



TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA INFORMÁTICA

IMPLEMENTACIÓN DE UN SISTEMA DE
MONITORIZACIÓN DE DISPOSITIVOS Y SERVICIOS EN
UNA RED DOMÉSTICA.

Autor

Pablo Manuel García Sánchez



ESCUELA SUPERIOR DE INGENIERÍA

Cádiz, Abril de 2021

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA INFORMÁTICA

IMPLEMENTACIÓN DE UN SISTEMA DE
MONITORIZACIÓN DE DISPOSITIVOS Y SERVICIOS EN
UNA RED DOMÉSTICA

Autor

Pablo Manuel García Sánchez

Director

Antonio Jesús Molina Cabrera



Escuela Superior
de Ingeniería

ESCUELA SUPERIOR DE INGENIERÍA

Cádiz, Abril de 2021

Fdo.: Pablo Manuel García Sánchez

DECLARACIÓN PERSONAL DE AUTORIA

Pablo Manuel García Sánchez con DNI 45382609K, estudiante del Grado de Ingeniería Informática en la Escuela Superior de Ingeniería de la Universidad de Cádiz, como autor de este documento académico titulado “Implementación de un sistema de monitorización de dispositivos y servicios en una red doméstica” y presentado como Trabajo Final de Grado.

DECLARO QUE

Es un trabajo original, que no copio ni utilizo parte de obra alguna sin mencionar de forma clara y precisa su origen tanto en el cuerpo del texto como en su bibliografía y que no empleo datos de terceros sin la debida autorización, de acuerdo con la legislación vigente. Asimismo, declaro que soy plenamente consciente de que no respetar esta obligación podrá implicar la aplicación de sanciones académicas, sin perjuicio de otras actuaciones que pudieran iniciarse.

En Cádiz, a 11/04/2021

Fdo: Pablo Manuel García Sánchez



IMPLEMENTACIÓN DE UN SISTEMA DE MONITORIZACIÓN DE DISPOSITIVOS Y SERVICIOS EN UNA RED DOMÉSTICA

ÍNDICE GENERAL

- **DATOS CLIENTE:** ESCUELA SUPERIOR DE INGENIERÍA, UNIVERSIDAD DE CÁDIZ.
AV. UNIVERSIDAD DE CÁDIZ, 10, 11519 PUERTO REAL, CÁDIZ
956 48 32 00
DIRECCION.ESI@UCA.ES
- **DATOS AUTOR:** PABLO MANUEL GARCÍA SANCHEZ
INGENIERO INFORMÁTICO
PABLO.GARCIASANCH@ALUM.UCA.ES

Cádiz, Abril de 2021

ÍNDICE GENERAL

ÍNDICE GENERAL	9
ÍNDICE DE ILUSTRACIONES	13
ÍNDICE DE TABLAS	15
MEMORIA	197
1 OBJETIVO	21
2 ANTECEDENTES	21
3 DESCRIPCIÓN DE LA SITUACIÓN ACTUAL	22
3.1 ARQUITECTURA RED DOMÉSTICA ESTÁNDAR	23
4 NORMAS Y REFERENCIAS	23
4.1 DISPOSICIONES LEGALES Y NORMAS APLICADAS	24
4.2 BIBLIOGRAFÍA	24
4.3 HERRAMIENTAS	25
4.3.1 Herramientas software	25
4.3.2 Herramientas hardware	26
4.4 OTRAS REFERENCIAS	28
4.4.1 Repositorio en GitHub	28
5 DEFINICIONES Y ABREVIATURAS	28
6 REQUISITOS INICIALES	29
7 ALCANCE DEL PROYECTO	29
8 ESTUDIO DE ALTERNATIVAS Y VIABILIDAD	30
8.1 ESTUDIO DEL SOFTWARE A UTILIZAR	30
8.1.1 Nagios	30
8.1.2 Zabbix	33
8.1.3 Cacti	36
8.1.4 Pandora FMS	38
8.1.5 Tabla comparativa de las herramientas	41
9 DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA	42
9.1 ARQUITECTURA DE LA SOLUCIÓN PROPUESTA	42
9.1.1 NMS (Raspberry Pi)	44
9.1.2 Agente SNMP	44
9.1.3 Dispositivos administrados	44
9.2 FUNCIONAMIENTO	45
9.2.1 Monitorización LAN	45
9.2.1.1 Ping	47
9.2.1.2 Get (request-response)	47
9.2.1.3 Trap	47
9.2.2 Monitorización WAN	47
9.3 GESTIÓN DE LA INFORMACIÓN	48
9.3.1 Base de datos Homemonitor_db	49
9.3.1.1 Tabla hosts	49
9.3.1.2 Tabla snmp_linux	49
9.3.1.3 Tabla snmp_windows	50
9.3.1.4 Tabla isp	51

9.4	DESCRIPCIÓN DE LA INTERFAZ	51
9.4.1	Hosts	52
9.4.2	SNMP Linux	52
9.4.2.1	CPU	53
9.4.2.2	Memoria RAM	54
9.4.2.3	Espacio en Disco	54
9.4.3	SNMP Windows	55
9.4.3.1	CPU	56
9.4.3.2	Memoria RAM	57
9.4.3.3	Espacio en Disco	57
9.4.4	ISP (Proveedor de Servicios de Internet)	58
9.4.4.1	Velocidad de Carga y Descarga.	59
9.4.4.2	Latencia	60
9.5	SISTEMA DE ALARMAS	61
9.5.1	Alertas ICMP	61
9.5.2	Alertas SNMP	61
9.5.3	Alertas ISP	62
10	PLANIFICACION TEMPORAL	63
10.1	METODOLOGÍA DE DESARROLLO	63
10.2	PLANIFICACIÓN TEMPORAL DEL PROYECTO	63
10.2.1	Especificación del proyecto.	64
10.2.2	Investigación y búsqueda de información.	64
10.2.3	Implementación de la solución.	64
10.2.4	Memoria del proyecto	64
11	EVALUACIÓN DE RIESGOS	66
11.1	LISTA DE RIESGOS	66
11.2	CATALOGACIÓN DE LOS RIESGOS	66
11.3	PLAN DE CONTINGENCIA	67
12	RESUMEN DEL PRESUPUESTO	67
	FUNDAMENTOS TEÓRICOS	69
13	CONCEPTOS BÁSICOS.	73
13.1	RED DE DATOS	73
13.2	TOPOLOGÍAS	73
13.2.1	Topología en Malla	73
13.2.2	Topología en Bus	74
13.2.3	Topología en Anillo	74
13.2.4	Topología en Estrella	75
13.2.5	Topologías en Árbol	76
13.2.6	Topologías Lógicas	76
13.3	MODELO OSI	77
13.4	MODELO TCP/IP	78
13.4.1	Acceso a red	79
13.4.2	Internet	79
13.4.3	Transporte	79
13.4.4	Aplicación	79
13.5	PROTOCOLOS	79
13.5.1	UDP	79
13.5.2	TCP	80
14	MONITORIZACIÓN	81

14.1	TIPOS DE MONITORIZACIÓN	81
14.1.1	Monitorización pasiva	81
14.1.2	Monitorización activa	81
14.1.3	Alarms	81
14.2	ELEMENTOS A MONITORIZAR	81
15	ICMP	82
15.1	FUNCIONAMIENTO	82
15.2	OPERACIONES	82
16	SNMP	83
16.1	HISTORIA	84
16.2	FUNCIONAMIENTO	84
16.2.1	Dispositivos administrativos	84
16.2.2	Agente	84
16.2.3	NMS (Network Management System)	84
16.2.4	Operaciones	86
16.2.4.1	Get	86
16.2.4.2	Get-Next	86
16.2.4.3	Get-Bulk	86
16.2.4.4	Set	87
16.2.4.5	Trap	87
16.3	COMUNIDADES	87
16.4	MIB	88
16.5	VERSIONES EXISTENTES	89
17	PROVEEDOR SERVICIOS DE INTERNET	89
17.1	ARQUITECTURA	90
18	RASPBERRY PI OS	91
19	OTROS SISTEMAS OPERATIVOS	91
19.1	EL SISTEMA OPERATIVO LINUX	91
19.1.1	Historia	91
19.1.2	Características generales	92
19.2	WINDOWS SERVER 2016	92
19.2.1	Características generales	93
	ANEXOS	95
20	MANUAL DE USUARIO	99
20.1	CONFIGURACIÓN INICIAL	99
20.1.1	Configuración /etc/hosts	99
20.1.2	Configuración servidor web	99
20.1.3	Configuración base de datos	100
20.1.4	Configuración lenguaje de programación	104
20.1.5	Configuración sistema de alarmas	104
20.1.6	Configuración entorno ICMP	107
20.1.7	Configuración entorno SNMP	108
20.1.7.1	Configuración EMS	108
20.1.7.2	Configuración Agente SNMP en Linux	111
20.1.7.3	configuración Agente SNMP en Windows	119
20.1.8	Configuración entorno ISP	125
20.1.9	Configuración de los demonios	126

20.1.10	Configuración config.py	127
20.1.11	Configuración interfaz gráfica	127
21	LIBRERÍAS Y CÓDIGOS RELEVANTES DEL SISTEMA	130
21.1	ICMP	130
21.1.1	Vigilante.py	130
21.2	SNMP	134
21.2.1	Get-Response	134
21.2.1.1	config.py	134
21.2.1.2	SnmpCPULinux.py	135
21.2.1.3	SnmpCPUWindows.py	136
21.2.1.4	SnmpRamLinux.py	138
21.2.1.5	SnmpRamWindows.py	139
21.2.1.6	SnmpDiskLinux.py	141
21.2.1.7	SnmpDiskWindows.py	142
21.2.1.8	SnmpLinux.py	144
21.2.1.9	SnmpWindows.py	145
21.2.2	TRAPS	147
21.2.2.1	trap_handler.py	147
21.3	ISP	148
21.3.1	Config.py	148
21.3.2	Isp_monitor.py	149
	ESPECIFICACIONES DEL SISTEMA	1553
22	OBJETIVOS DEL SISTEMA	157
23	DESCRIPCIÓN DE ACTORES	157
24	REQUISITOS FUNCIONALES	158
24.1	DIAGRAMAS CASOS DE USO	159
24.1.1	Login	159
24.1.2	Gestión de Dispositivos Administrados	159
24.1.3	Gestión de Alarmas	160
24.2	CASOS DE USO	160
	PRESUPUESTO	1675
25	PRESUPUESTO	169
25.1	INTRODUCCIÓN	169
25.2	PRESUPUESTO DETALLADO	169
25.2.1	Presupuesto Hardware	169
25.2.2	Presupuesto Software	169
25.2.3	Presupuesto del Personal	169
25.2.4	Resumen del presupuesto	170

ÍNDICE DE ILUSTRACIONES

Ilustración 1: Topología de red estándar	23
Ilustración 2: Router Movistar HGU	26
Ilustración 3: Raspberry PI	27
Ilustración 4: Asus Zenbook UX303LB	27
Ilustración 5: Arquitectura NRPE Nagios	31
Ilustración 6: Arquitectura NSClient++ Nagios	31
Ilustración 7: Interfaz Web Nagios	32
Ilustración 8: Gráficos Nagios	33
Ilustración 9: Arquitectura Agente-Servidor Zabbix	34
Ilustración 10: Interfaz Web Zabbix	36
Ilustración 11: Interfaz Web Cacti	37
Ilustración 12: Arquitectura Pandora FMS	39
Ilustración 13: Interfaz Web Pandora FMS	41
Ilustración 14: Arquitectura red doméstica	43
Ilustración 15: Management Information Base SNMP	45
Ilustración 16: Diseño del flujo de trabajo entre dispositivos LAN	46
Ilustración 17: Diseño del flujo de trabajo entre dispositivos WAN	47
Ilustración 18: Hosts registrados en la red	52
Ilustración 19: Tablero de los Dispositivos Administrados Linux	53
Ilustración 20: Selector de Dispositivos Linux	53
Ilustración 21: CPU dispositivos Linux	54
Ilustración 22: Memoria RAM dispositivos Linux	54
Ilustración 23: Espacio en Disco dispositivos Linux	55
Ilustración 24: Espacio en Disco dispositivos Linux	55
Ilustración 25: Tablero de los Dispositivos Administrados Windows	56
Ilustración 26: Selector de Dispositivos Windows	56
Ilustración 27: CPU dispositivos Windows	57
Ilustración 28: Memoria RAM dispositivos Windows	57
Ilustración 29: Espacio en Disco dispositivos Windows	58
Ilustración 30: Espacio en Disco dispositivos Windows	58
Ilustración 31: Tablero Proveedor Servicios de Internet	59
Ilustración 32: Velocidad Carga y Descarga ISP	59
Ilustración 33: Velocidad Descarga ISP	60
Ilustración 34: Velocidad Carga ISP	60
Ilustración 35: Latencia ISP	60
Ilustración 36: Notificación nuevo dispositivo	61
Ilustración 37 Notificación desconexión/reconexión dispositivo	61
Ilustración 38: Notificación evento Linux	62
Ilustración 39: Notificación evento Windows	62
Ilustración 40: Notificación calidad servicio ISP	63
Ilustración 41: Diagrama de Gantt de la planificación temporal del proyecto	65
Ilustración 42: Topología en Malla	73
Ilustración 43: Topología en Bus	74
Ilustración 44: Topología en Anillo	75
Ilustración 45: Topología en Estrella	76
Ilustración 46: Topología en Árbol	76
Ilustración 47: Modelo OSI	78
Ilustración 48: Modelo TCP/IP	78
Ilustración 49: Formato PDU UDP	79
Ilustración 50: Formato PDU TCP	80

Ilustración 51: Formato de mensaje ICMP	83
Ilustración 52: Formato de Mensaje SNMP	85
Ilustración 53: Formato PDU SNMP	85
Ilustración 54: Estado de Errores SNMP	85
Ilustración 55: Operación GET SNMP	86
Ilustración 56: Operación SET SNMP	87
Ilustración 57: Operación TRAP SNMP	87
Ilustración 58: MIB SNMP	88
Ilustración 59: Arquitectura ISP	90
Ilustración 60: Apache Debian Default Page	100
Ilustración 61: Configuración servidor phpMyAdmin	102
Ilustración 62: Configuración BD phpMyAdmin	102
Ilustración 63: Configuración contraseña phpMyAdmin	103
Ilustración 64: Importar BD phpMyAdmin	103
Ilustración 65: BotFather Telegram	105
Ilustración 66: Configuración BOT Telegram 1	105
Ilustración 67: Configuración BOT Telegram 2	106
Ilustración 68: Configuración SNMP Windows	120
Ilustración 69: Configuración Agente SNMP Windows	121
Ilustración 70: Configuración Traps SNMP Windows	122
Ilustración 71: Configuración Seguridad SNMP Windows	123
Ilustración 72: EvntWin Windows	124
Ilustración 73: Configuración EvntWin Windows 1	124
Ilustración 74: Configuración EvntWin Windows 2	125
Ilustración 75: Página inicio Grafana	128
Ilustración 76: Configuración Grafana	128
Ilustración 77: Configuración Data Source Grafana	129
Ilustración 78: Importar Tableros Grafana	130
Ilustración 79: Diagrama Caso de Uso Login	159
Ilustración 80: Diagrama Caso de Uso Gestión de Dispositivos Administrados	160
Ilustración 81: Diagrama Caso de Uso Gestión de Alarmas	160

ÍNDICE DE TABLAS

Tabla 1: Especificaciones Router (Movistar HGU) _____	27
Tabla 2: Especificaciones Raspberry PI _____	27
Tabla 3: Especificaciones Asus Zenbook UX303LB _____	28
Tabla 4: Tabla comparativa herramientas monitorización. _____	42
Tabla 5: Tabla hosts _____	49
Tabla 6: Tabla snmp_linux _____	50
Tabla 7: Tabla snmp_windows _____	50
Tabla 8: Tabla isp _____	51
Tabla 9: Catalogación Riesgos _____	67
Tabla 10: Planes de Contingencia _____	67
Tabla 11: Resumen del Presupuesto _____	67
Tabla 12: Objetivo 01: Elección del sistema administrador _____	157
Tabla 13: Objetivo 02: Elección de Agente _____	157
Tabla 14: Objetivo 03: Monitorización variables _____	157
Tabla 15: Objetivo 04: Almacenamiento de datos _____	157
Tabla 16: Objetivo 05: Interfaz Usable _____	157
Tabla 17: Objetivo 06: Sistema de Alertas _____	157
Tabla 18: Actor 01: Administrador _____	158
Tabla 19: Requisito funcional 01: Sistema Administrador de Red _____	158
Tabla 20: Requisito funcional 02: Dispositivo Administrado _____	158
Tabla 21: Requisito funcional 03: Disponibilidad _____	158
Tabla 22: Requisito funcional 04: Estado de los dispositivos _____	158
Tabla 23: Requisito funcional 05: Espacio en Disco _____	158
Tabla 24: Requisito funcional 06: Memoria Disponible _____	158
Tabla 25: Requisito funcional 07: Carga del procesador _____	159
Tabla 26: Requisito funcional 08: Estado ISP _____	159
Tabla 27: Requisito funcional 09: Sistema de Alarmas _____	159
Tabla 28: Caso de Uso 01: Login _____	160
Tabla 29: Caso de Uso 02: Añadir Dispositivo _____	161
Tabla 30: Caso de Uso 03: Editar Dispositivo _____	161
Tabla 31: Caso de Uso 04: Eliminar dispositivo _____	162
Tabla 32: Caso de Uso 05: Añadir Alarma _____	162
Tabla 33: Caso de Uso 06: Editar Alarma _____	162
Tabla 34: Caso de Uso 07: Eliminar Alarma _____	163
Tabla 35: Presupuesto Hardware _____	169
Tabla 36: Presupuesto Software _____	169
Tabla 37: Presupuesto del personal _____	169
Tabla 38: Resumen del presupuesto _____	170



IMPLEMENTACIÓN DE UN SISTEMA DE MONITORIZACIÓN DE DISPOSITIVOS Y SERVICIOS EN UNA RED DOMÉSTICA

MEMORIA

- **DATOS CLIENTE:** ESCUELA SUPERIOR DE INGENIERÍA, UNIVERSIDAD DE CÁDIZ.
AV. UNIVERSIDAD DE CÁDIZ, 10, 11519 PUERTO REAL, CÁDIZ
956 48 32 00
DIRECCION.ESI@UCA.ES
- **DATOS AUTOR:** PABLO MANUEL GARCÍA SANCHEZ
INGENIERO INFORMÁTICO
PABLO.GARCIASANCH@ALUM.UCA.ES

Cádiz, Abril de 2021

ÍNDICE

MEMORIA	197
1 OBJETIVO	21
2 ANTECEDENTES	21
3 DESCRIPCIÓN DE LA SITUACIÓN ACTUAL	22
3.1 ARQUITECTURA RED DOMÉSTICA ESTÁNDAR	23
4 NORMAS Y REFERENCIAS	23
4.1 DISPOSICIONES LEGALES Y NORMAS APLICADAS	24
4.2 BIBLIOGRAFÍA	24
4.3 HERRAMIENTAS	25
4.3.1 Herramientas software	25
4.3.2 Herramientas hardware	26
4.4 OTRAS REFERENCIAS	28
4.4.1 Repositorio en GitHub	28
5 DEFINICIONES Y ABREVIATURAS	28
6 REQUISITOS INICIALES	29
7 ALCANCE DEL PROYECTO	29
8 ESTUDIO DE ALTERNATIVAS Y VIABILIDAD	30
8.1 ESTUDIO DEL SOFTWARE A UTILIZAR	30
8.1.1 Nagios	30
8.1.2 Zabbix	33
8.1.3 Cacti	36
8.1.4 Pandora FMS	38
8.1.5 Tabla comparativa de las herramientas	41
9 DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA	42
9.1 ARQUITECTURA DE LA SOLUCIÓN PROPUESTA	42
9.1.1 NMS (Raspberry Pi)	44
9.1.2 Agente SNMP	44
9.1.3 Dispositivos administrados	44
9.2 FUNCIONAMIENTO	45
9.2.1 Monitorización LAN	45
9.2.1.1 Ping	47
9.2.1.2 Get (request-response)	47
9.2.1.3 Trap	47
9.2.2 Monitorización WAN	47
9.3 GESTIÓN DE LA INFORMACIÓN	48
9.3.1 Base de datos Homemonitor_db	49
9.3.1.1 Tabla hosts	49
9.3.1.2 Tabla snmp_linux	49
9.3.1.3 Tabla snmp_windows	50
9.3.1.4 Tabla isp	51
9.4 DESCRIPCIÓN DE LA INTERFAZ	51
9.4.1 Hosts	52
9.4.2 SNMP Linux	52
9.4.2.1 CPU	53

9.4.2.2 Memoria RAM	54
9.4.2.3 Espacio en Disco	54
9.4.3 SNMP Windows	55
9.4.3.1 CPU	56
9.4.3.2 Memoria RAM	57
9.4.3.3 Espacio en Disco	57
9.4.4 ISP (Proveedor de Servicios de Internet)	58
9.4.4.1 Velocidad de Carga y Descarga.	59
9.4.4.2 Latencia	60
9.5 SISTEMA DE ALARMAS	61
9.5.1 Alertas ICMP	61
9.5.2 Alertas SNMP	61
9.5.3 Alertas ISP	62
10 PLANIFICACION TEMPORAL	63
10.1 METODOLOGÍA DE DESARROLLO	63
10.2 PLANIFICACIÓN TEMPORAL DEL PROYECTO	63
10.2.1 Especificación del proyecto.	64
10.2.2 Investigación y búsqueda de información.	64
10.2.3 Implementación de la solución.	64
10.2.4 Memoria del proyecto	64
11 EVALUACIÓN DE RIESGOS	66
11.1 LISTA DE RIESGOS	66
11.2 CATALOGACIÓN DE LOS RIESGOS	66
11.3 PLAN DE CONTINGENCIA	67
12 RESUMEN DEL PRESUPUESTO	67

1 OBJETIVO

El objetivo de este proyecto es implementar un sistema de monitorización de dispositivos y servicios multiplataforma en cualquier red doméstica o pequeña empresa, con el fin de proveer al usuario administrador de red de herramientas de detección, prevención y análisis de intrusiones, caídas de servicio y vulnerabilidades tanto en servidores, servicios, dispositivos de red o estaciones de trabajo.

El sistema final implementado debe ser capaz de notificar al administrador de red cuando detecte tráfico indicativo de un intento de intrusión o caída de servicio y debe ser capaz, en la medida de lo posible, de ofrecer la información necesaria para mantener la red lo más segura posible mediante informes y alertas de detección de vulnerabilidades.

Los objetivos básicos que debe de cubrir el proyecto son tres:

- Monitorización de los equipos en tiempo real.
- Monitorización de la calidad de servicio ofrecida por el proveedor en tiempo real.
- Creación de un sistema de alarmas que permita alertar al administrador de red cuando algunos de los aspectos parametrizados superen los rangos establecidos.

2 ANTECEDENTES

El software de monitorización nos ha rodeado de una forma u otra desde los principios de la computación. La era de los primeros miniordenadores no sería inexacto categorizarla como la era de la casi ausencia de supervisión. Las herramientas de monitoreo basadas en software en el sentido contemporáneo del término eran primitivas, produciendo resultados que consistían en poco más que volcados de memoria (en caso de bloqueos) y logs.

Fue durante los 90's cuando las herramientas de monitorización en tiempo real se convirtieron en un estándar de la mayoría de los sistemas operativos de escritorio monitorizando algunos de los recursos y servicios de estos.

Al mismo tiempo, comenzó a gestarse la monitorización de redes con la contribución de herramientas como "nmon", "MRTG" y "Big Brother". Mientras que la monitorización de escritorio se centraba en la supervisión de un único sistema, la monitorización de redes enfrentaba otro reto más amplio supervisar la salud y el desarrollo de múltiples interfaces de comunicación física, así como, componentes hardware haciendo uso del tráfico para adecuarse y no interferir en la carga de trabajo solicitada por el usuario.

A final de los 90's la mayoría de las herramientas de monitorización fueron desarrollados con la asunción de que iban a ser usadas en redes locales o equivalentes, con un número limitado de usuarios en un entorno cerrado.

Sin embargo, al comienzo del siglo XXI se estaba haciendo evidente que las necesidades de monitoreo de los sitios web y servicios basados en Internet no eran compartidas por aquellas típicas redes LAN. En la actualidad se trabaja en el desarrollo de una generación de herramientas de monitorización estadísticas (Cacti, Nagios y Zabbix) que admiten protocolos de Internet estándar, pueden usarse en múltiples plataformas, a menudo son bastante escalables y, por lo general, tienen interfaces basadas en la web.

3 DESCRIPCIÓN DE LA SITUACIÓN ACTUAL

Hoy en día nuestras redes domésticas están haciendo cada vez más y más grandes. Casi cualquier dispositivo requiere de una conexión a internet para desempeñar sus funciones. Pero a medida que vamos extendiéndola se hace más grande la necesidad de monitorizar los eventos que ocurren dentro de ella. Ya sea para detectar una intrusión, identificar cualquier problema en la red o simplemente conocer qué dispositivo está haciendo mayor uso del ancho de banda.

Obtener diversa información de lo que ocurre en tu red presenta diferentes ventajas:

- Prevención de cortes de red antes de que ocurran

Una caída de servicio es uno de los problemas más graves que se pueden presentar en una red. Soluciones de monitoreo pueden ayudar a prevenir una caída antes de que ocurra buscando en la red cualquier comportamiento de rendimiento sospechoso que indique que está a punto de ocurrir una interrupción. Si un dispositivo o una parte de la red tiene un rendimiento retardado, su herramienta de monitoreo detectará el problema y lo alertará.

- Reducción del tiempo de caída de servicio.

Eventualmente, es muy probable que en un ámbito doméstico se descubra un problema de rendimiento de la red por sí mismo, pero podría llevar mucho tiempo solucionarlo una vez descubierto. La monitorización informará a su equipo de los problemas a medida que sucedan, reduciendo drásticamente el tiempo de detección de éste. Consecuentemente se reducirá el tiempo medio de reparación (MTTR) de los problemas de red. Existen muchas soluciones de monitoreo que además de identificar, aplican diagnósticos a los problemas que descubren facilitando así su remedio.

- Generación de reportes de red

Las herramientas de monitoreo brindan visualizaciones sobre métricas clave de rendimiento de una red. Consecuentemente son capaces de generar informes de desempeño que puedan ser revisados. Estos informes deben incluir datos tanto recientes como históricos para que pueda analizar el rendimiento de su red a lo largo del tiempo. Además, permiten personalizar los informes para centrarse en las métricas que son importantes para cada caso.

- Descubrimiento de amenazas de seguridad en tu red

La monitorización de eventos no solo es útil para rastrear el desempeño de sus sistemas, también ayudan a hacer frente a las amenazas de seguridad que invaden la red. Con una de estas herramientas es posible recibir alertas sobre eventos que indiquen si hay malware presente en su sistema (transferencias de datos anormales, acceso a ficheros protegidos, etc.).

- Mantenimiento de una visibilidad total de la red

Para comprender realmente el rendimiento de su red, se debe poder observar cada área de la misma. Las soluciones de monitoreo detectan automáticamente los dispositivos que se conectan a una red, dibujando y actualizando mapas visuales que muestran información sobre el desempeño de sus nodos de red.

Monitorizar no es sólo recoger y guardar información. Monitorizar es agregar datos, filtrar, analizar, tomar decisiones y actuar.

3.1 ARQUITECTURA RED DOMÉSTICA ESTÁNDAR

La contratación de los diferentes proveedores de servicios, así como, su posterior instalación de la red ha llevado a una gran generalización de las topologías de red domésticas.

A continuación, observamos un ejemplo comúnmente utilizado en estos ámbitos particulares.

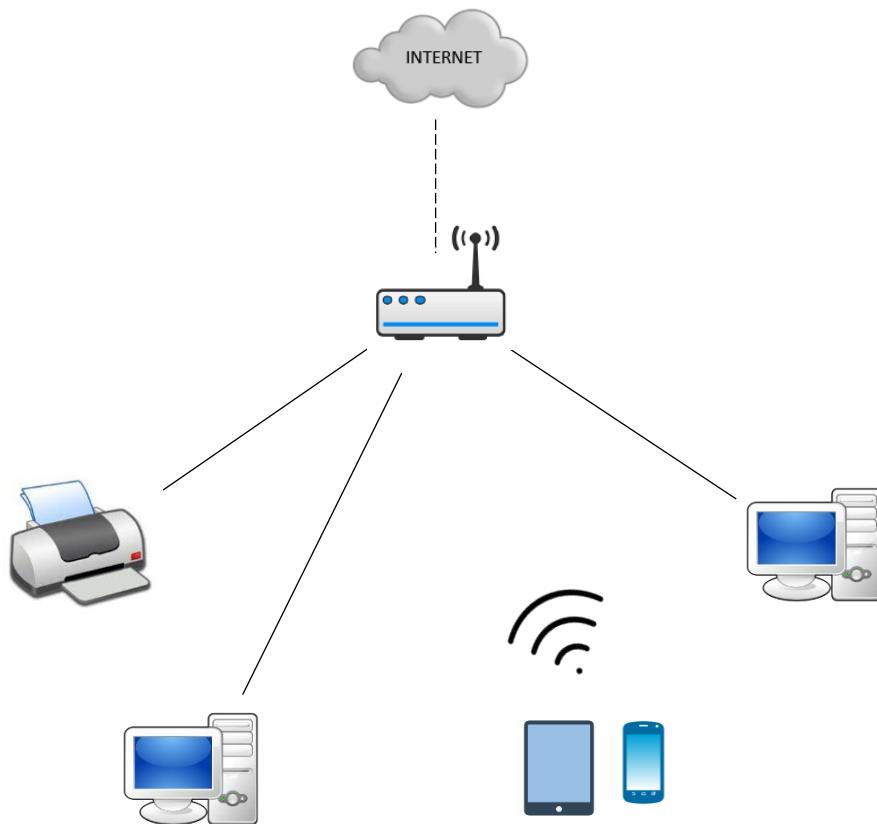


Ilustración 1: Topología de red estándar

En él podemos apreciar al router de operadora que hace las funciones de enrutador, switch y punto de acceso. Es el encargado de la salida hacia internet y al cual se conectan los diferentes dispositivos finales, tales como los ordenadores, móviles, o impresoras. Ya sea a través de un medio físico o inalámbrico.

Todas las ventajas comentadas anteriormente son aplicables a este tipo de arquitectura.

4 NORMAS Y REFERENCIAS

A lo largo de este apartado se recogerán todas las normas y referencias en las que se ha basado el proyecto. Así como, las fuentes y herramientas de las que se ha hecho uso para la elaboración del mismo.

4.1 DISPOSICIONES LEGALES Y NORMAS APLICADAS

- Norma UNE 157001:2014 - Criterios generales para la elaboración formal de los documentos que constituyen un proyecto técnico.

4.2 BIBLIOGRAFÍA

Davin, J., Case, J. D., Fedor, M., & Schoffstall, M. L. (1990). *Simple Network Management Protocol (SNMP)*. RFC 1157. <https://tools.ietf.org/html/rfc1157>

Douglas R. Mauro & Kevin J. Schmidt. (2005). «*Essential SNMP*» 2nd Edition.

Eben Upton & Gareth Halfacree. (2016). «*Raspberry Pi User Guide*», 4th Edition.

Estela Raffino, M. (2020). *Modelo OSI - Concepto, cómo funciona, para qué sirve y capas*.

<https://concepto.de/modelo-osi/>

Forouzan, B. (2019). «*Transmision De Datos Y Redes De Comunicaciones*» 4º Edición.

Grafana Labs. (2021). *Grafana documentation*. Grafana Labs.

<https://grafana.com/docs/grafana/latest/>

IONOS España S.L.U. (2019). *¿Qué es el ICMP? Aspectos destacados del protocolo de mensajes*.

IONOS Digitalguide. <https://www.ionos.es/digitalguide/servidores/know-how/que-es-el-protocolo-icmp-y-como-funciona/>

IONOS España S.L.U. (2020). *TCP/IP: El protocolo que hace posible Internet*. IONOS Digitalguide.

<https://www.ionos.es/digitalguide/servidores/know-how/tcpip/>

Les Cottrell. (2001). *Passive vs. Active Monitoring*.

<https://www.slac.stanford.edu/comp/net/wan-mon/passive-vs-active.html>

Liliana Allende, S., Alejandro Gibellini, F., Beatriz Sánchez, C., & Mariel Serna, M. (2019).

«*Sistema Operativo LINUX Teoría y Práctica*» 2ª Edición.

McCabe, J. (2016). *Introducing Windows Server 2016*. https://docs.microsoft.com/es-es/archive/blogs/microsoft_press/free-ebook-introducing-windows-server-2016

Nagios Enterprises, LLC. (2009). *Nagios—The Industry Standard In IT Infrastructure Monitoring*.

Nagios. <https://www.nagios.org/>

Net-SNMP. (2011). <http://www.net-snmp.org/>

Orange SA. (s. f.). *OID Repository—Home*. <http://oid-info.com/>

Pandora FMS. (2012). «*Pandora FMS 4.0.2 Manual de Usuario*». 1º Edición (España), 20 junio 2012. https://pandorafms.com/downloads/PDF/PandoraFMS_4.0.2_Manual_ES.pdf

Rose, M. T., & McCloghrie, K. (1991). *Concise MIB definitions. RFC 1212*.
<https://tools.ietf.org/html/rfc1212>

The Cacti Group, Inc. (2004). *Cacti®—The Complete RRDTool-based Graphing Solution*.
<https://www.cacti.net/>

Tutorials Point. (2016). *MariaDB tutorialspoint*.
https://www.tutorialspoint.com/mariadb/mariadb_tutorial.pdf

Villa Avila, L. H., & Villanueva Vivas, A. J. (2013). *Diseño e implementación de un ISP con acceso inalámbrico para soportar servicios de Internet Y Telefonía IP en el laboratorio de Telecomunicaciones de la Universidad Autónoma de Occidente*.

Zabbix SIA. (2001). *Zabbix Manual [Zabbix Documentation 3.0]*.
<https://www.zabbix.com/documentation/3.0/manual>

4.3 HERRAMIENTAS

4.3.1 Herramientas software

- **Python:** Lenguaje de programación de alto nivel. Necesario para desarrollar los scripts recolectores de información.
- **Bash Script:** Lenguaje de comandos interpretado por el shell de UNIX
- **Batch Script:** Lenguaje de comandos interpretado por el shell MSDOS (Windows)
- **Sublime Text:** Editor de texto especializado en código de programación
- **GanttProject:** Editor de diagramas de Gantt utilizado para la realización del diagrama de planificación del proyecto.
- **Microsoft Word 2013:** Editor de texto utilizado para la elaboración del documento final.
- **EvntWin:** Generador de SNMP Traps a partir de eventos preconfigurados en Windows. Usado para la generación y envío de traps en los sistemas de Microsoft.
- **Telegram:** Aplicación online de mensajería instantánea y VOIP. Necesaria para la notificación al usuario de cualquier evento generado.
- **RealVNC Viewer:** Aplicación de acceso remoto vía interfaz gráfica. Utilizado para acceder al NMS desde cualquier dispositivo en su misma red

- **phpMyAdmin:** Gestor de Base de Datos. Usado para gestionar la información de la base de datos relacional SQL.
- **Grafana:** Grafana es una herramienta de código abierto para el análisis y visualización de métricas. Se utiliza para la representación gráfica de la información almacenada.

