados como Microsoft Exchange e IBM Notes y sistemas de correo web como Outlook.com, Gmail y Yahoo! El correo puede usar protocolos no estándar internamente, pero todos usan SMTP cuando envían o reciben correos electrónicos desde fuera de sus propios sistemas. Los servidores SMTP suelen utilizar el Protocolo de control de transmisión en el puerto número 25. Los clientes de correo electrónico a nivel de usuario generalmente usan SMTP solo para enviar mensajes a un servidor de correo para retransmitirlos, generalmente envían correos electrónicos salientes al servidor de correo en el puerto 587 o 465 según RFC 8314. Para recuperar mensajes, IMAP y POP3 son servidores estándar, pero propietarios También a menudo implementamos protocolos propietarios, por ejemplo, Exchange ActiveSync.

2.1.4 Bot

Un bot o también conocido como agente de software, es una idea originada por John McCarthy a mediados de los años 50 y el concepto como tal fue acuñado, años después, por Oliver Selfridge en el MIT. McCarthy y Selfridge tenían en mente un sistema que, dado un objetivo, podía proporcionar detalles de las adecuadas operaciones computacionales, además de pedir ayuda en

ciertas circunstancias cuando tuviera dificultades. Se le llamó "soft-robot" debido a las semejanzas con el concepto de un robot tangible. (Kay, 1984).

Se puede decir, entonces, que un bot es un programa informático que actúa con un usuario en una relación de agencia, es decir, cumpliendo una tarea en nombre del usuario. Estos agentes pueden ser autónomos o trabajar en nombre de otros agentes o en conjunto con usuarios reales; por supuesto, para cumplir el propósito de ejecutar una tarea para el usuario, los bots deben poseer cualidades similares a las humanas como la comprensión del lenguaje natural y el habla, e inclusive cuando se trata de robots en sí, la personalidad de los humanos. (Nwana, 1996)

Entre las características más importantes de un agente autónomo se encuentran: el hecho de no necesitar ser invocados para una tarea sino poder activarse por ellos mismos; deben poder mantener su estado en un anfitrión, conservando el contexto; no requerir la interacción del usuario de manera directa y poder invocar otras tareas (u otros bots) de manera instantánea.

Si se desea hablar de bots (agentes) se debe mencionar la clara diferencia de aproximaciones que existe por parte de distintos autores, sin embargo, existen algunos conceptos que comúnmente se incluyen:

- Persistencia: el código del bot no se ejecuta por acción de un usuario, sino que se encuentra ejecutando continuamente y decide por sí mismo cuándo debe realizar alguna actividad.
- Autonomía: los agentes tienen la capacidad de selección de tareas, así como de priorizar dichos objetivos y, por supuesto, la toma de decisiones sin intervención de usuarios.

- Sociabilidad: los bots pueden ejecutar a otros a través de algún tipo de comunicación y coordinación para colaborar en alguna tarea.
- Reactividad: los agentes perciben el contexto en el que operan y reaccionan de manera apropiada. (Nwana, 1996)

2.1.5 A.P.I.

Una interfaz de programación de aplicaciones o API por sus siglas en inglés es una interfaz o protocolo de comunicación entre un cliente y un servidor destinado a simplificar la creación de software de lado del cliente. Se lo puede comparar con una especie de contrato entre el cliente y el servidor, en donde si el cliente realiza una solicitud en un formato específico, siempre recibirá una respuesta en un formato específico por parte del servidor o iniciará una acción definida. (Waldemar, Waldemar, Ierusalimschy, & Ierusalimschy, 1994)

Una API puede ser utilizada para un sistema basado en web, un sistema operativo, un sistema de base de datos, bibliotecas de software e inclusive el hardware de un equipo. Una especificación de una API puede tomar distintas formas, pero usualmente incluye instrucciones para rutinas, estructuras de datos, clases de objetos, variables y llamadas remotas. Por ejemplo, se tiene a POSIX como ejemplo de una API de Windows. La documentación de la API se proporciona con frecuencia para facilitar el uso y la implementación.

Una API con frecuencia está relacionada con una biblioteca de software. La API describe y prescribe el "comportamiento esperado" (una especificación) mientras que la biblioteca es una "implementación real" de este conjunto de reglas. Una API puede tener múltiples implementaciones (o ninguna si es abstracta) en forma de diferentes bibliotecas que comparten la misma interfaz de programación.

La separación de la API de su implementación puede permitir que los programas escritos en un idioma utilicen una biblioteca escrita en otro. Por ejemplo, debido a que Scala y Java compilan en código de bytes compatible, los desarrolladores de Scala pueden aprovechar cualquier API perteneciente a Java. (Waldemar, Waldemar, Ierusalimschy, & Ierusalimschy, 1994)

2.1.6 Linux

Es una familia de sistemas operativos de código abierto tipo Unix basados en el núcleo Linux lanzado por primera vez en septiembre de 1991 por Linus Torvalds. Usualmente el sistema operativo se lo encuentra con alguna distribución, las cuales incluyen el kernel de Linux, así como las bibliotecas y el software de soporte, muchos de los cuales son proporcionados por el Proyecto GNU. (Linux, s.f.)

Muchas de las distribuciones usan la palabra "Linux" en su nombre, sin embargo, la Free Software Foundation usa el nombre GNU/Linux para enfatizar la importancia del software GNU. Entre las distribuciones más populares de Linux se incluyen Debian, Fedora y Ubuntu. En cuanto a distribuciones comerciales, las más reconocidas son Red Hat Enterprise Linux y SUSE Linux Enterprise Server. Las distribuciones de escritorio de Linux incluyen un sistema de ventanas como X11 o Wayland, y un entorno de escritorio como GNOME, sin embargo, las distribuciones destinadas a servidores suelen omitir gráficos por completo o incluir una pila de soluciones como LAMP. Por supuesto, como Linux es libremente redistribuible, cualquiera puede crear una distribución para cualquier propósito. (Linux, s.f.)

En un inicio, Linux fue desarrollado para computadoras personales basadas en la arquitectura Intel x86, pero desde entonces se han realizado versiones para más plataformas que cualquier otro sistema operativo. Linux es el sistema operativo líder en servidores y otros grandes segmentos como las computadoras mainframe, algunas supercomputadoras e inclusive teléfonos

móviles, sin embargo, es apenas usado por el 2.3% de las computadoras de escritorio en el mercado.

Un sistema basado en Linux es un sistema operativo similar a Unix por lo que obtiene muchos de sus principios básicos de diseño de los principios establecidos en Unix en los 70 y 80. Dicho sistema utiliza un núcleo monolítico, el conocido núcleo Linux que maneja el control de procesos, las redes, el acceso a los periféricos y los sistemas de archivos. Los controladores de dispositivos se integran directamente con el kernel o se agregan como módulos que se cargan mientras el sistema se está ejecutando. El usuario GNU es una parte clave de la mayoría de los sistemas basados en el kernel de Linux, siendo Android la más notable excepción.

En el tema de los componentes preinstalados de un sistema Linux se pueden mencionar algunos de los más importantes:

- Gestor de arranque: por ejemplo, GNU GRUB o Gummiboot. Es el programa que carga el kernel de Linux en la memoria principal de la computadora; se ejecuta por la computadora cuando se enciende y después de que se realiza la inicialización del firmware.
- Bibliotecas de software: contienen código que puede ser utilizado por procesos en ejecución. En sistemas Linux que utilizan archivos ejecutables en formato ELF, el enlazador dinámico que gestiona el uso de bibliotecas dinámicas se conoce como: ld-linux.so. Si el sistema está configurado para que el usuario compile el software, también se incluirán archivos de encabezado para describir la interfaz de las bibliotecas instaladas.
- Sistema de gestión de paquetes: por ejemplo, dpkg y RPM. Alternativamente los paquetes se pueden compilar a partir de tarballs binarios.

- Programas de interfaz de usuario: por ejemplo, shells de comandos o entornos de ventanas. (Linux, s.f.)

El kernel de Linux está licenciado bajo la GNU General Public License (GPL) versión2. La GPL requiere que cualquier persona que distribuya software basado en el código fuente bajo esta licencia, debe poner a disposición del destinatario el código fuente original, así como cualquier modificación del mismo bajo los mismos términos. (Linux, s.f.)

2.1.7 Open Source

No hace falta mencionar que el término "open source" hace referencia a un acuerdo de libre uso de cierto producto de software, es decir, bien sea para modificar su estructura o para compartirlo con otros usuarios. Es evidente que, a pesar de haber sido acuñado con temas estrictos de programas computacionales, este término se ha ido ampliando para otra clase de productos (sobre todo aquellos relacionados con la Informática y la Cultura Digital). Sin embargo, muchas personas aún confunden este concepto con que un producto de software no tenga ningún valor comercial (es decir, que sea gratis para los usuarios). (OpenSource, 2019)

Varias organizaciones en la actualidad promueven el uso de herramientas digitales "open source" debido a ciertas ventajas, por ejemplo:

- El poder de la comunidad: cuando se habla de un programa distribuido con alguna licencia comercial y se trata de software propietario (es decir, el usuario debe pagar cierta cantidad de dinero para poder usar al programa) y su código fuente no está disponible, se pierde mucha de la retroactividad que se genera con las comunidades. Mientras más personas sean capaces de observar y entender el funcionamiento de los programas digitales, éstos pueden crecer rápidamente debido a la cantidad de

usuarios involucrados y que desean tener experiencias más limpias. Las comunidades de software open source siempre son extensas y muy útiles.

- Control sobre el software: si a la gran mayoría de usuarios les cuesta encontrar valor a algo intangible como un programa informático, es incluso peor cuando ellos no tienen la capacidad de observar a profundidad la estructura de estos programas y no pueden redistribuirlos.
- Los derivados: existen muchos derivados de las ventajas mencionadas previamente: como el aumento de seguridad debido a que la comunidad encuentra bugs y problemas de rendimiento de manera más constante, lo que también implica aumento en la estabilidad de los programas sin depender de los desarrolladores para ir actualizando dicho software. (OpenSource, 2019)

2.2 Herramientas

2.2.1 Zabbix

2.2.1.1 Introducción

Zabbix es una solución informática de código abierto que se especializa en el monitoreo que es generalmente empleada en ámbitos empresariales. Esta herramienta fue desarrollada, en un inicio, por Alexei Vladishev y ahora por la compañía Zabbix SIA, de la cual Vladishev es CEO y mantiene un constante acercamiento con la comunidad. La compañía ha tenido un crecimiento considerable hasta llegar a posicionarse como una de las principales opciones de monitoreo, adquiriendo varios aliados comerciales como Axel IT, Grupo SIA o Infocom. (Zabbix, 2019).

Zabbix se especializa en el monitoreo de numerosos parámetros de una red, así como la salud e integridad de servidores. Uno de los pilares de esta herramienta es la flexibilidad con la que opera a la hora de generar alertas basadas en el envío de correos electrónicos para

prácticamente cualquier evento. Como ya se mencionó en el Capítulo 1, esto permite una pronta acción de los operadores de una compañía con respecto a algún problema del servidor. Además, la herramienta se enfoca en la generación de informes y visualización de datos basados en la información almacenada, dándole relevancia y jerarquía a aquello que la compañía ha venido realizando durante un período de tiempo (datos de la compañía en sí misma), lo que convierte a Zabbix en una buena opción para la planificación de capacidad. (Zabbix, 2019).

El entorno que ofrece Zabbix considera tanto el sondeo como la captura (polling y trapping en inglés respectivamente), acciones que serán profundizadas posteriormente. Para acceder a los informes, estadísticas varias y parámetros de configuración de Zabbix, se necesita acceder a una interfaz web, lo cual se ha vuelto una práctica común en esta clase de software para poder evaluar estado de servidores desde cualquier ubicación. Con lo antes mencionado se puede intuir la relevancia de Zabbix para el monitoreo de infraestructura de TI y no necesariamente se habla de compañías grandes con multitud de servidores, sino también de organizaciones pequeñas con pocos servidores.

El costo de Zabbix es cero. El programa fue escrito y distribuido bajo la licencia GPL versión 2, lo que implica que el código fuente es distribuido gratuitamente y está disponible para el público en general.

2.2.1.2 Características de Zabbix

Zabbix es una solución de monitoreo de red altamente integrada, que ofrece una multiplicidad de funciones en un solo paquete, para hablar de todas sus características se realizará un breve resumen de las más importantes y que se muestra a continuación:

- Recolección de datos: incluye la verificación de disponibilidad y rendimiento, monitoreo de VMWare, personalización de intervalos para la recolección de datos específicos, verificaciones personalizadas, etcétera.
- Definiciones flexibles de umbrales: se pueden establecer umbrales flexibles de varios problemas llamados “triggers” que hacen referencia a valores de la base de datos de fondo.
- Alteras altamente configurables: el envío de notificaciones se puede personalizar tomando en cuenta el horario de escalación, destinatario, tipo de medio, además las notificaciones se pueden hacer significativas (o darles una importancia más elevada) y útiles utilizando variables macro. Las acciones automáticas incluyen comandos remotos.
- Amplias opciones de visualización: capacidad de crear gráficos personalizados que pueden combinar varios elementos en una sola vista, generación de mapas de red, pantallas personalizadas y presentaciones de diapositivas para obtener una descripción general como un tablero, generación de informes, etcétera.
- Almacenamiento de datos históricos: guardado de datos en una base de datos específica, historia configurable y procedimiento de limpieza incorporado.
- Fácil configuración: agregar dispositivos monitoreados como hosts, los hosts son recogidos para el monitoreo una vez que se encuentren en la base de datos y se pueden generar plantillas a dispositivos monitoreados.
- API de Zabbix: proporciona una interfaz programable a Zabbix para manipulación masiva, integración de software de terceros y otros fines.

- Sistema de permisos: autenticación segura de usuarios, limitación de vistas a ciertos usuarios. (Zabbix, 2019)

2.2.1.3 Arquitectura de Zabbix

Zabbix se compone de varios componentes de software, con distintas funciones cada uno, las cuales se detallan a continuación:

- Servidor Zabbix: es el componente central al que los agentes reportan disponibilidad, información de integridad y estadísticas. El servidor es el repositorio central en donde se almacenan todos los datos de configuración, estadísticas y datos operativos.
- Almacenamiento de base de datos: toda la información de configuración, así como los datos recopilados por Zabbix, se almacenan en una base de datos.
- Interfaz web: para un fácil acceso a Zabbix desde cualquier lugar y desde cualquier plataforma, se proporciona una interfaz basada en la web. La interfaz es parte del servidor Zabbix, y generalmente (pero no necesariamente) se ejecuta en la misma máquina física que la que ejecuta el servidor.
- Proxy: el proxy de Zabbix puede recopilar datos de rendimiento y disponibilidad en nombre del servidor Zabbix. Un proxy es una parte opcional de la implementación de Zabbix, sin embargo, puede ser muy beneficioso distribuir la carga de un único servidor Zabbix.
- Agente: los agentes de Zabbix se implementan en objetivos de monitoreo para controlar activamente los recursos y aplicaciones locales e informar los datos recopilados al servidor de Zabbix.

- Flujo de datos: aunque el flujo de datos como tal no es una parte "tangible" de la arquitectura de Zabbix, es importante mencionarlo para que se entienda el proceso de funcionamiento de esta plataforma. Para crear un elemento que reúna datos, primero se debe crear un host. Al pasar al otro extremo del espectro de Zabbix, primero se debe tener un elemento para crear un trigger. Se debe tener un trigger para crear una acción. Por lo tanto, si desea recibir una alerta de que la carga de un CPU es demasiada en el Servidor X, primero se debe crear una entrada de host para el Servidor X, seguido de un elemento para monitorear su CPU, luego un trigger que se activa si el uso de dicho CPU es demasiado alto, seguido de una acción que se encarga de enviar un correo electrónico. Si bien eso puede parecer muchos pasos, con el uso de plantillas realmente no lo es. Sin embargo, debido a este diseño, es posible crear una configuración muy flexible. (Zabbix, 2019).