4.3.2 Herramientas hardware

- **Router (Movistar HGU)**



Ilustración 2: Router Movistar HGU

Concepto	Valor
Fabricante	Mitrasstar
LAN	4 conectores RJ-45 para conexiones 10/100/1000 Mbps Gigabit Ethernet
Teléfono	Puerto de línea telefónica para terminales.
Wi-Fi(2.4G)	Soporte de Wireless N Interfaz 802.11n . 2 antenas internas . Canales soportados: del 1 al 13. Acceso sin encriptación, WEP (64 / 128bit), WPA2 con PSK, WPA + WPA2 modo mixto. Posibilidad de restringir acceso por dirección MAC.
Wifi Plus(5G)	4 antenas internas. IEEE 802.11ac. Ancho de banda configurable 20/40/80MHz. Canales soportados: 36, 40, 44, 48, 52, 56, 60, 64, 100, 104, 108 y 112. Acceso sin encriptación, o WPA2 con AES.
Routing	Routing entre interfaces LAN y WAN enrutadas. IPoE WAN con direccionamiento IP estático / DHCP. Soporte de VLAN tagging por cada IPoE WAN. Soporte de VLAN tagging por cada PPPoE WAN. Rutas IP estáticas. RIPv2 con modo pasivo. NAT simétrico/completo con soporte de ALG. Firewall con estado. Soporte de mapeo de puertos(Virtual server)/DMZ. DNS dinámico. Cliente DHCP con soporte de Opción 120, Opción 100/NTP. Servidor LAN DHCP con servidores DNS estáticos. Dirección IP secundaria. DHCP condicional sirviendo grupos para Vendor Class Id específicos.

	minAddress/maxAddress/Router/DNSservers/Option 240. Opciones 120/100/NTP para clients LAN. Control de paquetes CPU (LCP/DHCP) con prioridad alta. DNS proxy deshabilitado. IGMP proxy. Routing IPv6.
--	--

Tabla 1: Especificaciones Router (Movistar HGU)

- **Raspberry Pi**



Ilustración 3: Raspberry PI

Concepto	Valor
Procesador	ARM Cortex-A72
Frecuencia de reloj	1,5 GHz
GPU	VideoCore VI (con soporte para OpenGL ES 3.x)
Memoria	2GB DDR4 RAM
Conectividad	Bluetooth 5.0, Wi-Fi 802.11ac, Gigabit Ethernet
Puertos	GPIO 40 pines 2 x micro HDMI 2 x USB 2.0 2 x USB 3.0 CSI (cámara Raspberry Pi) DSI (pantalla táctil) Micro SD Conector de audio jack USB-C (alimentación)

Tabla 2: Especificaciones Raspberry PI

- **Asus Zenbook UX303LB**



Ilustración 4: Asus Zenbook UX303LB

Concepto	Valor
Pantalla	13.3" retroiluminación LED 1920 x 1080 / Full HD
Procesador	Intel Core i7-5500U 5 ^a Gen
Frecuencia de reloj	2.4 GHz (3 GHz) / 4 MB Caché
GPU	NVIDIA GeForce 940M - 2 GB DDR3 SDRAM
Memoria	8GB DDR3L SODIMM
Conectividad	Bluetooth 5.0, Wi-Fi 802.11ac
Puertos	1 x HDMI. 1 x combo audio. 1 x Mini Display Port. 3 x USB 3.0. Lector de Tarjetas
Sistema Operativo	Dual Boot: Ubuntu 18.04 LTS – Windows 10 (64 bits)

Tabla 3: Especificaciones Asus Zenbook UX303LB

4.4 OTRAS REFERENCIAS

4.4.1 Repositorio en GitHub

Github es una plataforma para alojar el código de las aplicaciones de cualquier desarrollador. Fue adquirida por Microsoft en 2018. Es una plataforma compleja que fomenta la colaboración entre desarrolladores aportando funcionalidad que permiten a los equipos de desarrollo trabajar juntos en el mismo proyecto y crear fácilmente nuevas versiones de software.

Es por ello que se ha optado por crear un repositorio donde alojar toda la documentación y software. De esta forma, se facilita la tarea de crear nuevas versiones del proyecto, así como la inclusión de algún otro nuevo desarrollador.

El enlace es el siguiente:

<https://github.com/pablogrsc/MONITORIZACION-DOMESTICA>

5 DEFINICIONES Y ABREVIATURAS

MTTR: Mean time to repair (Tiempo medio de reparación)

NDR: Network detection and response. (Capacidad de detección y respuesta en una red)

SNMP: Simple Network Management System (Protocolo Estándar de Administración y Gestión en Red)

ICMP: Internet Control Message Protocol (Protocolo de Control de Mensajes de Internet)

Ethernet: Estándar de red local para computadores (IEEE 802.3).

SSH: Secure Shell (Intérprete de órdenes seguro)

VNC: Virtual Network Computing (Computación Virtual en Red)

NMS: Network Management System (Sistema Administrador de Red)

ISP: Internet Service Protocol (Protocolo de Servicio de Internet)

SLA: Service Level Agreement. (Acuerdo Nivel de Servicio)

NFS: Network File System (Sistema de archivos en red)

SMTP: Simple Mail Transfer Protocol (Protocolo simple de transferencia de correo)

HTTP: Hypertext transfer protocol (Protocolo de transferencia de hipertexto)

SMS: Short Message Service (Servicio de Mensajes Cortos)

E/S: Entrada o salida

SQL: Structured Query Language (Lenguaje de consulta estructurado)

UDP: User Data Protocol (Protocolo de Datagrama de Usuario)

TCP: Transer Control Protocol (Protocolo de control de transmisión)

HA: High Availability (Gran Disponibilidad)

WMI: Windows Management Instrumentation (Instrumentación de gestión de Windows)

LAN: Local Area Network (Red de área local)

WAN: Wide Area Netowrk (Red de área amplia)

6 REQUISITOS INICIALES

Basándonos en el primer encuentro con el tutor, los requisitos iniciales acordados por ambas partes son los siguientes:

- Monitorización de los dispositivos activos en la red a través del protocolo ICMP
- Monitorización de los dispositivos activos en la red a través del protocolo SNMP
- Monitorización de la calidad del servicio ofrecido por el ISP
- Creación de un sistema de alarmas para notificar al Administrador de red.
- Visualización de los datos.

Estos requisitos son ampliamente analizados en el capítulo 5, “Especificaciones del Sistema”.

7 ALCANCE DEL PROYECTO

Este proyecto se aplica a la definición e implementación de un sistema de monitorización de dispositivos y servicios en una red doméstica.

Este proyecto incluye:

- Estudio teórico de una red doméstica común.
- Especificación de requisitos del sistema de monitorización y administración de eventos acordados entre el tutor y alumno.
- Estudio de una plataforma económica basada en una arquitectura novedosa.
- Estudio teórico sobre protocolo de red SNMP

- Estudio teórico sobre protocolo de red ICMP
- Análisis de las alternativas en el despliegue del proyecto.
- Descripción de las soluciones propuestas
- Guía de instalación y actualización del proyecto.
- Establecimiento de mediciones de recursos para la consecución del proyecto.
- Elaboración de un presupuesto para la consecución del proyecto.

8 ESTUDIO DE ALTERNATIVAS Y VIABILIDAD

En la actualidad se han creado múltiples programas y aplicaciones referentes a la monitorización de redes, algunos son desarrollos sencillos mientras que otros trabajan de forma más compleja. La forma de trabajo de estas herramientas puede variar, esto dependerá de los protocolos que usen para obtener la información requerida y la forma en que se despliega esta información, dependiendo de la complejidad de la herramienta.

8.1 ESTUDIO DEL SOFTWARE A UTILIZAR

En este capítulo se evalúan algunas de las herramientas que se encuentran en el mercado, algunas cuentan con licencia de software libre y otras son propietarias.

8.1.1 Nagios

Es un sistema de código abierto para la monitorización de redes y se ha convertido en una herramienta muy popular entre los administradores de red. Monitoriza servidores y servicios que le sean especificados notificando los cambios que se hayan producido en los dispositivos. Fue diseñado originalmente para sistemas Linux, pero también es usado en sistemas operativos Windows, Mac OS X, CentOS, etc. Tiene licencia GNU publicada por la Free Software Foundation (Nagios Enterprises, LLC, 2009).

Algunas de las características que monitoriza Nagios son los servicios de red, como los protocolos SMTP, POP313, HTTP, ICMP14, etc. y los recursos de los servidores (carga del procesador, porcentaje de uso de los discos duros, etc.). Tiene la facilidad de poder crear simples “plugins” (pequeño programa que interactúa con alguna aplicación del servidor para obtener una función o información específica), que permiten a los usuarios crear sus propios parámetros para ser verificados.

Así también tiene la posibilidad de definir la jerarquía de la red, permitiendo distinguir entre host caídos y host inaccesibles. Notifica vía correo electrónico, mensajes de texto SMS, o cualquier otro método definido por el usuario a un grupo de contactos cuando ocurre un problema con un servicio o en un host.

Contiene una interfaz web opcional, para observar el estado de la red actual, notificaciones, historial de problemas, archivos logs, etc.

- Funcionamiento

Como se mencionó, Nagios funciona a través de plugins. Para monitorizar un sistema Linux o Unix remoto, se utiliza el plugin NRPE (Nagios Remote Plugin Executor). Éste funciona para recopilar los datos de los recursos locales de la máquina que se desea monitorizar

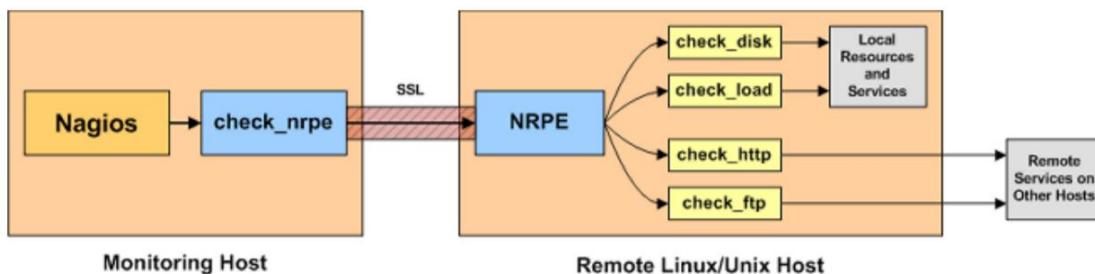


Ilustración 5: Arquitectura NRPE Nagios

Cuando Nagios necesita monitorizar uno de los recursos del sistema remoto ejecuta el módulo `check_nrpe` e indica que servicio necesita ser monitorizado. El `check_nrpe` contacta al demonio NRPE instalado en la máquina remota a través de una conexión SSL protegida. El demonio ejecuta el plugin apropiado para verificar el servicio o recurso solicitado. El resultado se pasa al demonio NRPE y de regreso al módulo `check_nrpe` que a su vez, entrega el resultado a Nagios.

Lo mismo sucede si se requiere monitorizar sistemas Windows, sólo que se necesita un módulo diferente, en este caso `check_nt`. Dentro de la máquina remota Windows se necesita un plugin para interactuar con el servidor Nagios, como puede ser NSClient, NC_Net o algun otro.

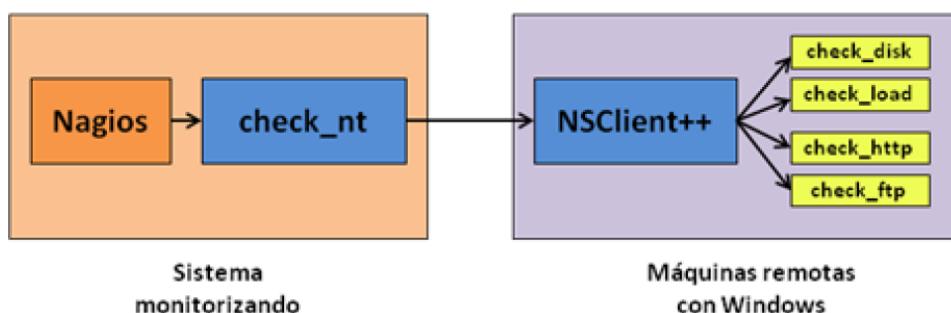


Ilustración 6: Arquitectura NSClient++ Nagios

La instalación y configuración de Nagios se realiza a través scripts en modo consola.

En la parte gráfica, Nagios ofrece un menú en la parte izquierda de la pantalla, donde se encuentran las diferentes características que ofrece Nagios, como el estado de los hosts, reportes con histogramas y las alertas. En la parte central de la venta, se muestra a detalle cada parámetro monitorizado, así como también un pequeño cuadro resumen, con el estado en el que se encuentran todos los dispositivos.

The screenshot displays the Nagios XI web interface with the following sections:

- Left Sidebar:**
 - Quick View: Home Dashboard, Tactical Overview, Birdseye, Operations Center, Operations Screen, Open Service Problems, Open Host Problems, All Service Problems, All Host Problems, Network Outages.
 - Details: Service Detail, Host Detail, Hostgroup Summary, Hostgroup Overview, Hostgroup Grid.
 - Metrics: BPI, Metrics.
 - Graphs: Performance Graphs, Graph Explorer.
 - Maps: BBmap, Google Map, Hypermap, Minemap, Nagvis, Network Status Map, Legacy Network Status Map.
 - Incident Management: Latest Alerts, Acknowledgments, Scheduled Downtime, Mass Acknowledge, Recurring Downtime, Notifications.
 - Monitoring Process: Process Info, Performance, Event Log.
- Host Status Summary:** Shows counts for Up (53), Down (81), Unreachable (3), Pending (0), Unhandled (64), Problems (64), and All (117). Last Updated: 2017-10-05 16:06:57.
- Service Status Summary:** Shows counts for Ok (228), Warning (12), Unknown (84), Critical (271), Pending (2), Unhandled (366), Problems (367), and All (595). Last Updated: 2017-10-05 16:06:57.
- Status Summary For All Host Groups:** A table showing host groups, hosts, and services status. Examples include:

Host Group	Hosts	Services
All EMC SAN Hosts (all_emc_hosts)	2 Up	4 Ok, 1 Critical
Firewalls (firewalls)	1 Up	1 Ok
Host Deadpool (host-deadpool)	1 Up	1 Down, 6 Ok, 1 Critical, 1 Unreachable
Linux Servers (linux-servers)	5 Up	52 Ok, 3 Warning, 9 Unknown, 6 Critical
new group (new group)	8 Up	58 Ok, 3 Warning, 9 Unknown, 21 Critical
Printers (printers)	3 Up	2 Ok, 2 Critical, 2 Unreachable
Websites (websites)	5 Up	20 Ok, 2 Warning, 2 Critical
Windows Servers (windows-servers)	2 Down	6 Critical
- Top Alert Producers Last 24 Hours:** A horizontal bar chart showing the top alert producers over the last 24 hours. The top three are Port:24-Gigabit---Level Bandwidth (Switch 1), Port:-1-Gigabit---Level Bandwidth (Switch 1), and Port:23-Gigabit---Level Bandwidth (vs1.nagios.com).
- Metrics Overview:** A table showing disk usage for localhost, vs1.nagios.com, and exchange.nagios.org.
- Disk Usage:** A table showing disk usage details for localhost, vs1.nagios.com, and exchange.nagios.org.
- Bottom Footer:** Nagios XI 5.4.10, Check for Updates, About, Legal, Copyright © 2008-2017 Nagios Enterprises, LLC.

Ilustración 7: Interfaz Web Nagios

Nagios también cuenta con un esquema gráfico donde se pueden ver los equipos respecto a su distancia y grupo al que pertenecen. Así mismo posee un historial donde se ven los errores, alertas y eventos no fallidos.

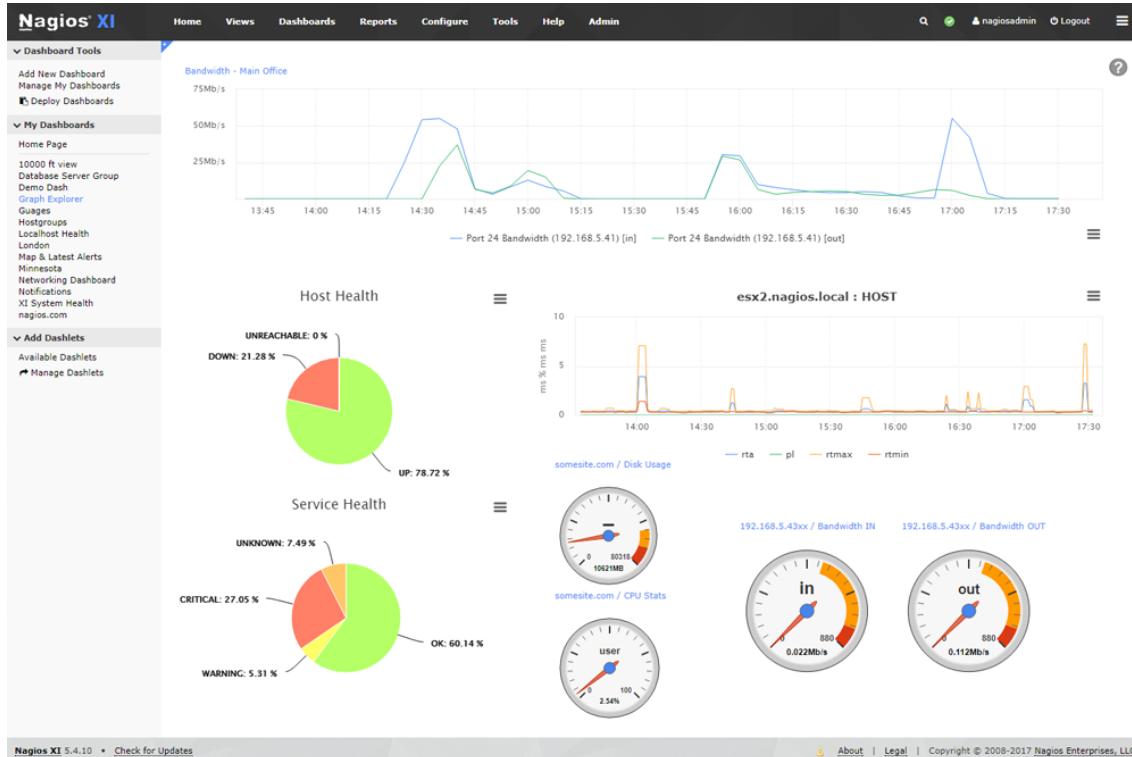


Ilustración 8: Gráficos Nagios

- Ventajas

Nagios es una herramienta de monitorización completa. Una de sus ventajas claras, es el uso de plugins prediseñados para revisar el estado de los servicios y la capacidad para crear los propios para monitorizar aspectos específicos de acuerdo a las necesidades del administrador. Tiene soporte para monitorizar miles de dispositivos y equipos. Envía alertas vía correo electrónico y mensajes SMS en caso de que algún equipo falle. Tiene una interfaz gráfica web nada compleja.

- Desventajas

La configuración de los equipos no es nada fácil, ya que todo se realiza vía consola mediante archivos de configuración donde se agregan, modifican, y dan de baja los equipos, siendo ésta su peor desventaja ya que no se pueden hacer estas modificaciones desde el entorno Web. Exige mayor trabajo por parte del usuario para su configuración y mantenimiento.

8.1.2 Zabbix

Zabbix es un software libre que monitoriza numerosos parámetros de una red y el rendimiento, disponibilidad e integridad de los servidores y equipos (Zabbix SIA, 2001).

Zabbix propone distintas formas de monitorización. Por un lado, ofrece recopilación de datos a través de agentes recolectores disponibles en todas las plataformas (Linux, Mac y Windows), por otro lado, ofrece integración con un servicio de monitorización más “nativo” a través de SNMP, JMX e IPMI.

Entre otras muchas cosas es capaz de descubrir los nodos en un rango de IPs mediante agentes SNMP. Puede monitorizar servicios remotos, tiene soporte para traps SNMP y SNMP v1, v2 y v3.

Además de proporcionar extensa información sobre la máquina: procesos de carga, actividad en la red, actividad en disco, parámetros del sistema operativo, servicios de red (SMTP, HTTP), E/S.

Esta herramienta además ofrece funciones de informes y visualización de datos basadas en la información almacenada en su base de datos compatible con MySQL, PostgreSQL, SQLite, Oracle o IBM DB2.

La forma de acceder a todos los informes y estadísticas de Zabbix, así como los parámetros de configuración, es a través de una interfaz web desarrollada en PHP y JavaScript.

Desde ella se puede realizar todas las configuraciones posibles que tiene Zabbix. Tiene una amplia gama de gráficos para poder utilizar en nuestros mapas personalizados, y además, podemos añadir tanto nuevos iconos como fondos.

Zabbix utiliza un mecanismo de notificación flexible que permite a los usuarios configurar alertas basadas en correo electrónico o SMS para prácticamente cualquier evento.

Otra de las opciones destacables de Zabbix es la capacidad de tomar acciones que le indiquemos. No sólo es capaz de enviarnos un SMS o un correo electrónico entre otras, sino que cuando algo falla, también es capaz de ejecutar un script para poder levantar un servicio caído.

Con una configuración adecuada, Zabbix puede desempeñar un papel importante en la monitorización de la infraestructura de TI, ya sean organizaciones pequeñas con algún servidor como grandes empresas con multitud de servidores.

- Funcionamiento

La arquitectura más comúnmente utilizada en los sistemas que hacen uso de Zabbix es una arquitectura Agente-Servidor, basada en un servidor que obtiene los datos a través de consultas ejecutadas sobre una serie de agentes. Un agente es simplemente un programa escrito en C e instalado en un host que se desea monitorizar.

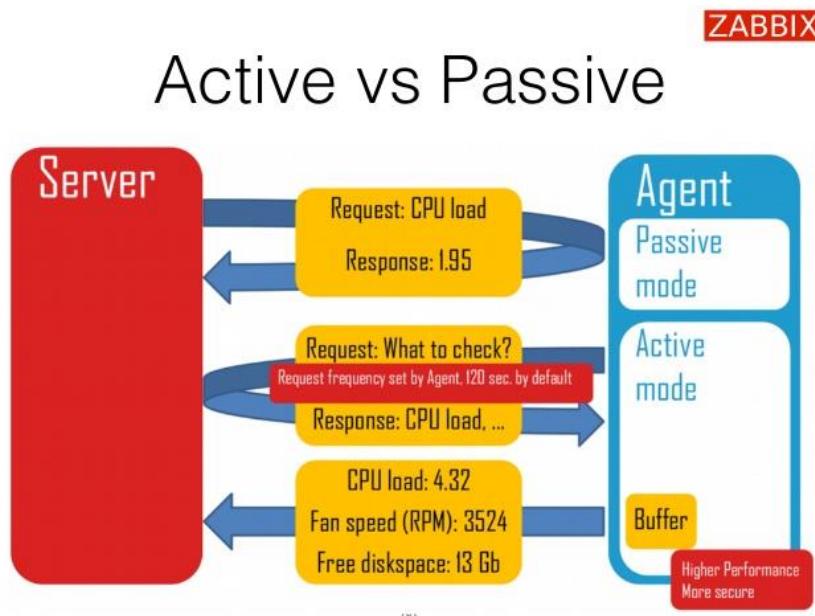


Ilustración 9: Arquitectura Agente-Servidor Zabbix

En cualquier caso, no siempre es un requisito indispensable disponer de un agente instalado en el host a monitorizar. De hecho, nos encontraremos con equipos en los que, dadas sus características, no sea posible la instalación del agente. En esos casos Zabbix puede llevar a cabo comprobaciones sencillas sobre el equipo, como puede ser un ping o la comprobación de servicios http o ftp a través de su puerto establecido.

Sin embargo, si realmente se quieren explotar las posibilidades de monitorización que Zabbix ofrece, lo ideal es tener instalado un agente en el host siempre que sea posible, pues de esta manera se podrán recopilar y tratar muchos más datos con los que obtener una información mucho más fiable. Una vez instalado, el agente funciona como un servicio más en el host y, a través de llamadas al sistema, recopila la información que se necesita para el proceso de monitorización desde el servidor principal.

La comunicación entre agente y servidor es sencilla, y se realiza a través de los puertos 10050 y 10051, ambos TCP. El host donde está instalado el agente “escucha” las peticiones que el servidor envía, y lo hace a través del puerto 10050. Por su parte, una vez recopilados los datos por el agente, el servidor recibe éstos por su puerto 10051 local.

Una vez definida la arquitectura “interna” sobre la que sea asienta el funcionamiento de Zabbix, es turno de explicar cómo el usuario final se comunica con la herramienta en los procesos de configuración, visualización de datos, etc.

Zabbix dispone de un “frontend” o interfaz web, escrito en PHP, que nos proporciona varias opciones para visualizar los datos recogidos, desde listas de problemas y gráficos simples hasta mapas de red e informes más elaborados. La información con la que se trabaja en el frontend no es más que los datos proporcionados por otra capa importante en la arquitectura de Zabbix, el “backend”. En líneas generales, el backend es la capa que toma la información que los agentes han enviado al servidor (típicamente almacenada en una Base de Datos MySQL) para después suministrarla al frontend en el que se visualizarán los datos y enviará las correspondientes alertas que se desencadenen en función de la información recopilada.

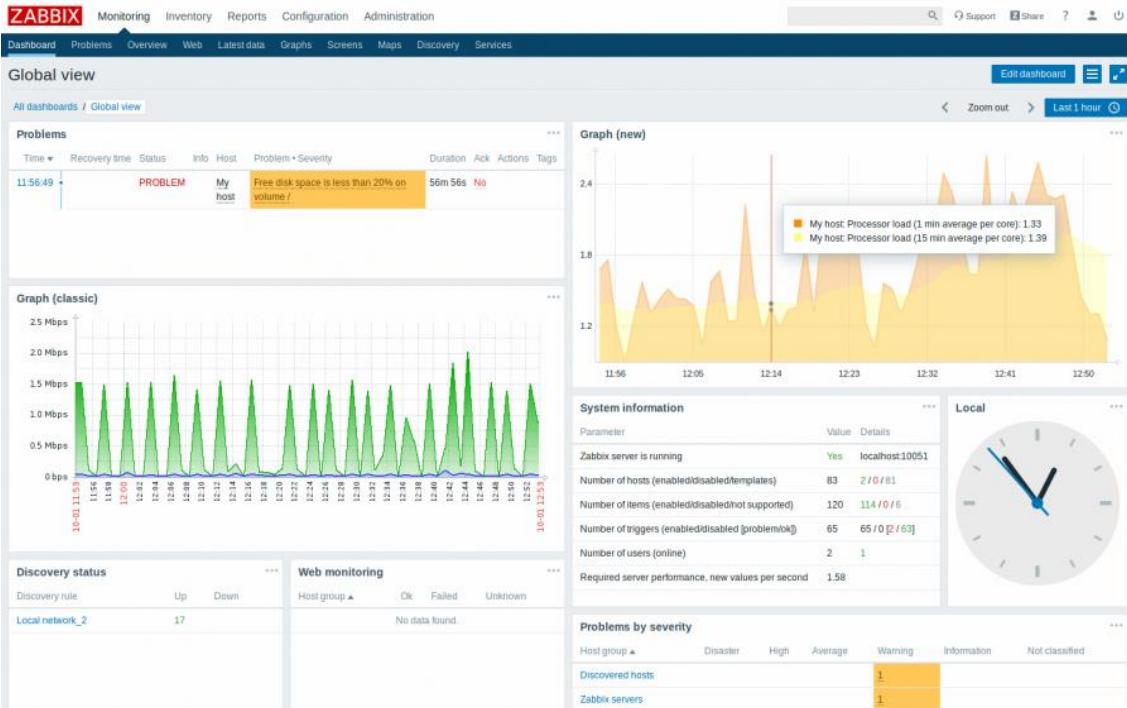


Ilustración 10: Interfaz Web Zabbix

- Ventajas

Una de las principales ventajas es que contiene distintas formas de monitorización SNMP, propio agente, JMX, IPMI.

No es necesario editar ficheros de configuración (a excepción de los ficheros de los agentes software y el servidor), cualquier parámetro de monitorización, cualquier alerta, cualquier notificación, se puede crear y modificar a través del frontend web.

Además, brinda grandes prestaciones para monitorización de sistemas Windows.

- Desventajas

La instalación inicial es un tanto complicada, desde que la comprensión inicial de los conceptos del funcionamiento de la herramienta exige un cierto tiempo hasta, por ejemplo, la necesidad de ejercer una configuración y ajustes para que la base de datos se encuentre operativa.

Por otra parte, la interfaz web tiene demasiada densidad de funcionalidades, causando confusión en la navegación para usuarios poco frecuentes.

El no disponer de una versión Enterprise hace que no crezca tanto su popularidad en cuanto a clientes importantes asociados a la herramienta

8.1.3 Cacti

Cacti es una aplicación muy vistosa que almacena toda la información necesaria para crear gráficas, estos datos provienen de una base de datos en MySQL. Esta aplicación es manejada por PHP y tiene soporte para SNMP que se complementa con RRDtool donde obtiene la información de los dispositivos mediante un recolector de datos que se ejecuta cada 5 minutos para posteriormente crear gráficas de todo tipo, desde el tráfico en la red hasta el uso de memoria de una computadora y sus particiones. Maneja usuarios, los cuales tienen distintos privilegios, como por ejemplo cambiar parámetros a las gráficas mientras que otros sólo pueden verlas (The Cacti Group, Inc., 2004)

- RRDtool

RRDtool es un sistema para almacenar y mostrar datos a través del tiempo. Tiene la característica que los datos se almacenan de manera compacta, ésta es la función de Round Robin. Algo de gran importancia es que la base de datos no crece con el tiempo. Con RRDtool se puede mostrar los datos fácilmente en forma de gráficos para distintos períodos de tiempo. De manera interna se recogen los datos a graficar con una frecuencia determinada y se almacenan en diferentes archivos, según su resolución.

En esta base de datos RRD (Round Robin Database) hay tres archivos RRA (Round-Robin Archive) para diferentes períodos de tiempo. Uno de los archivos recolecta la información generada durante los últimos 5 minutos. El archivo que contiene los valores para una hora se calculan a partir de 12 valores del anterior archivo RRA. Finalmente, el archivo para un día, tiene un valor que resume la medida para un día. Este se obtiene a partir de 24 valores del RRA de 1 hora. Cada uno de los archivos tiene un tamaño fijo, por lo que no crece en el tiempo. Existe un apuntador al último dato recogido. Los diferentes archivos se actualizan en paralelo a partir de las actualizaciones periódicas.

- Elementos de configuración

Cacti cuenta con una interfaz gráfica y una serie de menús amigables, la forma de configurarlo también es sencilla para el usuario, cuenta con distintos menús donde se puede configurar desde los usuarios y sus privilegios, agregar nuevos hosts, hasta agregar nuevos plugins y gráficas.

Básicamente Cacti cuenta con dos pestañas, la primera de ellas llamada “console” donde es posible administrar la herramienta y cambiar la configuración de ésta, y de “graphs” que es un menú en el cual podemos tener accesos a las gráficas de los hosts que ya se han configurado.

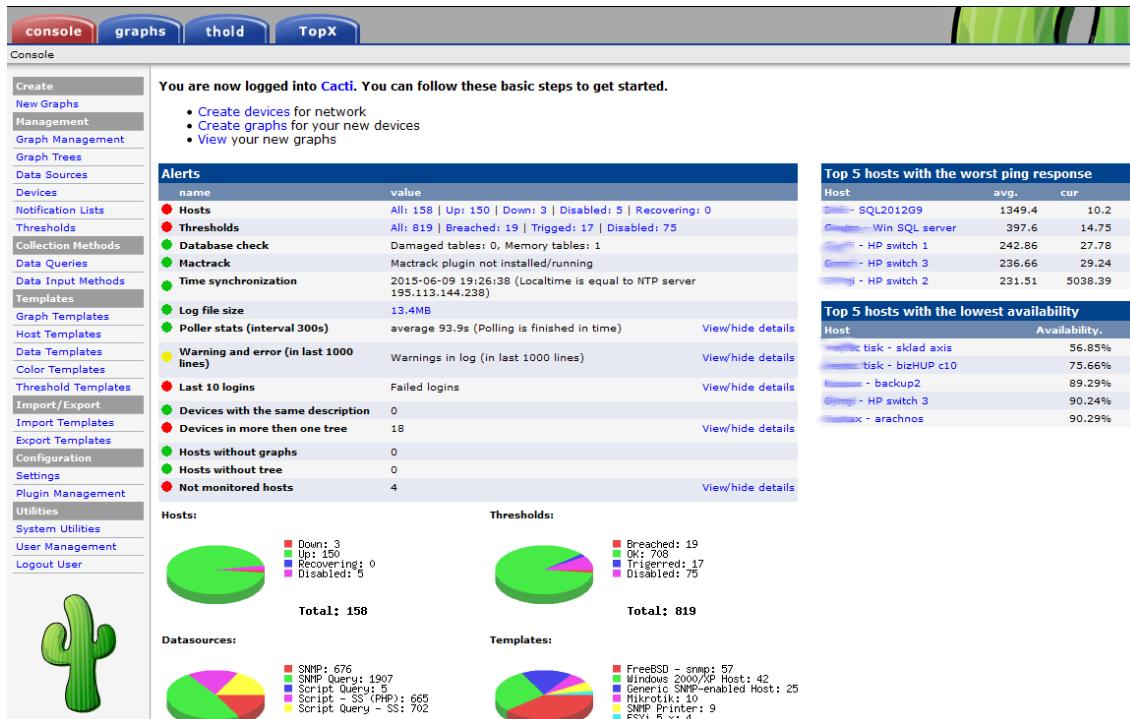


Ilustración 11: Interfaz Web Cacti

Cacti no sólo puede graficar los datos que nos proporcione el SNMP también puede graficar datos de diferentes scripts dándonos así una gran flexibilidad.

Los datos que nos proporciona Cacti son gráficas que tienen la característica de mostrar a detalle el dato que queremos con fecha, la hora exacta del evento, como los datos son proporcionados por un histórico, también nos puede almacenar los datos de todo un año, dándonos el pico máximo, el mínimo, el valor actual y el valor promedio. A todo esto, se le puede agregar que Cacti nos proporciona la opción de acercarnos al valor que se requiere, para ello utiliza un método de interpolación de datos muy exacto, que nos da una idea de cómo estaba el valor en ese momento.

- Ventajas

La principal ventaja que maneja Cacti, es su capacidad para desplegar la información de los dispositivos que se monitorizan de forma gráfica. Se puede ver el comportamiento, la disponibilidad, el almacenamiento, temperatura y otras características más que se manejan en esta aplicación.

Gracias al protocolo SNMP es posible que cualquier dispositivo sea monitorizado por Cacti. Se puede hacer uso de las tres versiones de este protocolo (SNMPv1, SNMPv2 y SNMPv3). Soporta el manejo de un gran número de dispositivos para ser monitorizados al mismo tiempo, más de 10,000. Tiene la capacidad de manejar usuarios para acceder al sistema y otorgar diferentes privilegios a los mismos, desde el usuario que sólo puede ver las gráficas hasta el que tiene el privilegio de administrador. Guarda un historial de cada parámetro de hasta un año para hacer un comparativo anual.

- Desventajas

Se mostró que Cacti tiene diversas cualidades y ventajas, pero también posee debilidades y carencias. La información desplegada en las gráficas no siempre llega a ser clara y se necesita de un procesamiento extra por parte del usuario o administrador de Cacti para interpretar los datos. Es cierto, las gráficas son muy bonitas gracias a las herramientas que se utilizan para ser concebidas, pero no sirven de mucho si no se sabe qué es realmente lo que nos muestran.

8.1.4 Pandora FMS

Pandora FMS es un software de monitorización orientado a todo tipo de entornos. FMS es el acrónimo de “Sistema de monitorización Flexible”. Se emplea para monitorizar sistemas, aplicaciones o dispositivos de red (Pandora FMS, 2012)

Esta herramienta está orientada a grandes entornos, y permite gestionar con y sin agentes, miles de sistemas, por lo que puede administrar grandes clusters, centros de datos y redes de todo tipo a través de un soporte multiusuario con diferentes perfiles.

Pandora FMS ofrece una monitorización de sistemas de forma remota mediante TCP/IP o de forma local a través de agentes. Los agentes son abiertos y se pueden adaptar para la monitorización de cualquier elemento. Cabe destacar la disponibilidad de una consola de recepción de traps SNMP en tiempo real.

Permite conocer el estado de cada elemento de un sistema a lo largo del tiempo, ya que almacena los datos recogidos en la monitorización en una Base de Datos relacional.

Al igual que las herramientas descritas anteriormente elabora gráficos combinados con datos de varios tipos y con la peculiaridad de que es capaz de generar informes SLA.

Otro aspecto común es la configuración de alertas para envío de mensajes vía e-mail o SMS.

Pandora FMS está publicado bajo licencia GPL2. Es Open Source aunque dispone de una versión específica para empresas, con una licencia comercial, llamada Enterprise.

- Funcionamiento

Pandora FMS es modular y descentralizado, siendo su parte más esencial la base de datos (MySQL es la base por defecto teniendo la posibilidad de instalar Bases de Datos PostgreSQL y Oracle) en donde se almacena la información. Todos los componentes de Pandora FMS se pueden replicar y funcionar en un entorno de HA puro (Activo/Pasivo) o en un entorno clusterizado (Activo/Activo con balanceo de carga).

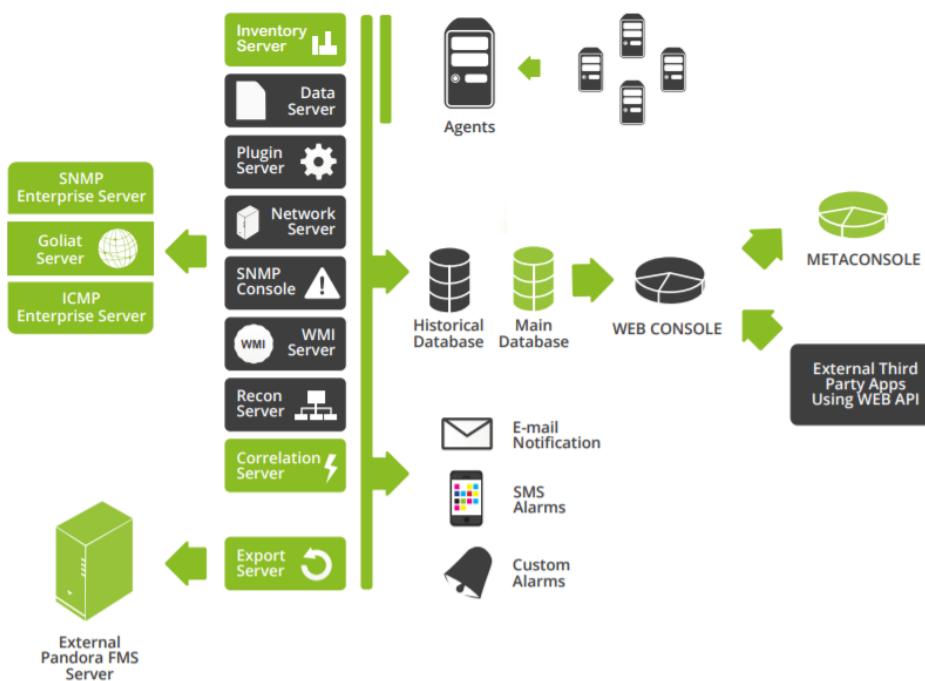


Ilustración 12: Arquitectura Pandora FMS

Pandora FMS consta de diversos elementos (agentes) que se encargan de recoger y procesar los datos de los servidores.