2.2.2 Zimbra

Es un software colaborativo que incluye un servidor de email y un cliente web. La compañía que desarrolló este software fue LyquidSys que luego cambió su nombre a Zimbra Inc. La primera versión de este paquete de software fue lanzada en 2005. Posteriormente atravesaron por distintas situaciones comerciales con la venta de toda la organización a Yahoo! a finales del año 2007 y en el 2010, asimismo, por parte de VMware. Situación que no se terminaría de resolver hasta el año 2015, cuando fueron adquiridos por Sycanor. Funciona sobre el sistema operativo Linux y cuenta con una licencia de distribución personalizada, pero se puede dividir a este software por tener dos versiones: una open-source y otra comercial con elementos cuyo código fuente no es libre, que son la Network Edition y la Open Source Edition. (Zimbra, 2018).

ZCS Web Client es el paquete principal y que admite funciones correo electrónico, calendarios grupales y uso compartido de documentos mediante una interfaz web Ajax, la cual permite sugerencias de herramientas, elementos “drag and drop” y menús de click derecho en la interfaz de usuario. También se incluyen capacidades de búsqueda avanzada y relaciones de fechas, autoría de documentos en línea, mashups "Zimlet" y una interfaz de usuario de administración completa.

El servidor ZCS utiliza varios segmentos de código abierto, pero también existen componentes cuyo código fuente no puede ser usado como por ejemplo el denominado “Messaging Application Programming Interface” para sincronizar contactos y calendario con Outlook. También es importante señalar la presencia de una interfaz de programación de aplicaciones SOAP en todas sus funciones y también la de un servidor IMAP y POP3. El servidor se ejecuta en muchas distribuciones de Linux. (Zimbra, 2018).

ZCS puede sincronizar correo, contactos y elementos de calendario con clientes de correo de código abierto como Evolution y también con clientes propietarios como Microsoft Outlook y Apple Mail, ya sea a través de conectores propietarios o utilizando el protocolo ActiveSync, ambos disponibles exclusivamente en la versión con soporte comercial. Zimbra también proporciona sincronización bidireccional nativa a muchos dispositivos móviles como Android o iOS. (Zimbra, 2018).

2.2.3 Telegram Bot API

Antes de hablar del funcionamiento de la API de la cual se hará uso en el presente trabajo, se deben mencionar dos puntos fundamentales: el concepto detrás de Telegram y el funcionamiento de sus bots.

2.2.3.1 Telegram

Tal y como menciona la propia compañía Telegram, su aplicación se basa en la mensajería orientada en la seguridad y velocidad, que es ofrecida al público de manera gratuita. Telegram tiene como uno de sus principales puntos de referencia la versatilidad a la hora de sincronizar los mensajes enviados por esta plataforma, sin importar si una de las cuentas creadas está asociada a un dispositivo o un número de teléfono. La popularidad de Telegram ha sido notable desde que se ha comprobado su eficiente manejo de seguridad, así como la disponibilidad de la API que permite construir aplicaciones basadas en Telegram de manera sencilla.

2.2.3.2 Características del bot de Telegram

Con el bot de Telegram se manejan aspectos tales como la obtención de notificaciones y noticias personalizadas. Como ya se dijo en secciones anteriores, un bot puede actuar como un periódico inteligente enviando contenido relevante tan pronto como se publique sin ninguna gestión por parte del usuario. También se debe tener en cuenta la integración con otros servicios; un bot puede enriquecer los chats de Telegram con contenido de servicios externos como un Bot de Gmail o un bot para obtener imágenes o incluso videos de repositorios tan populares como Youtube. No solamente se puede controlar temas netamente multimedia, sino también aspectos como el tema de la gestión de pagos virtuales; un bot puede ofrecer servicios de pagos o trabajar como escaparate virtual. Asimismo, la creación de herramientas personalizadas como la creación de alertas, pronósticos de tiempo, traducciones, entre otros, es parte importante de las funciones que un bot podría aportar en este sentido. Inclusive, se pueden crear pequeños juegos para una o varias personas debido a las ricas experiencias basadas en HTML5 que un bot puede ofrecer. Por último, un bot de Telegram puede fungir como conexión entre personas que estén buscando

conversaciones gracias a las interacciones de los usuarios y aspectos como proximidad o intereses comunes.

2.2.3.2 Funcionamiento de un bot de Telegram

En esencia, los bots de Telegram son cuentas especiales que no requieren un número de teléfono adicional para ser configuradas. Los usuarios pueden interactuar con bots de dos maneras:

- Se pueden enviar mensajes y comandos a los bots abriendo un chat con estas cuentas especiales o agregándolos a un grupo. Esto puede ser usado para bots de noticias, por ejemplo.
- Se pueden enviar solicitudes directamente desde el campo de entrada escribiendo el nombre de usuario del bot y una consulta en particular. Esto permite enviar contenido desde bots en línea directamente a cualquier chat o grupo.

Todos los cambios, configuraciones y más se encuentran en el sitio oficial de Telegram dedicado al Bot API.

2.2.4 Python

2.2.4.1 Introducción

Python es un lenguaje de programación interpretado, de alto nivel y de propósito general. Creado por Guido van Rossum y lanzado por primera vez en 1991, la filosofía de diseño de Python enfatiza la legibilidad del código con su uso notable de espacios en blanco significativos. Sus construcciones de lenguaje y su enfoque orientado a objetos tienen como objetivo ayudar a los programadores a escribir código claro y lógico para proyectos a pequeña y gran escala.

Python se escribe dinámicamente y se recolecta "basura". Es compatible con múltiples paradigmas de programación, incluida la programación procesal, orientada a objetos y funcional.

Python se describe a menudo como un lenguaje de "baterías incluidas" debido a su biblioteca estándar integral.

Python fue concebido a fines de la década de 1980 como un sucesor del lenguaje ABC. Python 2.0, lanzado en 2000, introdujo características como la comprensión de listas y un sistema de recolección de basura capaz de recolectar ciclos de referencia. Python 3.0, lanzado en 2008, fue una revisión importante del lenguaje que no es completamente compatible con versiones anteriores, y gran parte del código de Python 2 no se ejecuta sin modificaciones en Python 3. Debido a la preocupación sobre la cantidad de código escrito para Python 2, soporte para Python 2.7 (el último lanzamiento de la serie 2.x) se extendió hasta 2020. El desarrollador de idiomas Guido van Rossum asumió la responsabilidad exclusiva del proyecto hasta julio de 2018, pero ahora comparte su liderazgo como miembro de un consejo directivo de cinco personas.

El lenguaje Python 2, es decir, Python 2.7.x, estará "caducando" el 1 de enero de 2020, y el equipo de voluntarios de Python no solucionará problemas de seguridad ni proveerá mejoras de otras maneras después de esa fecha. Con el final de la vida útil, solo Python 3.6.xy posterior, p. Python 3.8, que se lanzará en octubre de 2019 (actualmente en versión beta), será compatible.

Los intérpretes de Python están disponibles para muchos sistemas operativos. Una comunidad global de programadores desarrolla y mantiene CPython, una implementación de referencia de código abierto. Una organización sin fines de lucro, Python Software Foundation, administra y dirige recursos para el desarrollo de Python y CPython.

2.2.4.2 Historia

Python fue concebido a fines de la década de 1980 por Guido van Rossum en Centrum Wiskunde & Informatica (CWI) en Holanda como un sucesor del lenguaje ABC (inspirado por SETL), capaz de manejar excepciones e interactuar con el Sistema operativo Amieva.

Su implementación comenzó en diciembre de 1989. Van Rossum continuó como desarrollador principal de Python hasta el 12 de julio de 2018, cuando anunció sus "vacaciones permanentes" de sus responsabilidades como “Dictador Benévolo de por Vida de Pyhton”, un título que la comunidad de Python le otorgó para reflejar su compromiso a largo plazo como jefe del proyecto y parte de la toma de decisiones. En enero de 2019, los desarrolladores activos de Python eligieron a Brett Cannon, Nick Coghlan, Barry Warsaw, Carol Willing y Van Rossum para un "Consejo Directivo" de cinco miembros para dirigir el proyecto.

Python 2.0 se lanzó el 16 de octubre de 2000 con muchas características nuevas importantes, que incluyen un recolector de basura con detección de ciclo y soporte para Unicode.

Python 3.0 se lanzó el 3 de diciembre de 2008. Fue una revisión importante del lenguaje que no es completamente compatible con versiones anteriores. Muchas de sus características principales se incorporaron a las series de versiones Python 2.6.x y 2.7.x. Las versiones de Python 3 incluyen la utilidad 2to3, que automatiza (al menos parcialmente) la traducción del código de Python 2 a Python 3.

La fecha de finalización de la vida útil de Python 2.7 se estableció inicialmente en 2015 y luego se pospuso hasta 2020 por preocupación de que un gran cuerpo de código existente no podría ser fácilmente transferido a Python 3. En enero de 2017, Google anunció el trabajo en un

“transcompilador” Python 2.7 to Go para mejorar el rendimiento bajo cargas de trabajo concurrentes.

2.2.4.3 Manejo de paquetes Pip

pip es un sistema de administración de paquetes estándar de facto utilizado para instalar y administrar paquetes de software escritos en Python. Se pueden encontrar muchos paquetes en la fuente predeterminada para paquetes y sus dependencias: Python Package Index (PyPI).

La mayoría de las distribuciones de Python vienen con pip preinstalado. Python 2.7.9 y posterior (en la serie python2), y Python 3.4 y posterior incluyen pip (pip3 para Python 3) de forma predeterminada.

“pip” es un acrónimo recursivo para "Paquetes de instalación de Pip". Una de las principales ventajas de pip es la facilidad de su interfaz de línea de comandos, que hace que la instalación de paquetes de software Python sea tan fácil como emitir un comando:

pip install some-package-name

Los usuarios también pueden eliminar fácilmente el paquete:

pip uninstall some-package-name

Lo más importante es que pip tiene una función para administrar listas completas de paquetes y números de versión correspondientes, que es posible a través de un archivo de "requisitos". Esto permite la recreación eficiente de un grupo completo de paquetes en un entorno separado (por ejemplo, otra computadora) o entorno virtual. Esto se puede lograr con un archivo con el formato correcto y el siguiente comando, donde require.txt es el nombre del archivo:

pip install -r requirements.txt

2.2.4.4 Biblioteca Requests

Requests es una biblioteca HTTP con licencia Apache2, escrita en Python. Está diseñado para ser utilizado por humanos para interactuar con el lenguaje. Esto significa que no se tiene que agregar manualmente cadenas de consulta a las URL ni codificar de forma sus datos POST.

La biblioteca Requests permite el envío de solicitudes HTTP / 1.1 utilizando Python. Con esto se puede agregar contenido como encabezados, datos de formulario, archivos multiparte y parámetros a través de bibliotecas Python simples. También se permite acceder a los datos de respuesta de Python de la misma manera.

En programación, una biblioteca es una colección o selección preconfigurada de rutinas, funciones y operaciones que un programa puede usar. Estos elementos a menudo se denominan módulos y se almacenan en formato de objeto.

Para la instalación de esta biblioteca se puede seguir el procedimiento que ofrece pip, es decir *pip install requests* y de esta manera tener la biblioteca disponible para ser usada en el resto de código, por supuesto, luego de su debida importación.

Es menester mencionar el significado de una "request" o solicitud. Cuando se hace un ping a un sitio web o algún portal en línea buscando alguna información en particular, se está generando una solicitud. Y eso es precisamente lo que esta biblioteca genera. Por ejemplo:

$$r = requests.get('https://www.google.com')$$

Antes de realizar cualquier acción con un sitio web en Python es aconsejable analizar el estado actual de ese portal. Para hacerlo se puede emplear algo como lo que se muestra en la Figura 1.

```
r = requests.get('https://github.com/timeline.json')
r.status_code
>>200

r.status_code == requests.codes.ok
>>> True

requests.codes['temporary_redirect']
>>> 307

requests.codes.teapot
>>> 418

requests.codes['o/']
>>> 200
```

Figura 2. Trabajando con el Response Code. Fuente: Python for Beginners (2013).

Para obtener el contenido como tal de la página web, se emplea lo mostrado en la Figura 2

```
import requests
r = requests.get('https://github.com/timeline.json')
print r.text

# The Requests library also comes with a built-in JSON decoder,
# just in case you have to deal with JSON data

import requests
r = requests.get('https://github.com/timeline.json')
print r.json
```

Figura 3. Obtener el contenido. Fuente: Python for Beginners (2013)

un servidor. Gracias a cómo funcionan la biblioteca Requests, se puede acceder a los encabezados utilizando las mayúsculas necesarias.

Si se realiza esta función, pero no existe un encabezado en la respuesta, el valor predeterminado será “none”.

La biblioteca Requests disminuirá automáticamente cualquier contenido extraído de un servidor, pero la mayoría de los juegos de caracteres Unicode están decodificados sin problemas de todos modos.

Cuando se realiza una solicitud a un servidor, la biblioteca Requests realiza una conjetura sobre la codificación de la respuesta, y lo hace en función de los encabezados HTTP. La codificación que se adivina se usará cuando acceda al archivo r.text.

A través de este archivo, se puede discernir qué codificación está utilizando la biblioteca Requests y cambiarla si es necesario. Esto es posible gracias a la propiedad r.encoding. Si se cambia el valor de codificación, las solicitudes utilizarán el nuevo tipo siempre que se llame a r.text en el código.

Hay una serie de excepciones y códigos de error con los que debe estar familiarizado al usar la biblioteca Requests en Python:

- Si hay un problema de red como una falla de DNS o una conexión rechazada, la biblioteca Requests generará una excepción “ConnectionError”.
- Con respuestas HTTP no válidas, las solicitudes también generarán una excepción HTTPError, pero no son muy comunes.
- Si se agota el tiempo de espera de una solicitud, se generará una excepción de tiempo de espera.
- Si una solicitud excede el número preconfigurado de redireccionamientos máximos, se generará una excepción TooManyRedirects.
- Cualquier excepción que surja de Solicitudes se heredará del objeto request.exceptions.RequestException.

2.3 Automatización de Procesos

2.3.1 Notificación de Eventos

Mediante la utilización de Zabbix como un entorno controlado en el que se pueda controlar la situación interna de la organización, se plantea la automatización de la notificación de eventos especiales con algunas de las tecnologías mencionadas previamente, las cuales son:

- Envío de mensajes críticos con información específica de algún evento, ya sean prevenciones o mensajes para informar del estado, equipo por equipo, de toda la infraestructura interna. Esto se lo conseguirá mediante el uso del bot de Telegram.
- Envío de correos electrónicos con información similar a la de los mensajes. Esto se lo conseguirá mediante el uso del API de Zimbra.
- Como se especificó previamente, el agente de Zabbix está en constante control de los datos generados por aplicaciones o equipos dentro de la organización, los cuales se encuentran recopilados en el servidor de Zabbix; este mismo agente está relacionado con la notificación de eventos a los administradores por estadísticas y datos específicos de algún equipo puntual. Serán el agente y el servidor Zabbix los que se encarguen de la apropiada comunicación con los API's involucrados para la automatización de este punto. (Zabbix, 2019)
- Posteriormente, en el capítulo de Implementación, se detallarán los aspectos principales de cómo fueron aplicados estos conceptos.

2.3.2 Escalamiento de Incidentes

El escalamiento de incidentes trata de evitar posibles inconvenientes relacionados con la jerarquía interna de una organización. En muchas ocasiones, los funcionarios no están totalmente

capacitados para resolver un problema y esto no implica que siempre sea por desconocimiento técnico o falta de experiencia, sino por falta de recursos y/o importancia suficiente para la toma de decisiones.

Por lo tanto, se plantea la notificación directa a los encargados/responsables que cuenten con todo lo necesario para solventar algún inconveniente. Esto se lo va a desarrollar de la misma manera que el punto anterior: un envío de mensaje usando Telegram y un correo electrónico usando Zimbra.