Los servidores de Pandora FMS son los elementos encargados de realizar las comprobaciones existentes. Ellos las verifican y cambian el estado de estas en función de los resultados obtenidos. También son los encargados de disparar las alertas que se establezcan para controlar el estado de los datos. Los servidores de Pandora FMS están siempre en funcionamiento y verifican permanentemente si algún elemento tiene algún problema y si está definido como alerta. Si ocurre esto, éste ejecuta la acción definida en la alarma, tal como enviar un SMS, un correo electrónico, o activar la ejecución de un script. Pandora FMS gestiona automáticamente el estado de cada servidor, su nivel de carga y otros parámetros. El usuario puede monitorizar el estado de cada servidor, a través de la sección de estado de servidores de la consola Web.

Al referirnos a un agente Pandora FMS, podemos distinguir 3 formas de obtener los datos:

- Agente: El agente de Pandora FMS es simplemente un elemento organizativo que se crea con la consola Web de Pandora FMS. El cual está asociado a un grupo de módulos o elementos individuales de monitorización. También permite que dicho agente pueda tener de forma opcional asociadas una o más direcciones IP.
- Agente de Software: El agente de Software es el agente que se instala en una máquina o equipo remoto, el mismo es completamente diferente al del servidor o al de la consola Web de Pandora FMS. El agente de Software permite obtener la información local del equipo donde se está ejecutando.
- Agente Físico o de Hardware.

Una vez analizadas las funcionalidades y arquitectura interna de Pandora FMS, procederemos a explicar cómo esta herramienta interacciona con el usuario final a través de su Consola Web.

Esta consola es la interfaz de usuario de Pandora FMS. Permite operar y administrar a diferentes usuarios, con diferentes privilegios, controlar el estado de los agentes, ver información estadística, generar gráficas y tablas de datos, así como gestionar incidencias con su sistema integrado. También es capaz de generar informes y definir de forma centralizada nuevos módulos, agentes, alertas y crear otros usuarios y perfiles.

La consola Web está programada en PHP y no requiere por parte del usuario final la instalación de ningún software adicional. No obstante, las gráficas también están disponibles en FLASH y para poder verlas en este formato será necesario el complemento de FLASH para su navegador; puede accederse desde cualquier plataforma moderna que soporte HTML y CSS.

La consola Web a su vez, puede ejecutarse en múltiples servidores, esto es, podemos tener tantas consolas Web como queramos, tanto para repartir carga como para facilitar el acceso por problemas logísticos (grandes redes, numerosos grupos de usuarios diferentes, diferencias geográficas, diferencias administrativas, etc.). Su único requisito es poder acceder al contenedor de datos donde Pandora FMS almacena todo: la base de datos y en el caso de la versión “Enterprise”, acceder al repositorio de configuraciones de los agentes de forma sincronizada (vía NFS).

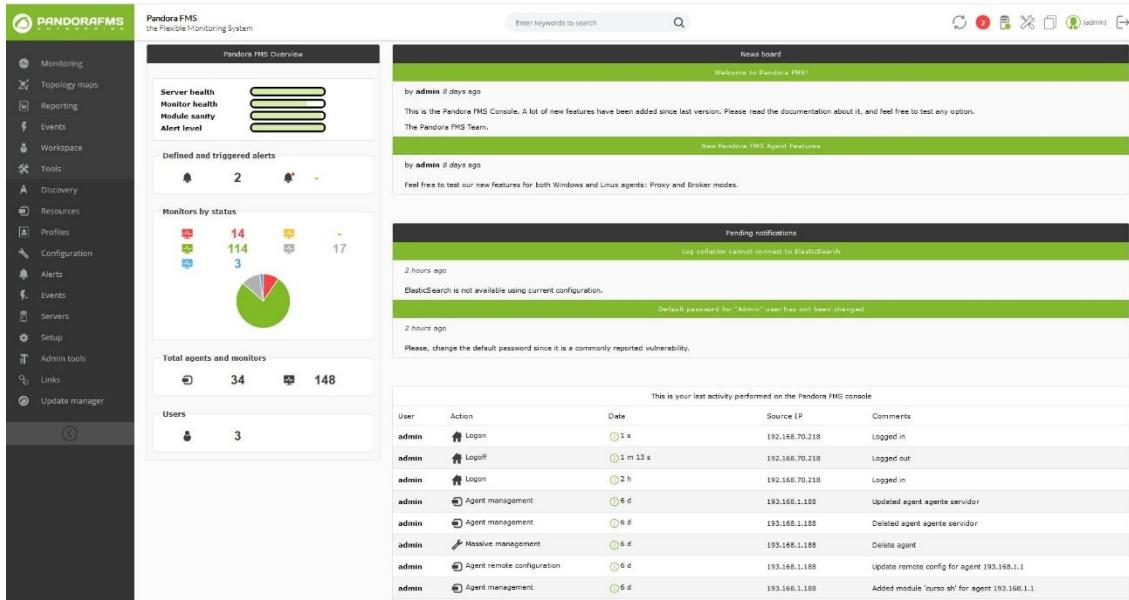


Ilustración 13: Interfaz Web Pandora FMS

- Ventajas

La principal característica de Pandora FMS es su alta disponibilidad y escalabilidad. Pandora FMS tiene redundancia sobre todos sus sistemas. Se puede crear cualquier cantidad de servidores o equipos. Los agentes también disponen de mecanismos para poder enviar a varios servidores, por si falla uno de ellos.

Otro aspecto destacable es la monitorización WMI. Se puede obtener información de cualquier Servidor Windows sin tener que instalar software. Esto incluye tanto disponibilidad, rendimiento como información de inventario.

Pandora soporta WMI de forma nativa, incluyendo el escaneo de servidores para obtener información común (CPU, Disco, Memoria) de forma automatizada.

- Desventajas

Quizás la carencia más notoria de este software sea que la gran cantidad de los elementos a monitorizar se establecen a través de módulos que se definen de forma textual en los ficheros de configuración de los agentes. Una tarea algo tediosa durante la configuración inicial del entorno.

8.1.5 Tabla comparativa de las herramientas

		Nagios	Zabbix	Cacti	Pandora FMS
Recolección de datos		ICMP, SNMP, SSL, SSH	ICMP, SNMP, JMX, IPMI, SSH, Agentes	ICMP, SNMP, RRDTool	ICMP, SNMP, WMI, Agentes
Alarmas	Correo	Sí	Sí	Sí	Sí
	SMS	Sí	Sí	Sí	Sí
Sistema Operativo		Linux Windows	Linux Windows	Linux Windows	Linux Windows

	Mac OS*	Mac OS		Mac OS*
Tipo de Software	Libre	Libre	Libre	Libre + Versión Empresarial
Permisos a usuarios	Sí	Sí	Sí	Sí
Generación de reportes	Sí	Sí	Sí	Sí

Tabla 4: Tabla comparativa herramientas monitorización.

*Se necesitan de plugins adicionales.

A lo largo de este apartado hemos analizado sólo una muestra del total de herramientas que existen actualmente, ya que sería imposible analizar todas las existentes al detalle. Estas herramientas funcionan de manera similar con sus propias características, algunas son más elaboradas que otras, unas de ellas con licencias comerciales mientras que otras con licencia de software libre, pero la mayoría contiene funcionalidades similares que nos ayudan a cuidar la integridad y disponibilidad de la red.

Tras el análisis de estas diferentes herramientas cabe destacar que ninguna se ajusta debidamente a los requerimientos de nuestro proyecto aportando funcionalidades excesivas, complejidades para el usuario final, consumo de recursos inmoderado y gasto energético desmedido.

Es por ello que finalmente se ha optado por crear una solución personalizada que se ajuste a los requerimientos del proyecto y no sobrepasen esos factores anteriormente mencionados.

9 DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

9.1 ARQUITECTURA DE LA SOLUCIÓN PROPUESTA

A lo largo de este apartado, comentaremos los aspectos más técnicos relativos a la arquitectura del proyecto.

Comenzaremos detallando la arquitectura de red desde un punto de vista general, para ir profundizando en cada uno de los apartados más significativos de la misma.

Partimos de la siguiente topología inicial con el detalle de nuestro dispositivo central que actuará como sistema administrador de red, NMS (por sus siglas en inglés “Network Management Station”).

A continuación, se muestra un ejemplo de la topología de la solución propuesta:

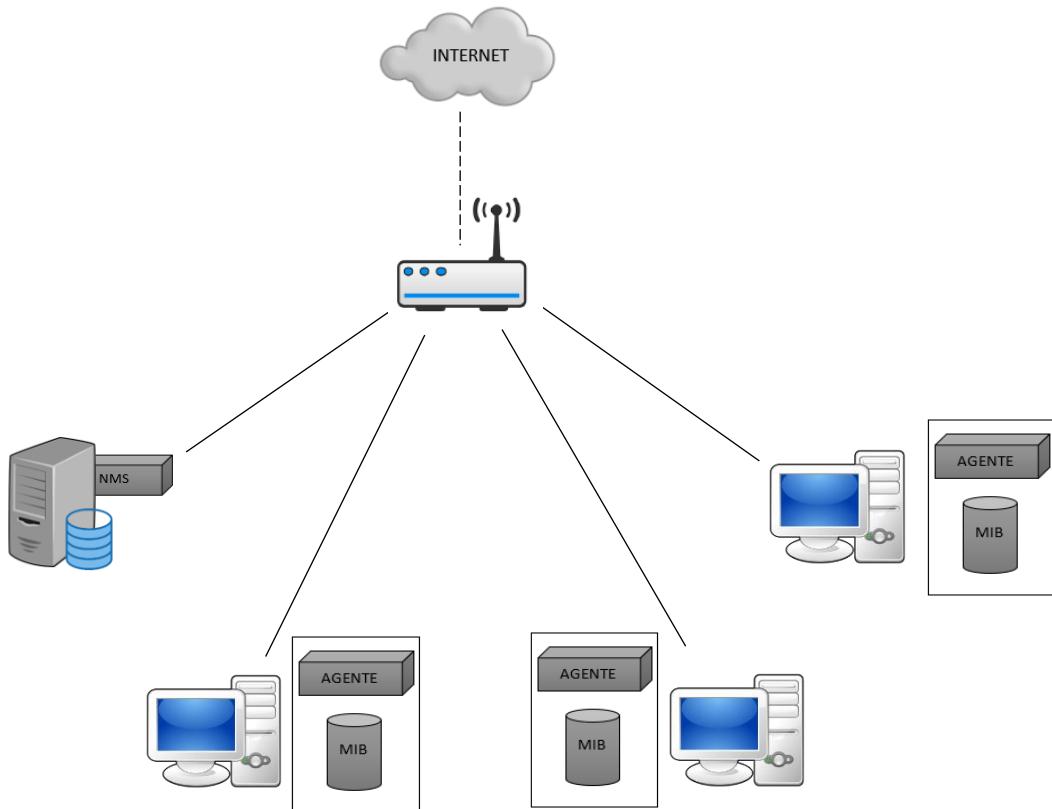


Ilustración 14: Arquitectura red doméstica

Nótese que se ha optado por un esquema de direccionamiento físico para así evitar posibles problemas con el direccionamiento dinámico que pudieran afectar al histórico de monitorización de un equipo o simplemente dificultando la tarea de identificación de los diferentes dispositivos activos en la red.

Para la consecución del proyecto se ha optado por una arquitectura basada en un modelo cliente-servidor.

Este modelo divide la carga de trabajo entre los proveedores de recursos o servicios (servidores) y los solicitantes de servicio (clientes).

En la arquitectura cliente-servidor, cuando el cliente manda una petición al servidor a través de la red, el servidor la acepta devolviendo los paquetes de datos al cliente. El cliente no comparte ninguno de sus recursos.

Extrapolando este modelo a nuestro proyecto nos encontramos con un modelo en el que nuestro NMS hace las funciones de servidor y los agentes SNMP instalados en los diferentes dispositivos monitorizados de cliente.

Sin embargo, por temas funcionales, no siempre actúa como servidor el NMS y como clientes los agentes de los dispositivos administrados, sino que por ejemplo para la monitorización de los nodos, es el dispositivo central, el que actúa como cliente realizando la petición de datos y estos como servidores devolviendo la información.

Por lo tanto, dentro de esta arquitectura de red podemos diferenciar tres actores principales:

- Sistemas administradores de red (NMS)
- Agentes
- Dispositivos administrados

9.1.1 NMS (Raspberry Pi)

Se trata del núcleo del proyecto, y consta de los scripts de Python encargados de medir nuestra calidad del ISP, realizar las peticiones ICMP/SNMP a los clientes y gestionar toda la información recolectada.

Desde nuestro sistema administrador de red podremos visualizar la información obtenida de nuestros hosts mediante peticiones directas SNMP a través de su servidor web instalado.

Por otro lado, también será el encargado de administrar el servicio de Traps SNMP por el cual los hosts alertan a esta estación central si alguno de los valores parametrizados supera el umbral establecido.

Además, velará por la seguridad de nuestra red comprobando que hosts están conectados en cada momento haciendo uso del protocolo ICMP.

Por último, gestionará un histórico de nuestra calidad de servicio ofrecido por nuestro proveedor de internet, ISP.

El acceso se lleva a cabo desde SSH o RFB desde la misma red interna. Tiene establecido unas credenciales de usuario para evitar posibles accesos no autorizados.

9.1.2 Agente SNMP

El agente de gestión se encuentra procesándose en cada uno de los dispositivos administrados. Es un software que se comunica con la entidad gestora para transmitirle información. Además, también permite manipular los objetos gestionados. Es decir, a través del agente, desde la entidad gestora se pueden ejecutar acciones en el dispositivo gestionado.

Evidentemente, habrá tantos agentes como dispositivos administrados se quieran tener. Estos dispositivos actúan con el rol de servidor de datos de administración al ponerlos a disposición del cliente NMS. Los datos los almacena el agente en forma de su propia base de datos MIB (Management Information Base).

Se comunican con el NMS a través de mensajes de solicitud-respuesta (solicitud del NMS por sondeo, respuesta del agente) y de “traps” del agente desencadenados por algún evento en el dispositivo administrado.

El agente en Linux es el demonio “snmpd” y se puede instalar con “net-SNMP”.

En Windows se tiene que activar un servicio del sistema, SNMP de WMI (Windows Management Instrumentation). WMI también es el Protocolo de Windows para WBEM (Web Based Enterprise Management).

La configuración del agente como del resto del proyecto es tratado en el apartado 23, “Manual de Usuario”.

9.1.3 Dispositivos administrados

Se trata de todos los dispositivos que se considera que hay que gestionar, y por lo tanto se necesita recabar información sobre ellos.

Los objetos administrados (o gestionados) son los elementos de los dispositivos administrados de los que se recoge la Información de administración. Puede tratarse, por ejemplo, de la ocupación de memoria RAM, del consumo de CPU, etc. Pueden ser tanto elementos de hardware como de software.

El dispositivo administrado tiene la responsabilidad de ir recolectando y almacenando la información de sus objetos administrados para poner esa Información a disposición del NMS. Toda esa Información se almacena en una base de datos MIB (Management Information Base) distribuida por todos los dispositivos administrados, donde cada uno es responsable del mantenimiento de la Información de sus propios objetos.

Cuando se define un objeto gestionado se deben incluir las operaciones permitidas sobre él. El estado o las propiedades del objeto pueden determinar dichas operaciones.

9.2 FUNCIONAMIENTO

Este proyecto ha sido desarrollado en base a un objetivo. Permitir que los administradores puedan administrar los nodos, como los servidores, las estaciones de trabajo, routers, etc. En una red IP doméstica. Durante este apartado se abordará el funcionamiento del mismo.

9.2.1 Monitorización LAN

El proceso de recolección de la información comienza con el agente SNMP instalado en el dispositivo administrado. Estos agentes recopilan información sobre los objetos y su funcionamiento almacenándola en su propia base de datos MIB (Management Information Base).

La MIB organiza variables de manera jerárquica. Las variables MIB permiten que el software de administración controle el dispositivo de red. Formalmente, la MIB define cada variable como una ID de objeto (OID).

Las OID identifican de forma exclusiva los objetos administrados en la jerarquía de la MIB. La MIB organiza las OID según estándares RFC en una jerarquía de OID, que se suele mostrar como un árbol.

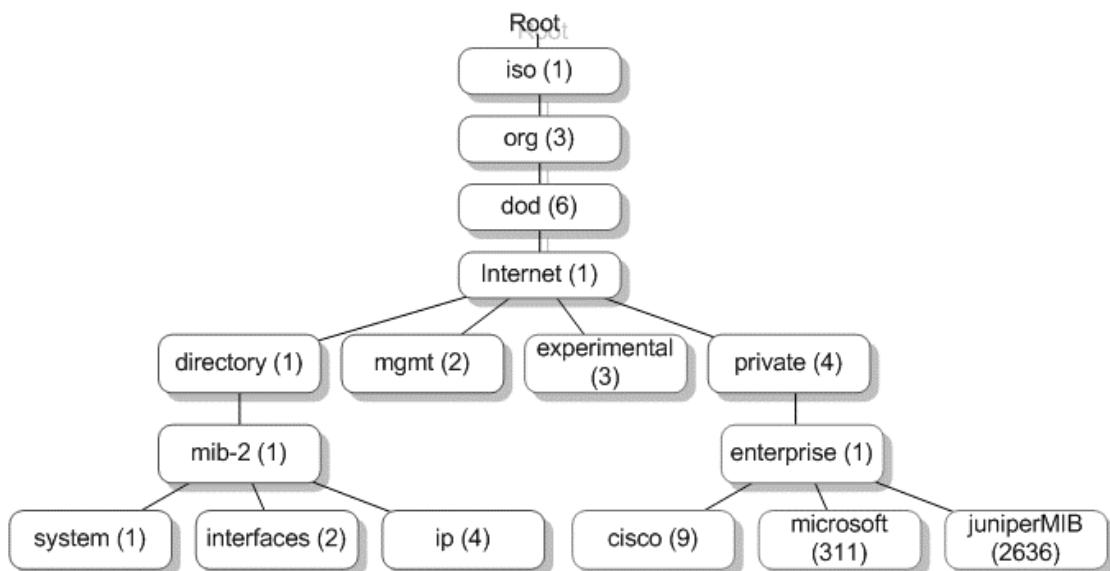


Ilustración 15: Management Information Base SNMP

El árbol de la MIB para un dispositivo determinado incluye algunas ramas con variables comunes a varios dispositivos de red y algunas ramas con variables específicas de ese dispositivo o proveedor.

El siguiente paso consiste en comunicar nuestro NMS con los diferentes dispositivos administrados con el fin de obtener y representar los diferentes datos recolectados por el agente.

En primer lugar, para que SNMP funcione NMS debe tener acceso a la MIB. Para asegurar que las solicitudes de acceso sean válidas, debe haber cierta forma de autenticación.

Para la consecución de nuestro proyecto hemos hecho uso de la versión 2c de SNMP, es decir, SNMPv2c. En esta versión la autenticación se lleva a cabo a través de cadenas de comunidad que controlan el acceso a la MIB. Las cadenas de comunidad son contraseñas de texto no cifrado.

Quizás no sea la versión más segura, pero teniendo en cuenta el entorno doméstico en el que nos encontramos se ha optado por la compatibilidad y facilidad de implementación en lugar de la seguridad.

Una vez definida nuestra cadena de comunidad con la que autenticar el NMS con los diferentes dispositivos administrados detallaremos las comunicaciones LAN.

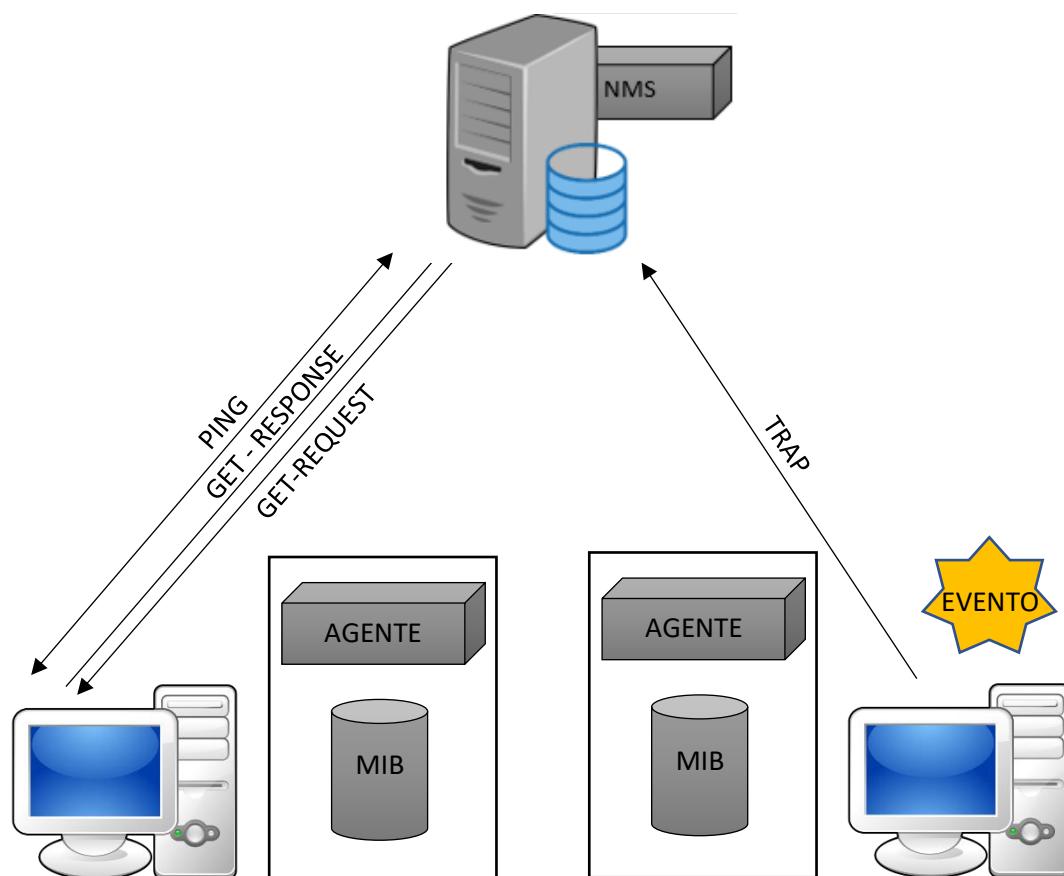


Ilustración 16: Diseño del flujo de trabajo entre dispositivos LAN

De estas comunicaciones se encargarán nuestros scripts Python instalados en el NMS haciendo uso de las operaciones PING, GET y TRAP de los protocolos ICMP y SNMP respectivamente.

9.2.1.1 *Ping*

Es la operación más simple de todas las de TCP/IP. Envía uno o más datagramas IP a un host de destino especificado pidiendo una respuesta y midiendo el tiempo de ida y vuelta. La emitimos de manera periódica desde nuestro EMS a todos los dispositivos de la red para tener un control, en todo momento, de que dispositivos están activos en nuestra red.

Por otra parte, también es utilizada de manera transversal para medir la latencia de nuestro ISP.

9.2.1.2 *Get (request-response)*

La petición get-request es iniciada por el NMS, el cual envía la petición al agente. El agente recibe la petición y la procesa. Si el agente es capaz de recolectar la información solicitada, recupera el valor de la variable de MIB solicitada y responde con get-response al NMS con ese valor.

9.2.1.3 *Trap*

Es el tipo de mensaje asíncrono que el agente SNMP envía al NMS para advertirle acerca de un evento que ha tenido lugar en un dispositivo administrado.

Este agente SNMP instalado en cada dispositivo administrado es el encargado de manejar los procesos relativos al sistema de alarmas. Esto es debido a que tanto las alertas como los avisos, no están controlados desde la aplicación central, sino que es cada nodo el que se responsabiliza de su propia monitorización en este aspecto. Es por ello que cada cierto intervalo de tiempo predefinido realiza las comprobaciones sobre su propio estado, comparándolo con los umbrales de alarma establecidos, y decidiendo si enviar un aviso, una alerta, o si todo está funcionando correctamente. En el caso de que alguno de los umbrales configurados se vea rebasado, el agente mandará un paquete UDP al NMS con la información relativa que ha hecho superar los valores predefinidos.

La filosofía de los trap es evitar el consumo excesivo de ancho de banda por los mensajes de administración. Por eso se envían sin acuse de recibo

9.2.2 Monitorización WAN

Detalladas las comunicaciones entre los diferentes dispositivos en nuestra red LAN pasaremos a explicar la tecnología con la que monitorizamos la calidad de servicio ofrecida por nuestro ISP en cada momento.

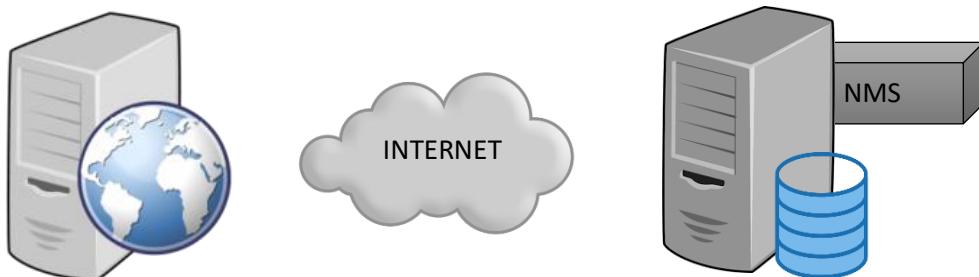


Ilustración 17: Diseño del flujo de trabajo entre dispositivos WAN

Básicamente hacemos un seguimiento del servicio de nuestro proveedor a través de pruebas de velocidad periódicas. Para ello, nos ayudamos de una utilidad denominada “speedtest-cli” la cuál es completamente de código abierto, publicada bajo Apache 2.0

Este es un cliente simple escrito en Python que vamos a poder utilizar para medir el ancho de banda bidireccional de nuestra conexión a internet y que utiliza la infraestructura de speedtest.net para darnos los resultados.

Al trabajar con Speedtest.net. Hay varios conceptos a tener en cuenta sobre este factor potencial:

- Speedtest.net ha pasado a usar pruebas de socket puro en lugar de pruebas basadas en HTTP.
- Esta aplicación está escrita en Python. Las diferentes versiones de Python ejecutarán ciertas partes del código más rápido que otras.
- La capacidad y la velocidad de la CPU y la memoria jugarán un papel importante en la inconsistencia entre Speedtest.net e incluso otras máquinas en la misma red.

Por defecto, el programa ejecuta una prueba contra el servidor speedtest.net más cercano.

Tras múltiples pruebas y análisis de los datos obtenidos no hemos encontrado ninguna inconsistencia siendo este cliente óptimo para los requerimientos del proyecto.

Adicionalmente, en caso de encontrar alguna inconsistencia debido a, por ejemplo, una sobrecarga del servidor escogido, es posible cambiar el servidor objetivo contra el que hacer las pruebas.

Como en casos anteriores, el encargado de ejecutar estas sucesivas pruebas de velocidad como de gestionar toda la información obtenida es uno de nuestros scripts Python instalados en el NMS.

9.3 GESTIÓN DE LA INFORMACIÓN

MariaDB ha sido el sistema gestor elegido para la implementación de la base de datos de la aplicación.

A parte de las características que de sobras conocidas que hacen de MariaDB un potente gestor, hay que mencionar la integración, compatibilidad y facilidad de uso que proporciona conjuntamente con el servidor Apache y el lenguaje Python, ya que, es una bifurcación de MySQL.

El motivo de escoger MariaDB en lugar de MySQL, que era el estándar hasta hace unos años, es porque en 2008 MySQL fue adquirida por un tercero y cambió su licencia. MariaDB tiene licencia GPL (Tutorials Point, 2016).

Sin embargo, no es solo la licencia, lo que hace que no decantemos por MariaDB. Existen algunas diferencias entre ambos servidores, entre las que se encuentra que MariaDB ha añadido tablas a nivel de sistema para mejorar la optimización de los datos. Por otro lado, MariaDB tiene un rendimiento superior a MySQL.

El único inconveniente que puedes tener utilizando MariaDB es de compatibilidad, a pesar de que ambos servidores presentan un elevado grado de compatibilidad. Es decir, si tu aplicación funciona con MySQL, también debería de funcionar sin ningún problema con MariaDB.

Debido al funcionamiento de la aplicación no ha sido necesario crear un complejo diseño, ni establecer relaciones, pues con un diseño simple cubre a la perfección las necesidades básicas. En el caso de posteriores ampliaciones sí que sería necesario modificar el diseño, estableciendo relaciones entre las tablas y añadiendo nuevas.

Se ha optado por un diseño en el que se diferencian los dispositivos UNIX de los dispositivos Windows, ya que, las MIB gestionadas por cada Agente SNMP difieren según el sistema operativo.

9.3.1 Base de datos Homemonitor_db

El conjunto de tablas que almacenan la totalidad de la información que debe manejar la aplicación del lado del servidor, se encuentran reunidas en la base de datos llamada Homemonitor_db.

A continuación, veremos el conjunto de tablas que forman Homemonitor_db, enumerando y describiendo los campos que las forman.

9.3.1.1 Tabla hosts

En esta tabla se almacena toda la información identificativa de cada host activo en la red.

Campo	Tipo	Null	Key	Extra
id	Int(11)	NO	Primaria	auto_increment
hostname	Varchar(50)	Sí	NO	NO
ip	Varchar(15)	Sí	NO	NO
mac	Varchar(17)	Sí	NO	NO

Tabla 5: Tabla hosts

- id: Identificador único. Es autoincremental, por lo que se incrementara de forma automática cada vez que añadamos un nuevo usuario en el sistema.
- hostname: Nombre de red del dispositivo.
- ip: Dirección IP del dispositivo.
- mac: Dirección MAC del dispositivo.

9.3.1.2 Tabla snmp_linux

En esta tabla se almacena todos los datos de monitorización de los dispositivos Unix existentes en la red.

Campo	Tipo	Null	Key	Extra
id	Int(11)	NO	Primaria	auto_increment
hour	Datetime	Sí	NO	NO
hostname	Varchar(50)	Sí	NO	NO
load1	double	Sí	NO	NO
load5	double	Sí	NO	NO
load15	double	Sí	NO	NO
user_cpu	double	Sí	NO	NO
system_cpu	double	Sí	NO	NO
idle_cpu	double	Sí	NO	NO
ram_total	double	Sí	NO	NO
ram_free	double	Sí	NO	NO
ram_buffer	double	Sí	NO	NO
ram_cache	double	Sí	NO	NO

disk_total	double	SÍ	NO	NO
disk_free	double	SÍ	NO	NO
disk_used	double	SÍ	NO	NO
disk_percent	double	SÍ	NO	NO

Tabla 6: Tabla *snmp_linux*

- id: Identificador único. Es autoincremental, por lo que se incrementara de forma automática cada vez que añadamos un nuevo usuario en el sistema.
- hour: Fecha y hora de la consulta sobre el estado del dispositivo.
- hostname: Nombre de red del dispositivo.
- load1: Porcentaje de uso de la CPU en el último minuto.
- load5: Porcentaje de uso de la CPU en los últimos 5 minutos.
- load15: Porcentaje de uso de la CPU en los últimos 15 minutos.
- user_cpu: Porcentaje de uso CPU por el usuario en el último minuto.
- system_cpu: Porcentaje de uso CPU por el sistema en el último minuto.
- idle_cpu: Porcentaje de inactividad de la CPU en el último minuto.
- ram_total: Espacio en Gigabytes de RAM instalada en el dispositivo.
- ram_free: Espacio en Gigabytes de RAM libre en el dispositivo.
- ram_buffer: Espacio en Gigabytes del búffer de RAM en el dispositivo
- ram_cache: Espacio en Gigabytes de la caché de RAM en el dispositivo
- disk_total: Espacio en disco en Gigabytes instalado en el dispositivo.
- disk_free: Espacio en disco en Gigabytes libre en el dispositivo.
- disk_used: Espacio en disco en Gigabytes usado en el dispositivo.
- disk_percent: Porcentaje de espacio en disco usado en el dispositivo.

9.3.1.3 Tabla *snmp_windows*

En esta tabla se almacena todos los datos de monitorización de los dispositivos Windows existentes en la red.

Campo	Tipo	Null	Key	Extra
id	Int(11)	NO	Primaria	auto_increment
hour	Datetime	SÍ	NO	NO
hostname	Varchar(50)	SÍ	NO	NO
cpu_usage	double	SÍ	NO	NO
ram_total	double	SÍ	NO	NO
ram_used	double	SÍ	NO	NO
disk_total	double	SÍ	NO	NO
disk_used	double	SÍ	NO	NO
disk_free	double	SÍ	NO	NO
disk_percent	double	SÍ	NO	NO

Tabla 7: Tabla *snmp_windows*

- id: Identificador único. Es autoincremental, por lo que se incrementara de forma automática cada vez que añadamos un nuevo usuario en el sistema.
- hour: Fecha y hora de la consulta sobre el estado del dispositivo.
- hostname: Nombre de red del dispositivo.
- cpu_usage: Porcentaje de uso de la CPU en el último minuto.

- **ram_total:** Espacio en Gigabytes de RAM instalada en el dispositivo.
- **ram_used:** Espacio en Gigabytes de RAM en uso en el dispositivo.
- **disk_total:** Espacio en disco en Gigabytes instalado en el dispositivo.
- **disk_free:** Espacio en disco en Gigabytes libre en el dispositivo.
- **disk_used:** Espacio en disco en Gigabytes usado en el dispositivo.
- **disk_percent:** Porcentaje de espacio en disco usado en el dispositivo.

9.3.1.4 Tabla isp

En esta tabla se almacena todos los datos de la calidad de servicio ofrecido por nuestro ISP.

Campo	Tipo	Null	Key	Extra
id	Int(11)	NO	Primaria	auto_increment
hour	Datetime	Sí	NO	NO
latency	double	Sí	NO	NO
download	double	Sí	NO	NO
download_ratio	double	Sí	NO	NO
upload	double	Sí	NO	NO
upload_ratio	double	Sí	NO	NO

Tabla 8: Tabla isp

- **id:** Identificador único. Es autoincremental, por lo que se incrementara de forma automática cada vez que añadamos un nuevo usuario en el sistema.
- **hour:** Fecha y hora de la consulta sobre el estado del dispositivo.
- **latency:** Latencia en milisegundos del ISP.
- **download:** Velocidad de bajada en Megabytes/segundo del ISP
- **download_ratio:** Porcentaje del ratio entre la velocidad de bajada contratada y la real.
- **upload:** Velocidad de subida en Megabytes/segundo del ISP
- **upload_ratio:** Porcentaje del ratio entre la velocidad de subida contratada y la real.

9.4 DESCRIPCIÓN DE LA INTERFAZ

A continuación, se nos hace necesario una forma de visualizar los datos que recogemos de una manera eficaz, concreta y sencilla. Para ello utilizaremos Grafana. Es un software libre basado en licencia de Apache 2.0, que permite la visualización y el formato de datos métricos. Permite crear cuadros de mando y gráficos altamente personalizables a partir de múltiples fuentes, incluidas entre otras, bases de datos como MariaDB (Grafana Labs, 2021).

Grafana hace uso de tableros para representar a través de sus gráficos la información almacenada en la Base de Datos que se ha sincronizado.

Se ha optado por crear cuatro tableros diferentes para visualizar de la forma más ordenada posible los datos de los diferentes dispositivos administrados:

- **Hosts:** Tablero que muestra el listado de todos los dispositivos que alguna vez estuvieron conectados en nuestra red.
- **SNMP Linux:** Tablero dedicado a los diferentes dispositivos Linux.
- **SNMP Windows:** Tablero dedicado a los diferentes dispositivos de Windows.
- **ISP:** Tablero dedicado a la calidad del servicio ofrecido por el proveedor de Internet.

9.4.1 Hosts

Este tablero se compone únicamente de una tabla donde podemos observar el histórico de dispositivos que se ha conectado a nuestra red.