Capítulo 3: Especificación de Requerimientos

3.1 Introducción

Antes de empezar la etapa de implementación se deben tener en cuenta todas las consideraciones impuestas tanto por la parte tecnológica (herramientas, productos de software y derivados) como por la parte operativa (la institución/organización en donde se realizará el proceso). Por lo tanto, se llevará a cabo un análisis de requerimientos o requisitos que funciona como un acuerdo entre las dos partes, es decir, la empresa que pasa a ser cliente y mi persona que viene a ser el proveedor.

Si bien el presente trabajo no necesariamente se lo puede ver como una construcción de un producto de software como tal, en esta fase previa de levantamiento de información y requisitos será tratado precisamente como un software detallando tanto las necesidades del lugar de implantación tanto inmediatas como a largo plazo. Sin embargo, debido a la estructura particular de este estudio, no se presenta a continuación un documento detallado de requerimientos (como sí debería hacerse en un desarrollo de software convencional). Por lo tanto, la estructura de esta sección quedará compuesta por un Análisis de Requerimientos general que englobe todos los requisitos propuestos por la organización y sus implicaciones, mientras que en el aspecto de la infraestructura se destina el espacio de Requisitos Técnicos, donde se tratarán las distintas necesidades tanto de hardware como de software que poseen todas las herramientas tecnológicas descritas en capítulos previos.

El propósito de este capítulo es el de dejar completamente claro cómo se procederá para evitar conflictos internos y externos en el sitio donde se implantará el sistema global. Analizar los requerimientos es fundamental para establecer límites al estudio y plantear vías alternas si es que

existiese algún conflicto con ciertos elementos de la organización, como falta de activos, presupuesto, etcétera.

No se prefijan lineamientos de desarrollo de software de ningún tipo ni se tendrán en cuenta patrones de diseño debido a las características de este estudio, como ya se mencionó, sin embargo lo que el Análisis de Requerimientos provee sigue siendo un medio para facilitar las pruebas y revisiones debido a las pautas generales que fueron establecidas, además de presentar el alcance del trabajo, que permitirá establecer cuándo el mismo ha sido concluido o si es que es necesario incrementar algo adicional. Para el tema del desarrollo de las pruebas del sistema general, sí se establecerán ciertas reglas a seguir de acuerdo a la metodología de desarrollo que se planteará posteriormente y cuya elección es personal.

3.2 Análisis de Requerimientos

3.2.1 Requisitos de Hardware/Requisitos Físicos

La organización plantea como primer requisito el tener en consideración el número de equipos disponibles dentro de la misma, ya sean éstos: computadores personales de los empleados, servidores internos y asimismo la cantidad de servidores externos (de parte de los clientes).

Este punto no implica analizar si los servidores, por ejemplo, soportan las plataformas que serán necesarias para el monitoreo de equipos, sino más bien, analizar el número de los mismos.

Luego de la revisión correspondiente de los datos facilitados por la organización, se establece que todo lo que las plataformas de software establezcan como requisitos (que se especificará en la siguiente sección) están cubiertas por parte de los equipos internos, en especial en el tema de los servidores.

Un dato adicional es que, de acuerdo a como se plantea la estructura de este estudio, se necesita contar con algunos teléfonos móviles que puedan recibir mensajes en la plataforma Telegram, esto tampoco resulta un problema de acuerdo a los estándares de comunicación de empleados de la organización que provee un teléfono inteligente para cada colaborador de la empresa. Dejando de lado el hecho de que la organización es la que ha solicitado en primer lugar realizar una revisión de que los equipos tecnológicos cumplan en número y forma, es menester mencionar que éste no deja de ser un requisito general que debe establecerse en casi cualquier proyecto de este tipo.

3.2.2 Requisitos propios de la organización

La organización ha manifestado su opinión sobre la implementación práctica del presente proyecto mediante los puntos que se detallan a continuación:

Uso general de software Open Source: uno de los objetivos de la organización es el migrar todos sus productos de software hacia aquellos que sean Open Source. Es por esto que, todas las herramientas que se usen en este estudio deben cumplir con este requisito, caso contrario no podrán ser implementadas en la organización.

Búsqueda de alternativas menos costosas/gratuitas: la organización, además, está en un proceso de reducción de costos gradual que implica el uso de herramientas alternativas. Por ejemplo, en la actualidad la organización tiene como estándar el uso de Office 365 para todo el tema de correos, sin embargo, el costo que tiene activar cada una de las licencias de este software es elevado. Por lo que se platea en este estudio una alternativa con la herramienta de correo electrónico introducida previamente.

Todo mecanismo que sea implantado deberá aportar en la automatización de procesos antiguos: la organización busca mejorar mediante la eliminación de trabajo manual en ciertos aspectos de trabajo común o diario, es por esto que se plantea que todo nuevo elemento que sea implementado en el núcleo de la empresa deberá suplir o eliminar cierta sección de la cadena de trabajo.

Previa aplicación en ambientes controlados: la organización, como es evidente, no está dispuesta a correr ningún riesgo en sus actividades comunes, por la implementación de un nuevo sistema o mecanismo tecnológico, por lo tanto, requiere que todo aquello que vaya a ser implantado pase por una ejecución en un ambiente controlado, que debe ser aprobado por la misma organización y que no afecte en ningún nivel las operaciones establecidas.

Pruebas: ligado al punto anterior, la organización requiere que se extiendan resultados aceptables de varias clases de pruebas para la instalación/implementación de cualquier software. De este modo se espera reducir cualquier clase de inconveniente técnico cuando un nuevo sistema entre en producción y así asegurar que la calidad sea un factor común en todos los productos de software de la organización.

La organización cuenta con certificación 9001 y 27001, por lo tanto, es fundamental llevar con control estricto con los respaldos de equipos que forman parte del core de negocio.

3.3.3 Enfoque del proyecto.

Uno de los requisitos subyacentes es el relacionado con el enfoque que debe tener la implementación de este estudio. La organización no desea un simple producto que le permita reducir el número de pasos en un determinado proceso, sino el de mirar más allá. El requisito primordial en este sentido es el de apostar por la proactividad en lugar de la reactividad, en otras

palabras, poder adelantarse a los hechos o problemas para tener una ventaja que permita establecer planes de contingencia o planes rápidos de mejora.

El hecho de pensar en un sistema global tecnológico que permita monitorear distintos equipos para encontrar fallas podría identificarse como un medio puramente funcional en el momento en el que algún equipo monitoreado falle. Sin embargo, la idea de este trabajo es usar todas las herramientas tecnológicas necesarias para estar siempre un paso delante de los posibles eventos que puedan suscitarse. Además, la organización también plantea como fundamento la línea de pensamiento que se explica a través del Zen de Python:

> "*Bello es mejor que feo. Explícito es mejor que implícito. Simple es mejor que complejo. Complejo es mejor que complicado. Plano es mejor que anidado. Espaciado es mejor que denso. La legibilidad es importante. Los casos especiales no son lo suficientemente especiales como para romper las reglas. Sin embargo, la practicidad le gana a la pureza. Los errores nunca deberían pasar silenciosamente. A menos que se silencien explícitamente. Frente a la ambigüedad, evitar la tentación de adivinar. Debería haber una, y preferiblemente solo una, manera obvia de hacerlo. A pesar de que esa manera no sea obvia a menos que seas Holandés. Ahora es mejor que nunca. A pesar de que nunca es muchas veces mejor que *ahora* mismo. Si la implementación es difícil de explicar, es una mala idea. Si la implementación es fácil de explicar, puede que sea una buena idea. Los espacios de nombres son una gran idea, ¡tengamos más de esos!"* (Python, 1999).

Todos los principios planteados por el Zen de Python son lineamientos generales que la organización se ha planteado y no necesariamente implican la utilización del lenguaje de programación relacionado, sino el atractivo de sus elementos.

El hecho de tener estas ideas detrás del desarrollo e implementación del sistema no suponen un impedimento o algún factor que promueva cambios, por el contrario, están en la línea de lo que se plantea y esas doctrinas pueden ser fácilmente asimiladas en este estudio. Es importante señalar nuevamente que, la elección del lenguaje de programación no ha sido impuesta de ningún modo, por lo que la decisión será exclusivamente personal sin ninguna clase de presión por parte de la organización (más allá de los lineamientos propuestos previamente).

Adicionalmente, se establece una clara diferencia entre la proactividad y el actuar deprisa. La proactividad es una característica que puede estar apegada tanto a un modo de trabajo como a un propio individuo; en este caso, la proactividad hace referencia a los comportamientos que se pueden tener para actuar frente a determinada situación siempre acudiendo a la calma y planificación. La proactividad suele dar paso a acciones premeditadas que pueden confundirse con decisiones tomadas sobre la marcha que no han sido estudiadas. La organización demanda que la proactividad sea parte de los principios de sus empleados.

3.3 Requisitos Técnicos

3.3.1 Zabbix

3.3.1.1 Hardware

- 128 MB de RAM mínimo. Se recomienda más memoria dependiendo de la plataforma a usar.
- 256 MB de espacio en disco (esto puede aumentar considerablemente de acuerdo al número de hosts y/o parámetros que se estén monitoreando).
- CPU con una frecuencia alta de reloj y varios núcleos debido a la gran cantidad de recursos necesarios por Zabbix dependiendo del número de parámetros a monitorear y del motor de base de datos elegido.

- Se requiere un puerto de comunicación serial y un módem GSM en serie para utilizar el soporte de notificación por SMS en Zabbix.
- Revisión de puertos de red para evitar conflictos con otros servicios en los equipos a configurar. (Zabbix, 2019)

3.3.1.2 Plataformas soportadas

Debido a los requisitos de seguridad y la naturaleza de misión crítica del servidor de monitoreo, UNIX es el único sistema operativo que puede brindar de manera consistente el rendimiento, la tolerancia a fallas y la resistencia necesarios. Sin embargo, se puede decir que Zabbix opera en las siguientes plataformas: Linux, IBM AIX, FreeBSD, NetBSD, OpenBSD, HP-UX, macOS, Solaris y todas las versiones de Windows desde XP. (Zabbix, 2019)

3.3.1.3 Software

Zabbix está construido alrededor de un servidor web Apache moderno, motores de bases de datos líderes y lenguaje PHP. A continuación, se listan una serie de productos que son requisitos para el funcionamiento adecuado de Zabbix:

Tabla 1. Requerimientos de Software de Zabbix

Software	Versión	Comentarios
MySQL	5.0.3-8.0.x	Se requiere si MySQL se usa como base de datos backend para Zabbix. Se requiere el motor InnoDB.
Oracle	10g o posterior	Requerido si Oracle se usa como base de datos backend de Zabbix.
Otros motores de BDD	Depende del motor	Requeridos si se usan como bases de datos backend de Zabbix.
Apache	1.3.12 o posterior	Requerido para el frontend.
PHP	5.4.0 o posterior	Requerido para el frontend.
Extensiones PHP	/	Algunas extensiones de PHP son requeridas.
Cookies y Javascript habilitados	/	En el navegador web del lado del cliente.

Biblioteca “libpcre”	8.x	Servidor. Se requiere la biblioteca PCRE para el soporte de Expresión regular compatible con Perl (PCRE). Los nombres pueden diferir según la distribución de GNU / Linux, por ejemplo, 'libpcre3' o 'libpcre1'.
Biblioteca “libevent”	1.4 o superior	Servidor. Necesario para soporte métrico masivo y monitoreo de IPMI.
Biblioteca “libpthread”	/	Necesario para el soporte de bloqueo de mutex y lectura-escritura.
Biblioteca “zlib”	/	Necesario para el soporte de compresión.
Espacio en disco para BDD	/	Consultar el soporte oficial para conocer la fórmula y determinar el espacio necesario en base a la configuración de Zabbix.

Fuente: Manual de Zabbix, 2019.

3.3.2 Zimbra

Este software tiene varios requerimientos técnicos divididos entre el servidor y el cliente. A continuación, se muestran los requisitos más importantes:

Para servidores:

1. Evaluación y Pruebas:

- Intel/AMD 64-bit CPU 1.5GHz.
- Para instalaciones de un solo servidor se requieren 8GB de RAM. Es recomendable contactar con Zimbra para solicitar información en caso contrario.
- GB de espacio en disco para instalaciones, actualizaciones y documentación varia relacionada.
- Espacio en disco extra reservado para el correo como tal. (Zimbra, 2018)

2. Entorno de Producción

- Intel/AMD 2.0 GHZ+ 64-bit CPU
- 8 GB en RAM para instalaciones de un solo servidor.

- Alrededor de 20 GB de espacio en disco para distintos archivos.
- Desactivar todos los firewalls de la máquina.

3. Sistemas Operativos Compatibles

- Red Hat® Enterprise Linux® 7 (64-bit)
- CentOS Linux® 7 (64-bit)
- Red Hat Enterprise Linux 6 (64-bit).
- CentOS Linux 6 (64-bit).
- Ubuntu 14.04 LTS Server Edition (64-bit)
- Ubuntu 12.04.4 LTS Server Edition (64-bit)

4. Otros

- Resolución mínima del monitor de administración: 1024 x 768
- Navegadores webs compatibles para los usuarios finales del cliente web: casi todos los navegadores populares de la actualidad. Internet Explorer es soportado desde su versión 8.x en adelante.
- Velocidad de Internet mínima: 128 kbps.
- Revisión de puertos de red para evitar conflictos con otros servicios en los equipos a configurar.
- Zimbra Mobile – Sistemas Operativos Compatibles: iOS 6 o superior; Android 6 o superior. (Zimbra, 2018)

3.3.3 Bot de Telegram

A continuación, se muestra el único requisito para la instalación de la app de Telegram:

- Sistema operativo iOS 8.0 o superior, Android 4.1 o superior, Windows Pone o versiones de escritorio disponibles en Windows 10, macOS y Linux
- Para la creación de un nuevo bot:
- Descargar e instalar la app de Telegram y seguir los pasos de configuración inicial.
- Iniciar una nueva conversación con @BotFather que es un bot que se encarga de la creación de bots.
- Esperar un momento y confirmar que se ha recibido un mensaje igual al que se muestra a continuación: (Telegram)

```
You: /newbot
BotFather: Alright, a new bot. How are we going to call it? Please
choose a name for your bot. It must end in `bot`. Like this, for
example: TetrisBot or tetris_bot.
You: SMTFirstBot
BotFather: Done! Congratulations on your new bot. You will find it at
telegram.me/SMTFirstBot. You can now add a description, about section
and profile picture for your bot, see /help for a list of commands.
By the way, when you've finished creating your cool bot, ping our Bot
Support if you want a better username for it. Just make sure the bot
is fully operational before you do this.

Use this token to access the HTTP API:
2764xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

Figura 4. Creación de un nuevo bot. Fuente: Manual de Telegram.

Una vez que el bot esté creado se necesita configurarlo de acuerdo a las necesidades correspondientes. La API del bot es una interfaz basada en HTTP, creada para desarrolladores interesados en construir bots para Telegram. (Telegram).

Capítulo 4: Implementación

4.1 Metodología elegida y aclaraciones

La metodología que se plantea para este proyecto es la que se conoce como Cascada. Esta fue una de las primeras metodologías de implementación de proyectos que fueron establecidas. A pesar de su simpleza en comparación con metodologías más modernas, Cascada sigue siendo usada en proyectos cortos y con una linealidad marcada, lo que se ajusta a lo que el presente estudio requiere.

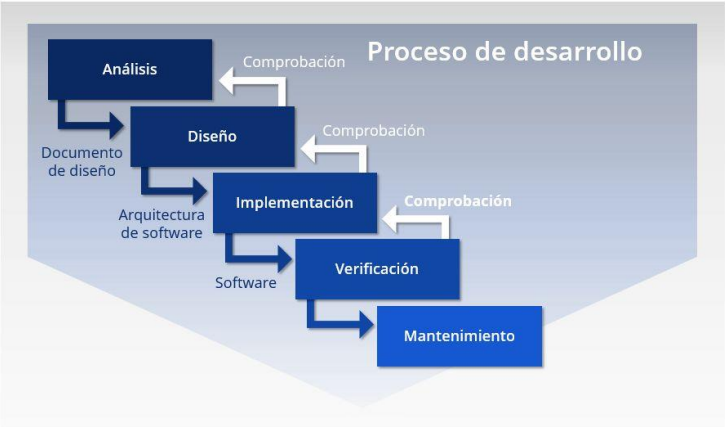


Figura 5. Metodología Cascada. Fuente: Waterfall Model of System Development. (2010)

Esta metodología se basa en el cumplimiento de ciertas fases o etapas de manera consecutiva, en un orden detallado con anterioridad, que usualmente va del Análisis de Requerimientos hasta Mantenimiento. Sin embargo, la aplicación de este marco de trabajo no se hará de manera estricta, sino más bien con algunas concesiones:

- La aplicación de Cascada se realizará sobre la instalación de cada una de las herramientas en un orden establecido. Primero se empezará con Zabbix, luego Zimbra y posteriormente el resto.
- La fase de Requerimientos comprende todo lo desarrollado en el Capítulo 3 del presente estudio.

- La fase de Diseño se establecerá como la creación de un diagrama de la arquitectura de red y se omitirán diagramas comunes debido a la naturaleza de la codificación necesaria.
- La fase de Implementación arranca con el proceso de instalación de las herramientas necesarias, para luego establecer los componentes programáticos de cada sección.
- La fase de Pruebas se realiza para cada herramienta y posteriormente se realizará para la integración de cada una de las partes en el proyecto global.

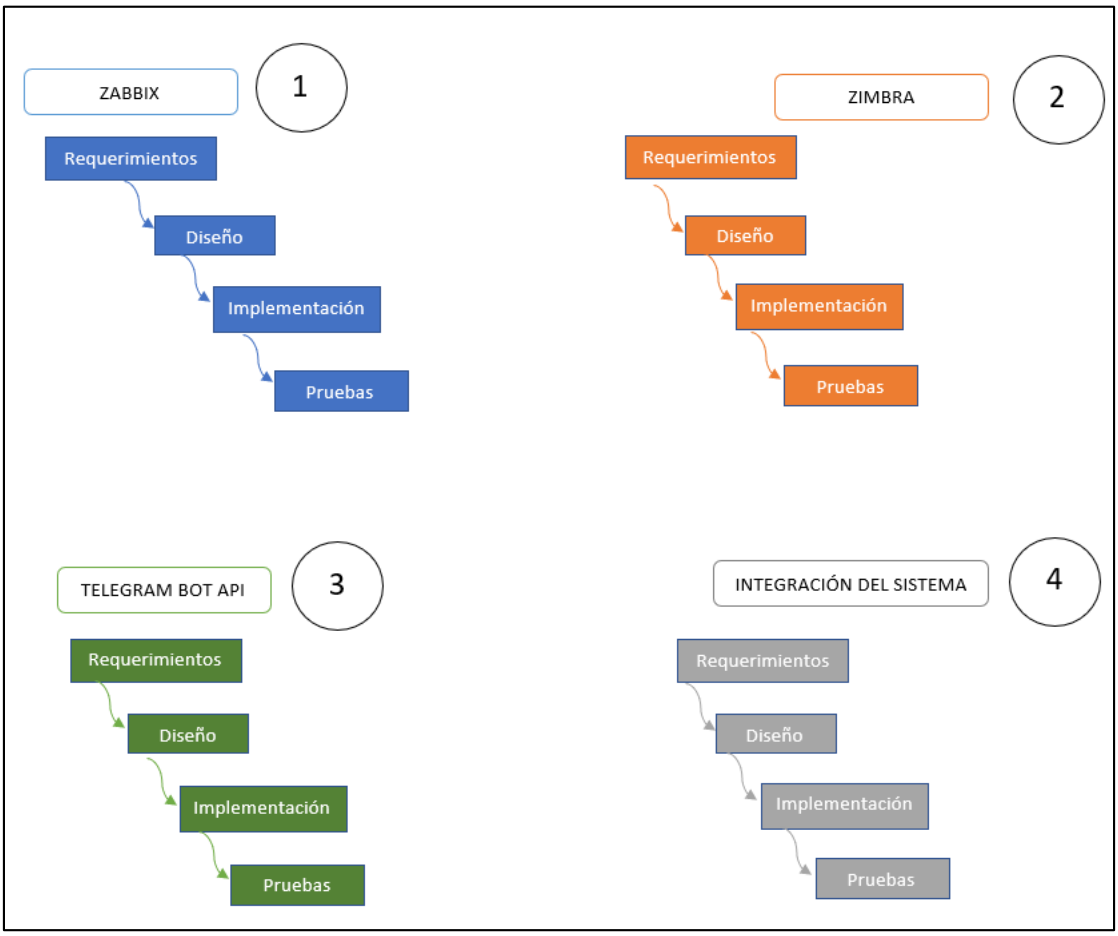


Figura 4. Esquema general de la implementación de Cascada en este proyecto.

4.2 Aplicación de Cascada con Zabbix.

4.2.1 Análisis y Diseño

La necesidad de usar esta herramienta, así como las respectivas descripciones de sus requerimientos fueron tratadas en capítulos anteriores, por lo que no serán repasadas en este apartado nuevamente.

4.2.2 Implementación: Instalación y Configuración de Zabbix

Previo a la configuración de Zabbix, se necesitan algunas herramientas para trabajar correctamente.

En primer lugar, es necesaria la instalación del motor de base de datos MariaDB

```
[root@localhost ~]# yum install mariadb-server -y
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * base: mirror.uta.edu.ec
 * extras: mirror.uta.edu.ec
 * updates: mirror.uta.edu.ec
Resolving Dependencies
--> Running transaction check
---> Package mariadb-server.x86_64 1:5.5.64-1.el7 will be installed
--> Processing Dependency: mariadb(x86-64) = 1:5.5.64-1.el7 for package: 1:maria
db-server-5.5.64-1.el7.x86_64
--> Processing Dependency: perl-DBI for package: 1:mariadb-server-5.5.64-1.el7.x
86_64
--> Processing Dependency: perl-DBD-MySQL for package: 1:mariadb-server-5.5.64-1
.el7.x86_64
--> Processing Dependency: perl(DBI) for package: 1:mariadb-server-5.5.64-1.el7.
x86_64
--> Running transaction check
---> Package mariadb.x86_64 1:5.5.64-1.el7 will be installed
---> Package perl-DBD-MySQL.x86_64 0:4.023-6.el7 will be installed
---> Package perl-DBI.x86_64 0:1.627-4.el7 will be installed
--> Processing Dependency: perl(RPC::PlServer) >= 0.2001 for package: perl-DBI-1
.627-4.el7.x86_64
--> Processing Dependency: perl(RPC::PlClient) >= 0.2000 for package: perl-DBI-1
```

Figura 5. Instalación MariaDB.

Luego se procede con la instalación del servidor web Apache

```
[root@localhost ~]# yum install httpd -y
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * base: mirror.uta.edu.ec
 * extras: mirror.uta.edu.ec
 * updates: mirror.uta.edu.ec
Package httpd-2.4.6-90.el7.centos.x86_64 already installed and latest version
Nothing to do
```

Figura 6. Instalación de Apache

Instalación de dependencias PHP

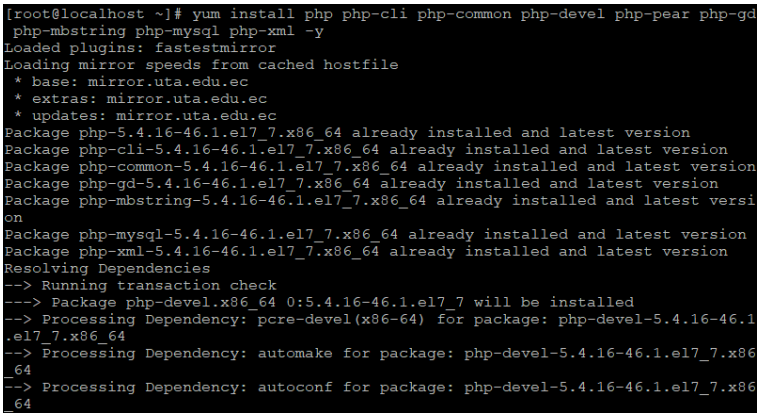


Figura 7. Dependencias PHP

Posteriormente, se debe iniciar Apache y MariaDB en el arranque, como muestra la siguiente figura:

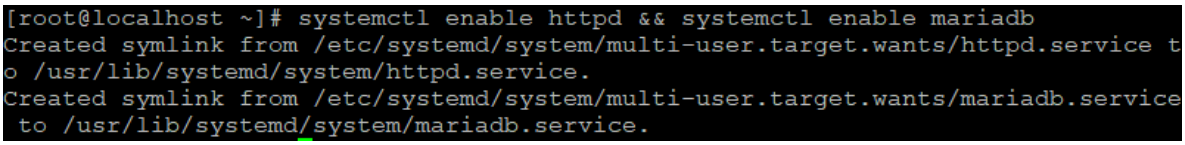


Figura 8. Iniciación de Apache y MariaDB en el arranque

Luego se deben iniciar los servicios de Apache y MariaDB

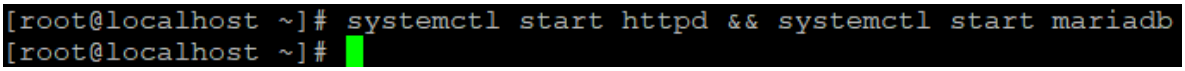


Figura 9. Iniciación de servicios de Apache y MariaDB

La configuración inicial MariaDB debe ser realizada tal y como lo muestra la figura 6*:

```
[root@localhost ~]# mysql_secure_installation

NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB
      SERVERS IN PRODUCTION USE!  PLEASE READ EACH STEP CAREFULLY!

In order to log into MariaDB to secure it, we'll need the current
password for the root user.  If you've just installed MariaDB, and
you haven't set the root password yet, the password will be blank,
so you should just press enter here.

Enter current password for root (enter for none):
OK, successfully used password, moving on...

Setting the root password ensures that nobody can log into the MariaDB
root user without the proper authorisation.

Set root password? [Y/n] y
New password:
Re-enter new password:
Password updated successfully!
Reloading privilege tables..
 ... Success!
```

Figura 10. Configuración inicial de MariaDB

4.2.3 Implementación: Instalación de Zabbix Server

Como primer paso se debe agregar el repositorio de Zabbix para CentOS 7, tal y como se muestra en la siguiente imagen:

```
[root@localhost ~]# rpm -ivh https://repo.zabbix.com/zabbix/3.4/rhel/7/x86_64/za
bbix-release-3.4-2.el7.noarch.rpm
Retrieving https://repo.zabbix.com/zabbix/3.4/rhel/7/x86_64/zabbix-release-3.4-2
.el7.noarch.rpm
warning: /var/tmp/rpm-tmp.HqiTEd: Header V4 RSA/SHA512 Signature, key ID a14fe59
1: NOKEY
Preparing...                          ################################# [100%]
Updating / installing...
   1:zabbix-release-3.4-2.el7         ################################# [100%]
```

Figura 11. Repositorio de Zabbix para CentOS

Una vez realizado esto, se procede con la instalación de la herramienta.

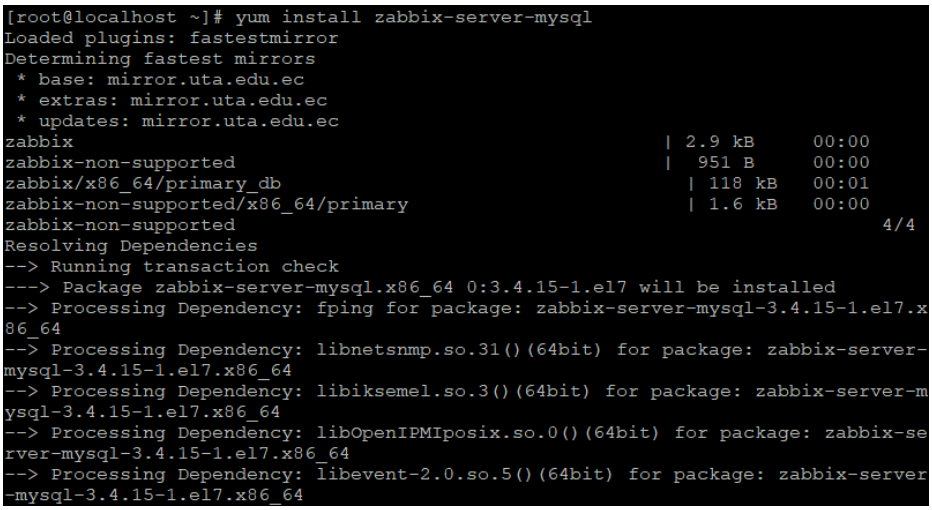


Figura 12. Instalación de Zabbix

A continuación, se procede a instalar la parte visual de Zabbix, es decir, el frontend.

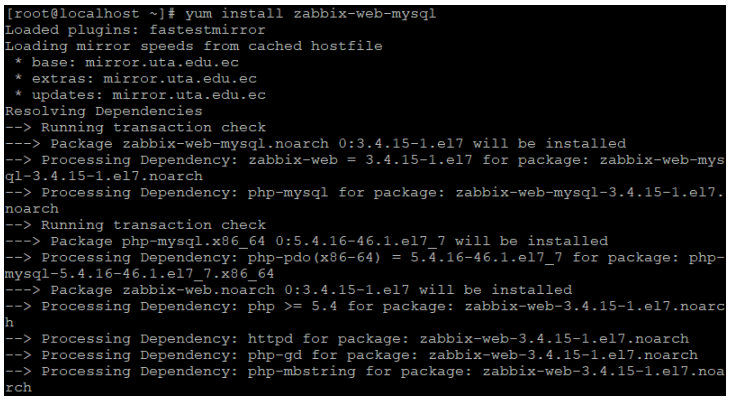


Figura 13. Instalación del frontend de Zabbix

Una vez completada dicha instalación se procede a crear una base de datos usando MariaDB que fue instalado previamente. Luego, se procede a la creación de un usuario en el motor de base de datos y se le concede acceso a la base creada con anterioridad. Finalmente, se procede a importar el esquema de la base de datos. Este proceso se muestra en las Figuras 10, 11 y 12.

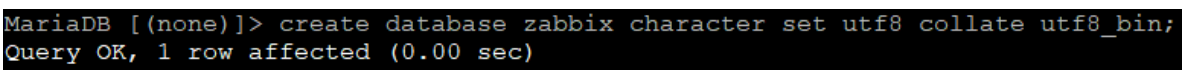


Figura 14. Creación de la base de datos usando MariaDB

```
MariaDB [(none)]> grant all privileges on zabbix.* to zabbix@localhost identifie
d by '          ';
Query OK, 0 rows affected (0.00 sec)
```

Figura 15. Creación de usuario con acceso a la base creada.

```
[root@localhost ~]# zcat /usr/share/doc/zabbix-server-mysql*/create.sql.gz | mys
ql -uzabbix -p zabbix
Enter password:
```

Figura 16. Importación del esquema de la base de datos.

Una vez configurada la estructura de la base de datos, se debe modificar el archivo de configuración de Zabbix Server con la contraseña del usuario de la base de datos creada, tal y como se muestra en la Figura 12 y 13.

```
[root@localhost ~]# vi /etc/zabbix/zabbix_server.conf
```

Figura 17. Accediendo al archivo de configuración de Zabbix

```
#       Comment this line if no password is used.
#
# Mandatory: no
# Default:
DBPassword=
```

Figura 18. Añadiendo la contraseña del usuario de la base de datos creada.

Una vez realizado esto, se procede a habilitar el inicio del Servicio Zabbix Server en el arranque:

```
[root@localhost ~]# systemctl enable zabbix-server
Created symlink from /etc/systemd/system/multi-user.target.wants/zabbix-server.s
ervice to /usr/lib/systemd/system/zabbix-server.service.
```

Figura 19. Habiltando el servicio de Zabbix en el arranque.

Se inicia el servicio de Zabbix:

```
[root@localhost ~]# systemctl start zabbix-server
```

Figura 20. Iniciación del servicio de Zabbix.

Antes de continuar, se debe realizar un cambio de zona horaria en PHP, como se muestra en la siguiente imagen:

```
#
# Zabbix monitoring system php web frontend
#

Alias /zabbix /usr/share/zabbix

<Directory "/usr/share/zabbix">
    Options FollowSymLinks
    AllowOverride None
    Require all granted

    <IfModule mod_php5.c>
        php_value max_execution_time 300
        php_value memory_limit 128M
        php_value post_max_size 16M
        php_value upload_max_filesize 2M
        php_value max_input_time 300
        php_value max_input_vars 10000
        php_value always_populate_raw_post_data -1
        php_value date.timezone America/Guayaquil
    </IfModule>
</Directory>

-- INSERT --
```

Figura 21. Cambio de zona horaria en PHP

A continuación, se debe realizar la configuración de SELinux:

```
[root@localhost ~]# setsebool -P httpd_can_connect_zabbix on
```

Figura 22. Configuración de SELinux

Lo que resta es la configuración de Zabbix mediante la web.



Figura 23. Configuración de Zabbix vía web – Primera pantalla.

El primer paso del instalador es el de revisar los prerrequisitos, que fueron instalados previamente, es decir, no se debería perder tiempo en este apartado.

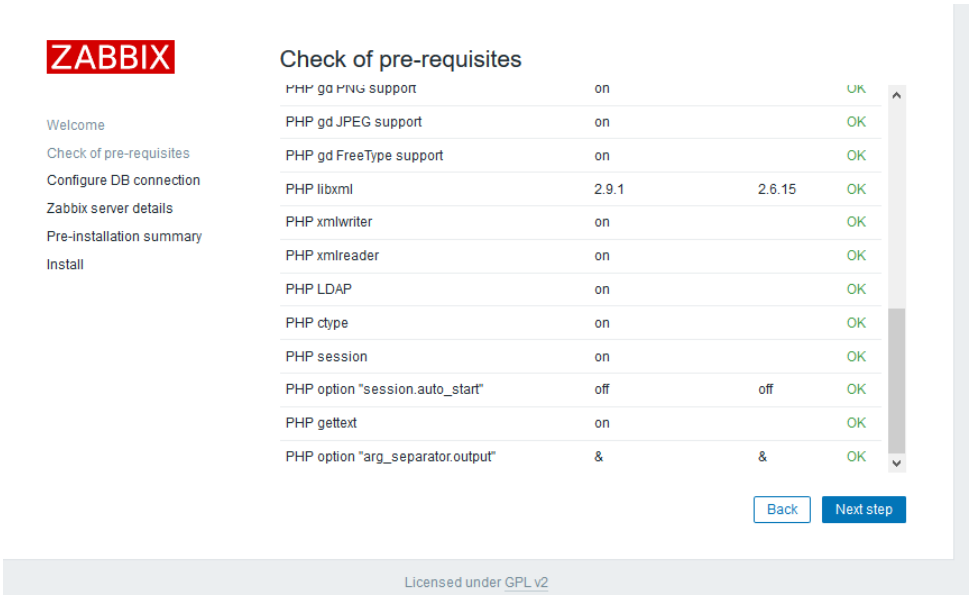


Figura 24. Configuración de Zabbix vía web - Revisión de prerrequisitos

El siguiente paso consiste en la configuración de la Base de Datos y su respectiva conexión.

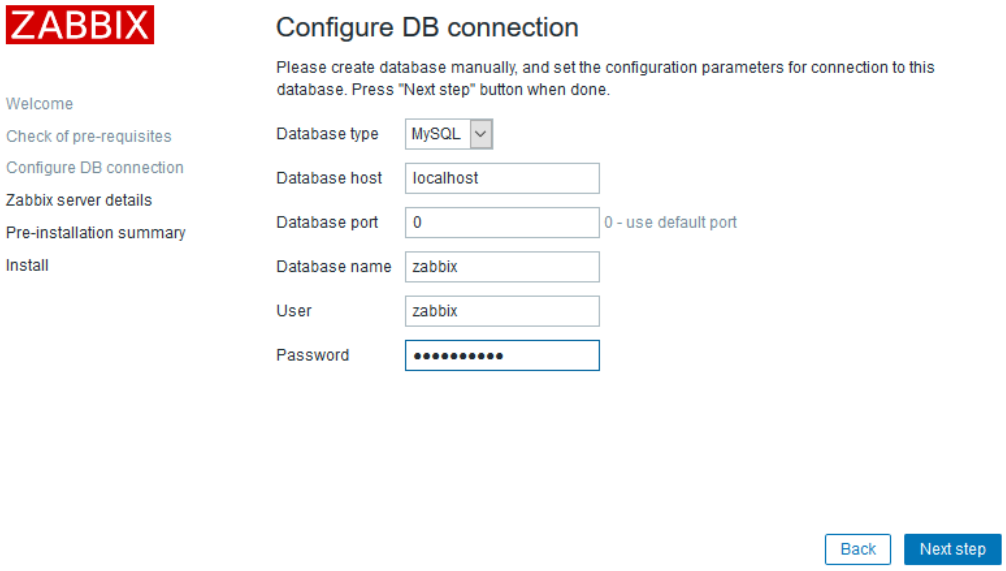


Figura 25. Configuración de Zabbix vía web - Conexión con la Base de Datos

El siguiente paso es la instalación per sé que culmina con una pantalla igual a la de la mostrada en la Figura 21:

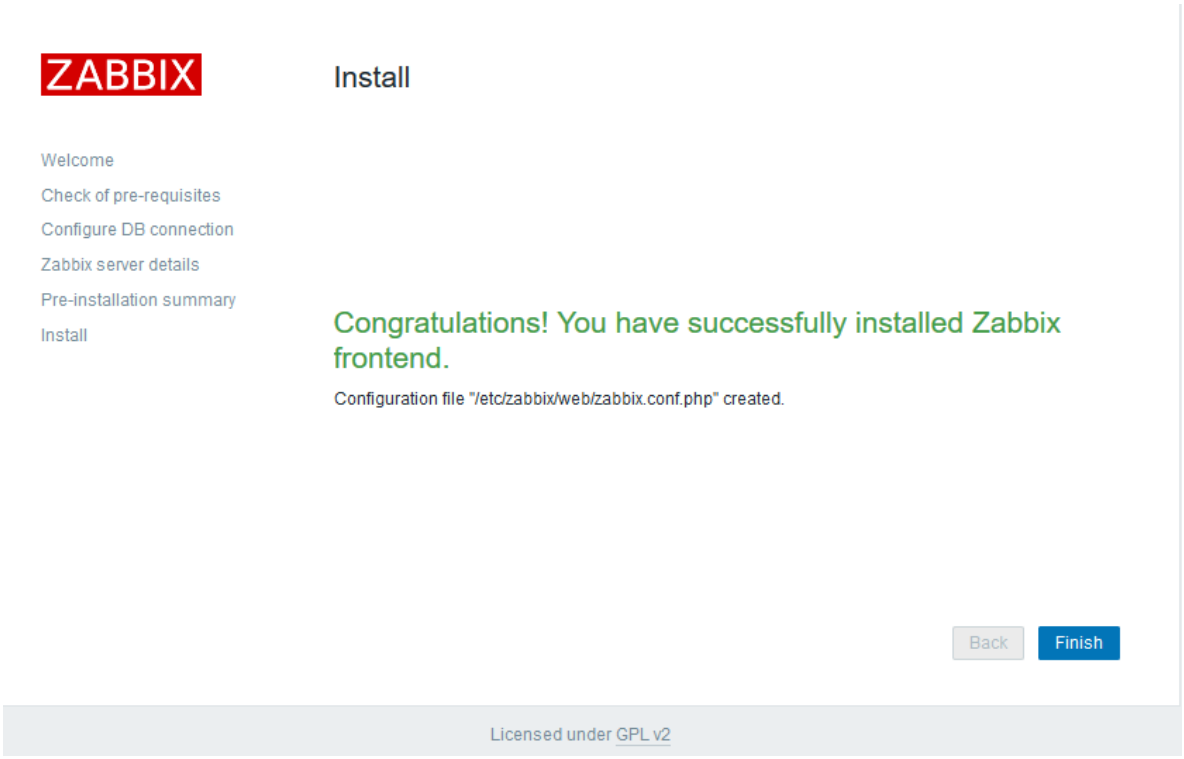


Figura 26. Configuración de Zabbix vía web - Finalización del proceso de instalación

4.2.4 Pruebas: Prueba de Monitoreo de Equipo

Para las pruebas de monitoreo con Zabbix, es necesario configurar un host en la herramienta. En este caso, el nombre elegido para el host es el de "Securesoft -Sensor Humedad – Temperatura", tal y como se muestra en la siguiente figura:

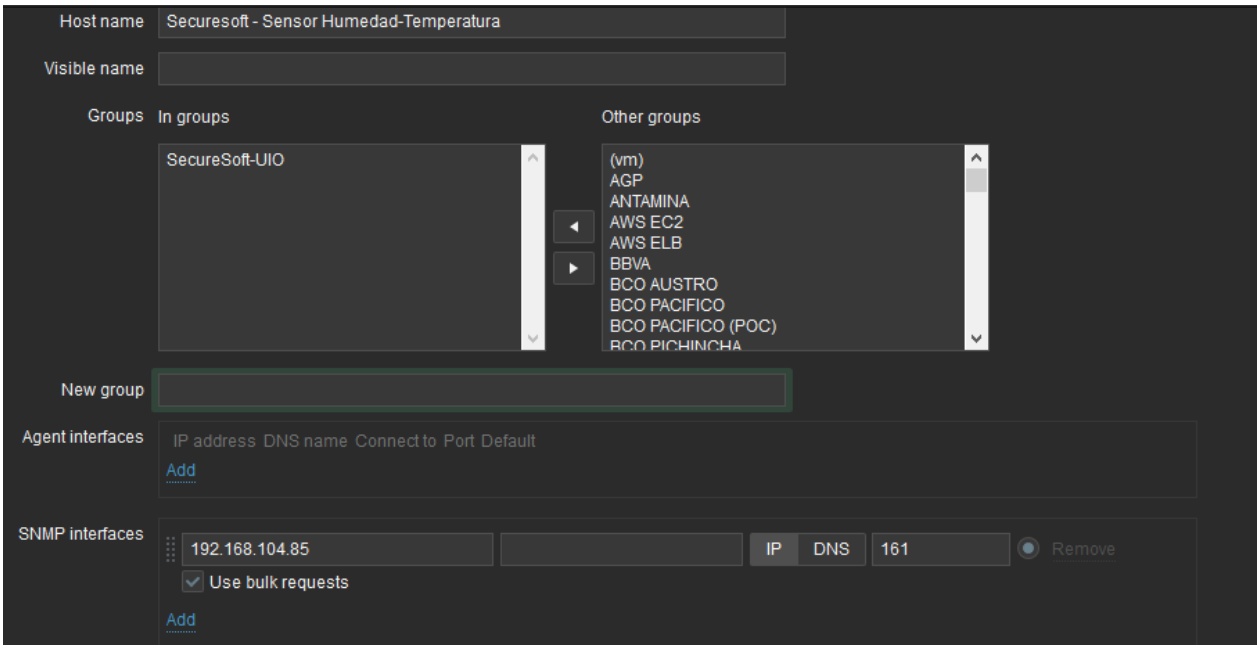


Figura 27. Configuración de un host en Zabbix.

Posteriormente, se puede realizar el monitoreo de la manera esperada y sin ningún contratiempo.

Template Module Generic SNMPv3: Device name		system.name	3600	14d		SNMPv3 agent	General	Enabled
Template Module Generic SNMPv3: Device uptime V3	Triggers 1	system.uptime	30	14d	0d	SNMPv3 agent	Status	Enabled
Template AKCP: Temperatura	Triggers 1	akcp.temperatura	60s	90d	365d	SNMPv3 agent	temperature	Enabled
Template AKCP: Humedad	Triggers 2	akcp.humedad	60s	90d	365d	SNMPv3 agent	humidity	Enabled

Figura 28. Monitoreo con Zabbix (1)

General (4 Items)		
Device contact details	2019-11-12 11:14:45	System Contact
Device description	2019-11-12 11:14:45	SP2+ 1.0.4615 Dec 3 2018 1...
Device location	2019-11-12 11:14:45	System Location
Device name	2019-11-12 11:14:45	System Name
humidity (1 Item)		
Humedad	2019-11-12 12:09:53	47
Status (4 Items)		
Device uptime V3	2019-11-12 12:09:53	3 days, 15:01:35
ICMP loss	2019-11-12 12:09:45	0 %
ICMP ping	2019-11-12 12:09:45	Up (1)
ICMP response time	2019-11-12 12:09:45	0.7ms
temperature (1 Item)		
Temperatura	2019-11-12 12:09:53	22

Figura 29. Monitoreo con Zabbix (2)

4.3 Aplicación de Cascada con Zimbra

4.3.1 Análisis y Diseño

La necesidad de usar esta herramienta, así como las respectivas descripciones de sus requerimientos fueron tratadas en capítulos anteriores, por lo que no serán repasadas en este apartado nuevamente.

4.3.2 Implementación: Instalación y Configuración de Zimbra

Como requisito se tiene el hecho de que el sistema operativo sobre el que se trabaje debe ser Linux, en este caso se procede a la instalación de Ubuntu Server 16.04.

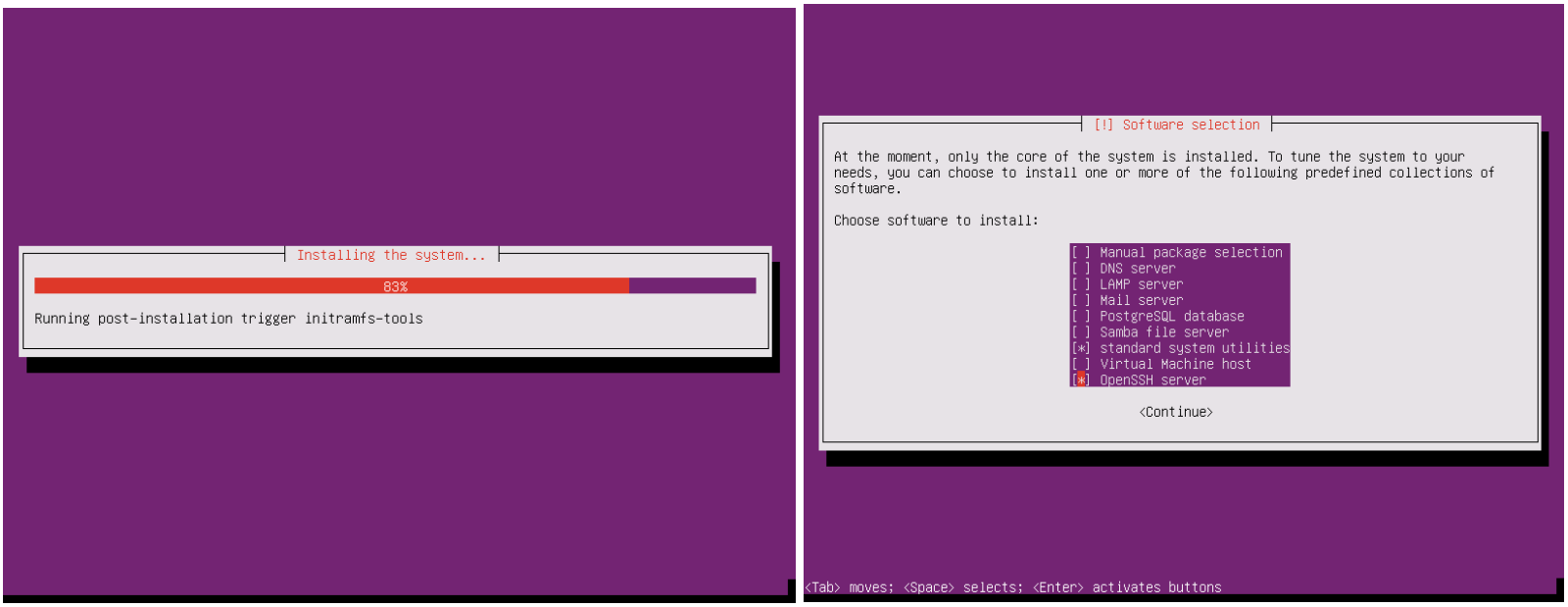


Figura 30. Instalación de Ubuntu Server

A continuación, se deben realizar las configuraciones de registros DNS en un servidor externo.

Type	Host	Value	TTL
A Record	@	179.49.60.243	Automatic
[redacted]			
[redacted]			
CNAME Record	mail	securesoftec.com.	Automatic
[redacted]			

Figura 31. Configuración de DNS en servidor externo.

El siguiente paso es el de instalar todas las dependencias necesarias para este proceso:

```
root@ubuntu-srv:~# apt install libperl5.22 libgmp10 libaio1 unzip pax sysstat sqli
te3
Reading package lists... Done
Building dependency tree
Reading state information... Done
libgmp10 is already the newest version (2:6.1.0+dfsg-2).
libperl5.22 is already the newest version (5.22.1-9ubuntu0.6).
The following additional packages will be installed:
  libsensors4
Suggested packages:
  lm-sensors sqlite3-doc isag zip
The following NEW packages will be installed:
  libaio1 libsensors4 pax sqlite3 sysstat unzip
0 upgraded, 6 newly installed, 0 to remove and 0 not upgraded.
Need to get 1,030 kB of archives.
After this operation, 3,788 kB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Figura 32. Instalación de dependencias

Lo que viene a continuación es la configuración de una IP estática, de la manera en que se muestra en la siguiente figura:

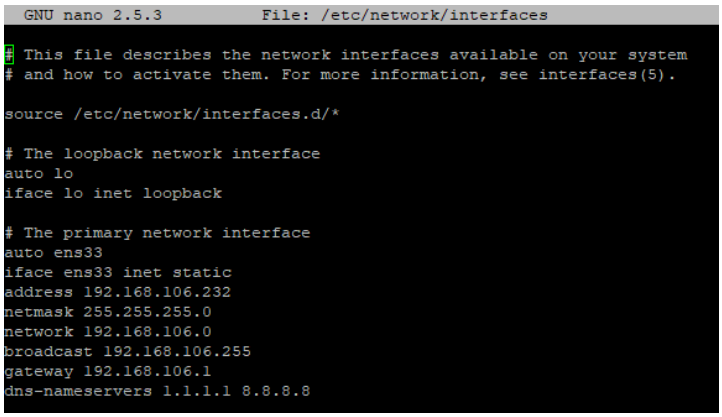


Figura 33. Configuración IP estática

En este punto es necesario reiniciar el servicio de red (networking), mediante el comando mostrado en la siguiente figura:

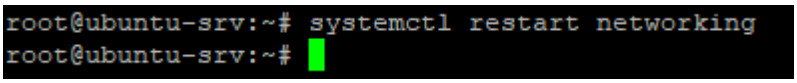


Figura 34. Reiniciando el servicio de red en Ubuntu.

Como recomendación de Zimbra, se procede a remover Apparmor, que es un módulo de seguridad del kernel de Linux.

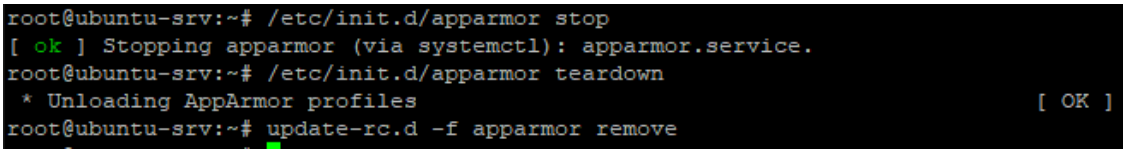


Figura 35. Removiendo Apparmor

Posteriormente se debe instalar un servidor DNS, como se muestra en la siguiente figura:

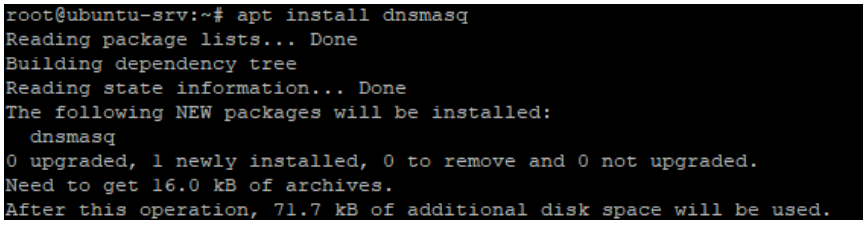


Figura 36. Instalación servidor DNS

Asimismo, se debe configurar un servidor DNS reverso:

```
 GNU nano 2.5.3            File: /etc/dnsmasq.conf

#txt-record=example.com,"v=spf1 a -all"

#Example zeroconf
#txt-record=_http._tcp.example.com,name=value,paper=A4

# Provide an alias for a "local" DNS name. Note that this _only_ works
# for targets which are names from DHCP or /etc/hosts. Give host
# "bert" another name, bertrand
#cname=bertand,bert

# For debugging purposes, log each DNS query as it passes through
# dnsmasq.
#log-queries

# Log lots of extra information about DHCP transactions.
#log-dhcp

# Include another lot of configuration options.
#conf-file=/etc/dnsmasq.more.conf
#conf-dir=/etc/dnsmasq.d

# Include all the files in a directory except those ending in .bak
#conf-dir=/etc/dnsmasq.d,.bak

# Include all files in a directory which end in .conf
#conf-dir=/etc/dnsmasq.d/,*.conf

server=1.1.1.1
server=8.8.8.8
domain=securesoftec.com
mx-host=securesoftec.com,mail.securesoftec.com,5
listen-address=127.0.0.1
```

Figura 37. Configuración servidor DNS reverso

El siguiente paso es la validación de resolución de dominios tanto de manera externa como interna. Esto se muestra en las siguientes figuras:

```
root@mail:~# dig securesoftec.com mx

; <<>> DiG 9.10.3-P4-Ubuntu <<>> securesoftec.com mx
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 39562
;; flags: qr aa rd ra ad; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 2

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1280
;; QUESTION SECTION:
;securesoftec.com.              IN      MX

;; ANSWER SECTION:
securesoftec.com.       0       IN      MX      5 mail.securesoftec.com.

;; ADDITIONAL SECTION:
mail.securesoftec.com.  0       IN      A       192.168.106.232

;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Tue Nov 12 11:40:15 -05 2019
;; MSG SIZE  rcvd: 98
```

Figura 38. Validación de resolución de dominios de manera interna.

```
C:\Users\didac>nslookup mail.securesoftec.com
Server:  win-og4r3dejb60.securesoftcorp.ec
Address:  192.168.104.92

Non-authoritative answer:
Name:    securesoftec.com
Address:  179.49.60.243
Aliases:  mail.securesoftec.com
```

Figura 39. Validación de resolución de dominios de manera externa.

Lo que resta es la instalación per se de Zimbra, empezando por la descompresión de los archivos de instalación:

```
tar -xzvf zcs-8.8.7_GA_1964.UBUNTU16_64.20180223145016.tgz
```

Figura 40. Descompresión de archivos de instalación.

Posteriormente, se debe navegar hasta el directorio descomprimido e instalar la herramienta.

```
cd zcs-8.8.7_GA_1964.UBUNTU16_64.20180223145016/
ls
sudo ./install.sh
```

Figura 41. Instalación de la herramienta desde el directorio descomprimido.

Para validar la instalación se comprueba el acceso a la consola de administración de la herramienta:

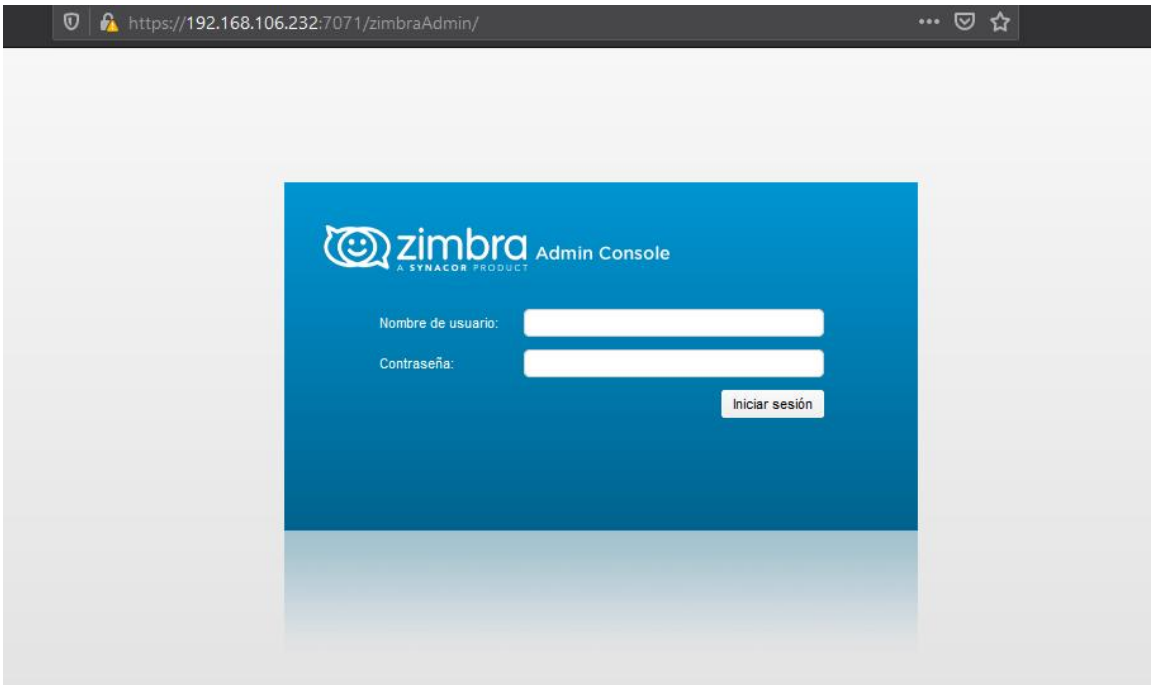


Figura 42. Validación de la instalación.

4.3.3 Pruebas: Prueba de Envío de Correo electrónico.

Como se muestra en las siguientes figuras, el envío del correo electrónico, utilizando la herramienta Zimbra, resulta exitoso. La prueba consistió en tanto el envío como recibimiento del correo electrónico.

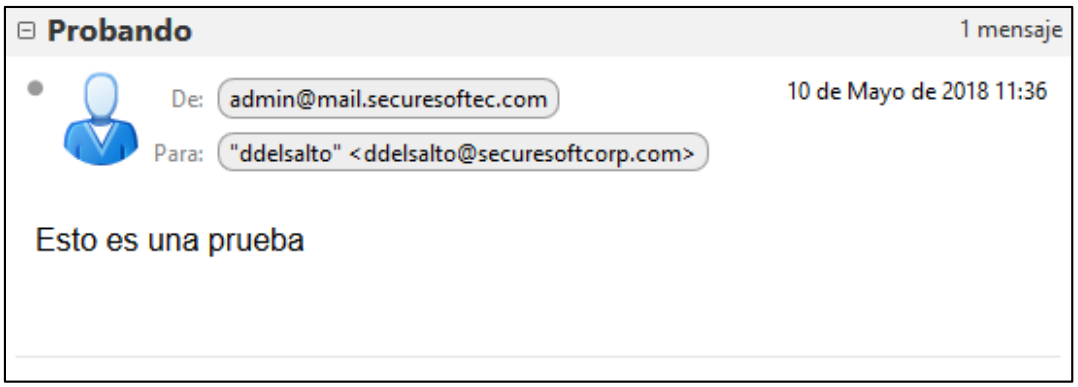


Figura 43. Comprobación del envío del correo electrónico.

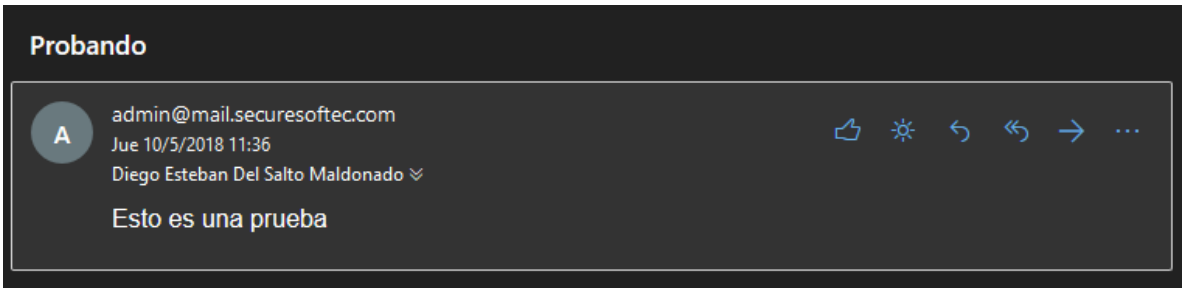


Figura 44. Comprobación del recibimiento del correo electrónico.

4.4 Aplicación de Cascada con Telegram

4.4.1 Análisis

La necesidad de usar esta herramienta, así como las respectivas descripciones de sus requerimientos fueron tratadas en capítulos anteriores, por lo que no serán repasadas en este apartado nuevamente.

4.4.2 Diseño: Diagrama de Paquetes

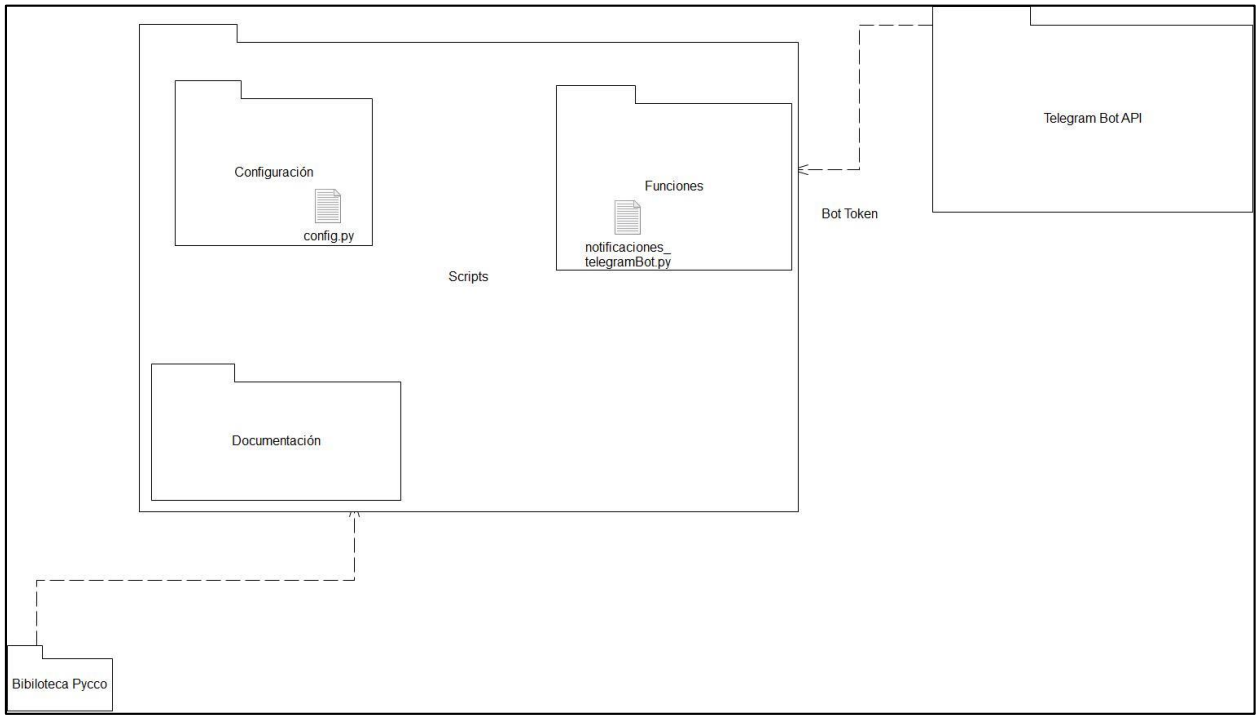


Figura 45. Diagrama de paquetes de la implementación

4.4.3 Implementación: Código Fuente

Tal y como ya se había establecido previamente, se usará Python para realizar la implementación necesaria del Bot de Telegram.