Cada dispositivo lo identificamos a través de estos tres parámetros:

- Hostname
- IP
- MAC

Hosts		
Hostname	IP	MAC
router	192.168.1.1	00:0C:29:00:00:00
Asus-Pablo	192.168.1.1	00:0C:29:00:00:00
ARRIS_VIP5242_C85261337F95	192.168.1.1	00:0C:29:00:00:00
OnePlus6T	192.168.1.1	00:0C:29:00:00:00
TL-WPA4220	192.168.1.1	00:0C:29:00:00:00
localhost	192.168.1.1	00:0C:29:00:00:00
OnePlus6T	192.168.1.1	00:0C:29:00:00:00
OnePlus6T	192.168.1.1	00:0C:29:00:00:00
Galaxy-S6-edge	192.168.1.1	00:0C:29:00:00:00
Galaxy-S6-edge	192.168.1.1	00:0C:29:00:00:00
POCOPHONEF1-POCOPHON	192.168.1.1	00:0C:29:00:00:00
iPad-de-Pascual	192.168.1.1	00:0C:29:00:00:00
android-4800cc78739f90ec	192.168.1.1	00:0C:29:00:00:00
Google-Home-Mini	192.168.1.1	00:0C:29:00:00:00
RedmiNote8Pro-RedmiN	192.168.1.1	00:0C:29:00:00:00
Asus-Pablo	192.168.1.1	00:0C:29:00:00:00

Ilustración 18: Hosts registrados en la red

9.4.2 SNMP Linux

En este tablero hemos creado cuatro gráficos en los que se puede consultar el estado de la CPU, Memoria RAM y Espacio en Disco de los diferentes dispositivos UNIX administrados.

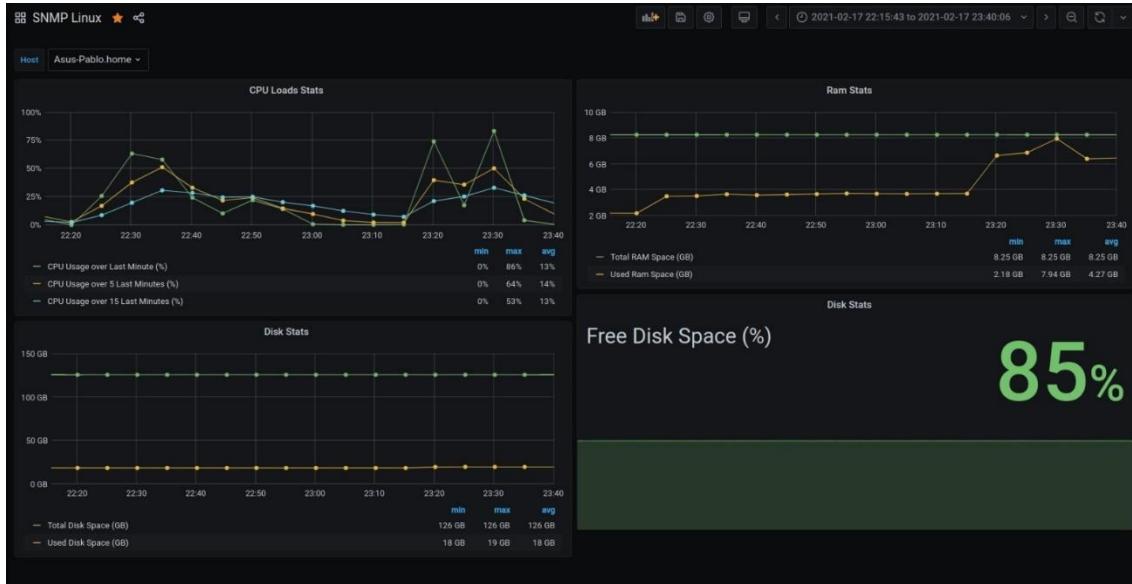


Ilustración 19: Tablero de los Dispositivos Administrados Linux

Para cambiar entre los diferentes dispositivos UNIX administrados desplegamos la lista de Hosts.

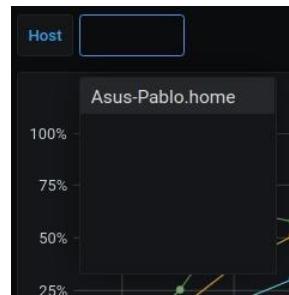


Ilustración 20: Selector de Dispositivos Linux

9.4.2.1 CPU

En este gráfico monitorizamos la carga de la CPU del dispositivo mediante 3 parámetros:

- Porcentaje de la carga de la CPU en el último minuto.
- Porcentaje de la carga de la CPU en los últimos 5 minutos.
- Porcentaje de la carga de la CPU en los últimos 15 minutos.

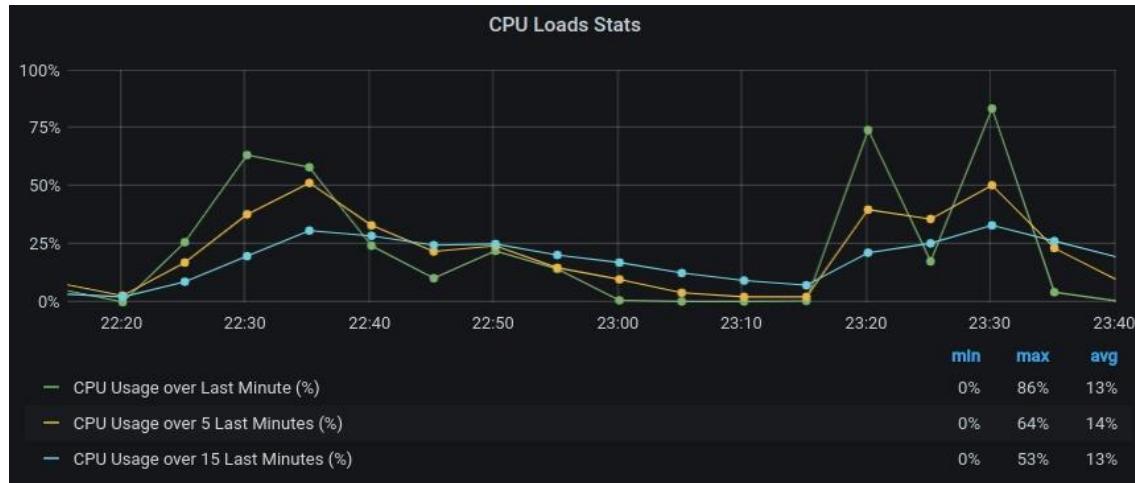


Ilustración 21: CPU dispositivos Linux

9.4.2.2 Memoria RAM

En este gráfico representamos la Memoria RAM disponible a través de 2 parámetros:

- Espacio de Memoria RAM Total en Gigabytes.
- Espacio de Memoria RAM en Uso en Gigabytes.

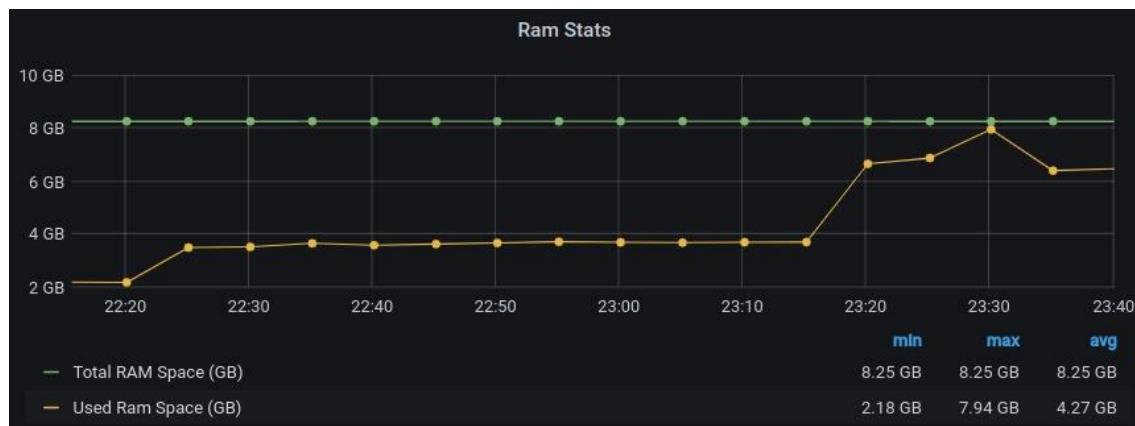


Ilustración 22: Memoria RAM dispositivos Linux

9.4.2.3 Espacio en Disco

Para la monitorización de esta variable hacemos uso de dos gráficos.

En el primero representamos:

- Espacio en Disco Total en Gigabytes
- Espacio en Disco Ocupado en Gigabytes.

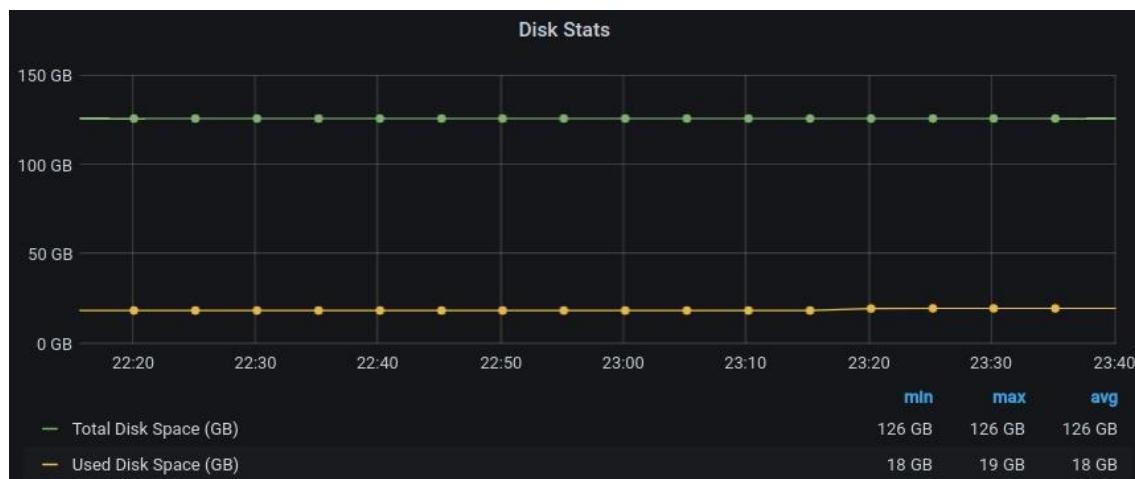


Ilustración 23: Espacio en Disco dispositivos Linux

En el segundo representamos:

- Espacio en Disco Libre en Gigabytes

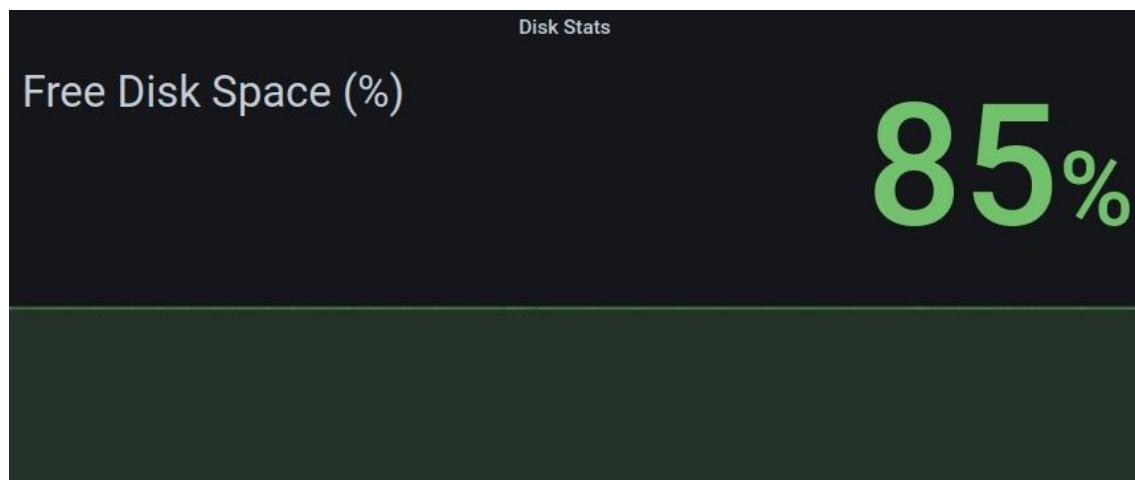


Ilustración 24: Espacio en Disco dispositivos Linux

9.4.3 SNMP Windows

En este tablero hemos creado cuatro gráficos en los que se puede consultar el estado de la CPU, Memoria RAM y Espacio en Disco de los diferentes dispositivos Windows administrados.



Ilustración 25: Tablero de los Dispositivos Administrados Windows

Para cambiar entre los diferentes dispositivos Windows administrados desplegamos la lista de Hosts.

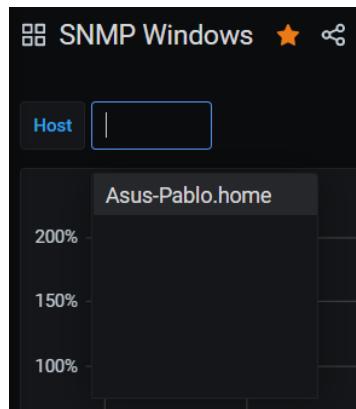


Ilustración 26: Selector de Dispositivos Windows

9.4.3.1 CPU

En este gráfico monitorizamos la carga de la CPU del dispositivo mediante:

- Porcentaje de la carga de la CPU en el último minuto

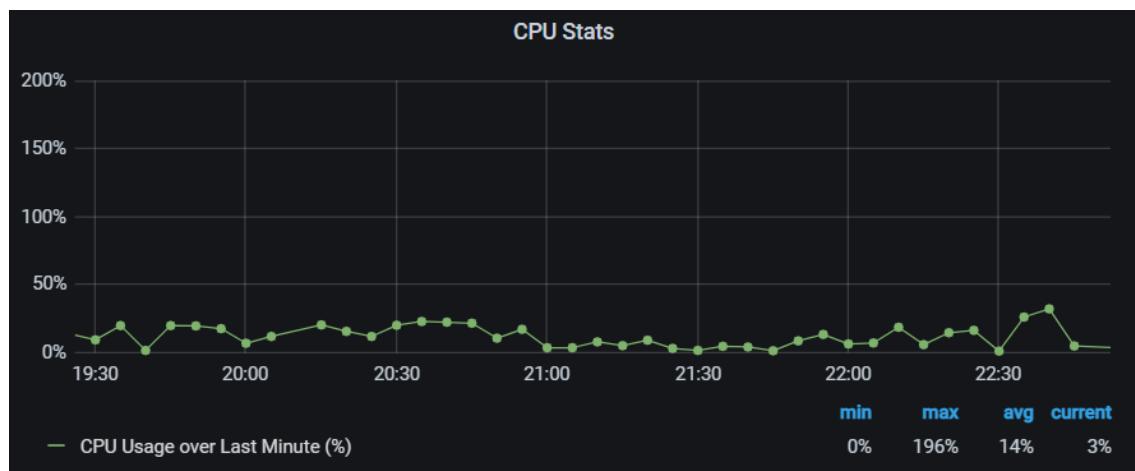


Ilustración 27: CPU dispositivos Windows

9.4.3.2 Memoria RAM

En este gráfico representamos la Memoria RAM disponible a través de:

- Espacio de Memoria RAM Total en Gigabytes.
- Espacio de Memoria RAM en Uso en Gigabytes

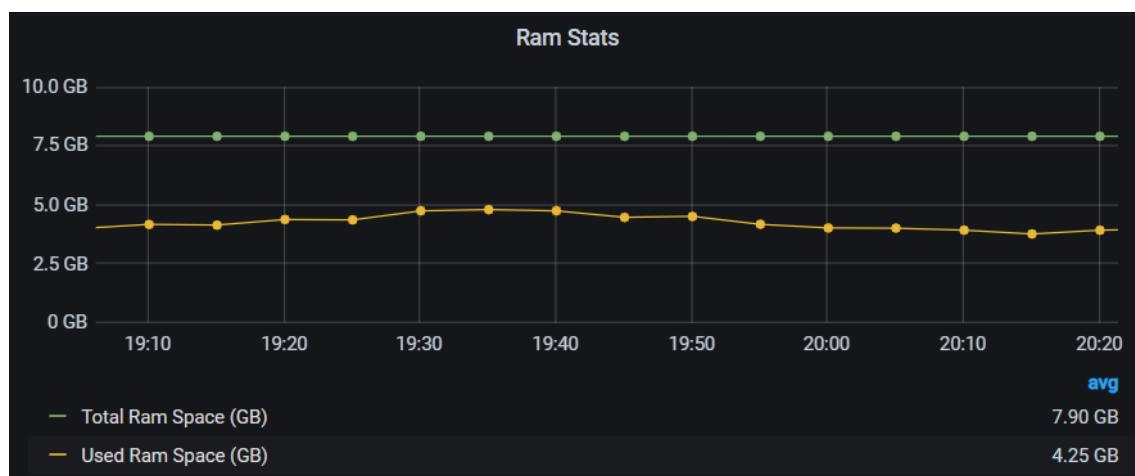


Ilustración 28: Memoria RAM dispositivos Windows

9.4.3.3 Espacio en Disco

Para la monitorización de esta variable hacemos uso de dos gráficos.

En el primero representamos:

- Espacio en Disco Total en Gigabytes
- Espacio en Disco Ocupado en Gigabytes.

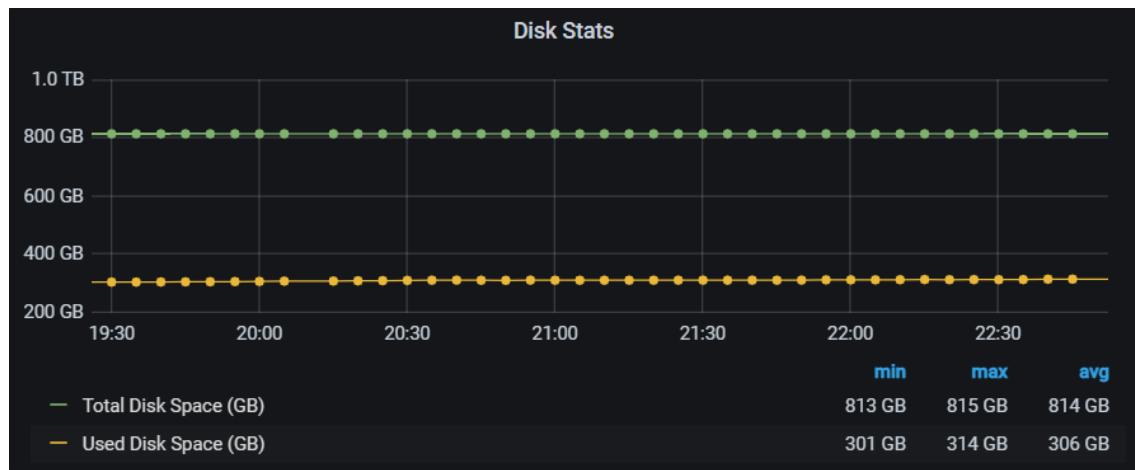


Ilustración 29: Espacio en Disco dispositivos Windows

En el segundo representamos:

- Porcentaje de Espacio Libre en Disco

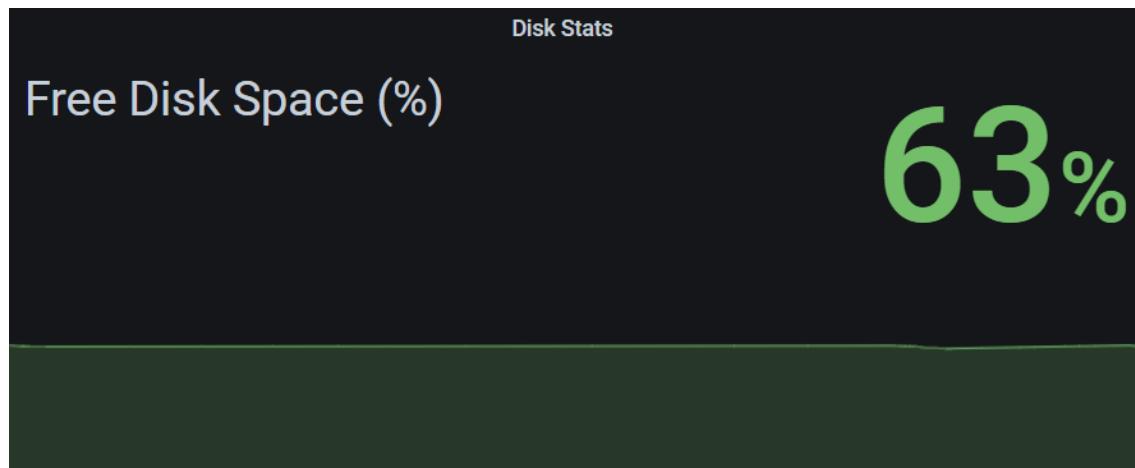


Ilustración 30: Espacio en Disco dispositivos Windows

9.4.4 ISP (Proveedor de Servicios de Internet)

En este tablero hemos creado cuatro gráficos en los que se puede consultar la calidad de servicio ofrecida por el proveedor contratado en cada momento.



Ilustración 31: Tablero Proveedor Servicios de Internet

9.4.4.1 Velocidad de Carga y Descarga.

En este gráfico monitorizamos la velocidad de subida y bajada del proveedor mediante:

- Megabytes por Segundo de Velocidad de Descarga
- Megabytes por Segundo de Velocidad de Carga.

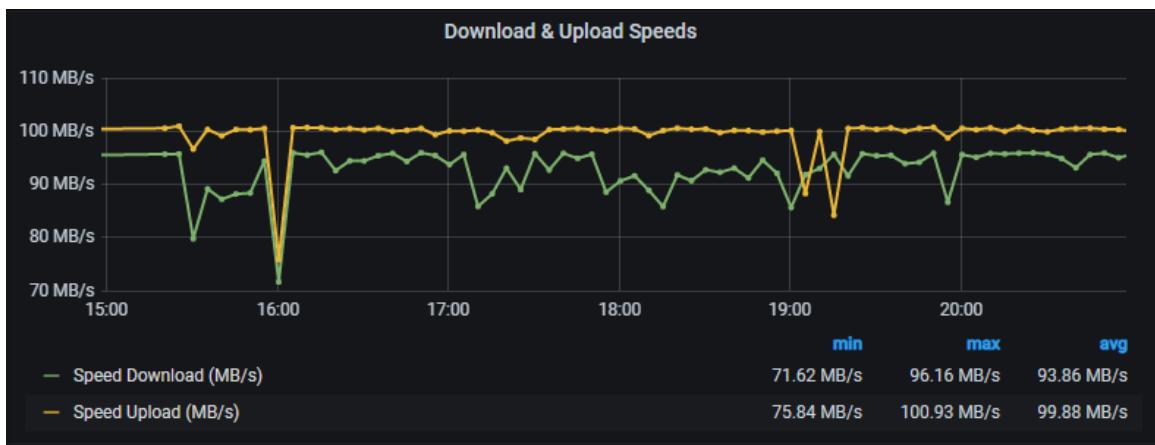


Ilustración 32: Velocidad Carga y Descarga ISP

De esta forma, obtenemos un mejor seguimiento del histórico de datos. Observándose de manera inmediata bajadas de velocidad o caídas de servicio.

Además, utilizamos dos gráficos más, en el que representamos estos mismos parámetros de una forma más precisa para consultar el estado actual del servicio con más detalle.

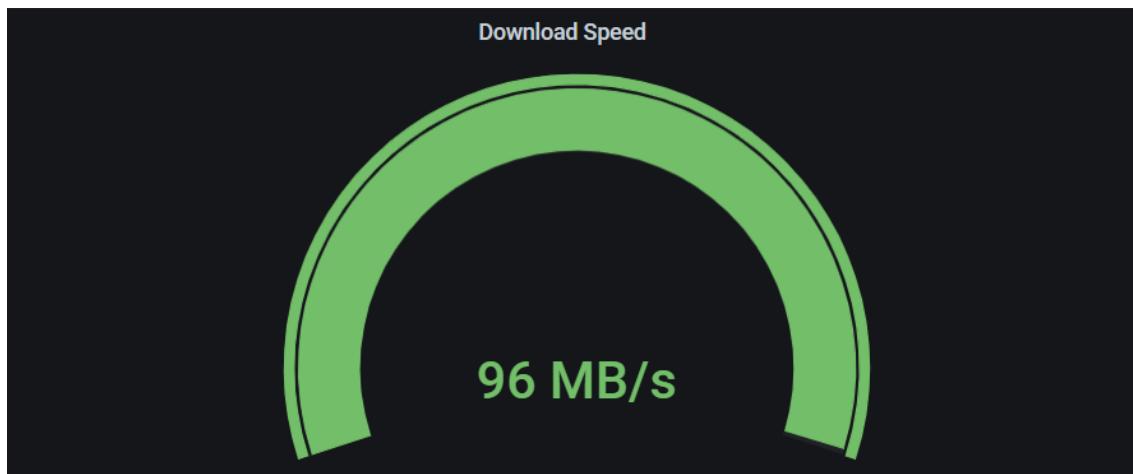


Ilustración 33: Velocidad Descarga ISP

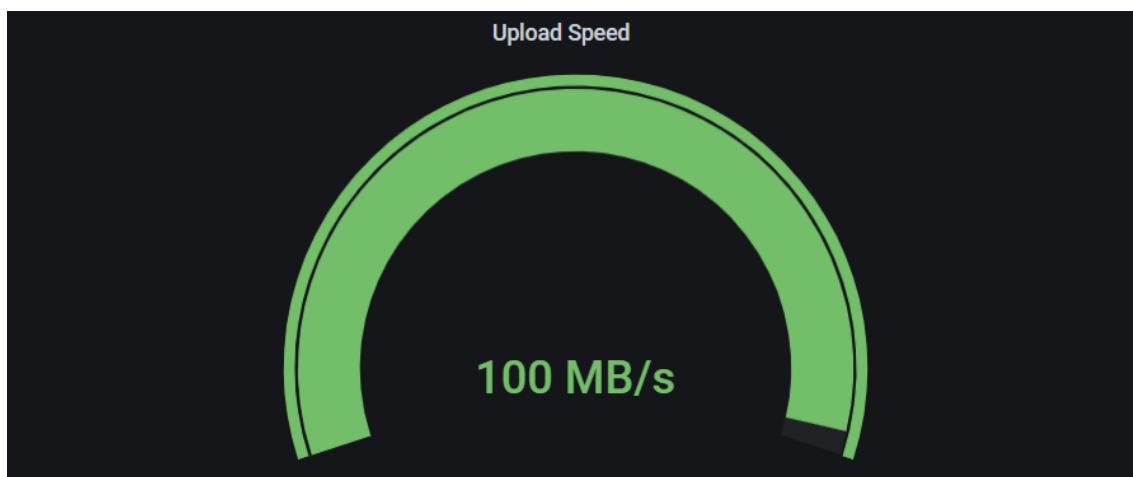


Ilustración 34: Velocidad Carga ISP

9.4.4.2 Latencia

Para representar la latencia hacemos uso de un gráfico que muestra:

- Latencia en Milisegundos

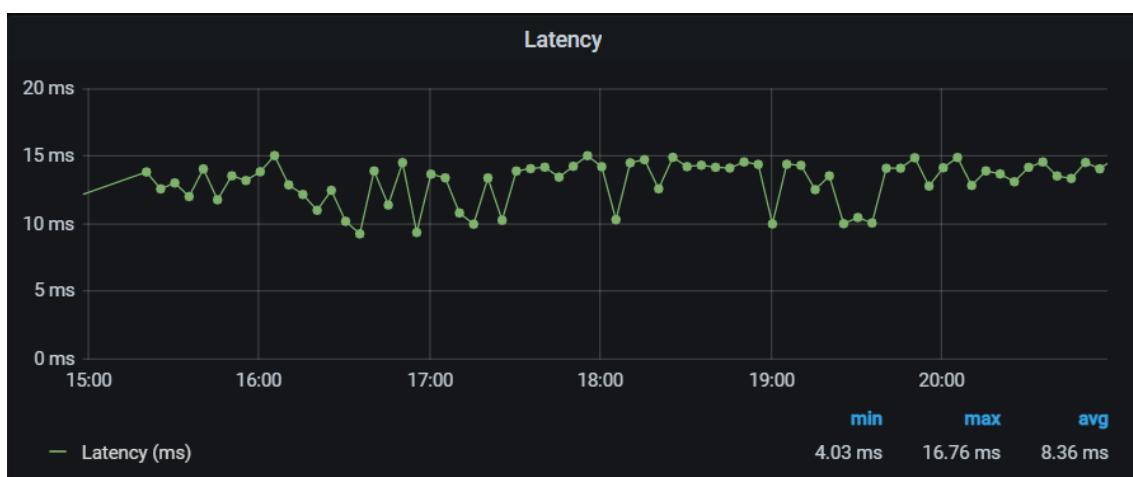


Ilustración 35: Latencia ISP

9.5 SISTEMA DE ALARMAS

Por último, unos de los requisitos fundamentales para la completitud del proyecto es el de crear un sistema de Alarmas que notifiquen a través de la forma más valiosa para el administrador de red cuando se produzca algún evento que requiera de su atención inmediata.

Se ha decidido hacer uso de la aplicación de Telegram como medio de notificación. Telegram es una aplicación de mensajería instantánea Open Source basado en licencia GNU GPL.

En un momento en el que las aplicaciones de mensajería instantánea se han instaurado como la opción preferida y más eficaz de muchos usuarios para ser notificados se ha optado para utilizarlas también como medio de aviso de cualquier evento monitorizado.

Para ello, se ha hecho uso de un Canal de mensajes autoadministrado por un BOT en el que reflejamos toda la información histórica de cualquier evento que se produzca a lo largo del tiempo.

En este canal recibiremos tres tipos de mensajes principalmente. Los relacionados con los protocolos ICMP, SNMP además de los encargados de avisar de la calidad de conexión del ISP.

9.5.1 Alertas ICMP

Cómo ya hemos comentado anteriormente, el sistema de monitorización hace un seguimiento de todos los dispositivos activos en la red por lo que se ha configurado un sistema de alarmas en el que se avise al administrador de:

- La conexión a la red de un nuevo dispositivo.
- La desconexión de un dispositivo cuando no se haya detectado actividad de este en los últimos 30 minutos.
- La reconexión de un dispositivo al detectar nueva actividad habiendo pasado 30 minutos de desconexión.

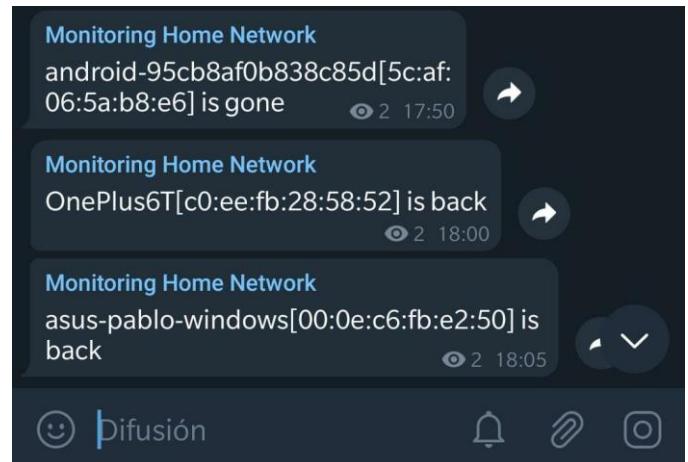
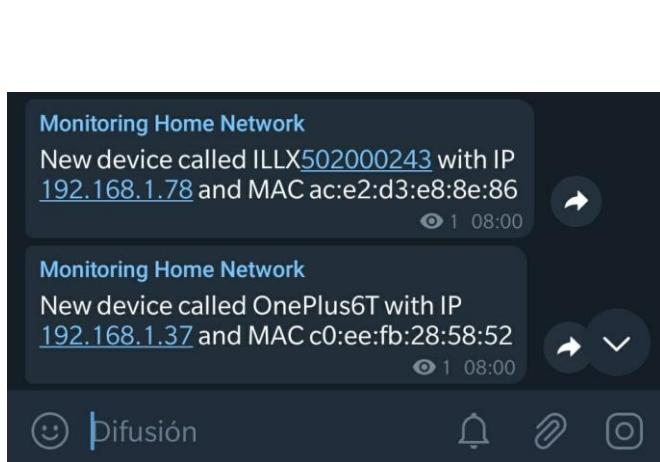


Ilustración 36: Notificación nuevo dispositivo

Ilustración 37 Notificación desconexión/reconexión dispositivo

9.5.2 Alertas SNMP

Para abordar este tema se ha decidido hacer uso de una funcionalidad nativa de SNMP, los SNMP Traps.

Por un lado, contamos con el Agente SNMP instalado en el dispositivo administrado encargado de generar el paquete UDP con la información del evento que ha hecho saltar la alerta.

Por otro lado, tenemos al NMS queda a la escucha en el puerto 162 para recibir cualquier paquete UDP que genere el Agente SNMP limitándose a volcar dicha información en un archivo de logs y reenviar la información del evento al Canal de Telegram alertando instantáneamente al administrador de red en su dispositivo móvil de cualquier evento que necesite de atención inmediata.

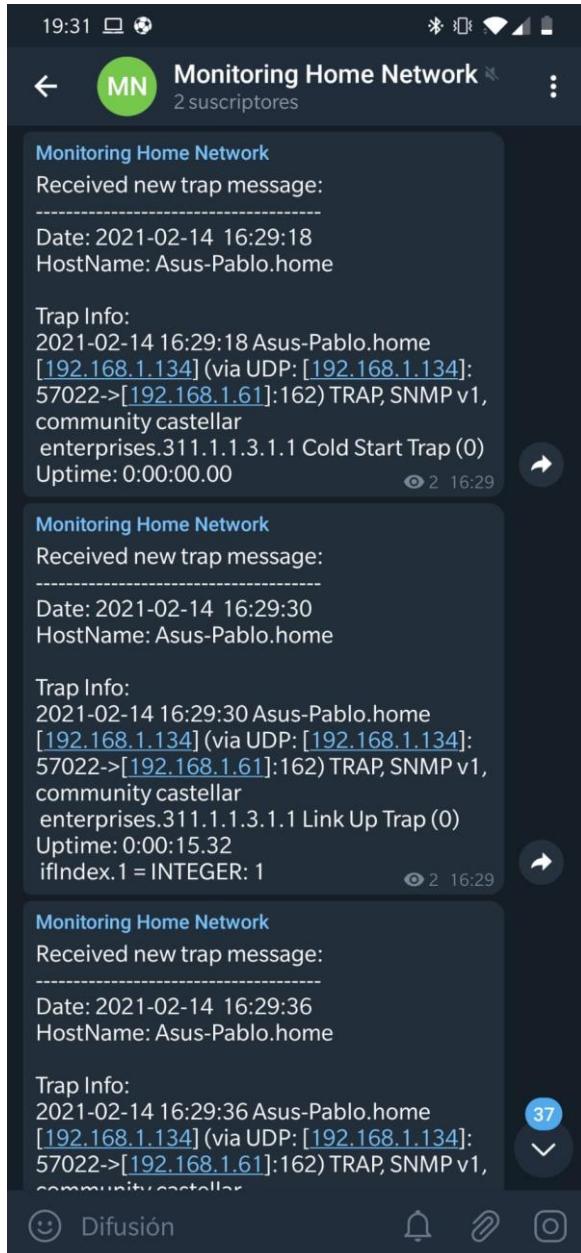


Ilustración 39: Notificación evento Windows

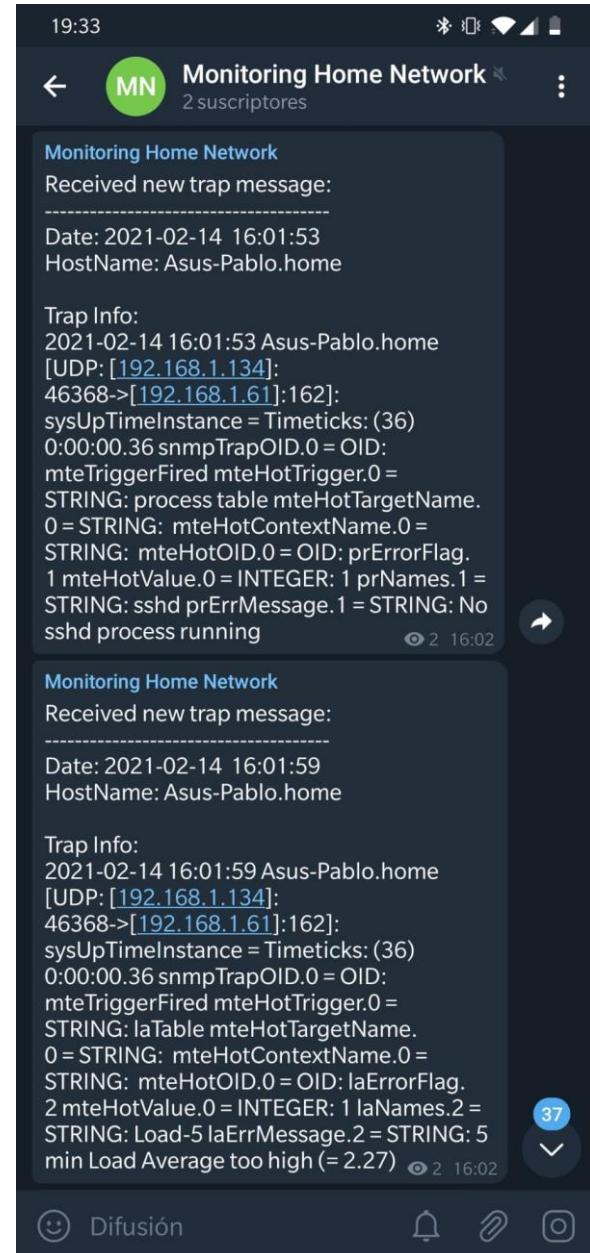


Ilustración 38: Notificación evento Linux

9.5.3 Alertas ISP

Por último, nuestro sistema de Alarmas nos ofrece una funcionalidad más. A través de él podremos detectar instantáneamente si se produce una caída del servicio ofrecido por nuestro

proveedor de servicios de internet, o simplemente nos está dando un servicio bastante empobrecido en relación con nuestra prestación contratada.

El NMS enviará un mensaje a nuestro canal si la ratio entre la velocidad de conexión proporcionada en ese instante y la velocidad contratada es menor al 60%.

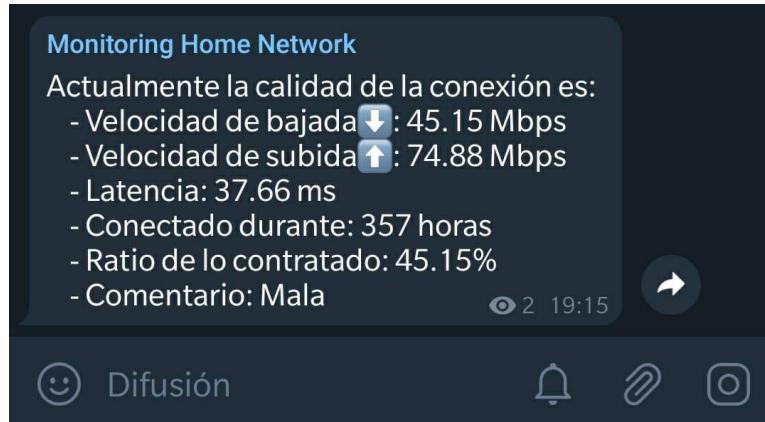


Ilustración 40: Notificación calidad servicio ISP

10 PLANIFICACION TEMPORAL

10.1 METODOLOGÍA DE DESARROLLO

Se ha decidido por una metodología de desarrollo ágil. Esta metodología hace referencia a un grupo de técnicas de desarrollo software basadas en un modelo iterativo donde los requerimientos y las soluciones evolucionan a través de la colaboración entre equipos multifuncionales autoorganizados

Concretamente, se ha optado por la metodología de desarrollo ágil SCRUM. Esta es una metodología de trabajo iterativa e incremental basada en la división y desarrollo de funcionalidades que se verifican y corrigen conforme se avanza en la consecución del proyecto reduciendo así las posibilidades de cometer errores.

Los procesos SCRUM permiten adaptarse sin problemas a los requisitos cambiantes y producir un producto que cumpla con los objetivos en evolución.

Es la más extendida debido a que incrementa significativamente la productividad y reduce el tiempo de obtención de beneficios respecto a otras metodologías de desarrollo donde se debe realizar primero un prototipo funcional con el que realizar las pruebas suponiendo un mayor riesgo, gasto de tiempo y coste.

10.2 PLANIFICACIÓN TEMPORAL DEL PROYECTO

En este apartado trataremos la planificación temporal del proyecto desde la primera reunión hasta la finalización del período de pruebas y memoria del proyecto.

10.2.1 Especificación del proyecto.

En primer lugar, se llevará a cabo el estudio de la viabilidad junto con la especificación de objetivos y alcance del proyecto.

Se pretende determinar unos requisitos iniciales, objetivos y estudiar la viabilidad de estos.

10.2.2 Investigación y búsqueda de información.

En esta segunda etapa, investigaremos y recopilaremos información acerca de la red, los protocolos por los que obtendremos la información de los dispositivos y estudiaremos las alternativas que surgen para la consecución del proyecto.

10.2.3 Implementación de la solución.

Durante esta fase implementará todo el desarrollo software necesario para el alcance de la solución propuesta. Abarcará desde la implementación de la base de datos, desarrollo de la aplicación central, configuración de los agentes, diseño de la interfaz gráfica y pruebas del sistema.

10.2.4 Memoria del proyecto

La última etapa consiste en la realización de la memoria del proyecto. Esta etapa es transversal durante todo el proyecto, desde que la investigación sobre los fundamentos teóricos hasta la finalización de las pruebas.

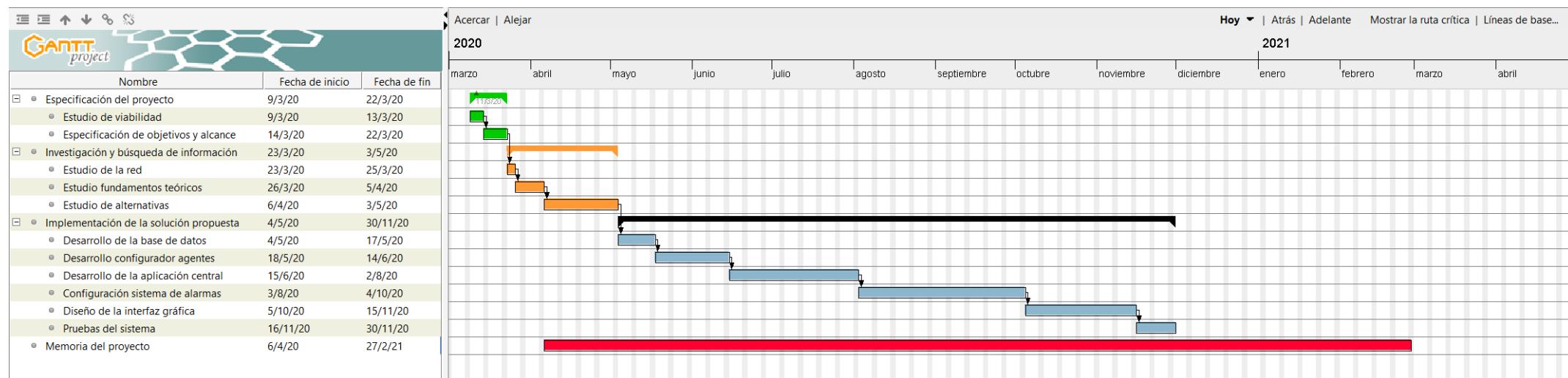


Ilustración 41: Diagrama de Gantt de la planificación temporal del proyecto

11 EVALUACIÓN DE RIESGOS

11.1 LISTA DE RIESGOS

R1. Planificación temporal

Especificación del proyecto. No se finaliza en la fecha prevista, aumentan los recursos.

R2. No consecución de algún requisito.

Especificación del proyecto. No se cumplen los objetivos del proyecto.

R3. Requisitos cambiantes.

Especificación del proyecto. Retraso en el desarrollo y consecuentemente en la obtención del producto.

R4. Falta personal en el equipo del proyecto.

Especificación del proyecto. Retraso en las entregas, no se cumplen los objetivos del proyecto.

R5. Herramientas de desarrollo inadecuadas.

Implementación de la solución. Retraso en la finalización del proyecto, menor calidad de este.

R6. Fase de pruebas.

Implementación de la solución. Falta de calidad, deficiencias funcionales, insatisfacción del cliente, pérdidas económicas.

R7. Incumplimiento de alguna norma, reglamento o legislación.

Todas las fases. Repercusiones legales, no se cumplen los objetivos.

R8. Falta de implantación de medidas de seguridad.

Especificación del proyecto, implementación de la solución. Pérdida de información, incumplimiento legal, pérdidas económicas.

R9. Abandono del proyecto antes de la finalización.

Todas las fases. Pérdidas económicas y reputacionales, posibles repercusiones legales.

11.2 CATALOGACIÓN DE LOS RIESGOS

Riesgo	Probabilidad	Impacto
R1	Alta – 70%	Crítico
R2	Alta – 80%	Crítico
R3	Alta – 70%	Marginal
R4	Media – 60%	Crítico
R5	Baja – 30%	Crítico
R6	Media – 60%	Crítico

R7	Baja – 30%	Crítico
R8	Alta – 70%	Crítico
R9	Baja – 10%	Catastrófico.

Tabla 9: Catalogación Riesgos

11.3 PLAN DE CONTINGENCIA

Riesgo	Soluciones que adoptar
R1	Aplazar alguna funcionalidad, afrontar posibles perdidas
R2	Revisar especificación requisitos, modificar la planificación, afrontar posibles pérdidas.
R3	Revisar contrato, aplazar funcionalidades, modificar planificación y presupuesto
R4	Aumentar personal, solicitar un aplazamiento, modificar planificación, afrontar posibles pérdidas.
R5	Mejorar la formación del equipo, prevenir herramientas alternativas.
R6	Diseñar test con antelación, realizar test automáticos, revisar contrato mantenimiento, afrontar posibles pérdidas.
R7	Revisar las normas y legislación, consultar a un experto, afrontar posibles repercusiones de carácter penal.
R8	Revisar seguridad en cada fase, aplicar políticas de seguridad activas.
R9	-

Tabla 10: Planes de Contingencia

12 RESUMEN DEL PRESUPUESTO

A continuación, mostraremos el resumen del presupuesto para la implementación de un sistema de monitorización de dispositivos y servicios en una red doméstica. Podemos encontrar un desglose más detallado del mismo en el Capítulo 4 Presupuestos.

Artículo / Servicio	Precio
Presupuesto Hardware	88,08€
Presupuesto Software	0€
Presupuesto del Personal	11100€
TOTAL	11188,08€

Tabla 11: Resumen del Presupuesto



IMPLEMENTACIÓN DE UN SISTEMA DE MONITORIZACIÓN DE DISPOSITIVOS Y SERVICIOS EN UNA RED DOMÉSTICA

FUNDAMENTOS TEÓRICOS

- **DATOS CLIENTE:** ESCUELA SUPERIOR DE INGENIERÍA, UNIVERSIDAD DE CÁDIZ.
AV. UNIVERSIDAD DE CÁDIZ, 10, 11519 PUERTO REAL, CÁDIZ
956 48 32 00
DIRECCION.ESI@UCA.ES
- **DATOS AUTOR:** PABLO MANUEL GARCÍA SÁNCHEZ
INGENIERO INFORMÁTICO
PABLO.GARCIASANCH@ALUM.UCA.ES

ÍNDICE

13 CONCEPTOS BÁSICOS.	73
13.1 RED DE DATOS	73
13.2 TOPOLOGÍAS	73
13.2.1 Topología en Malla	73
13.2.2 Topología en Bus	74
13.2.3 Topología en Anillo	74
13.2.4 Topología en Estrella	75
13.2.5 Topologías en Árbol	76
13.2.6 Topologías Lógicas	76
13.3 MODELO OSI	77
13.4 MODELO TCP/IP	78
13.4.1 Acceso a red	79
13.4.2 Internet	79
13.4.3 Transporte	79
13.4.4 Aplicación	79
13.5 PROTOCOLOS	79
13.5.1 UDP	79
13.5.2 TCP	80
14 MONITORIZACIÓN	81
14.1 TIPOS DE MONITORIZACIÓN	81
14.1.1 Monitorización pasiva	81
14.1.2 Monitorización activa	81
14.1.3 Alarmas	81
14.2 ELEMENTOS A MONITORIZAR	81
15 ICMP	82
15.1 FUNCIONAMIENTO	82
15.2 OPERACIONES	82
16 SNMP	83
16.1 HISTORIA	84
16.2 FUNCIONAMIENTO	84
16.2.1 Dispositivos administrativos	84
16.2.2 Agente	84
16.2.3 NMS (Network Management System)	84
16.2.4 Operaciones	86
16.2.4.1 Get	86
16.2.4.2 Get-Next	86
16.2.4.3 Get-Bulk	86
16.2.4.4 Set	87
16.2.4.5 Trap	87
16.3 COMUNIDADES	87
16.4 MIB	88
16.5 VERSIONES EXISTENTES	89
17 PROVEEDOR SERVICIOS DE INTERNET	89
17.1 ARQUITECTURA	90
18 RASPBERRY PI OS	91

19	OTROS SISTEMAS OPERATIVOS	91
19.1	EL SISTEMA OPERATIVO LINUX	91
19.1.1	Historia	91
19.1.2	Características generales	92
19.2	WINDOWS SERVER 2016	92
19.2.1	Características generales	93

13 CONCEPTOS BÁSICOS.

13.1 RED DE DATOS

Una red de datos es un conjunto de computadoras y dispositivos conectados entre sí mediante una o más vías de transmisión. La red existe para cumplir un determinado objetivo que es la transmisión de datos donde se realiza el intercambio de información entre dos dispositivos a través de algún medio de transmisión. Esta transmisión se considera local si los dispositivos se encuentran en un área geográfica delimitada y se considera remota si los dispositivos están separados por una distancia considerable (Forouzan, 2019). Este intercambio de datos es la base de muchos servicios, basados en redes de computadoras y se han convertido en una parte indispensable de los negocios, la industria y el entretenimiento.

13.2 TOPOLOGÍAS

El término topología se define como la forma en que está diseñada la red ya sea de manera física o lógica. La topología de una red es la representación geométrica de la relación entre todos los enlaces y los dispositivos que los enlazan entre sí. Existen topologías básicas como son la de malla, estrella, árbol, bus y anillo.

13.2.1 Topología en Malla

En este tipo de configuración cada dispositivo está conectado a uno o más de los otros dispositivos (también llamados nodos) y así, es posible llevar los datos de un nodo al otro por diferentes caminos. Una diferencia importante con las otras topologías es que no se requiere tener un concentrador o servidor central.

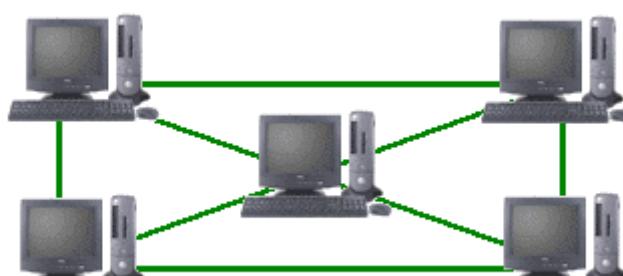
Entre las ventajas que se tienen con esta topología son:

- El sistema completo no se inhabilita si un enlace falla.
- La seguridad de saber que cuando un mensaje viaja a través de un enlace, solamente lo ve el receptor adecuado.
- Facilidad para detectar y aislar fallos.

Entre las desventajas:

- Cantidad de cable necesario y el número de puertos de entrada/salida.
- Coste para implementar esta configuración.

Por estas razones, la topología en malla se utiliza junto con otras topologías para formar una topología híbrida.



13.2.2 Topología en Bus

Considerada la topología más básica y fácil de implementar. Contiene un solo canal de comunicaciones para todos los dispositivos de red, este canal recibe el nombre de bus o backbone.

En los extremos del cable, que permite la comunicación y es único, contiene una resistencia de acople denominada terminador; este acople nos indica que no hay más equipos en los extremos y el cierre del bus por medio del acople de impedancias.

Ventajas de esta topología:

- Simplicidad

Desventajas:

- Límite en la cantidad de equipos conectados
- Es de complicada reconfiguración
- Colisión de mensajes
- Pérdidas en la transmisión
- Degradación de la señal.

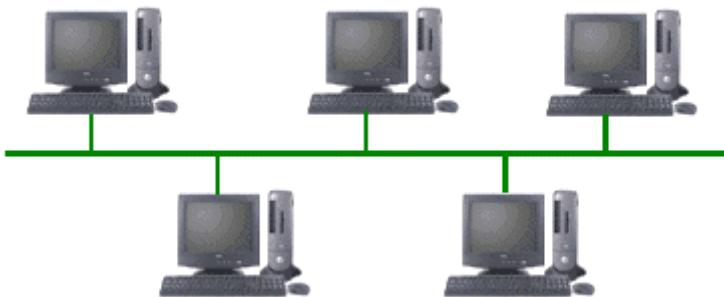


Ilustración 43: Topología en Bus

13.2.3 Topología en Anillo

Se caracteriza por tener una estación conectada a la siguiente y así hasta que la última estación se conecta con la primera dando forma de un círculo. Para poder repetir la señal cada nodo tiene un transmisor y un receptor.

Entre sus ventajas tenemos que:

- Es la que menos colisiones presenta entre las topologías por usar el algoritmo token ring

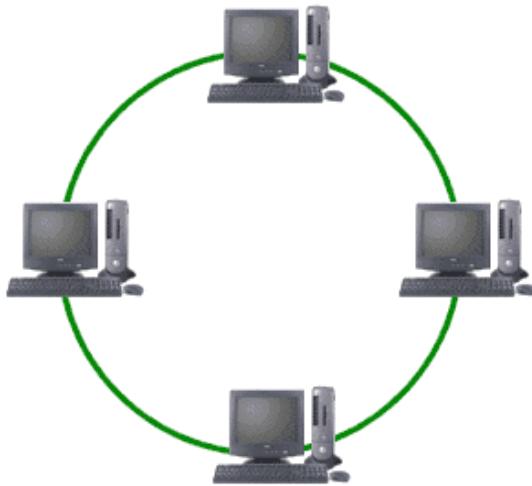


Ilustración 44: Topología en Anillo

13.2.4 Topología en Estrella

Es una de las topologías más usadas. La topología en estrella tiene un punto central al cual todos los dispositivos están conectados, al ser de esta manera la conexión toda comunicación, distribución, conmutación y control deberá pasar por este nodo central. Podemos encontrar topologías en estrellas con nodos activos que previenen el eco y lo soluciona y otras que carecen de esta función.

Entre sus ventajas está:

- Si uno de los terminales falla no afecta al resto
- Fácil escalabilidad de la red

Sin embargo, su gran desventaja:

- Si un nodo es el que falla la red se desconecta.

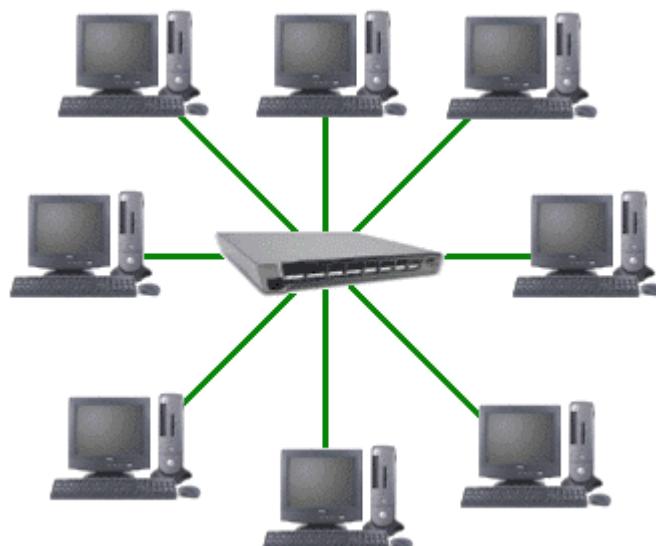


Ilustración 45: Topología en Estrella

13.2.5 Topologías en Árbol

La topología en árbol es una variante de la de estrella. Los nodos del árbol están conectados a un concentrador central que controla el tráfico de la red. Sin embargo, no todos los dispositivos se conectan directamente al concentrador central. La mayoría de los dispositivos se conectan a un concentrador secundario que, a su vez, se conecta al concentrador central que generalmente es un hub3 o switch.

Las ventajas y desventajas son:

- Similares a las de la Topología en Estrella.

Sin embargo, el incluir concentradores secundarios puede incrementar la distancia a la que puede viajar la señal entre dos dispositivos y permite a la red aislar y priorizar las comunicaciones de distintas computadoras.



Ilustración 46: Topología en Árbol

13.2.6 Topologías Lógicas

La topología lógica abarca las configuraciones de red de las entidades que intervienen en la comunicación.

Según su tamaño:

- Redes de Área Personal “PAN”: son aquellas redes que su área de acción comprende solo el entorno del usuario con aquellos dispositivos que interactúe. Entre las tecnologías que se pueden implementar tenemos Wifi o bluetooth.
- Redes de Área Local “LAN”: Redes que su alcance llega hasta unos cuantos kilómetros de cobertura, se las considera redes privadas por lo que su estructura suele ser una empresa, una casa, etc. Operan entre los 10 y 100 Mbps de velocidad.

- Redes de Área Metropolitana “MAN”: aunque en la actualidad no son muy usadas, son aquellas redes LAN que están interconectadas en la misma ciudad o ciudades próximas.
- Redes de Área extensa “WAN”: cuando unimos varias redes LAN que están separadas por miles de kilómetros reciben el nombre de WAN.

Según su carácter:

- Redes privadas: aquellas redes que son propiedad de una empresa, adecuada según las necesidades de esta.
- Redes públicas: Se las utiliza para la unión de redes pequeñas, caracterizadas por su precio económico.
- Redes mixtas: redes privadas que hacen uso de una red pública.

Según la tecnología que usan para la transmisión:

- Redes punto a punto: redes cuya comunicación está dada de equipo a equipo (origen y destino).
- Redes broadcast: cuando todos los equipos de la red comparten un canal de comunicación se le denomina broadcast.

13.3 MODELO OSI

El Modelo OSI (interconexión de sistemas abiertos) fue lanzado en la década de 1980, para permitir dividir las funciones de la comunicación en siete capas y así cada capa se comunicará con la anterior y la siguiente (Estela Raffino, 2020).

Cada capa tiene una función que será detallada brevemente a continuación:

- Capa física: Cada equipo que interviene en la comunicación contiene especificaciones eléctricas, mecánicas y funcionales. Esta información se define en la capa física.
- Capa de enlace a datos: Esta capa se encarga de la respectiva transferencia de datos entre una entidad y la siguiente, de la detección y corrección de errores en el proceso.
- Capa de red: se encarga de direccionar las cadenas de datos entre las diferentes entidades de la red o de una red distinta y de asignar una IP.
- Capa de transporte: Segmenta las cadenas de datos a transmitir y las transporta de una entidad a otra.
- Capa de sesión: Controla el enlace que se estableció en la capa anterior, el mantenimiento y restablecer el enlace si se llega a interrumpir.
- Capa de presentación: representación de la información que se transmitió.

- Capa de aplicación: Proporciona el acceso a los servicios de las capas anteriores y establece los protocolos para el intercambio de la información.



Ilustración 47: Modelo OSI

13.4 MODELO TCP/IP

La familia de protocolos TCP/IP se desarrolló antes que el modelo OSI. Por lo tanto, los niveles de Protocolo de Control de Transmisión/Protocolo de Red (TCP/IP) no coinciden exactamente con los del modelo OSI.

Está compuesta por cuatro niveles: acceso a red, internet, transporte y aplicación (IONOS España S.L.U., 2020).



Ilustración 48: Modelo TCP/IP

13.4.1 Acceso a red

Especifica información detallada de cómo se envían físicamente los datos a través de la red, que incluye cómo se realiza la señalización eléctrica de los bits mediante los dispositivos de hardware que conectan directamente con un medio de red, como un cable coaxial, un cable de fibra óptica o un cable de cobre de par trenzado. Algunos protocolos usados en este nivel son Ethernet, Token Ring, FDDI, etc.

13.4.2 Internet

Empaque los datos en datagramas IP, que contienen información de las direcciones de origen y destino utilizada para reenviar los datagramas entre hosts y a través de redes. Realiza el enrutamiento de los datagramas IP. Se usan los protocolos IP, ICMP, ARP, RARP.

13.4.3 Transporte

Permite administrar las sesiones de comunicación entre equipos host. Define servicio y el estado de la conexión utilizada al transportar datos. Los protocolos que se manejan en este nivel son TCP, UDP.

13.4.4 Aplicación

Define los protocolos de aplicación TCP/IP y cómo se conectan los programas de host a los servicios del nivel de transporte para utilizar la red. Se usan los protocolos HTTP, Telnet, FTP, TFTP, SNMP, DNS, SMTP, X Windows y otros protocolos de aplicación.

13.5 PROTOCOLOS

13.5.1 UDP

El protocolo de datagramas de usuario (User Datagram Protocol, por sus siglas en inglés), es un protocolo a nivel de transporte que va de extremo a extremo y que añade sólo direcciones de puertos, control de errores mediante sumas de comprobación y la información de longitud de datos del nivel superior.

El paquete producido por el protocolo UDP se denomina datagrama de usuario:

Bit #	0	7	8	15	16	23	24	31
0			Source Port				Destination Port	
32			Length				Header and Data Checksum	

Ilustración 49: Formato PDU UDP

- Dirección del puerto origen: Es la dirección del programa de aplicación que ha creado el mensaje.
- Dirección del puerto destino: Es la dirección del programa de aplicación que recibirá el mensaje.
- Longitud total: Este campo define la longitud total del datagrama de usuario en bytes.
- Suma de comprobación: Esta suma de comprobación es un campo de 16 bits utilizado para la detección de errores. UDP proporciona sólo las funciones básicas necesarias para la entrega extremo a extremo de una transmisión. No ofrece funciones de secuenciación ni de reordenación y no puede especificar el paquete dañado cuando se informa de un error.

13.5.2 TCP

El protocolo de control de transmisión TCP (Transmission Control Protocol por sus siglas en inglés.) proporciona servicios completos de transporte a las aplicaciones.

TCP es un protocolo de transporte puerto a puerto que ofrece un flujo fiable, es decir, que está orientado a conexión: se debe de establecer una conexión entre ambos extremos de la transmisión antes de poder enviar datos.

Comienza cada transmisión informando al receptor de que hay datagramas en camino y finaliza cada transmisión con una terminación de conexión. De esta forma, el receptor conoce la transmisión entera en lugar de un único paquete.

Además, es responsable de la entrega fiable del flujo entero de bits contenido en el mensaje inicialmente generado por la aplicación emisora. La fiabilidad se asegura mediante la detección de errores y la retransmisión de las tramas con errores; todos los segmentos deben ser recibidos y confirmados antes de que la transmisión se considere completa.

Debido a que UDP utiliza un tamaño de trama más pequeño es mucho más rápido que TCP, pero menos fiable.

Un ejemplo datagrama del protocolo TCP lo vemos a continuación:

Bit #	0	7	8	15	16	23	24	31				
0	Source Port				Destination Port							
32	Sequence Number											
64	Acknowledgment Number											
96	Data Offset	Res	Flags		Window Size							
128	Header and Data Checksum				Urgent Pointer							
160...	Options											

Ilustración 50: Formato PDU TCP

- Dirección del puerto origen: Esta dirección define el programa de aplicación de la computadora origen.
- Dirección del puerto destino: Este campo define el programa de aplicación de la computadora destino.
- Número de secuencia: Un flujo de datos del programa de aplicación se puede dividir en dos o más segmentos TCP. El campo con el número de secuencia indica la posición de los datos en el flujo de datos original.
- Número de confirmación: Se utiliza para confirmar la recepción de datos desde otro dispositivo que participa en la comunicación.
- Longitud de la cabecera: Este campo de cuatro bits indica el número de palabras de 32 bits de la cabecera TCP. Los cuatro bits pueden definir hasta 15 que al multiplicarlo por cuatro se obtiene el número total de bytes de la cabecera. Reservado. Este campo de seis bits está reservado para uso futuro. Control. Cada bit del campo de control de seis bits funciona de forma individual e independiente. Un bit puede definir el uso de un segmento o servir como una comprobación de la validez de otros campos. El bit urgente valida el campo de puntero urgente. El bit ACK valida el campo con el número de confirmación. El bit PSH se utiliza para informar al emisor de que se necesita un mayor ancho de banda. El bit RST se utiliza para reiniciar la conexión cuando hay confusión en los números de secuencia en tres tipos de segmentos: petición de conexión, confirmación de conexión y la recepción de confirmación. El bit FIN se utiliza en la terminación de la conexión.

- Tamaño de la ventana: Este campo de 16 bits define el tamaño de la ventana deslizante.
- Suma de comprobación: Este campo de 16 bits se utiliza para la detección de errores. Puntero urgente. Este es el último campo requerido en la cabecera. Su valor es válido sólo si el bit URG se encuentra activado. En este caso, el emisor está informando al receptor que hay datos urgentes en la porción de datos del segmento.
- Opciones y relleno: El resto de la cabecera TCP define campos opcionales. Se utilizan para evitar información adicional al receptor o para alineamiento.

14 MONITORIZACIÓN

La monitorización de red es la acción que nos permite verificar sistemáticamente el desempeño y la disponibilidad de los dispositivos críticos dentro de la red, a través de la identificación y detección de posibles problemas.

14.1 TIPOS DE MONITORIZACIÓN

Existen varias formas de monitorización. Las dos más comunes son la monitorización pasiva y la activa. Ambas tienen ventajas y desventajas (Les Cottrell, 2001).

14.1.1 Monitorización pasiva

La monitorización pasiva solo está al tanto de lo que pasa sin modificar ningún parámetro u objeto. Usa dispositivos para ver el tráfico que está circulando a través de la red, estos dispositivos pueden ser sniffers¹⁰ o en su defecto sistemas incluidos en los switches y ruteadores.

Ejemplos de estos sistemas son la monitorización remota (RMON) y el protocolo simple de administración de red (SNMP). Una de las características de la monitorización pasiva es que no incrementa el tráfico en la red para poder realizar las lecturas. Se puede capturar el tráfico teniendo un puerto espejo (mirror) en un dispositivo de red o un dispositivo intermedio que esté capturando el tráfico.

14.1.2 Monitorización activa

La monitorización activa tiene la capacidad de injectar paquetes de prueba dentro de la red o enviar paquetes a servidores con determinadas aplicaciones, siguiéndolos para medir los tiempos de respuesta. Por lo tanto, genera tráfico extra lo suficiente para recabar datos precisos. Este tipo de monitorización permite el control explícito en la generación de paquetes para realizar las mediciones, como el control en la naturaleza de generar tráfico, las técnicas de muestreo, frecuencia, tamaño de los paquetes entre otras.

14.1.3 Alarmas

Una alarma es un aviso o señal de cualquier tipo, que advierte de la proximidad de un peligro. Son consideradas como eventos fuera de lo común y por lo tanto, necesitan atención inmediata para mitigar la posible falla detectada. Las alarmas son activadas cuando un parámetro ha alcanzado cierto nivel o se tiene un comportamiento fuera de lo normal.

14.2 ELEMENTOS A MONITORIZAR

Existen funciones críticas dentro de los dispositivos de red que necesitan ser monitorizados constantemente y las alarmas que se lleguen a presentar deben ser atendidas tan pronto como sea posible cuando un evento ocurra.

Algunos de los parámetros más comunes son la utilización de ancho de banda, el consumo de CPU, consumo de memoria, el estado físico de las conexiones, el tipo de tráfico que manejan los ruteadores, switches, hubs, firewalls y los servicios de web, correo, base de datos entre otros.

También se debe estar pendientes de las bitácoras de conexión al sistema y configuración de los sistemas, ya que aquí se presenta información valiosa cuando un evento sucede. Estos parámetros deben de ser monitorizados muy de cerca para detectar posibles cambios y tendencias que afecten al usuario final.

15 ICMP

El Protocolo de Control de Mensajes de Internet (por sus siglas en inglés, Internet Control Message Protocol) es un protocolo de internet que provee de mensajes para reportar errores y otra información relacionada con el procesamiento de paquetes IP (IONOS España S.L.U., 2019).

15.1 FUNCIONAMIENTO

El protocolo ICMP desempeña varias funciones entre las que encontramos:

- Anunciar errores de red, tales como que un host o una parte completa de la red se encuentra inalcanzable debido a algún tipo de falla. Un paquete TCP/UDP dirigido a un puerto determinado sin ningún receptor conectado también se informa a través de ICMP.
- Anunciar congestión en la red. Cuando un router comienza a almacenar demasiados paquetes en el búfer, debido a la incapacidad de transmitirlos tan rápidos como se reciben, generará mensajes “ICMP Source Quench”. Dirigidos al remitente, estos paquetes deberían hacer que se reduzca la velocidad de transmisión de los paquetes. Por supuesto, generar demasiados mensajes de este tipo congestionaría aún más la red, por lo que, se usan con moderación.
- Ayudar en la solución de problemas. ICMP admite una función de eco, que simplemente envía un paquete en un viaje de ida y vuelta entre dos hosts. Ping, es una herramienta de administración de red común está basada en esta funcionalidad. Transmite una serie de paquetes midiendo los tiempos promedios de ida y vuelta y calculando los porcentajes de pérdida.
- Anunciar Timeouts. Si el campo TTL de un paquete IP obtiene el valor 0, el enrutador que descarta el paquete a menudo generará un paquete ICMP que avisa de este hecho. Traceroute otra herramienta ampliamente usada en mapeo de rutas de red trabaja de esta forma, enviando paquetes con pequeños valores TTL y observando los tiempos de espera ICMP.

15.2 OPERACIONES

ICMP genera bastantes tipos de mensajes muy útiles, tales como, Destination Unreachable, Echo Request y Reply, Redirect y Time Exceeded.

- ICMP Destination Unreachable (Destino inalcanzable): Cuando un router envía un mensaje ICMP de destino inalcanzable, significa que el enrutador no puede enviar el paquete a su destino final. Luego, el enrutador descarta el paquete original. Existen dos razones por las que un destino puede ser inalcanzable. Por lo general, el host de origen ha especificado una dirección inexistente o, con menos frecuencia, el enrutador no tiene una ruta hacia el destino.

Los mensajes de destino inalcanzable incluyen cuatro tipos básicos: red inalcanzable, host inalcanzable, protocolo y puerto inalcanzables.

- Los mensajes de red inalcanzable generalmente significan que se ha producido una falla en el enrutamiento o direccionamiento de un paquete.
- Los mensajes de host inalcanzable generalmente indican un error de entrega, como una máscara de subred incorrecta.
- Los mensajes de protocolo inalcanzable generalmente significan que el destino no admite el protocolo de capa superior especificado en el paquete.
- Los mensajes de puerto inalcanzable implican que el puerto o socket TCP no está disponible
- ICMP Echo Request y Reply (Mensaje de solicitud y respuesta): Cualquier host envía un mensaje de solicitud ICMP, generado por el comando ping, para probar la accesibilidad del nodo a través de una red. El mensaje de respuesta de ICMP indica que se puede acceder al nodo correctamente.
- ICMP Redirect (Redirección): El router envía un mensaje de redireccionamiento ICMP al host de origen para estimular un enrutamiento más eficiente. Además, el router, aún reenvía el paquete original al destino. Los redireccionamientos ICMP permiten que las tablas de enrutamiento de host permanezcan pequeñas porque es necesario conocer la dirección de un solo enrutador, incluso si ese enrutador no proporciona la mejor ruta. Incluso después de recibir un mensaje de redireccionamiento ICMP, algunos dispositivos pueden continuar usando la ruta menos eficiente.
- ICMP Time-exceeded (Tiempo excedido): El router envía un mensaje ICMP Tiempo excedido si el campo Tiempo de vida de un paquete IP (expresado en saltos o segundos) llega a cero. El campo Tiempo de vida evita que los paquetes circulen continuamente por la red si esta contiene un bucle de enrutamiento. Luego, el router descarta el paquete original.

Los diferentes mensajes ICMP se transmiten a través de tramas que cumplen con el siguiente formato:

	Bit 0-7	Bit 8-15	Bit 16-23	Bit 24-31
0	Tipo	Código	Suma de verificación	
32		Datos sobre la cabecera		

Ilustración 51: Formato de mensaje ICMP

- Tipo: Determina el tipo de mensaje al que hace referencia el paquete ICMP correspondiente.
- Código: Especifica este tipo de mensaje. Un tipo de mensaje ICMP puede obtener varios valores.
- Suma de Verificación: Garantiza la exactitud del mensaje
- Datos sobre la cabecera: Datos del protocolo ICMP que se crean y estructuran de manera muy diferente en función del tipo y de la instancia desencadenante. A menudo también se especifican aquí la cabecera IP y los primeros 64 bits del paquete de datos que es responsable del mensaje de error o de la solicitud de estado.

16 SNMP

El Protocolo Simple de Administración de Red (Simple Network Management Protocol) es un protocolo de internet para el manejo de dispositivos dentro de redes IP y pertenece a la capa de aplicación (Douglas R. Mauro & Kevin J. Schmidt, 2005).

16.1 HISTORIA

El Protocolo SNMP se usa para intercambiar información de gestión entre los dispositivos de la red. Su principal tarea consiste en monitorizar y administrar redes manteniendo un esquema común entre los dispositivos gestionados.