Como se muestra en la Figura 38, el primer paso es importar todas las librerías y módulos necesarios.

```
import sys
from datetime import datetime

import telebot as bot

from config import botToken as token
```

Figura 46. Importación de librerías y módulos necesarios

Una vez que todos los recursos están disponibles para trabajar, se procede a construir una función que permita enviar el mensaje en sí mismo. Para esto, primero es necesario recibir el token del bot (que es una cadena alfanumérica única para cada bot. Asimismo, se requiere el destinatario y tanto el título como el cuerpo del mensaje a enviar. Esto luego se envía a la API de Telegram. El procedimiento se observa en la Figura 39.

```
# === enviarMsg ===
def enviarMsg(t,d,s,m):
    zabbixBot = bot.TeleBot(t)
    try:
        zabbixBot.send_message(d,s + '\n' + m, disable_web_page_preview=True, parse_mode='HTML')
    except Exception as e:
        escribirLog(e)
```

Figura 47. Función para enviar el mensaje

En las últimas líneas de la función de envío del mensaje, se muestra un llamado a otra función denominada "escribirLog". Esta función sirve para manejar posibles errores en el envío del mensaje. Su construcción se puede observar en la Figura 40.

```
# === escribirLog ===
def escribirLog(msg):
    ahora = datetime.now()
    fecha = ahora.strftime("%d%m%Y%H%M%S")
    try:
        log = open("errores.log".format(fecha),"a+")
        log.write("{} : ERROR : {}\n".format(fecha,msg))
    except Exception:
        pass
    finally:
        log.close()
```

Figura 48. Función para lidiar con posibles errores al enviar el mensaje.

Finalmente, se construye la función "main" donde se escribe la lógica principal para la ejecución de este script. En la Figura 41 se muestra cómo se construye el mensaje definiendo los parámetros necesarios y luego se llama a la función para enviar dicho mensaje.

```
# === main ===
def main():
    destino = sys.argv[1]
    titulo = sys.argv[2]
    mensaje = sys.argv[3].replace('/n','\n')
    enviarMsg(token,destino,titulo,mensaje)
```

Figura 49. Función main del script.

También es importante mostrar el archivo de configuración, que es el que contiene el token del bot, necesario para la comunicación con la API.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

############################################################
## Author: Diego Del Salto
## Email: ddelsalto315@puce.edu.ec
############################################################

"""
Archivo de configuración
"""

# Token del bot de Telegram
botToken='1041054394:AAG5C1cp8VucuD9nqYOH2Gt7j1eB3NxU5cM'
```

Figura 50. Archivo de configuración.

4.4 Pruebas: Envío de mensajes de pruebas a cuenta personal de Telegram

Para esta fase de pruebas se creará un bot con el cual se pueda comprobar si es que es la implementación fue exitosa. Para esto, se debe escribir al BotFather el cual es un bot que permite la creación de otros bots. El primer paso es darle un nombre al bot, en este caso se llamará Zabbix-SSEC. Luego, el BotFather nos provee del token para acceder a la API.

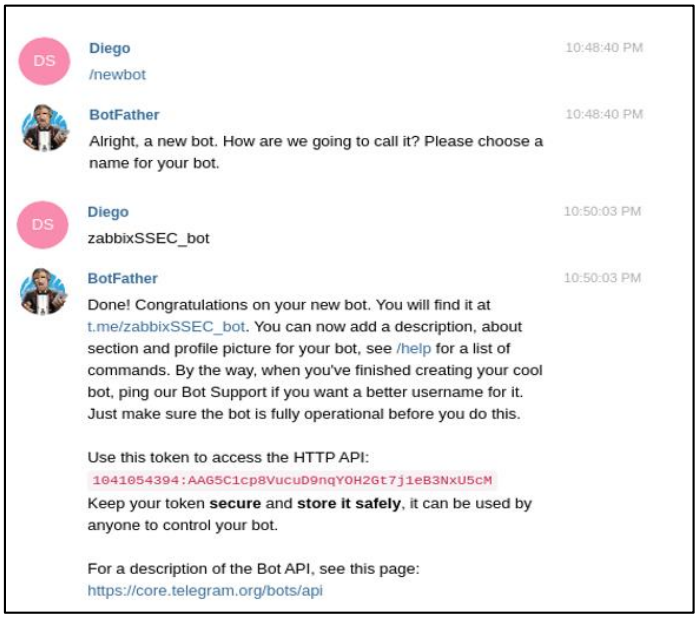


Figura 51. Creación de bot.

Luego, se realizan pruebas de script.

```
In [31]: runfile('/home/diego/Documents/notificaciones_telegramBot.py',
args='380858005 Probando Prueba', wdir='/home/diego/Documents')
Reloaded modules: config

In [32]: runfile('/home/diego/Documents/notificaciones_telegramBot.py',
args='38085800555 Probando Prueba', wdir='/home/diego/Documents')
Reloaded modules: config
```

Figura 52. Pruebas de script

Se evidencia la escritura en el archivo de log en caso de haber algún problema al conectarse con la API

errores.log ×

```
17112019123952 : ERROR : A request to the Telegram API was unsuccessful. The server returned HTTP 400 Bad Request. Response body:
[{"ok":false,"error_code":400,"description":"Bad Request: chat not found"}]
```

Figura 53. Archivo de log

Se comprueba la recepción del mensaje de manera exitosa en el buzón de entrada.



Figura 54. Comprobación del envío del mensaje

De manera adicional, se procede a instalar una biblioteca de Python que sea útil para la generación de documentación. En este caso, se ha elegido Pycco.

```
diego@diego-HP-Pavilion-g4-Notebook-PC:~$ sudo -H pip install pycco
[sudo] password for diego:
DEPRECATION: Python 2.7 will reach the end of its life on January 1st, 2020. Ple
ase upgrade your Python as Python 2.7 won't be maintained after that date. A fut
ure version of pip will drop support for Python 2.7. More details about Python 2
 support in pip, can be found at https://pip.pypa.io/en/latest/development/relea
se-process/#python-2-support
Collecting pycco
  Downloading https://files.pythonhosted.org/packages/77/30/41f68164be44bdb5c35d
b0826451923497d319b16bb83084331ca9042679/Pycco-0.6.0-py2-none-any.whl
Collecting markdown
  Downloading https://files.pythonhosted.org/packages/c0/4e/fd492e91abdc2d2fcb70
ef453064d98068876207939 7f779758e055f6575/Markdown-3.1.1-py2.py3-none-any.whl (87
kB)
    |                                | 92kB 1.2MB/s
```

Y finalmente se crea un archivo de documentación usando la biblioteca instalada.

```
diego@diego-HP-Pavilion-g4-Notebook-PC:~/Documents$ pycco *.py -p
pycco: config.py -> docs/config.html
pycco: notificaciones_telegramBot.py -> docs/notificaciones_telegramBot.html
```

Figura 56. Generación de archivos de configuración en base a los scripts realizados.

4.5 Aplicación General de Cascada

4.5.1 Análisis

En este apartado se pretende usar la Metodología de Cascada para todo el proyecto en general, reuniendo todos los entregables obtenidos a lo largo de la aplicación individual de la metodología, con el objetivo de que, al realizar las pruebas de esta aplicación se encuentre una versión funcional de todo el sistema.

4.5.2 Diseño: Diagrama de Red

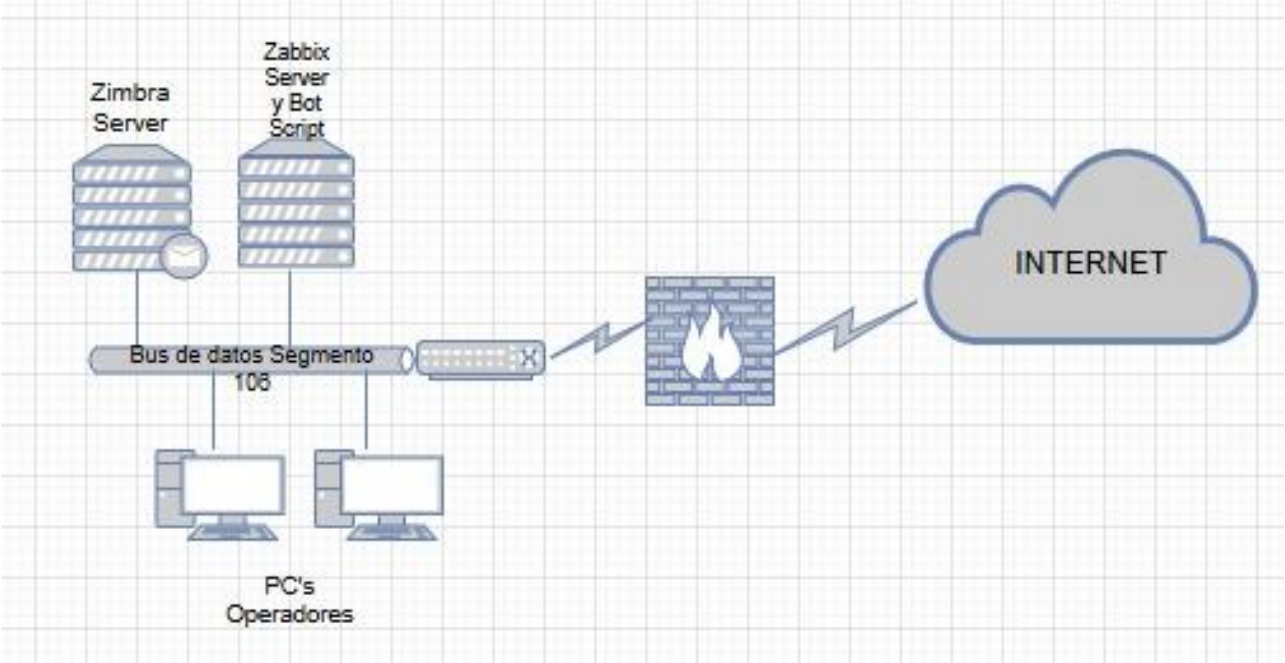


Figura 57. Diagrama de red del sistema

4.5.3 Integración: De los entregables de las 3 cascadas anteriores

En primer lugar, es necesario copiar el script desarrollado en el directorio de alertas de Zabbix