Hasta la llegada del Protocolo SNMP, la gestión de red había sido propietaria y los productos se desarrollaban por cada fabricante, complicando la gestión de redes heterogéneas, además, dada la dificultad de desarrollar este tipo de productos y el mercado restringido al que iban dirigidos, los productos eran caros y complejos. Con el crecimiento de la popularidad de TCP/IP, apareció un mercado lo suficientemente atractivo para que la IETF propusiera un estándar de gestión.

SNMP fue publicado inicialmente en 1989 pero las primeras aplicaciones no aparecieron hasta 1990. SNMP en su Versión 2 apareció en mayo de 1993 añadiendo nuevos comandos para reducir el tráfico de red, especialmente en redes grandes, además ofrece nuevas capacidades de notificación de errores, introduce la definición de nuevos objetos, más contadores, mejores herramientas de gestión y añadidos para garantizar la seguridad y la autenticación. SNMP en su última versión (SNMPv3) data de 2002 y posee cambios significativos con relación a sus predecesores, sobre todo en aspectos de seguridad.

16.2 FUNCIONAMIENTO

La funcionalidad del protocolo SNMP sigue una arquitectura cliente-servidor y se construye a partir de un sistema cuyos componentes trabajan de manera conjunta. Estos componentes son los siguientes:

16.2.1 Dispositivos administrativos

Elementos de una red administrada que contienen un agente SNMP. Tales como routers, switches, servidores de acceso, computadores, impresoras, hubs, bridges.

16.2.2 Agente

Es un componente de software que se ejecuta en el dispositivo a gestionar. Es un elemento pasivo y no origina mensajes, al contrario, responde a las peticiones del NMS. Únicamente iniciará la comunicación cuando deba comunicar una alarma porque el sistema se ha reiniciado o por fallos de seguridad en el sistema.

16.2.3 NMS (Network Management System)

Ejecuta aplicaciones que supervisan y controlan a los dispositivos administrados mediante SNMP. Estos NMS's conectan con los agentes SNMP para proporcionar volumen de recursos de procesamiento y memoria requeridos para la administración de la red. Cuando un NMS envía una solicitud, el agente devuelve la información solicitada desde el MIB .

La comunicación entre los dispositivos administrados y el NMS requiere de un servicio de comunicación que en el caso de SNMP utiliza un protocolo no orientado a conexión (UDP) para enviar un pequeño grupo de mensajes (PDUs) entre el NMS y los agentes. La utilización de un mecanismo de este tipo asegura que las tareas de administración de red no afectarán al rendimiento global de la red, ya que se evita la utilización de mecanismos de control y recuperación como los de un servicio orientado a conexión como TCP.

Los puertos comúnmente utilizados para SNMP son el 161 para mensajes SNMP de consulta y configuración SNMP; y el puerto 162 para alertas tipo traps que envía el dispositivo administrado a través del agente (Douglas R. Mauro & Kevin J. Schmidt, 2005).

Los paquetes utilizados para enviar consultas y respuestas SNMP poseen el formato indicado:

SNMP Message Format		
SNMP Version	SNMP Community	SNMP PDU

Ilustración 52: Formato de Mensaje SNMP

En los tres campos se contiene información referente a la versión, comunidad y el tipo de dato solicitado.

- Versión: Toma el valor en base de la versión del protocolo SNMP que se está utilizando.
- Comunidad: Nombre o palabra clave que se usa para la autenticación. Generalmente existe una comunidad de lectura llamada “public” y una comunidad de escritura llamada “private”.
- PDU: Contenido de la unidad de datos del protocolo, que depende de la operación que se ejecute.

Los mensajes del campo PDU tienen una estructura predeterminada que incluye los siguientes campos:

PDU				
PDU Type	Request-id	Error Status	Error Index	Variable Bindings

Ilustración 53: Formato PDU SNMP

- Tipo: Cualquier tipo de mensaje SNMP: Get, Get-Next, Get-Bulk, Set y Trap.
- Identificador: Se utiliza para relacionar peticiones y respuestas. El emisor asigna números de manera que cada consulta pendiente al mismo agente es identificada de manera inequívoca de modo que la aplicación SNMP puede correlacionar las respuestas emitidas con las peticiones pendientes y hacer frente a PDU duplicadas por un servicio de transporte inseguro como UDP.
- Estado e índice de error: Se emplea para indicar que ha ocurrido una anomalía mientras se procesaba una consulta.
- Enlazado de variables: Es una serie de nombres de variables con sus valores correspondientes (codificados en ASN.1).

Estado de error	Nombre	Significado
0	NoError	No hay error
1	tooBig	Demasiado grande
2	noSuchName	No existe esa variable
3	badValue	Valor incorrecto
4	readOnly	El valor es de solo lectura
5	genErr	Error genérico

Ilustración 54: Estado de Errores SNMP

16.2.4 Operaciones

16.2.4.1 Get

La petición get es iniciada por el NMS, el cual envía la petición al agente. El agente recibe la petición y la procesa. Algunos dispositivos sometidos a mucha carga, como lo son los ruteadores, pueden no ser capaces de responder a la petición y tendrán que rechazarla. Si el agente es capaz de recolectar la información solicitada, éste envía de vuelta un get-response al NMS, donde es procesada.

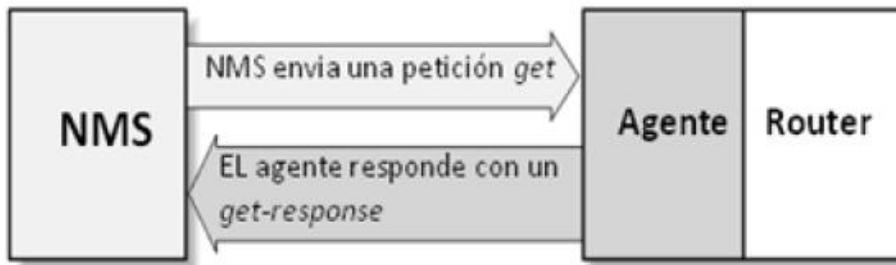


Ilustración 55: Operación GET SNMP

16.2.4.2 Get-Next

Este mensaje es usado para recorrer una tabla de objetos. Una vez que se ha usado un mensaje get para recoger el valor de un objeto, puede ser utilizado el mensaje get-next para repetir la operación con el siguiente objeto de la tabla. Siempre el resultado de la operación anterior será utilizado para la nueva consulta. De esta forma un NMS puede recorrer una tabla de longitud variable hasta que haya extraído toda la información para cada fila existente.

16.2.4.3 Get-Bulk

Este mensaje es usado por un NMS, que utiliza la versión 2 del protocolo SNMP normalmente cuando es requerida una larga transmisión de datos, tal como la recuperación de largas tablas. En este sentido es similar a la operación get-next usado en la versión 1 del protocolo, sin embargo, get-bulk es un mensaje que implica un método mucho más rápido y eficiente, ya que a través de un solo mensaje es posible solicitar la totalidad de la tabla.

16.2.4.4 Set

El comando set es usado para cambiar el valor de un objeto administrado o para crear una nueva fila en una tabla. Objetos que están definidos en la MIB como lectura y escritura o sólo escritura pueden ser alterados o creados usando este comando.

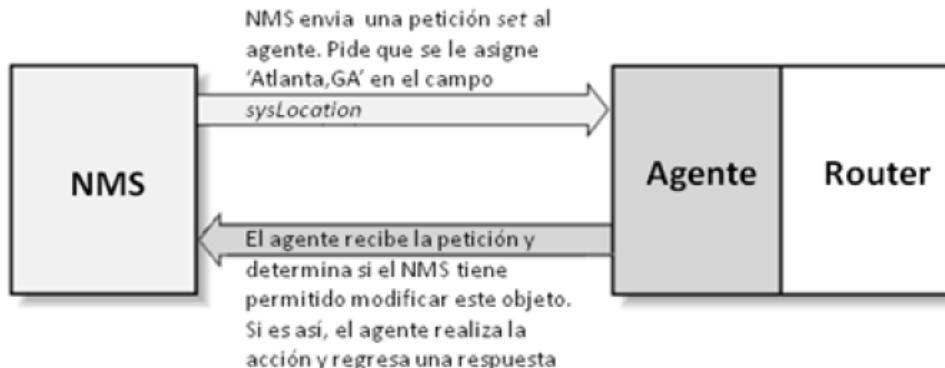


Ilustración 56: Operación SET SNMP

16.2.4.5 Trap

Un trap es la forma de comunicarse el agente con el NMS para decirle que algo malo ha sucedido. El trap se origina desde el agente y es enviado al destino, que normalmente es la dirección IP del NMS (Davin et al., 1990).

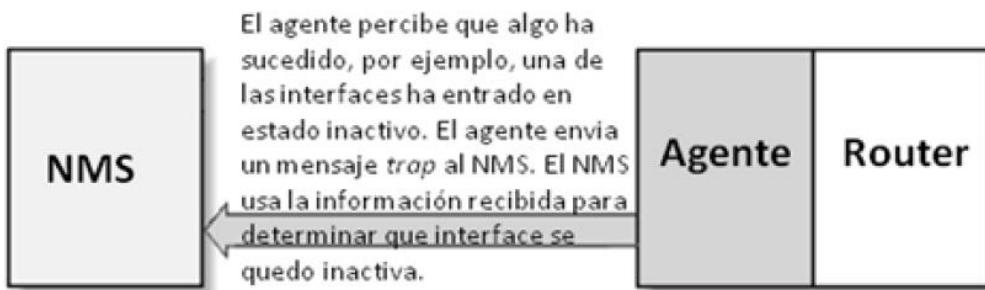


Ilustración 57: Operación TRAP SNMP

16.3 COMUNIDADES

El servicio SNMP ofrece una forma rudimentaria de seguridad mediante la utilización de nombres de comunidad y capturas de autenticación.

Los nombres de comunidad permiten autenticar los mensajes SNMP y, por lo tanto, proporcionan un esquema de seguridad rudimentario para el servicio SNMP.

No existe ninguna relación entre los nombres de comunidad y los nombres de dominio o de grupo de trabajo.

Un nombre de comunidad puede considerarse una contraseña compartida por las consolas de administración de SNMP y los equipos administrados. Éstos pueden ser configurados con el acceso de sólo lectura o lectura y escritura.

16.4 MIB

Una MIB (Management Information Base) es una base de información de administración, lo cual quiere decir que es una colección de información que está organizada jerárquicamente. Las MIB's son accedidas usando un protocolo de administración de red, como, por ejemplo, SNMP (Rose & McCloghrie, 1991).

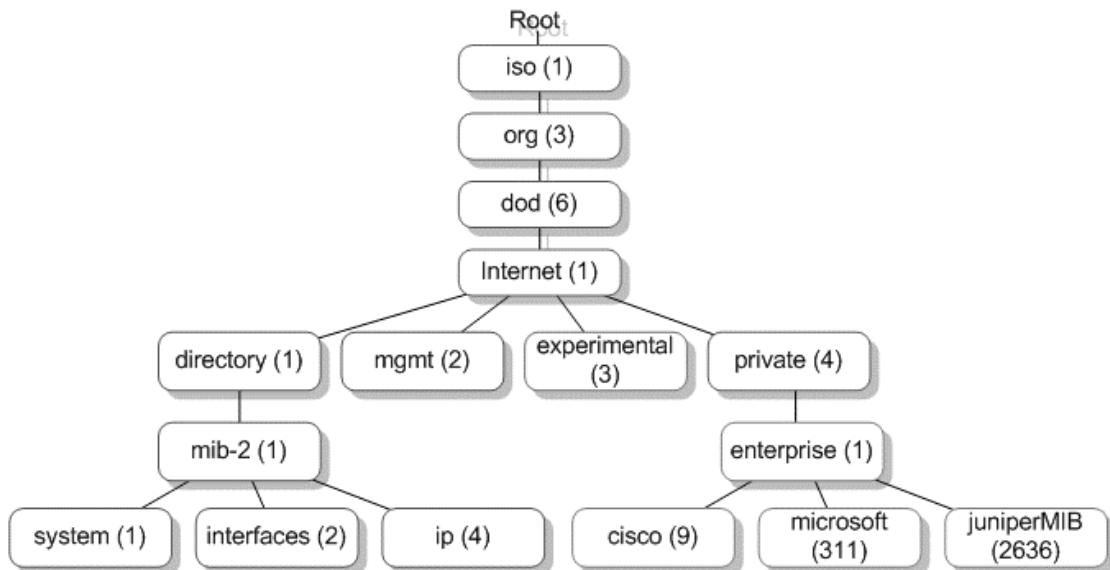


Ilustración 58: MIB SNMP

Un objeto administrado (algunas veces llamado objeto MIB, objeto, o MIB) es un número de características específicas de un dispositivo administrado. Los objetos administrados están compuestos de una o más instancias de objeto, que son esencialmente variables.

Existen dos tipos de objetos administrados: escalares y tabulares:

- Los objetos escalares definen una simple instancia de objeto.
- Los objetos tabulares definen múltiples instancias de objeto relacionadas que están agrupadas conjuntamente en tablas MIB.

Un identificador de objeto (object ID) únicamente identifica un objeto administrado en la jerarquía MIB. La jerarquía MIB puede ser representada como un árbol con una raíz anónima y los niveles, que son asignados por diferentes organizaciones. La base del árbol MIB se encuentra compuesto de varios grupos de objetos, los cuales en su conjunto son llamados mib-2. Los grupos son los siguientes:

- System: de este nodo cuelgan objetos que proporcionan información genérica del sistema gestionado.
- Interfaces: En este grupo está la información de las interfaces de red presentes en el sistema. Incorpora estadísticas de los eventos ocurridos en el mismo.
- At (address translation o traducción de direcciones): este nodo es obsoleto, pero se mantiene para preservar la compatibilidad con la MIB-I. En él se almacenan las direcciones de nivel de enlace correspondientes a una dirección IP.
- Ip: En este grupo se almacena la información relativa a la capa IP, tanto de configuración como de estadísticas.
- Icmp: En este nodo se almacenan contadores de los paquetes ICMP entrantes y salientes.

- Tcp: En este grupo está la información relativa a la configuración, estadísticas y estado actual del protocolo TCP.
- Udp: En este nodo está la información relativa a la configuración, estadísticas del protocolo UDP.
- Egp: Aquí está agrupada la información relativa a la configuración y operación del protocolo EGP.
- Transmission: De este nodo cuelgan grupos referidos a las distintas tecnologías del nivel de enlace implementadas en las interfaces de red del sistema gestionado.

16.5 VERSIONES EXISTENTES

El protocolo SNMP tiene varias versiones a través de las cuales un agente puede comunicarse:

- SNMPv1 (versión 1): el modelo de seguridad de esta primera versión es poco sofisticado. Todos los dispositivos que se comunican sobre SNMPv1 utilizan el string de la comunidad para verificar si la solicitud se puede llevar a cabo. Por defecto, el string de la comunidad private permite tanto leer como escribir información, mientras que el string de la comunidad public sólo permite operaciones de lectura.
- SNMPv2 (versión 2): introduce mejoras en términos de funcionamiento y de seguridad. Permite recuperar todas las entradas de una tabla en una sola operación y soluciona los problemas de monitorización y de carga de tráfico de la versión 1. La implementación más común para la versión 2 es SNMPv2c, que utiliza las características de la versión 2 sin implementar el nuevo modelo de seguridad, sino utilizando el mecanismo de string de la comunidad introducido en la versión 1.
- SNMPv3 (versión 3): la versión 1 utiliza como mecanismo de autenticación el parámetro referente a la comunidad, por lo que, si el agente y el administrador lo conocen, pueden interactuar entre ellos. Este tipo de protección es muy débil porque el texto se transmite en claro y puede explotarse mediante fuerza bruta. Para evitar esa falta de seguridad en las transmisiones, se creó una capa como complemento a las versiones 1 y 2, añadiendo a los mensajes SNMPv1 o SNMPv2 una cabecera adicional, dando lugar a lo que conocemos como la versión 3 del protocolo SNMP.

17 PROVEEDOR SERVICIOS DE INTERNET

El Proveedor de Servicios de Internet, ISP (por sus siglas en inglés, Internet Service Provider), es una compañía que suministra el servicio de internet a sus usuarios, a través de diferentes tecnologías como DSL, Cable modem, Dial-up y Wifi. Un ISP también ofrece servicios como email, Web hosting, DNS, FTP, voz sobre IP (VoIP), mensajería multimedia entre otros (Villa Avila & Villanueva Vivas, 2013).

Entre los objetivos de estas compañías se tiene:

- Mantener siempre la conectividad entre internet y sus clientes, para ello se debe disponer de un sistema de gestión de fallas.
- Mantener siempre los servicios básicos de un ISP.

Entre las consideraciones principales a tener en cuenta para diseñar un ISP a pequeña y mediana escala se encuentran:

- Cuál es el número de clientes conmutados y dedicados.

- Cuál es el ancho de banda asignado a los clientes.
- Qué servicios se prestarán en forma local desde la red interna, y cuáles desde Internet.
- Cuál es la estimación absoluta y porcentual de tráfico local y externo.
- Qué nivel de tolerancia a fallas se desea para el sitio.
- Qué tiempo promedio, y mínimo entre fallos se espera.
- Cuál será el tiempo de recuperación de en fallos.
- Qué alternativas de redundancia se utilizarán.

La mayoría de las compañías u organizaciones obtiene sus bloques de direcciones IPv4 de un ISP. Un ISP generalmente suministrará una pequeña cantidad de direcciones IPv4 utilizables (6 ó 14) a sus clientes como parte de los servicios. En cierto sentido, el ISP presta o alquila estas direcciones a la organización. Si se elige cambiar la conectividad de Internet a otro ISP, el nuevo ISP suministrará direcciones de los bloques de direcciones que ellos poseen, y el ISP anterior devuelve los bloques prestados a su asignación para prestarlos nuevamente a otro cliente.

17.1 ARQUITECTURA

Los proveedores de servicios de internet ofrecen los recursos necesarios tanto de hardware como de software, para que a través de ellos se permita el acceso a la red o a una conexión a internet.

La arquitectura de un ISP se puede organizar en dos secciones:

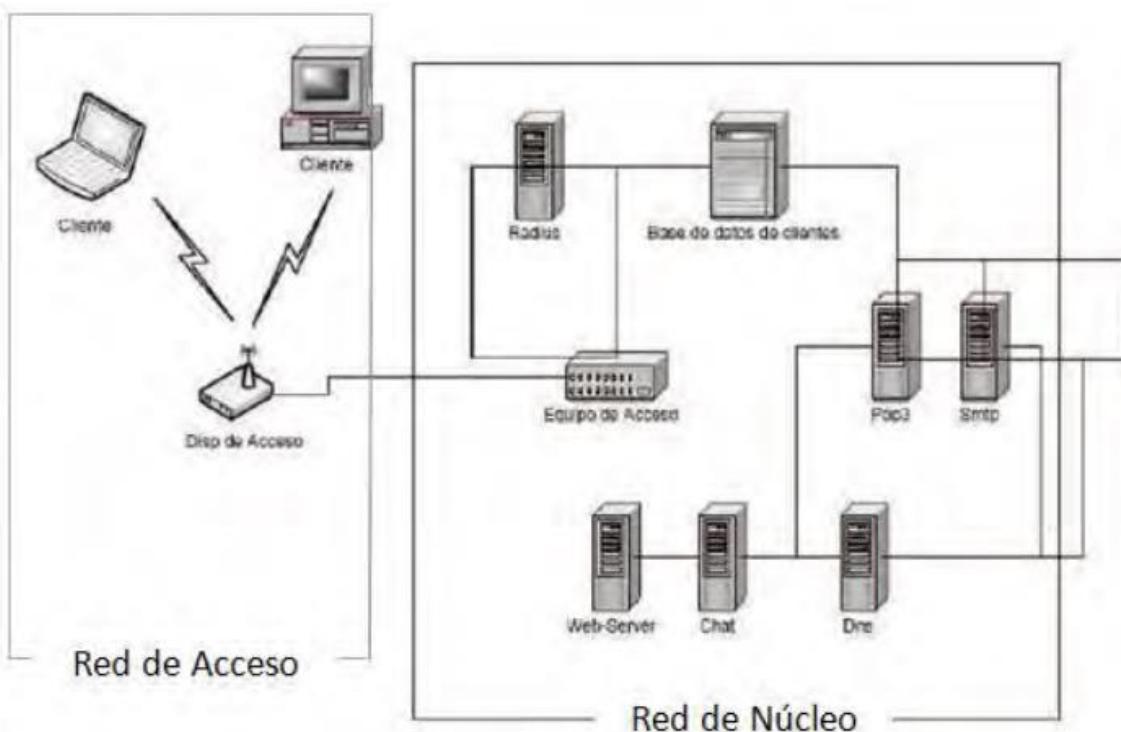


Ilustración 59: Arquitectura ISP

- **Red de núcleo:** El núcleo es donde se localizan los equipos de alta capacidad de transmisión. En este bloque se encuentran los elementos centrales de red, los cuales son capaces de administrar y gestionar. Aquí se encuentran los servidores AAA, la plataforma de servicio, la red IMS y sistemas de cobros, servicio de correo electrónico,

etc. La cantidad de tráfico que circula por esta red depende de la densidad de routers de concentración ya que las exigencias de las redes actuales son cada vez mayores. Igualmente, cuando la velocidad de acceso se incrementa, su rendimiento mejora, siendo la densidad de routers la que logra evitar que se produzcan picos en partes de la red.

- Red de acceso: La red de acceso es la que conecta al usuario directamente con la red, donde los únicos dispositivos de red más allá de la capa de acceso pueden ser teléfonos IP, puntos de acceso inalámbrico u otros en las instalaciones del cliente.

18 RASPBERRY PI OS

Raspberry Pi OS, antes conocido como Raspbian, es un sistema operativo Open Source lanzado en 2015 optimizado para funcionar en procesadores ARM, concretamente en el del Raspberry Pi, no generando fallos ni errores y pudiendo sacar todo el potencial a estas CPUs. Se deriva de Debian Linux y utiliza el entorno de escritorio LXDE de forma predeterminada (Eben Upton & Gareth Halfacree, 2016).

El sistema operativo Raspberry Pi viene con más de 35,000 paquetes: software precompilado incluido en un formato agradable para una fácil instalación en su Raspberry Pi.

Raspberry Pi OS ahora incluye imágenes de 32 y 64 bits. La versión de 64 bits, cuando se acopla a una placa Pi de 8 GB o incluso a una Pi de 4 GB, debería mostrar el potencial de la placa para las capacidades informáticas generales y multitarea. Dado que está basado en Linux, el sistema operativo Raspberry Pi se puede personalizar fácilmente para casos de uso individuales.

Raspberry Pi OS es un proyecto comunitario en desarrollo activo, con énfasis en mejorar la estabilidad y el rendimiento de tantos paquetes Debian como sea posible.

19 OTROS SISTEMAS OPERATIVOS

19.1 EL SISTEMA OPERATIVO LINUX

19.1.1 Historia

Linux, es un sistema operativo. Es una implementación de libre distribución UNIX para computadoras personales (PC), servidores y estaciones de trabajo. Es la denominación de un sistema operativo tipo-Unix y el nombre de un núcleo. Es uno de los paradigmas más prominentes del software libre y del desarrollo del código abierto, cuyo código fuente está disponible públicamente, para que cualquier persona pueda libremente usarlo, estudiarlo, redistribuirlo y, con los conocimientos informáticos adecuados, modificarlo (Liliana Allende et al., 2019).

Linux es usado como sistema operativo en una amplia variedad de plataformas de hardware y computadores, incluyendo las computadoras de escritorio (PCs x86 y x86-64, y Macintosh y PowerPC), servidores, supercomputadoras, mainframes, y dispositivos empotrados, así como teléfonos celulares.

En 1983 Richard Stallman fundó el proyecto GNU, con el fin de crear sistemas operativos parecidos a UNIX y compatibles con POSIX. Dos años más tarde creó la "Fundación del Software

"Libre" y escribió la GNU General Public License para posibilitar el software libre en el sistema de copyright.

El software GNU se extendía muy de prisa y en poco tiempo una multitud de programas fueron escritos, de manera que ya a principios de 1990 había bastante software GNU como para hacer un sistema operativo propio, pero faltaba el Kernel¹¹.

A principios de los años 1990, no había un sistema operativo libre completo. A pesar de que el proyecto GNU era desarrollado constantemente, no disponía sin embargo de ningún buen Kernel basado en UNIX, por el contrario, era un número de proyectos de softwares libres que podían ser traducidos en las variantes UNIX mediante el compilador de GNU.

El 5 de octubre de 1991, Linus anunció la primera versión "Oficial" de Linux, - versión 0.02. Con esta versión Linus pudo ejecutar Bash (GNU Bourne Again Shell) y gcc (Compilador GNU de C) pero no hubo una mejoría en su funcionamiento. En este estado de desarrollo ni se pensaba en los términos soporte, documentación, distribución. Después de la versión 0.03, Linus saltó en la numeración hasta la 0.10, más programadores a lo largo y ancho del internet empezaron a trabajar en el proyecto y después de revisiones, Linus incrementó el número de versión hasta la 0.95 (marzo 1992). En diciembre de 1993 el núcleo del sistema estaba en la versión 0.99 y la versión 1.0, llegó el 14 de marzo de 1994.

19.1.2 Características generales

- Multitarea. Linux es un sistema operativo que permite ejecutar varios procesos de manera simultánea. Como punto de referencia, MS-DOS es un sistema monotarea, y todos los sistemas Windows son multitarea.
- Multiusuario. Linux permite tener varios usuarios trabajando en la misma máquina y al mismo tiempo. En la actualidad el acceso se hace mediante otras máquinas. Por ejemplo, cuando una persona revisa su correo en un servidor, al mismo tiempo que otras, estamos siendo usuarios conectados en el servidor a la vez, cada uno viendo sus archivos y realizando sus tareas. Multiplataforma. Linux funciona en varias arquitecturas de procesadores, como Intel, Sparc, PowerPC, a diferencia de otros sistemas operativos, que sólo funcionan en una arquitectura determinada. Soporte de tecnologías venideras (multiprocesador). Linux en su versión de kernel más reciente soporta hasta 16 procesadores. Ya existen las versiones de 64 bits para Linux, manejo de grandes volúmenes de memoria RAM, etc. Es un software que actúa de sistema operativo.
- Memoria virtual. Usando paginación a disco, a una partición o un archivo en el sistema de archivos, o ambos, con la posibilidad de añadir más áreas de intercambio sobre la marcha. Un total de 16 zonas de intercambio de 128 Mb de tamaño máximo pueden ser usadas en un momento dado con un límite teórico de 2 Gb para intercambio. Este límite se puede aumentar fácilmente con el cambio y compilación de unas cuantas líneas en el código. Shells programables. El shell es un programa que se encuentra en el sistema que interpreta los comandos que son la interfaz con el sistema operativo y facilitan su control, el shell es el encargado de comunicaciones con el kernel.
- Consolas virtuales múltiples. Varias sesiones a través de la consola entre las que se puede cambiar con las combinaciones adecuadas de teclas (independiente del hardware de video). Se crean dinámicamente y es posible hasta tener 64.

19.2 WINDOWS SERVER 2016

Diseñado para medianas y grandes empresas, Windows Server 2016 es el sistema operativo recomendado para los servidores que ejecuten aplicaciones como sistemas de red, mensajería, inventario, bases de datos, sitios web y servidores de archivos e impresión. Proporciona alta

confiabilidad, rendimiento y un gran valor empresarial. Fue lanzado el 26 de septiembre de 2016 y fue una notable actualización al Windows 2012 Server. Está basado en tecnología NT (McCabe, 2016).

19.2.1 Características generales

Windows Server 2016 permite aumentar el rendimiento y la capacidad de un servidor mediante la adición de procesadores y memoria. Este enfoque para aumentar la capacidad del servidor es conocido como escalado vertical.

Es uno de los sistemas operativos más seguros que Windows haya lanzado al mercado. Protege las redes ante un posible código poco elaborado o malintencionado. Tiene avances en la funcionalidad de seguridad, permitiendo hacer un “blindaje” de máquinas virtuales y ofrece un servicio de “guardián de dispositivos” para mayor seguridad de los datos.

Es fácil de usar y administrar. Contiene herramientas que facilitan la configuración de funciones específicas de servidor y de las tareas habituales de administración de servidores. Permite una mayor productividad para administradores de tecnologías de la información y usuarios, a través de funciones avanzadas de administración de sistemas y almacenamiento.

Esta nueva versión de Windows Server presume de mejoras en 3 áreas fundamentales:

1. Seguridad: La mejora en la seguridad y la reducción de riesgos para el negocio se consigue integrando varias capas de seguridad en el sistema operativo.
2. CPD definido por software: Inspirados por la tecnología de Microsoft Azure (Servicio en la nube de Microsoft), aseguran una reducción de costes y una mayor flexibilidad gracias a las redes, el almacenamiento y los procesos definidos por software.
3. Innovación: Esta nueva versión de Windows Server incluye nuevas tecnologías como los “Contenedores” de Windows o Nano Server, que permiten nuevas formas de implementar y ejecutar aplicaciones, tanto locales como basadas en la nube.



IMPLEMENTACIÓN DE UN SISTEMA DE MONITORIZACIÓN DE DISPOSITIVOS Y SERVICIOS EN UNA RED DOMÉSTICA

ANEXOS

- **DATOS CLIENTE:** ESCUELA SUPERIOR DE INGENIERÍA, UNIVERSIDAD DE CÁDIZ.
AV. UNIVERSIDAD DE CÁDIZ, 10, 11519 PUERTO REAL, CÁDIZ
956 48 32 00
DIRECCION.ESI@UCA.ES
- **DATOS AUTOR:** PABLO MANUEL GARCÍA SANCHEZ
INGENIERO INFORMÁTICO
PABLO.GARCIASANCH@ALUM.UCA.ES

Cádiz, Abril de 2021

ÍNDICE

20	MANUAL DE USUARIO	99
20.1	CONFIGURACIÓN INICIAL	99
20.1.1	Configuración /etc/hosts	99
20.1.2	Configuración servidor web	99
20.1.3	Configuración base de datos	100
20.1.4	Configuración lenguaje de programación	104
20.1.5	Configuración sistema de alarmas	104
20.1.6	Configuración entorno ICMP	107
20.1.7	Configuración entorno SNMP	108
20.1.7.1	Configuración EMS	108
20.1.7.2	Configuración Agente SNMP en Linux	111
20.1.7.3	configuración Agente SNMP en Windows	119
20.1.8	Configuración entorno ISP	125
20.1.9	Configuración de los demonios	126
20.1.10	Configuración config.py	127
20.1.11	Configuración interfaz gráfica	127
21	LIBRERÍAS Y CÓDIGOS RELEVANTES DEL SISTEMA	130
21.1	ICMP	130
21.1.1	Vigilante.py	130
21.2	SNMP	134
21.2.1	Get-Response	134
21.2.1.1	config.py	134
21.2.1.2	SnmpCPULinux.py	135
21.2.1.3	SnmpCPUWindows.py	136
21.2.1.4	SnmpRamLinux.py	138
21.2.1.5	SnmpRamWindows.py	139
21.2.1.6	SnmpDiskLinux.py	141
21.2.1.7	SnmpDiskWindows.py	142
21.2.1.8	SnmpLinux.py	144
21.2.1.9	SnmpWindows.py	145
21.2.2	TRAPS	147
21.2.2.1	trap_handler.py	147
21.3	ISP	148
21.3.1	Config.py	148
21.3.2	lsp_monitor.py	149

20 MANUAL DE USUARIO

En este manual se recopila toda la información relativa al uso y configuración del sistema.

20.1 CONFIGURACIÓN INICIAL

Esta guía únicamente se deberá de seguir la primera vez que se conecte la Raspberry a nuestra red doméstica con el objetivo de monitorizar los dispositivos existentes en ella.

Nótese que este manual parte de la base de disponer de una Raspberry Pi previamente configurada con algún sistema UNIX.

20.1.1 Configuración /etc/hosts

Nuestro sistema administrador de red va a manejar una red con el protocolo TCP/IP.

El protocolo TCP/IP reconoce a las máquinas conectadas en red solamente por la dirección IP. Sin embargo, recordar los números de las máquinas no es fácil, motivo por el cual apareció el sistema de nombres. Es por ejemplo más fácil referirnos a las máquinas por “hostnames” que por direccionamiento IP.

Para facilitar el manejo de esta nomenclatura, TCP/IP maneja un servicio llamado resolución de nombres. Con él nos va a permitir que nos refiramos a los nombres de las máquinas y el sistema de resolución de nombres trasladará ese nombre a su dirección IP correspondiente para que TCP/IP pueda hacer su trabajo.

Esto lo hace el sistema a través de una tabla que reside en el archivo /etc/hosts de Linux. Allí debemos crear las equivalencias entre nombres y direcciones IP.

Por lo tanto, accedemos al fichero /etc/hosts y lo modificamos añadiendo una resolución de nombre a cada dirección IP monitorizada.

```
sudo nano /etc/hosts
```

1. 127.0.0.1	localhost
2. ::1	localhost ip6-localhost ip6-loopback
3. ff02::1	ip6-allnodes
4. ff02::2	ip6-allrouters
5.	
6. 127.0.1.1	raspberrypi
7. <<ip>>	<<hostname>>

20.1.2 Configuración servidor web

Lo primero que vamos a instalar es el servidor web. Este nos servirá para infinidad de cosas, y no solo para nuestra infraestructura. Simplemente con colocar una página web estática en el sitio adecuado lo podremos ver desde nuestra red local.

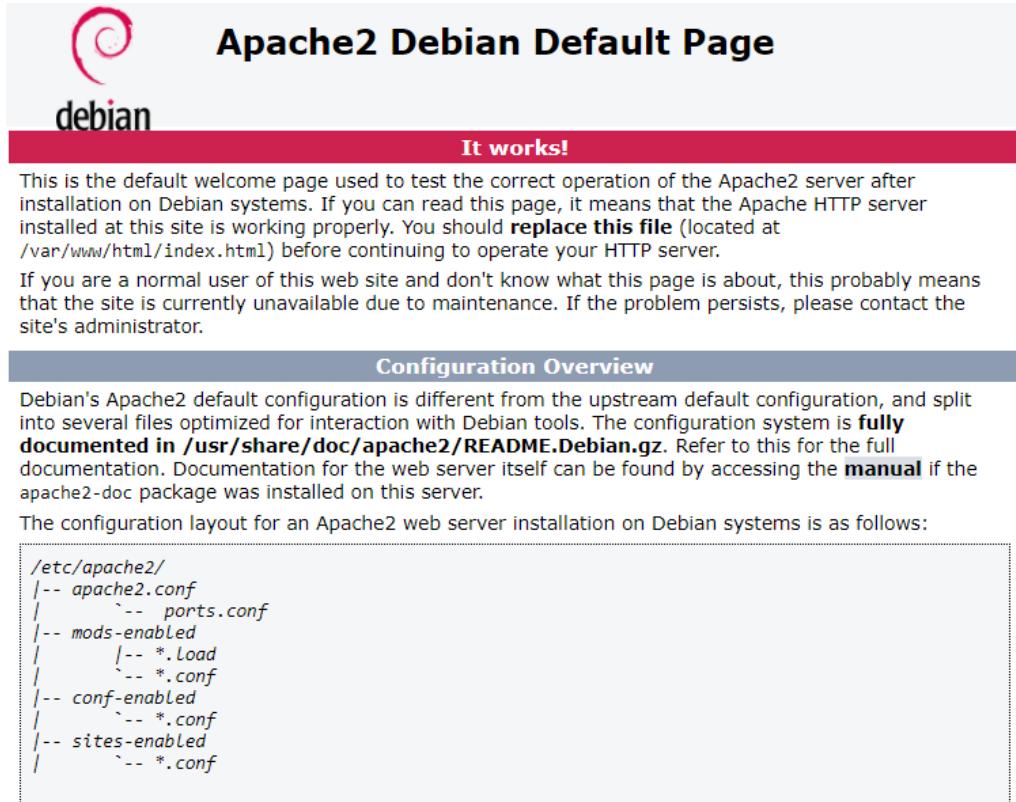
Para instalar el servidor web, solo tienes que ejecutar la siguiente orden:

```
sudo apt install apache2
```

Una vez completada la instalación, podemos comprobar si todo ha ido correctamente, abriendo el navegador de internet y escribiendo la dirección IP de nuestra Raspberry Pi. En nuestro caso particular:

```
http://<>ip>>
```

Si todo funciona como se espera tienes que ver una página como la que se muestra a continuación:



The configuration layout for an Apache2 web server installation on Debian systems is as follows:

```
/etc/apache2/  
|-- apache2.conf  
|   '-- ports.conf  
|-- mods-enabled  
|   '-- *.Load  
|   '-- *.conf  
|-- conf-enabled  
|   '-- *.conf  
|-- sites-enabled  
|   '-- *.conf
```

- `apache2.conf` is the main configuration file. It puts the pieces together by including all remaining configuration files when starting up the web server.
- `ports.conf` is always included from the main configuration file. It is used to determine the listening ports for incoming connections, and this file can be customized anytime.
- Configuration files in the `mods-enabled/`, `conf-enabled/` and `sites-enabled/` directories contain particular configuration snippets which manage modules, global configuration fragments, or virtual host configurations, respectively.
- They are activated by symlinking available configuration files from their respective `*-available/` counterparts. These should be managed by using our helpers `a2enmod`, `a2dismod`, `a2ensite`, `a2dissite`, and `a2enconf`, `a2disconf`. See their respective man pages for detailed information.
- The binary is called `apache2`. Due to the use of environment variables, in the default configuration, `apache2` needs to be started/stopped with `/etc/init.d/apache2` or `apache2ctl`. **Calling /usr/bin/apache2 directly will not work** with the default configuration.

Ilustración 60: Apache Debian Default Page

20.1.3 Configuración base de datos

Respecto al servidor de bases de datos escogeremos MariaDB por las razones ya comentadas anteriormente.

Para instalar MariaDB, solo tienes que ejecutar la siguiente orden en un emulador de terminal:

```
sudo apt install mariadb-server
```

Tras esto, y con independencia de la aplicación que utilicemos para gestionar bases de datos vamos a crear un usuario, que será el que utilicemos en las distintas aplicaciones. Para crear el usuario, lo primero que haremos será iniciar una sesión de MariaDB. Para ello, ejecutamos la siguiente orden:

```
sudo mariadb
```

Una vez dentro de la sesión de MariaDB, ejecutaremos las siguientes ordenes:

```
1. CREATE USER 'usuario'@'localhost' IDENTIFIED  
   BY 'contraseña';  
2. GRANT ALL PRIVILEGES ON * . * TO 'usuario'@'localhost';  
3. FLUSH PRIVILEGES;  
4. quit  
5.
```

En nuestro caso hemos creado el usuario pi para la base de datos con totales privilegios.

Existen diferentes herramientas que nos permiten gestionar bases de datos. Sin embargo, de entre las que se suelen utilizar para gestionar MySQL y MariaDB, hay que destacar phpMyAdmin. El uso de esta aplicación web está muy extendido.

phpMyAdmin es una aplicación web implementada en PHP y que nos permite gestionar bases de datos de MySQL/MariaDB desde un navegador de internet. Nos permite crear, modificar y eliminar bases de datos. Por supuesto, también nos permite, crear, modificar y eliminar tablas.

A continuación, procederemos a instalar phpMyAdmin, así, en una terminal, ejecutaremos la siguiente orden:

```
sudo apt install -y -t phpmyadmin
```

El proceso de instalación es muy sencillo. Durante el proceso, nos hará algunas preguntas que debemos contestar. La primera, debes elegir el servidor web que se debe configurar para que funcione phpMyAdmin. En nuestro caso elegiremos apache2.

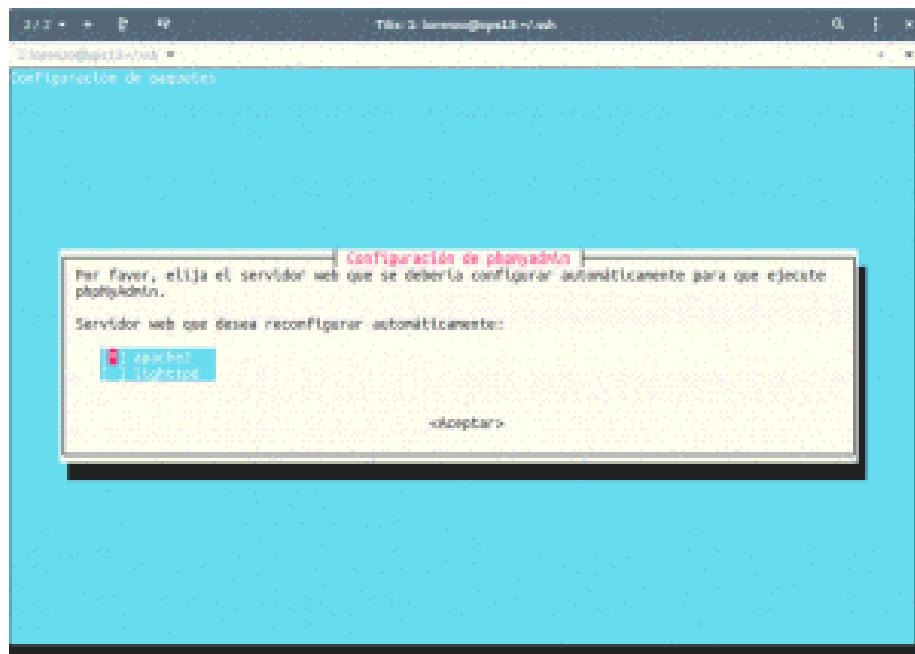


Ilustración 61: Configuración servidor phpMyAdmin

La siguiente pregunta va en relación con una base de datos que debe tener phpmyadmin. Simplemente contestamos que “Sí”.

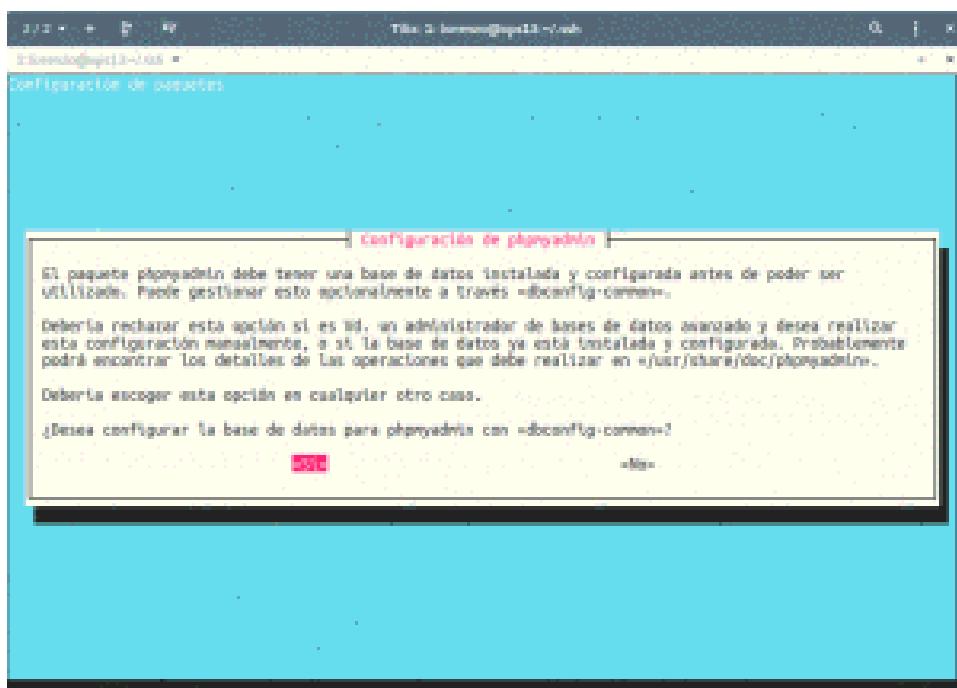


Ilustración 62: Configuración BD phpMyAdmin

Lo siguiente es que establezcas la contraseña para que phpmyadmin se registre con el servidor de bases de datos.

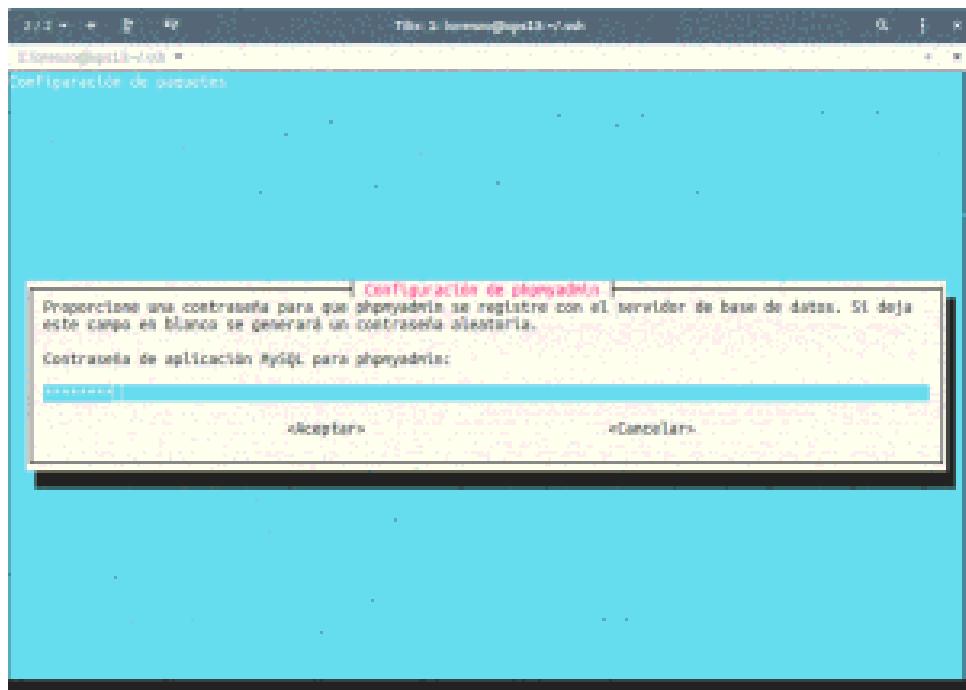


Ilustración 63: Configuración contraseña phpMyAdmin

Por último, deberemos confirmar la contraseña que has introducido en el paso anterior.

Terminado esto, debe funcionar, pero para evitar algún problema, reiniciamos el servidor Apache. Para esto, ejecutamos la orden:

```
sudo systemctl restart apache2
```

Finalizada la instalación de la herramienta, el siguiente paso es crear las diferentes tablas.

Para agilizar la tarea y no hacer necesario detallar el proceso de creación de cada una de las tablas de las que hacemos uso nos ayudaremos de un fichero sql el cuál encontraremos en el repositorio del proyecto:

```
https://github.com/pablogrsc/MONITORIZACION-DOMESTICA
```

Desde phpMyAdmin seleccionamos la funcionalidad de Importar y seleccionamos el fichero descargado.



Ilustración 64: Importar BD phpMyAdmin

20.1.4 Configuración lenguaje de programación

Para la consecución del proyecto se ha decidido utilizar Python como lenguaje de programación.

Python es un lenguaje de programación de alto nivel, de propósito general y muy popular. La última versión disponible en este momento es Python 3.8. Este lenguaje de programación se está utilizando en desarrollo web, aplicaciones de aprendizaje automático, junto con toda la tecnología de vanguardia en la industria del software.

Antes de instalar Python 3.8 hay algunas dependencias que necesitamos instalar.

Utilizaremos el siguiente comando para instalar cada una de ellas:

```
sudo apt-get install -y build-essential tk-dev libncurses5-dev libncursesw5-dev libreadline6-dev libdb5.3-dev libgdbm-dev libsqlite3-dev libssl-dev libbz2-dev libexpat1-dev liblzma-dev zlib1g-dev libffi-dev tar wget vim
```

Tras las dependencias requeridas, descargaremos Python. Lo podremos hacer desde su web oficial o utilizar la siguiente orden:

```
wget https://www.python.org/ftp/python/3.8.0/Python-3.8.0.tgz
```

Para extraer el paquete e instalarlo desde la fuente:

```
sudo tar zxf Python-3.8.0.tgz  
cd Python-3.8.0  
sudo ./configure --enable-optimizations  
sudo make -j 4  
sudo make altinstall
```

Con esto tendremos Python instalado, podemos comprobar nuestra versión utilizando:

```
python3.8 -V
```

20.1.5 Configuración sistema de alarmas

A continuación, veremos lo indispensable para crear nuestro propio bot para Telegram.

Lo primero, y el paso imprescindible para crear nuestro bot para Telegram es recurrir a @BotFather. Para esto solo tienes escribir en tu navegador de internet la dirección:

```
https://telegram.me/botfather
```

Otra opción es buscar directamente en Telegram botfather.

Una vez llegado a BotFather, verás lo siguiente:



Ilustración 65: BotFather Telegram

Pulsaremos en INICIAR en la parte inferior de la aplicación Telegram. Esto te lleva a las diferentes acciones que puedes emprender con BotFather. En general, cualquier bot para Telegram, que esté medianamente bien programado, debería mostrarte una ayuda como la que te muestra BotFather.

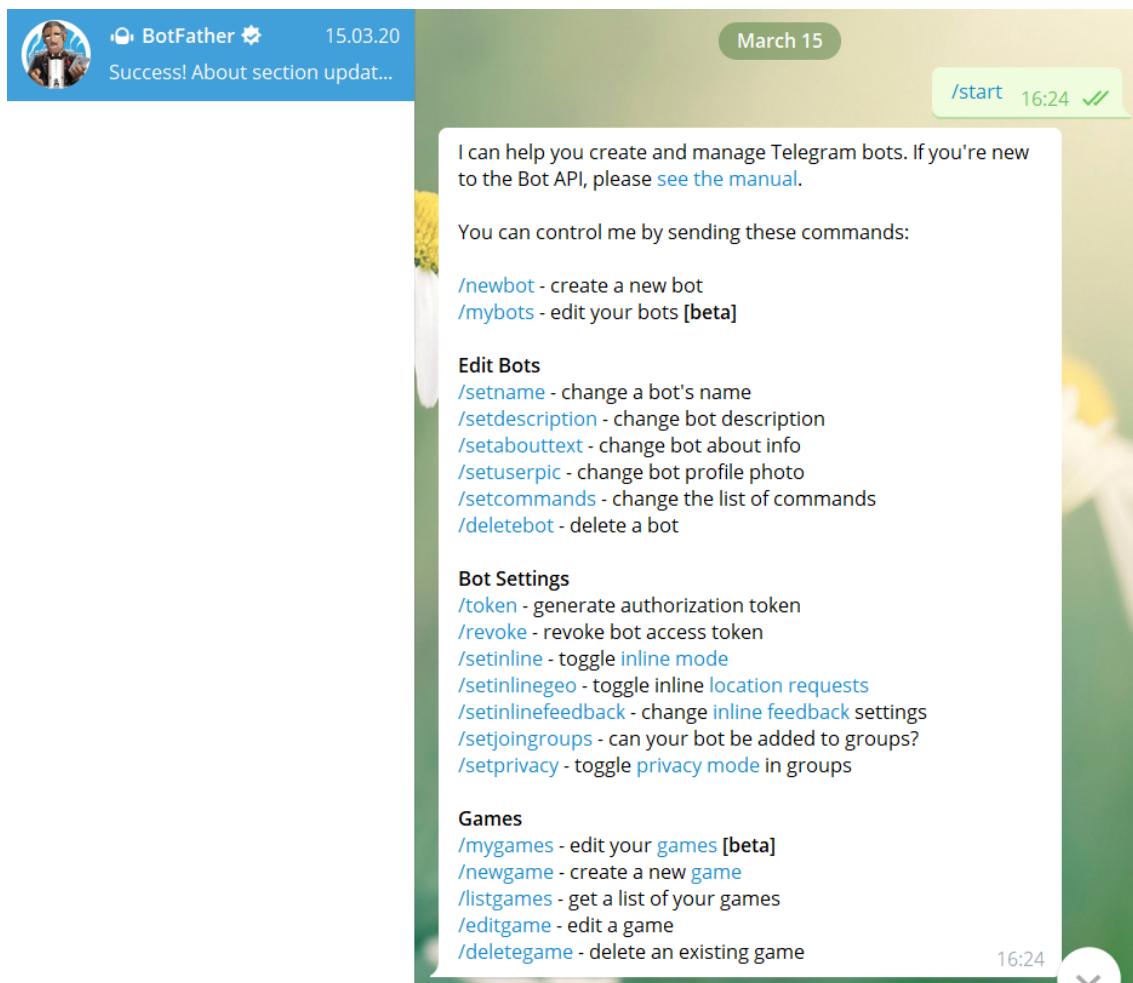


Ilustración 66: Configuración BOT Telegram 1

Llegados a este punto vamos a crear nuestro bot para Telegram. Para ello escribiremos un mensaje que sea /newbot. Esto nos devolverá un mensaje preguntando por el nombre de nuestro bot.

En nuestro caso, le llamaremos “CastellarBot”. Ahora nos pide que le pongamos un nombre de usuario, y debe cumplir con la condición de terminar en bot, por ejemplo “RaspihomeBot”.

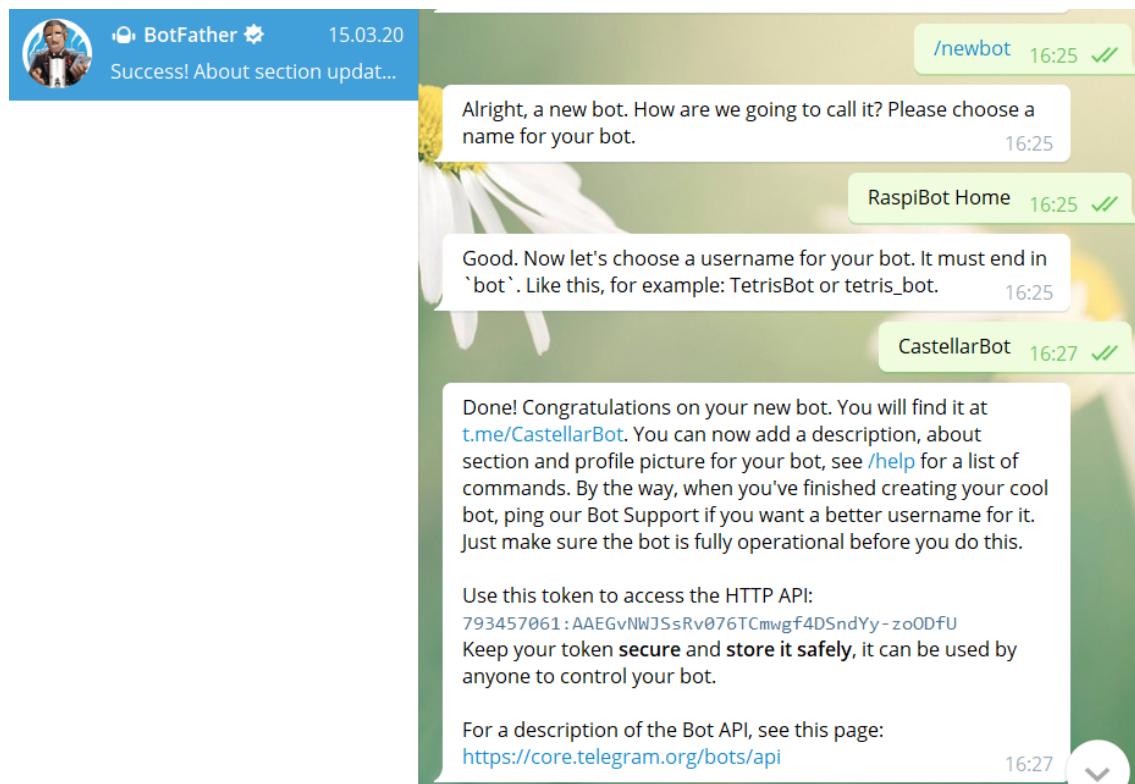


Ilustración 67: Configuración BOT Telegram 2

Perfecto, ya tenemos nuestro bot, y además nos han asignado un token que será el que utilizaremos en nuestras aplicaciones. Además, nos ha dado un enlace a nuestro bot:

<http://t.me/CastellarBot>

El siguiente paso es completar el perfil de nuestro bot para ayudar a los usuarios una vez lo tengamos en marcha. De nuevo desde el chat con BotFather, comenzaremos a editar el perfil de nuestro bot, utilizando los siguientes comandos:

- /setname para cambiar el nombre al bot
- /setdescription para cambiar la descripción de nuestro bot
- /setabouttext para cambiar la información que aparece en él acerca de
- /setuserpic para cambiar la imagen de perfil de nuestro bot

Hay otros comandos, pero ahora mismo, con estos ya tenemos más que suficiente.

Una vez creado nuestro bot, es fundamental que apuntemos el TOKEN. Lo vamos a utilizar en los siguientes pasos.

A continuación, crearemos un canal en Telegram. Este va a ser el lugar donde el bot te irá informando de lo que suceda.

Una vez creado el canal, añadimos a nuestro nuevo bot al canal, con permisos de administrador. Hecho esto, necesitaremos saber el número de identificación del canal. Para ello, el primer paso es que escribamos algo en nuestro canal, una frase o lo que consideremos. Despues, abrimos nuestro navegador y nos dirigimos a:

```
https://api.telegram.org/bot<TOKEN>/getUpdates/
```

sustituyendo <TOKEN>, por el token obtenido en el apartado anterior.

Aquí encontrarás un json parecido al que mosramos a continuación:

```
ok": true,  
  
"result": [  
  
    {  
  
        "update_id": 123456789,  
  
        "channel_post": {  
  
            "message_id": 1,  
  
            "chat": {  
  
                "id": -1234567890123,  
  
                "title": "Un canal",  
  
                "type": "channel"  
  
            },  
  
            "date": 1556000291,  
  
            "text": "texto de ejemplo"  
  
        }  
  
    }  
  
]
```

Con esto obtenemos el número del canal correspondiente a chat->id.

20.1.6 Configuración entorno ICMP

Este apartado del manual no requiere de ningún tipo de configuración previa. Todo funciona utilizando un sencillo script de Python el cuál analizaremos más adelante, junto con los demás scripts, en el apartado de Librerías y códigos del sistema.

20.1.7 Configuración entorno SNMP

Antes de empezar pasaremos a explicar cada uno de los paquetes que vamos a utilizar:

- **snmp:** Es un cliente de SNMP, el cual permite efectuar consultas a otros equipos con el servicio de snmpd activo
- **snmp-mibs-downloader:** La Base de Información Gestionada (Management Information Base o MIB) es un tipo de base de datos que contiene información jerárquica, estructurada en forma de árbol, de todos los dispositivos gestionados en una red de comunicaciones. Define las variables usadas por el protocolo SNMP para supervisar y controlar los componentes de una red
- **snmpd:** (Daemon incluído en paquete Net-SNMP) es un servicio (UDP puerto 161) que espera peticiones de cliente snmp, al recibir una solicitud, se procesa, recoge la información solicitada y / o lleva a cabo la operación solicitada (s) y devuelve la información al remitente
- **snmptrapd:** (Daemon incluído en el paquete NET-SNMP) es un servicio (UDP Puerto 162) que genera “traps” (notificaciones SNMP) cuando alguno de los procesos que monitoriza supera el umbral previamente establecido.

20.1.7.1 Configuración EMS

En primer lugar, debemos instalar los paquetes linux:

```
sudo apt-get install snmp  
sudo apt-get install snmp-mibs-downloader  
sudo download-mibs  
sudo apt-get install snmptrapd
```

El paquete snmp ofrece un conjunto de herramientas de líneas de comando para emitir solicitudes de SNMP a los agentes.

El paquete snmp-mibs-downloader ayudará a instalar y administrar los archivos de base de información gestionada (MIB), que realizan un seguimiento de los objetos de red.

Tras esto es importante modificar los archivos /etc/snmp/snmp.conf y /etc/default/snmpd para asegurarse que las herramientas de snmp puedan utilizar los datos MIB adicionales que instaló. Para ello:

```
sudo nano /etc/snmp/snmp.conf  
sudo nano /etc/default/snmpd
```

Para que el server pueda importar los archivos MIB, ingrese un comentario en la línea “mibs:”

```
/etc/snmp/snmp.conf
```

```
# As the snmp packages come without MIB files due to license  
reasons, loading  
  
# of MIBs is disabled by default. If you added the MIBs you  
can reenable  
  
# loading them by commenting out the following line.  
  
#mibs :
```

Y otro en la línea “export MIBS=”

```
/etc/default/snmpd  
  
# This file controls the behaviour of /etc/init.d/snmpd  
# but not of the corresponding systemd service file.  
# If needed, create an override file in  
# /etc/systemd/system/snmpd.service.d/local.conf  
# see man 5 systemd.unit and man 5 systemd.service  
  
# Don't load any MIBs by default.  
# You might comment this lines once you have the MIBs  
downloaded.  
#export MIBS=  
# snmpd options (use syslog, close stdin/out/err).  
SNMPDOPTS=' -Lswd -Lf /dev/null -u Debian-snmp -g Debian-snmp  
-p /run/snmpd.pid'
```

A continuación, vamos a configurar el servidor para recibir traps. Haremos uso del daemon snmptrapd comentado anteriormente ya que snmp no dispone de una herramienta específica para manejar traps.

Nótese que este paquete vuela por defecto los traps recibidos en el archivo de logs predeterminado de Ubuntu “/var/log/syslog”. Para cambiarlo, abrimos el archivo *etc/default/snmpd* y lo modificamos haciendo uso de la instrucción -Lf:

```
sudo nano /etc/default/snmptrapd
```

```
etc/default/snmptrapd

# This file controls the activity of snmptrapd

# snmptrapd control (yes means start daemon). As of net-snmp
version

# 5.0, master agentx support must be enabled in snmpd before
snmptrapd

# can be run. See snmpd.conf(5) for how to do this.

TRAPDRUN=yes

# snmptrapd options (use syslog).

TRAPDOPTS=' -Lf var/log/traps.log -Lsd -p /run/snmptrapd.pid'
```

De esta forma conseguimos que los traps entrantes se guarden en un archivo de logs separado. Concretamente en var/log/traps.log.

Debido a un error desconocido, a veces, se puede dar el caso que el daemon de snmptrapd no cargue su configuración inicial de etc/default/snmptrapd al ser iniciado.

Por lo que puede suceder que el servicio iniciado no vuelque el log en /var/log/traps.log y sólo lo haga en /var/log/syslog.

Para solucionarlo, en primer lugar, paramos el servicio de snmptrapd iniciado por defecto sino dará error de que el puerto UDP:162 está ocupado, ya que, tendremos el servicio duplicado:

```
sudo service snmptrapd stop

sudo netstat -lnp | grep 162 //Procesos que ocupan el udp:162
```

En segundo lugar, arrancamos un servicio nuevo de snmptrapd desde el shell obligándolo a que utilice traps.log con esta instrucción:

```
sudo snmptrapd -A -Lf /var/log/traps.log
```

Lo siguiente es configurar el daemon snmptrapd para que invoque al script de python encargado de mandar el último trap entrante a través de Telegram. Para ello, abrimos, etc/snmp/snmptrapd.conf:

```
sudo nano etc/snmp/snmptrapd.conf
```

etc/snmp/snmptrapd.conf

```
authCommunity log,execute,net castellar

format2 %.4y-%.2m-%.21 %.2h:%.2j:%.2k %B [%b] :\n%v\n

outputOption s

traphandle default ../../Monitoring/SNMP/trap_handler.sh
```

20.1.7.2 Configuración Agente SNMP en Linux

En primer lugar, debemos instalar el paquete linux:

```
sudo apt-get install snmpd
```

Como verdadero sistema servidor-cliente, el agente (cliente) no tiene ninguna de las herramientas externas necesarias para configurar sus propios ajustes de SNMP. Puede modificar algunos archivos de configuración para realizar algunos cambios, pero la mayoría de los que son necesarios se harán mediante la conexión al servidor agente del servidor administrador.

Cómo ya hemos comentado anteriormente, en este proyecto, se usará la versión 2c del protocolo SNMP. Nótese que más adelante crearemos un usuario SNMPv3 por requerimientos del proyecto.

Para comenzar, en el servidor agente, abrimos el archivo de configuración del demonio con privilegios sudo:

```
sudo nano /etc/snmp/snmpd.conf
```

Dentro de este, deberá realizar algunos cambios. Estos se utilizarán principalmente para iniciar la configuración, de modo que pueda administrarla desde su otro servidor.

En primer lugar, estableceremos el syslocation y syscontact. El primero, se refiere a el lugar físico donde usualmente se encuentra el sistema y el segundo, a la información de contacto del administrador.

/etc/snmp/snmpd.conf

```
#####
# SECTION: System Information Setup
#
# syslocation: The [typically physical] location of the system.
#   Note that setting this value here means that when trying to
#   perform an snmp SET operation to the sysLocation.0 variable
# will make
#   the agent return the "notWritable" error code.  IE, including
```

```
#      this token in the snmpd.conf file will disable write access
#      to
#      the variable.
#      arguments: location_string
sysLocation    <<location>>
sysContact     <<email>>

# syservices: The proper value for the sysServices object.
#      arguments: syservices_number
sysServices    72
```

Segundo, configuraremos el modo de operar del agente. Cambiaremos la directiva agentAddress. Por el momento, está configurada para permitir solo las conexiones que se originan desde la computadora local. Deberá excluir la línea actual y quitar el comentario de la que está debajo de ella, que es la que permite todas las conexiones.

```
/etc/snmp/snmpd.conf

#####
# SECTION: Agent Operating Mode
#
# This section defines how the agent will operate when it
# is running.

# agentaddress: The IP address and port number that the agent will
listen on.
# By default the agent listens to any and all traffic from any
# interface on the default SNMP port (161). This allows you to
# specify which address, interface, transport type and port(s)
that you
# want the agent to listen on. Multiple definitions of this
token
# are concatenated together (using ':'s).
# arguments: [transport:]port[@interface/address],...

# Listen for connections from the local system only
#agentAddress  udp:127.0.0.1:161
# Listen for connections on all interfaces (both IPv4 *and* IPv6)
agentAddress  udp:161,udp6:[::1]:161
```

Por otro lado, haremos que todos los dispositivos de nuestra red actuen como “SNMP Master Agents”, esto es, para que cada dispositivo monitorizado se comunique directamente con nuestro EMS.

Una estructura jerárquica SNMP donde cada master-agent almacena información de los múltiples sub-agents que tenga a su cargo tiene sentido en redes medianamente grandes lo cuál no se adapta a nuestro escenario.

```
# master: Should the agent operate as a master agent or not.
# Currently, the only supported master agent type for this token
# is "agentx".
```

```
#  
#     arguments: (on|yes|agentx|all|off|no)  
  
master agentx
```

Tercero definiremos el acceso al servicio SNMP de nuestro dispositivo.
Para ello definiremos las vistas donde especificaremos a que OIDs va a estar restringido el acceso en cada una de ellas.
En nuestro caso, únicamente nos crearemos una, con acceso total a todos los OIDS.

```
/etc/snmp/snmpd.conf  
  
#####  
### SECTION: Access Control Setup  
#  
#     This section defines who is allowed to talk to your running  
#     snmp agent.  
  
# Views  
#     arguments viewname included [oid]  
  
view systemview included .1 80
```

A continuación, vamos a definir la comunidad la cuál tendrá acceso a la información SNMP almacenada por el agente:

```
# rocommunity: a SNMPv1/SNMPv2c read-only access community name  
#     arguments: community [default|hostname|network/bits] [oid |  
-V view]  
  
# Read-only access to everyone to the systemonly view  
rocommunity <<community>> <<ip>> -V systemview
```

Por último definiremos un usuario SNMPv3, a pesar de estar utilizar la versión 2c del protocolo SNMP, ya que es necesario para utilizar el servicio DisMan encargado de la gestión de los traps SNMP.

```
# SNMPv3 AUTHENTICATION  
# SNMPv3 doesn't use communities, but users with (optionally) an  
# authentication and encryption string. This user needs to be  
created  
# with what they can view with rouser/rwuser lines in this file.  
#  
# createUser username (MD5|SHA|SHA-512|SHA-384|SHA-256|SHA-224)  
authpassphrase [DES|AES] [privpassphrase]  
# e.g.  
# createuser authPrivUser SHA-512 myauthphrase AES myprivphrase  
#  
# This should be put into /var/lib/snmp/snmpd.conf
```

```
#  
# rouser: a SNMPv3 read-only access username  
#         arguments: username [noauth|auth|priv] [OID | -V VIEW  
[CONTEXT]]]  
  
#An SNMPv3 username is required to authorize the DisMan service.  
createUser internalUser  
iquerySecName internalUser  
rouser internalUser
```

Tras la configuración inicial básica añadiremos algunas directivas para monitorizar a través de GET-RESPONSE la información relacionada con el espacio en disco, carga de CPU y memoria RAM.

```
/etc/snmp/snmpd.conf  
  
#####  
#####  
# SECTION: GET-RESPONSE Monitoring  
#  
# This section defines how the agent will operate when it  
# receives a snmp-get instruction  
  
# Process Monitoring  
#  
# Checks to see if processes are running on the agent  
# machine. An error flag (1) and a description message are then  
# passed to the 1.3.6.1.4.1.2021.2.1.100 and  
# 1.3.6.1.4.1.2021.2.1.101 MIB columns (respectively) if the  
# program is not found in the process table as reported by  
# "/bin/ps -e".  
  
# make sure sshd is running  
proc sshd  
  
#  
# Disk Monitoring  
#  
# Checks the disks mounted at PATH for available disk  
# space. If the disk space is less than MINSPACE (kB) if speci-  
# fied or less than MINPERCENT (%) if a % sign is specified, or  
# DEFDISKMINIMUMSPACE (kB) if not specified, the associated entry  
# in the 1.3.6.1.4.1.2021.9.1.100 MIB table will be set to (1)  
# and a descriptive error message will be returned to queries of  
# 1.3.6.1.4.1.2021.9.1.101.  
#Check the / partition and make  
sure it contains at least 10 gig  
disk      /    976563  
  
# Walk the UCD-SNMP-MIB::dskTable to see the resulting output  
# Note that this table will be empty if there are no "disk"  
entries in the snmpd.conf file  
  
#
```

```
# System Load
#
# Checks the load average of the machine and returns an error
# flag (1), and an text-string error message to queries of
# 1.3.6.1.4.1.2021.10.1.100 and 1.3.6.1.4.1.2021.10.1.101
# (respectively) when the 1-minute, 5-minute, or 15-minute aver-
# ages exceed the associated maximum values.
#           # Unacceptable 1-, 5-, and 15-
# minute load averages
load 2 1.8 1.5
```

Por último, como también queremos que nuestro agente nos actualice a través de Traps de cualquier cambio que le pueda ocurrir, añadimos estas directrices al fichero de configuración del daemon.

```
/etc/snmp/snmpd.conf

#####
# SECTION: TRAP Monitoring
#
# This section defines how the agent will operate when it
# sends a trap event notification.

# send SNMPv2c traps
trap2sink <><> <><>

#
# Event MIB - automatically generate alerts
#
# generate traps on UCD error
conditions
defaultMonitors      yes
# generate traps on
linkUp/Down
linkUpDownNotifications yes
#generate a trap when free memory drops
below 1,000,000KB. The free memory trap also includes the amount
of total real memory.
monitor -r 60 RamAlmostFull -o memTotalReal memAvailReal <
1000000
#generate a trap when the 1 minute interval
reaches 90%, the 5 minute interval reaches 70% or the 15 minute
interval reaches 50%.
monitor -r 30 ProcessorTooBusy -o laNames -o laErrMessage
"laTable" laErrorFlag !=0
#generate a trap when a disk is 90% full.
monitor -r 60 DiskAlmostFull dskPercent > 90
#generate authentication failure traps
authtrapenable 1
```

Resultando el archivo /etc/snmp/snmpd.conf de la siguiente forma:

```
/etc/snmp/snmpd.conf
```

```
#####
#####
#
# snmpd.conf
# An example configuration file for configuring the Net-SNMP agent
('snmpd')
# See snmpd.conf(5) man page for details
#
#####
#
# SECTION: System Information Setup
#
#
# syslocation: The [typically physical] location of the system.
#   Note that setting this value here means that when trying to
#   perform an snmp SET operation to the sysLocation.0 variable
will make
#   the agent return the "notWritable" error code. IE, including
#   this token in the snmpd.conf file will disable write access
to
#   the variable.
#   arguments: location_string
sysLocation    <<location>>
sysContact     <<email>>

# syservices: The proper value for the sysServices object.
#   arguments: syservices_number
sysServices    72

#####
#####
#
# SECTION: Agent Operating Mode
#
#
# This section defines how the agent will operate when it
is running.

# agentaddress: The IP address and port number that the agent will
listen on.
#   By default the agent listens to any and all traffic from any
#   interface on the default SNMP port (161). This allows you to
#   specify which address, interface, transport type and port(s)
that you
#   want the agent to listen on. Multiple definitions of this
token
#   are concatenated together (using ':'s).
#   arguments: [transport:]port[@interface/address],...

# Listen for connections from the local system only
#agentAddress  udp:127.0.0.1:161
# Listen for connections on all interfaces (both IPv4 *and* IPv6)
agentAddress  udp:161,udp6:[::1]:161

# master: Should the agent operate as a master agent or not.
#   Currently, the only supported master agent type for this token
```

```
#      is "agentx".  
#  
#      arguments: (on|yes|agentx|all|off|no)  
  
master agentx  
  
#####  
#####  
# SECTION: Access Control Setup  
#  
#      This section defines who is allowed to talk to your running  
#      snmp agent.  
  
# Views  
#      arguments viewname included [oid]  
  
view systemview included .1 80  
  
# rocommunity: a SNMPv1/SNMPv2c read-only access community name  
#      arguments: community [default|hostname|network/bits] [oid |  
-V view]  
  
# Read-only access to everyone to the systemonly view  
rocommunity <><> -V systemview  
  
# SNMPv3 AUTHENTICATION  
# SNMPv3 doesn't use communities, but users with (optionally) an  
# authentication and encryption string. This user needs to be  
created  
# with what they can view with rouser/rwuser lines in this file.  
#  
# createUser username (MD5|SHA|SHA-512|SHA-384|SHA-256|SHA-224)  
authpassphrase [DES|AES] [privpassphrase]  
# e.g.  
# createuser authPrivUser SHA-512 myauthphrase AES myprivphrase  
#  
# This should be put into /var/lib/snmp/snmpd.conf  
#  
# rouser: a SNMPv3 read-only access username  
#      arguments: username [noauth|auth|priv] [OID | -V VIEW  
[CONTEXT]]]  
  
#An SNMPv3 username is required to authorize the DisMan service.  
createUser internalUser  
iquerySecName internalUser  
rouser internalUser  
  
#####  
#####  
# SECTION: GET-RESPONSE Monitoring  
#  
#      This section defines how the agent will operate when it  
#      receives a snmp-get instruction
```

```
# Process Monitoring
#
# Checks to see if processes are running on the agent
# machine. An error flag (1) and a description message are then
# passed to the 1.3.6.1.4.1.2021.2.1.100 and
# 1.3.6.1.4.1.2021.2.1.101 MIB columns (respectively) if the
# program is not found in the process table as reported by
# "/bin/ps -e".
#
# make sure sshd is running
proc sshd

#
# Disk Monitoring
#
# Checks the disks mounted at PATH for available disk
# space. If the disk space is less than MINSPACE (kB) if speci-
# fied or less than MINPERCENT (%) if a % sign is specified, or
# DEFDISKMINIMUMSPACE (kB) if not specified, the associated entry
# in the 1.3.6.1.4.1.2021.9.1.100 MIB table will be set to (1)
# and a descriptive error message will be returned to queries of
# 1.3.6.1.4.1.2021.9.1.101.
#Check the / partition and make
#sure it contains at least 10 gig
disk      /    976563

#
# System Load
#
# Checks the load average of the machine and returns an error
# flag (1), and an text-string error message to queries of
# 1.3.6.1.4.1.2021.10.1.100 and 1.3.6.1.4.1.2021.10.1.101
# (respectively) when the 1-minute, 5-minute, or 15-minute aver-
# ages exceed the associated maximum values.
# Unacceptable 1-, 5-, and 15-
# minute load averages
load 2 1.8 1.5

#####
## SECTION: TRAP Monitoring
#
# This section defines how the agent will operate when it
# sends a trap event notification.

# send SNMPv2c traps
trap2sink <><> <><>
#
# Event MIB - automatically generate alerts
#
```

```
# generate traps on UCD error
conditions
defaultMonitors      yes
linkUp/Down           # generate traps on
linkUpDownNotifications yes
#generate a trap when free memory drops
below 1,000,000KB. The free memory trap also includes the amount
of total real memory.
monitor -r 60 RamAlmostFull -o memTotalReal memAvailReal <
1000000
#generate a trap when the 1 minute interval
reaches 90%, the 5 minute interval reaches 70% or the 15 minute
interval reaches 50%.
monitor -r 30 ProcessorTooBusy -o laNames -o laErrMessage
"laTable" laErrorFlag !=0
#generate a trap when a disk is 90% full.
monitor -r 60 DiskAlmostFull dskPercent > 90
#generate authentication failure traps
authtrapenable 1
```

20.1.7.3 configuración Agente SNMP en Windows

A continuación, especificaremos los pasos necesarios para activar el Agente SNMP en dispositivos con Windows Server 2003.