```
[root@zabbixProv notificaciones_TelegramBot]# cat /etc/zabbix/zabbix_server.conf | grep AlertScript
### Option: AlertScriptsPath
# AlertScriptsPath=${datadir}/zabbix/alertscripts
AlertScriptsPath=/usr/lib/zabbix/alertscripts
[root@zabbixProv notificaciones_TelegramBot]# ls
config.py  docs  notificaciones_telegramBot.py
[root@zabbixProv notificaciones_TelegramBot]# cp *.py /usr/lib/zabbix/alertscripts/ -v
'config.py' -> '/usr/lib/zabbix/alertscripts/config.py'
'notificaciones_telegramBot.py' -> '/usr/lib/zabbix/alertscripts/notificaciones_telegramBot.py'
```

Figura 58. Copiando el script desarrollado en el directorio de alertas de Zabbix.

Posteriormente se necesita realizar la configuración de medio de notificación por Telegram

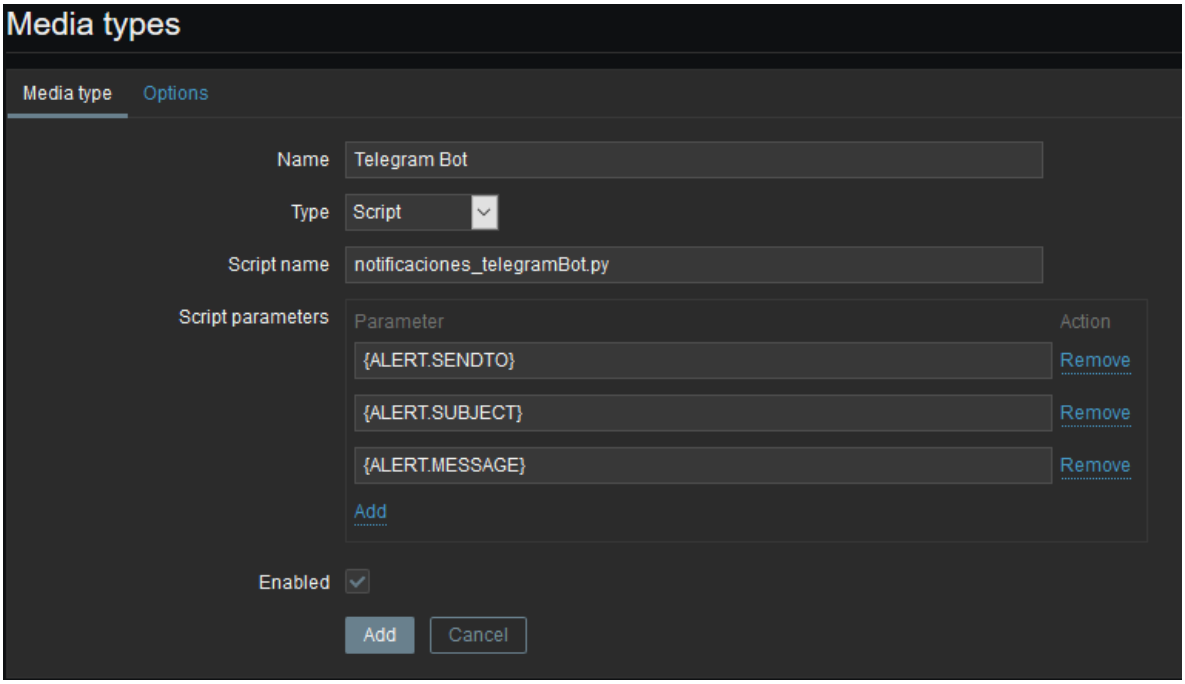


Figura 59. Configuración de medio de notificación por Telegram

Luego, se realiza la configuración de Telegram para envío de notificaciones para un usuario y se define el nivel de severidad de eventos que se van a notificar.

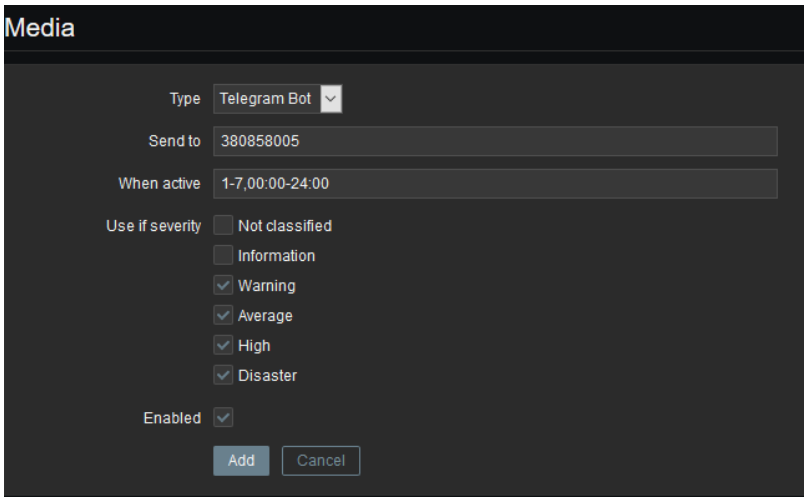


Figura 60. Configuración de Telegram para envío de notificaciones

Luego se realiza la configuración de medio de notificación por correo.

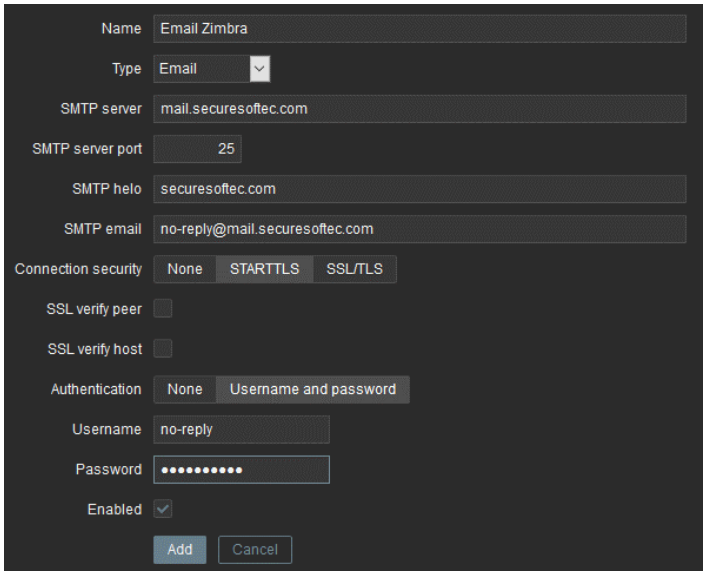


Figura 61. Configuración de medio de notificación por correo; configuración de medio de notificación por correo.

Luego, se realiza la configuración de correo para envío de notificaciones para un usuario y se define el nivel de severidad de eventos que se van a notificar.

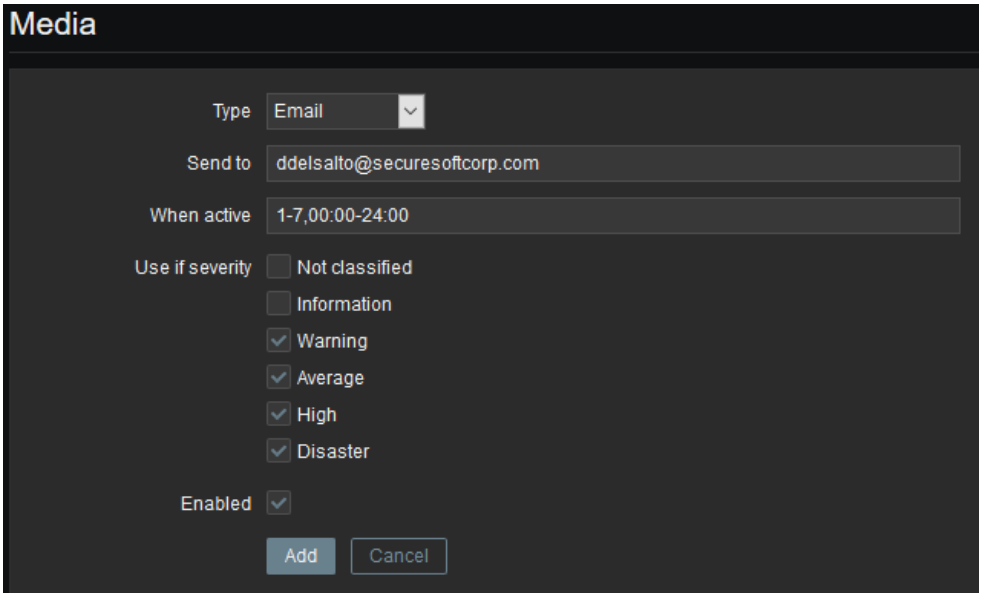


Figura 62. Configuración de correo para envío de notificaciones; nivel de severidad de eventos definido.

Luego, se realiza la configuración de acciones a realizar cuando se dispara una alerta en Zabbix. Las condiciones que se tienen que cumplir se muestran en la siguiente figura:

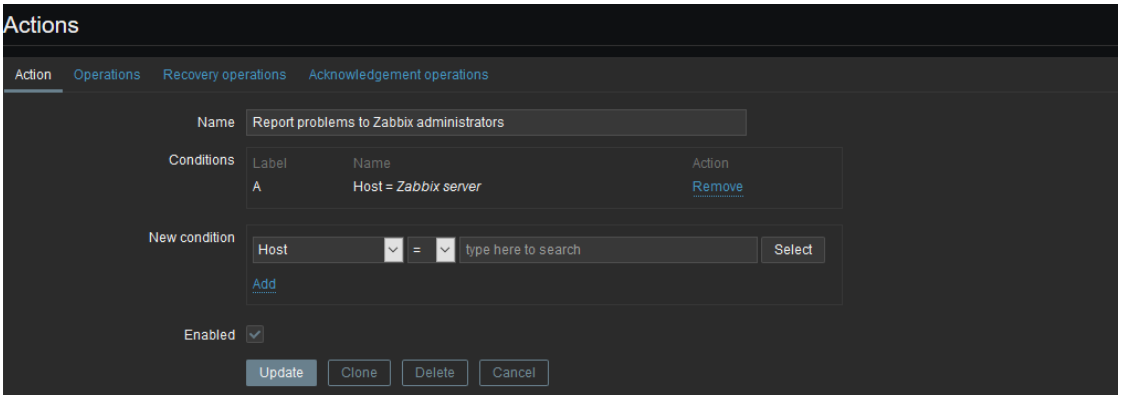


Figura 63. Condiciones por cumplir para disparar una alerta en Zabbix.

Se construye el prototipo del mensaje a enviar

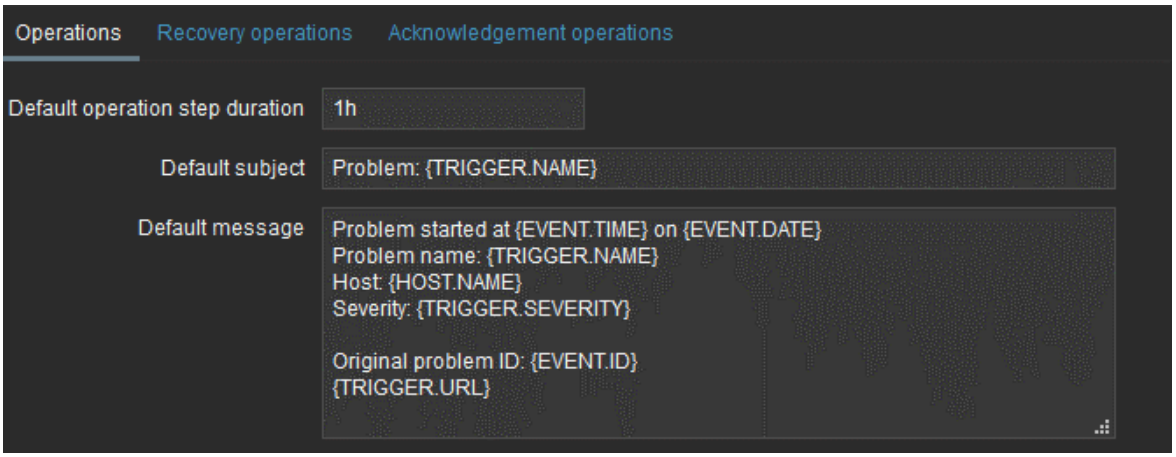


Figura 64. Prototipo del mensaje a enviar.

Luego se tienen que atener el detalle de las operaciones a realizar. Para este caso se define que cuando las condiciones se cumplan, se enviará un mensaje al grupo de "Zabbix administrators" por los medios configurados (Correo y Telegram).

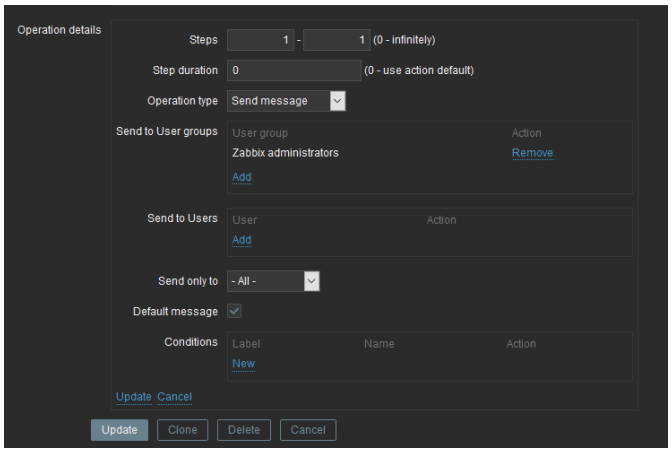


Figura 65. Detalle de las operaciones a realizar.

Se establecen los permisos de firewall para el envío de alertas al bot de Telegram



Figura 66. Permisos de firewall para envío de alertas al bot de Telegram.

4.5.4. Pruebas: Envío de notificación de Zabbix vía correo y Telegram

A continuación, se muestran una serie de capturas con las respectivas pruebas de que, en efecto, el sistema está funcionando como se esperaba.



Figura 67. Notificación de evento en Zabbix

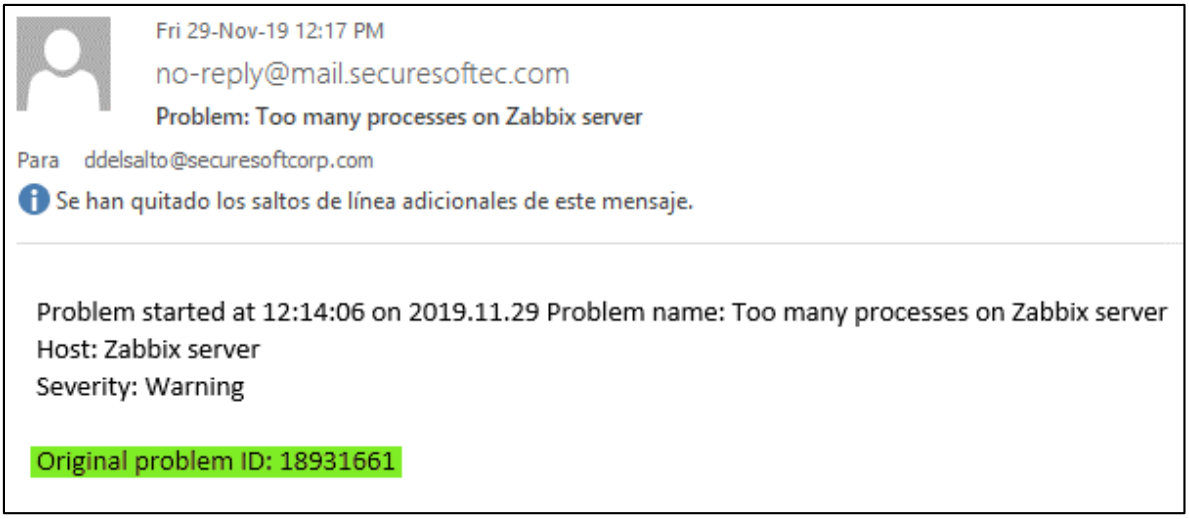


Figura 68. Envío de notificación por correo electrónico.

Envío notificación por Telegram



Figura 69. Envío de Notificación por Telegram

Capítulo 5: Conclusiones y Recomendaciones

5.1 Conclusiones

A continuación, se muestran las conclusiones obtenidas luego de la culminación del presente estudio:

- El envío de notificaciones a los equipos móviles de los operadores en la organización disminuyó de forma notable el tiempo de respuesta entre el descubrimiento de alguna novedad y la acción preventiva o correctiva de los empleados. A continuación, se muestra un cuadro comparativo de los tiempos de respuesta:

Tiempo de Respuesta (desde la aparición de algún evento hasta su corrección)	Antes de la aplicación del presente estudio	Después de la aplicación del presente estudio
Horas	Entre 72 y 120 horas aprox.	Entre 24 y 60 horas aprox.

- Al no necesitar un buzón de entrada para el envío de notificaciones, se bloquea el puerto externo de comunicación en el firewall, por lo cual el servidor de correo no se encuentra en una DMZ.
- El uso de una API facilita la construcción de software. La comunicación con la API provee de variadas funciones que pueden ser utilizadas con diversos fines por el programador, reduciendo así la complejidad de ciertos desarrollos.
- Se concluye que, al estar ambos servidores en el mismo segmento de red, no es necesario tener en cuenta permisos adicionales en el firewall.

- Fue necesario crear una política de SELinux para permitir que el servidor web apache pueda ser ejecutado desde un equipo diferente al localhost.
- Al tener históricos de rendimiento de la infraestructura tecnológica se puede planear y justificar la renovación y adquisición de equipos con los altos mandos de la organización de manera acertada.

5.2 Recomendaciones

- Se recomienda el uso de software libre tal y como se planteó desde el inicio de este estudio. Existen diversas ventajas con el uso de software libre, tal y como lo son las amplias comunidades de desarrolladores interesados en ayudar sin fines de lucro y las constantes mejoras y actualizaciones que los programas reciben por dichas comunidades.
- La elección de Pyhton como lenguaje de programación se mantiene como uno de los aciertos para conseguir lo propuesto en los objetivos iniciales, debido a la naturaleza detrás de Python que orienta a los desarrolladores a generar código corto y conciso, además de las múltiples bibliotecas que pueden ser usadas para distintos fines.
- Asimismo, es necesario mencionar las ventajas obtenidas con el uso del sistema operativo Ubuntu que permitió la ejecución de las herramientas y la configuración general de manera simple, sin ningún detalle adicional. Esto mismo se puede convertir en un problema si se trata de emular el sistema aquí desarrollado en ambientes distintos.
- Para la adquisición del dominio se recomienda que el servicio de hosting permita la redirección del puerto de email, así como la gestión de un servidor DNS.

- Se recomienda el uso de software de monitoreo o un servicio de monitoreo de la infraestructura de TI, si se tiene equipos críticos para el core del negocio, que almacene data histórica para el posterior análisis para la planeación de mantenimiento y mejoramiento de la infraestructura tecnológica.
- Se recomienda dejar habilitado SELinux para CentOS, ya que este se encarga de definir controles de accesos para aplicaciones, procesos y archivos en el sistema. Lo que brinda una capa más de seguridad.
- Se recomienda no enviar correos a otras entidades de correo oficiales (ej: Yahoo, Gmail, etc) si no se tiene bien configurado el registro MX y el DNS reverso, ya que esto puede causar que la IP pública ingrese en listas negras.
- Se recomienda mantener actualizado tanto el sistema operativo como los aplicativos de software libre utilizados (Zabbix, Zimbra, Python).

Bibliografía

Linux. (n.d.). *Linux*. Retrieved from Linux: https://www.linux.org/

Nwana, H. (1996). The Knowledge Engineering Review. In H. Nwana, *Software agents: An overview. The Knowledge Engineering Review* (pp. 205-244).

OpenSource. (2019). *Open Source*. Retrieved from What is Open Source: https://opensource.com/resources/what-open-source

Telegram. (n.d.). FAQ - Bots FAQ.

Waldemar, C., Waldemar, R., Ierusalimschy, L., & Ierusalimschy, F. (1994). The Design and Implementation of a Language for Extending Applications. . In *The Design and Implementation of a Language for Extending Applications.* .

Zabbix. (2019, 4 2). *Zabbix Documentation 4.2*. Retrieved from https://www.zabbix.com/documentation/4.2/manual/introduction/overview

Zimbra. (2018, 12 17). *Zimbra Documentation*. Retrieved from https://www.zimbra.com/documentation/