Está característica viene instalada de forma nativa, únicamente habría que activarla, para ello:

En primer lugar, nos dirigimos a Inicio, seleccionamos Panel de control, herramientas administrativas y, a continuación, hacemos clic en Administración de equipos.

En segundo lugar, En el árbol de la consola, expandimos servicios y aplicaciones y, a continuación, hacemos clic en servicios. Tras esto, buscamos el servicio SNMP.

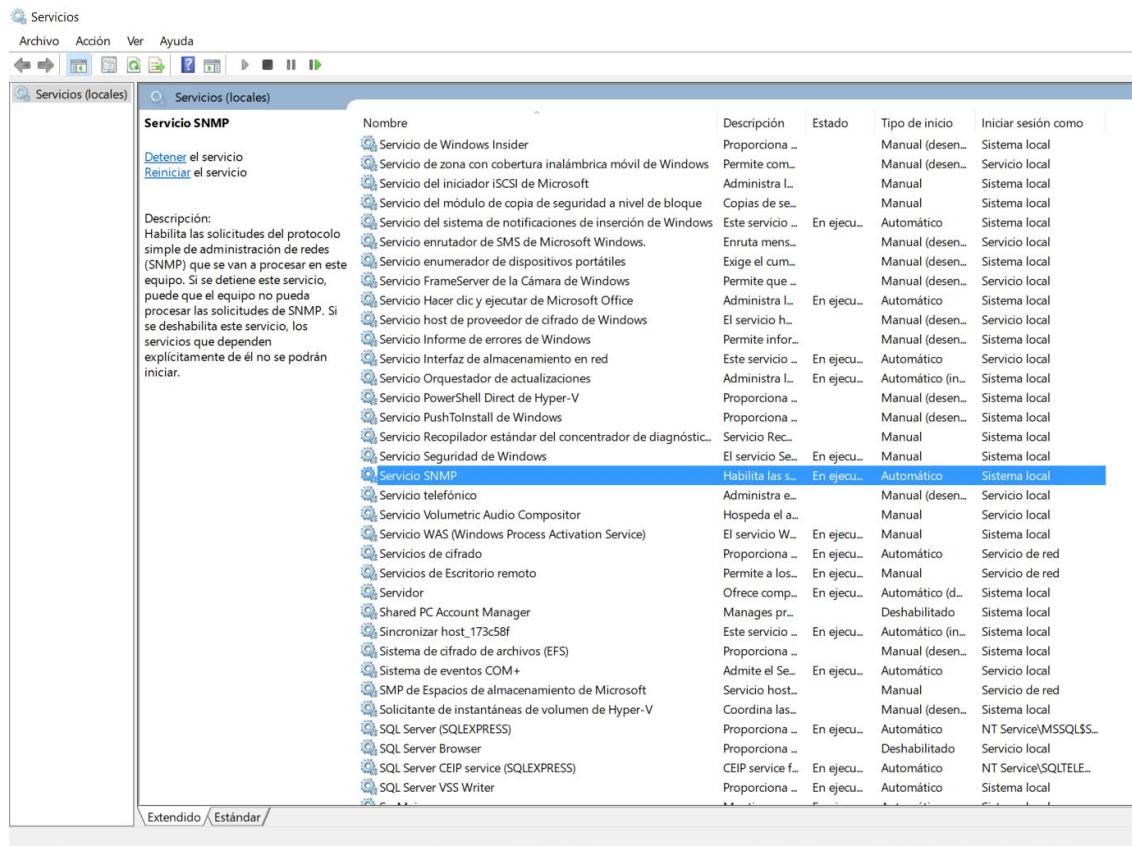


Ilustración 68: Configuración SNMP Windows

Una vez localizado nuestro servicio, nos dirigimos a la pestaña agente y escribimos el nombre del usuario o administrador del equipo en el cuadro contacto y, a continuación, la ubicación física del equipo o contacto en el cuadro Ubicación.

Estos comentarios se tratan como texto y son opcionales.

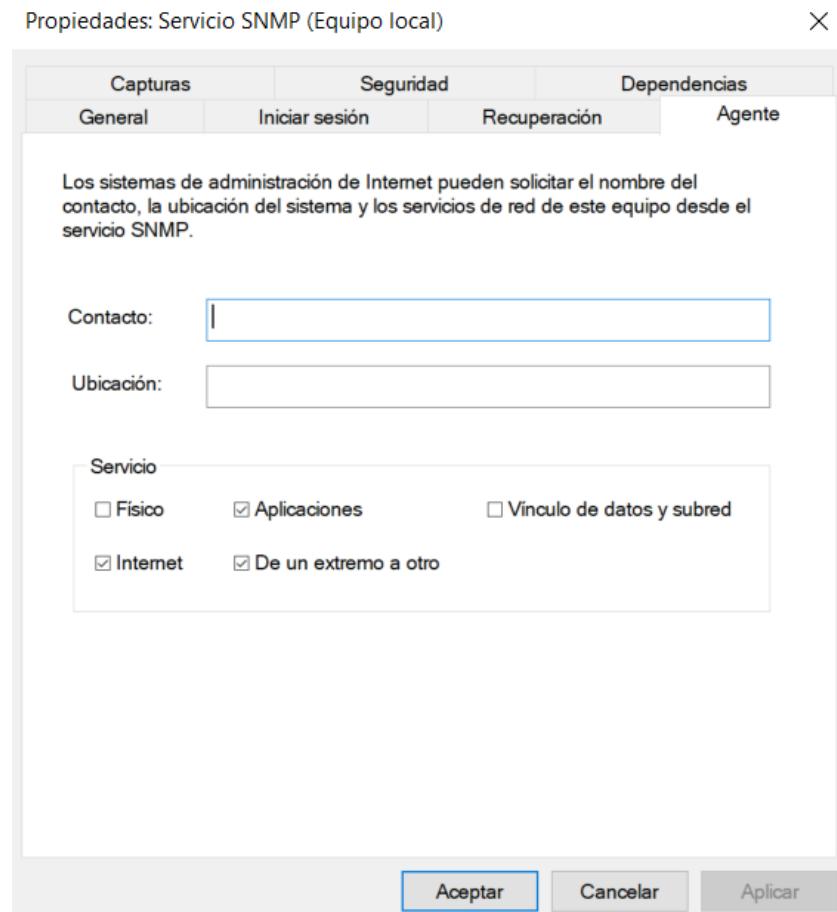


Ilustración 69: Configuración Agente SNMP Windows

En tercer lugar, abrimos la pestaña capturas.

En el cuadro Nombre de la comunidad, escribimos el nombre de la comunidad con distinción entre mayúsculas y minúsculas al que este equipo enviará mensajes de captura y, a continuación, hacemos clic en Agregar a la lista.

En destinos de captura, haga clic en Agregar.

En el cuadro nombre de host, dirección IP o IPX , escribimos el nombre, la dirección IP y, a continuación, haga clic en Agregar.

El nombre de host o la dirección aparecen en la lista destinos de interrupción .

Repetiremos los pasos 5 a 7 para agregar las comunidades y los destinos de captura que deseé.

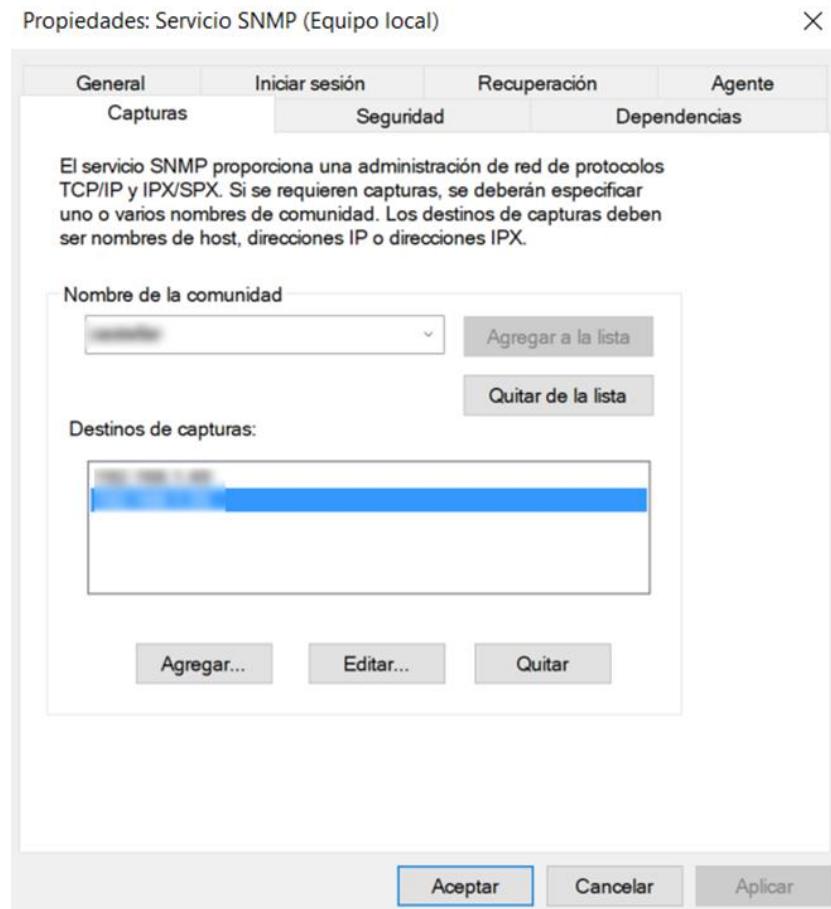


Ilustración 70: Configuración Traps SNMP Windows

Por último, definiremos la seguridad de la comunidad. Para ello, nos dirigimos a la pestaña seguridad.

En nombres de comunidad aceptados, hacemos clic en Agregar.

Para especificar cómo procesa el host las solicitudes SNMP desde la comunidad seleccionada, hacemos clic en el nivel de permisos que desee en el cuadro derechos de la comunidad.

En el cuadro nombre de comunidad, escribimos el nombre de comunidad con distinción de mayúsculas y minúsculas y, a continuación, hacemos clic en Agregar.

Especificamos si se va a aceptar paquetes SNMP desde un host. Para ello, realice una de las siguientes acciones:

- Para aceptar solicitudes SNMP desde cualquier host de la red, independientemente de la identidad, haremos clic en Aceptar paquetes SNMP de cualquier host.
- Para limitar la aceptación de paquetes SNMP, haremos clic en Aceptar paquetes SNMP de estos hosts, en Agregar y, a continuación, escribiremos el nombre de host, la dirección IP o la dirección IPX apropiados en el cuadro nombre de host, dirección IP o IPX .

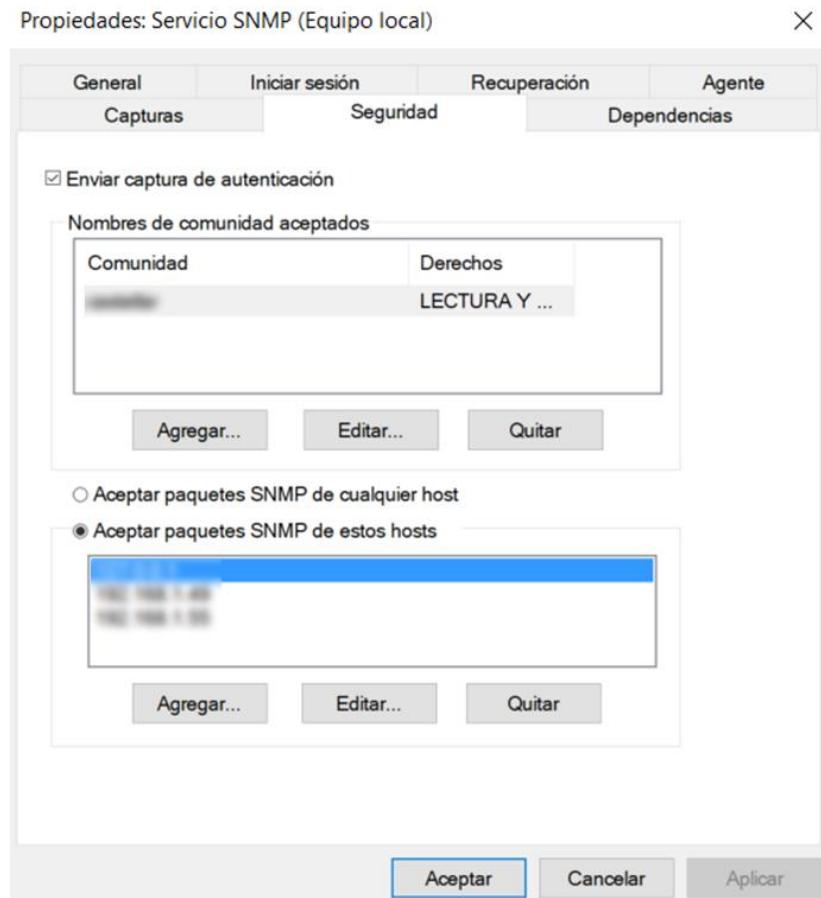


Ilustración 71: Configuración Seguridad SNMP Windows

Una vez aplicados todos los cambios, reiniciaremos el servicio y ya tendremos activado nuestro agente SNMP en Windows.

El siguiente paso es configurar el sistema de eventos por el cual el agente debe notificar al NMS dado un suceso determinado.

Para ello, haremos uso de la utilidad de Windows evntwin, de su acrónimo en inglés Windows Event to SNMP Trap translator.

Nos dirigimos a la barra de Windows y buscamos el launcher Ejecutar. Luego, escribimos evntwin.exe y lo lanzamos.

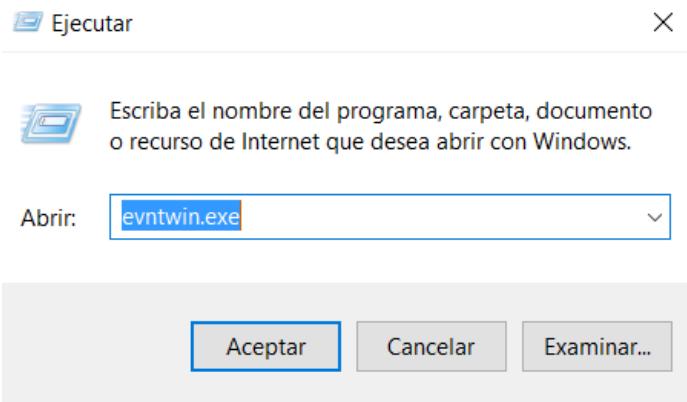


Ilustración 72: EvntWin Windows

Esto lanzará el traductor de eventos a Traps. Ahora añadiremos que eventos nos gustaría que generaran Traps SNMP para mandarlo a nuestro EMS. Seleccionamos la opción Personalizada y a continuación, Edición.

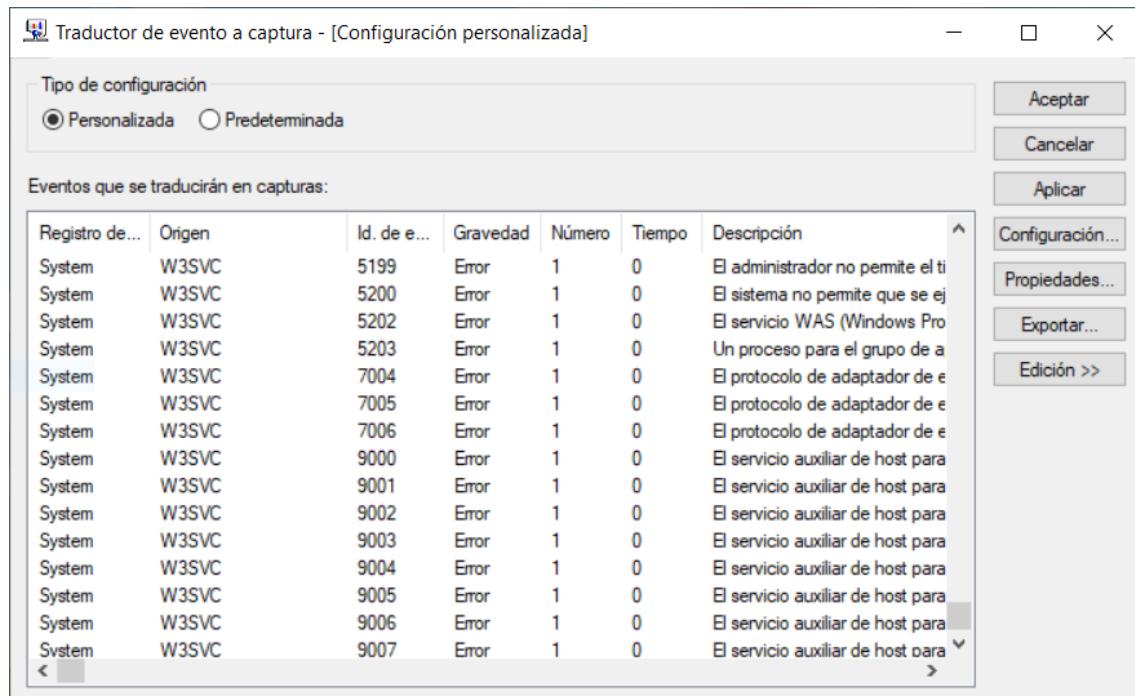


Ilustración 73: Configuración EvntWin Windows 1

Podemos observar que en la lista de eventos aparecen casi todos los eventos que el sistema es capaz de generar. Buscamos específicamente los eventos que queramos monitorizar y clickamos en Agregar.

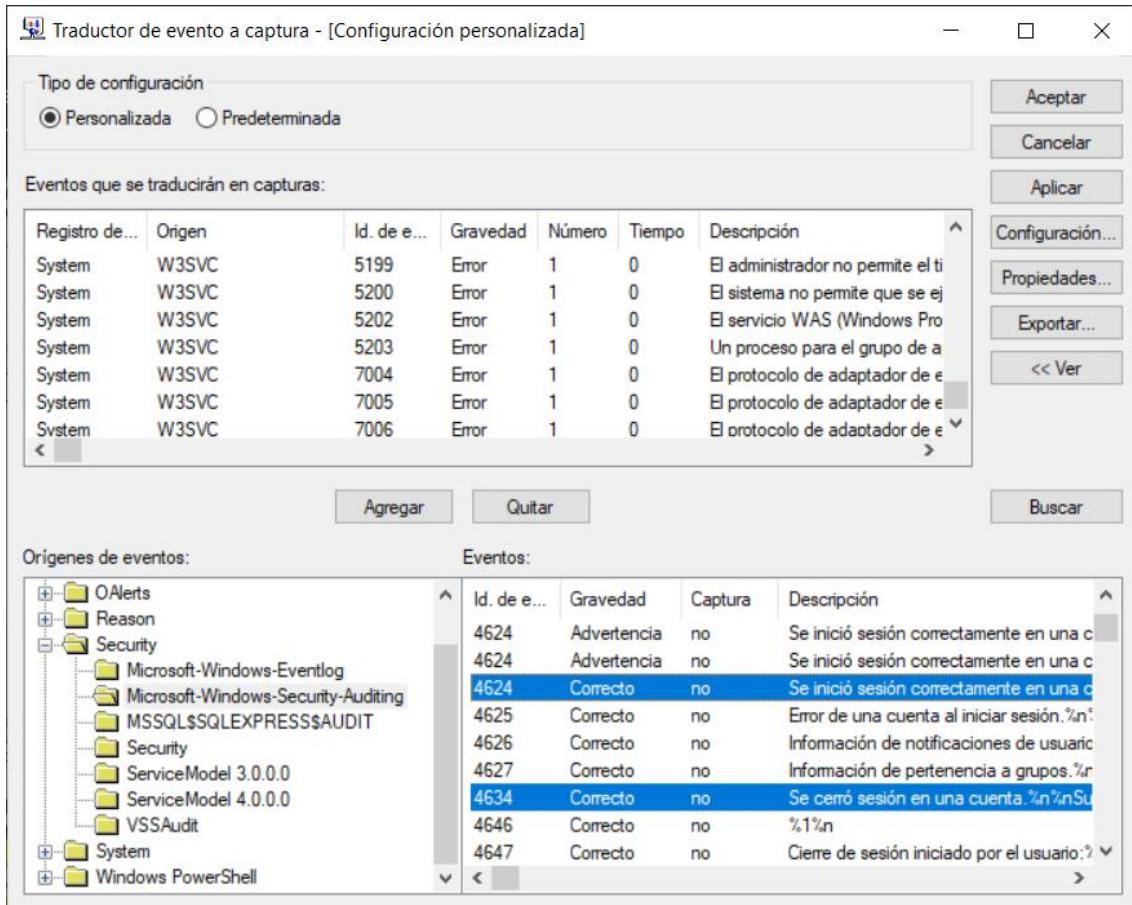


Ilustración 74: Configuración EvntWin Windows 2

Tras esto, aplicamos los cambios y reiniciaremos el servicio SNMP. Con esto tendríamos configurado nuestro Agente SNMP en los dispositivos Windows.

En el repositorio del proyecto he dejado con un ejecutable Windows .bat que importa automáticamente todos los eventos que hemos añadido para la consecución de nuestro proyecto.

```
https://github.com/pablogrsc/MONITORIZACION-DOMESTICA
```

20.1.8 Configuración entorno ISP

De la comprobación de la calidad de servicio que ofrece nuestro ISP en cada momento se encargará uno de nuestros scripts en Python. Dicho script posee una dependencia la cuál debemos de instalar antes de su ejecución.

Como hemos visto anteriormente, esta herramienta es la denominada speedtest-cli.

Para su instalación:

```
sudo apt install speedtest-cli
```

Como la herramienta se ha escrito en Python, también vamos a poder instalarla a través de pip de manera sencilla. Si ya tenemos pip instalado en nuestro equipo, solo tendremos que escribir lo siguiente en la terminal:

```
sudo pip install speedtest-cli
```

Independientemente de la opción escogida ya tendríamos instalado todo lo necesario para la ejecución de nuestro script y consecuentemente una monitorización satisfactoria de nuestro ISP.

20.1.9 Configuración de los demonios

Llegados a este punto ya tendríamos configurados todos los requisitos y dependencias necesarios para la ejecución de nuestros scripts. Los encargados de recolectar y almacenar toda la información de cada dispositivo haciendo uso de los protocolos de red previamente mencionados.

Seguidamente debemos de programar sus ejecuciones de manera periódica para así estar continuamente recabando información y poder visualizar el estado de cada equipo en tiempo real.

Para ello nos ayudaremos del administrador regular de procesos en segundo plano de Unix, "cron". Es el encargado de ejecutar procesos o guiones a intervalos regulares. Los procesos que deben ejecutarse y la hora a la que deben hacerlo se especifican en el archivo crontab.

Para configurar este demonio, vamos al shell de Unix y escribimos:

```
crontab -e
```

Establecemos las siguientes directrices que se encargarán de ejecutar los scripts basados en los protocolos ICMP y SNMP cada 5 minutos, y de igual forma, el script encargado de monitorizar el ISP cada 30 minutos.

```
/tmp/crontab.xzno9i/crontab

# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless
# redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and
# cron(8)
#
# m h dom mon dow   command
*/5 * * * * ..../ICMP/cron_vigilante.sh
*/30 * * * * ..../ISPMonitor/cron_isp.sh
```

```
*/5 * * * * ..../SNMP/cron_snmp.sh
```

20.1.10 Configuración config.py

Para facilitar la tarea de configuración de los scripts Python ejecutados en nuestro NMS se ha creado un fichero de configuración en “..../SNMP/config.py” dónde se unifican todos los parámetros a modificar para que estos programas funcionen en nuestro entorno.

Los parámetros a modificar son:

- Comunidad
- Direcciones IP de los dispositivos Linux a monitorizar.
- Direcciones IP de los dispositivos Windows a monitorizar.

```
..../SNMP/config.py
community = '<<community>>'
linux_target_addresses = ['<<ip>>']
windows_target_addresses = ['<<ip>>']
```

Análogamente se ha creado otro fichero de configuración en “..../ISP/config.py” donde se modificarán los parámetros necesarios para el correcto funcionamiento de los scripts encargados de medir la calidad de servicio de nuestro proveedor de Internet.

Los parámetros a modificar son:

- Velocidad de descarga contratada en MB/s
- Velocidad de carga contratada en MB/s
- Proveedor de servicios

```
..../ISP/config.py
# Internet Speeds
down_speed = 100.0
up_speed = 100.0

provider = '@<<provider>>'
```

20.1.11 Configuración interfaz gráfica

Empezaremos instalando Grafana en nuestro EMS. Para ello, en una primera instancia añadiremos la fuente donde descargar los paquetes oficiales.

```
wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key
add -
echo "deb https://packages.grafana.com/oss/deb stable main" |
sudo tee -a /etc/apt/sources.list.d/grafana.list
```

Una vez hecho esto, procederemos a la propia instalación de la herramienta.

```
sudo apt-get install -y grafana
```

A continuación, debemos hacer que Grafana se ejecute una vez se active nuestra Raspberry.

```
sudo /bin/systemctl enable grafana-server
sudo /bin/systemctl start grafana-server
```

Por último vamos a la siguiente dirección por el puerto 3000 dónde encontraremos el login inicial de Grafana.

```
http://<<ip>>:3000
```

En esta primera ejecución debemos de acceder con las credenciales por defecto, usuario: admin y contraseña: admin. Serán cambiadas tras el inicio de sesión.

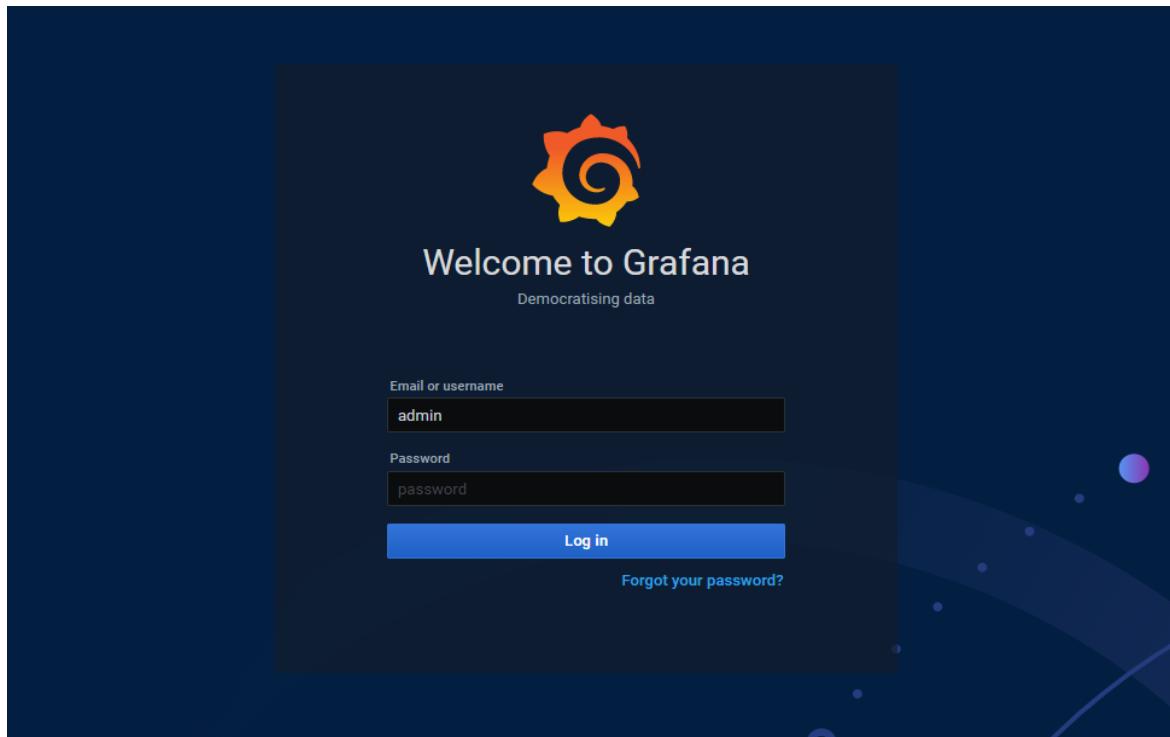


Ilustración 75: Página inicio Grafana

Llegados a este punto habríamos finalizado la instalación y ejecución de la herramienta.

En segundo lugar, debemos añadir la conexión con nuestra base de datos para tener una fuente de información a partir de la cual crear nuestros gráficos.

De esta forma, navegamos hasta Configuration, Data Source.

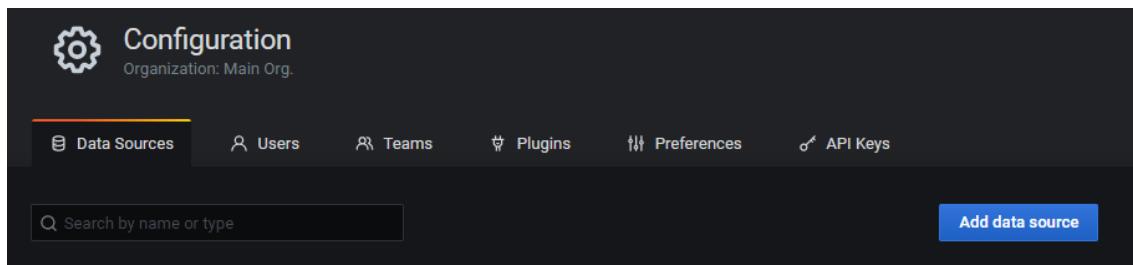


Ilustración 76: Configuración Grafana

En nuestro caso añadimos la siguiente conexión con MariaDB.

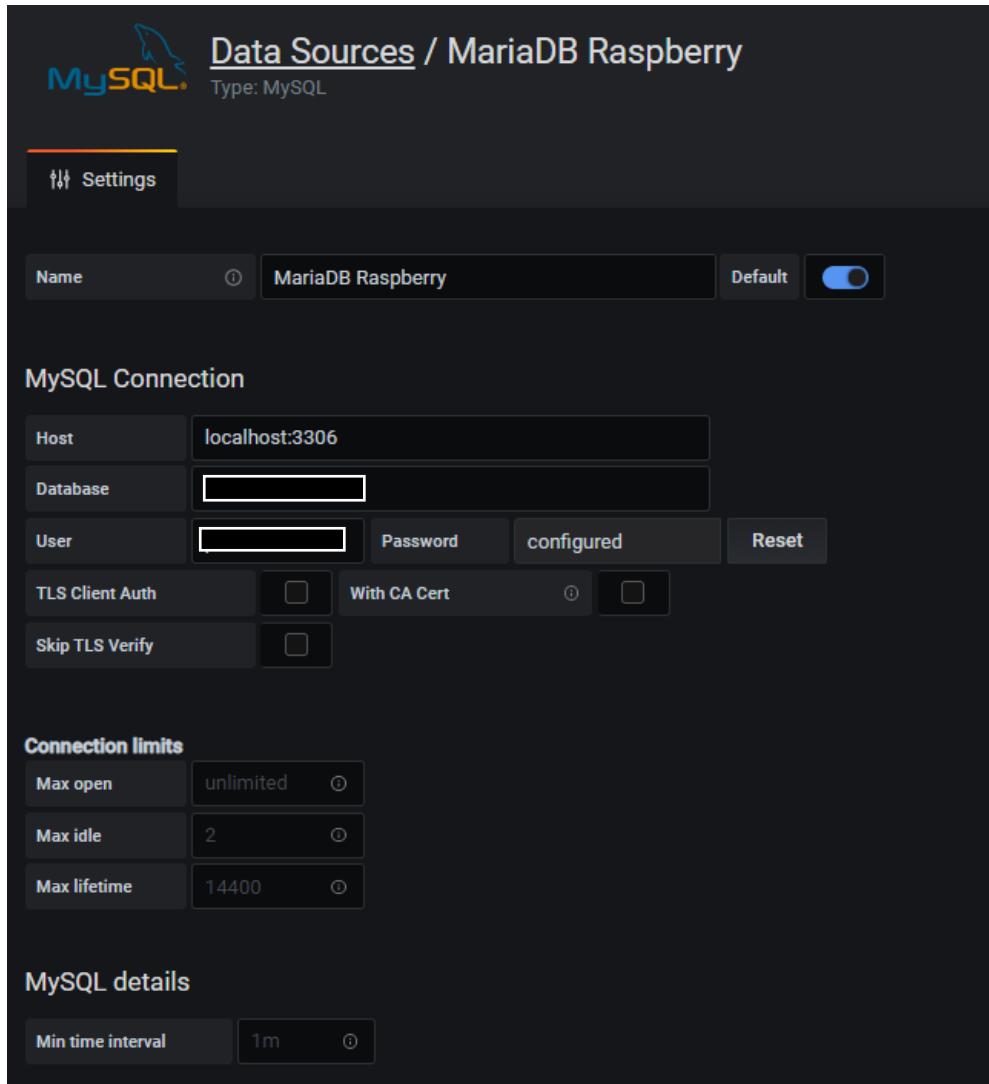


Ilustración 77: Configuración Data Source Grafana

Por último, debemos crear nuestros diferentes “DashBoards”. Grafana ofrece una gran variedad de gráficos para representar nuestros datos, por lo que, esta última parte es muy personalizable.

Para agilizar el proceso y no hacer necesario detallar el proceso de creación de cada uno de los diferentes DashBoard, nos ayudaremos de la funcionalidad de Importación de Grafana.

Los archivos correspondientes a cada DashBoard serán descargables desde el repositorio del proyecto:

<https://github.com/pablogrsc/MONITORIZACION-DOMESTICA>

Para la importación de los archivos, seleccionamos el icono + y a continuación import. Seleccionando cada uno de los json descargados.

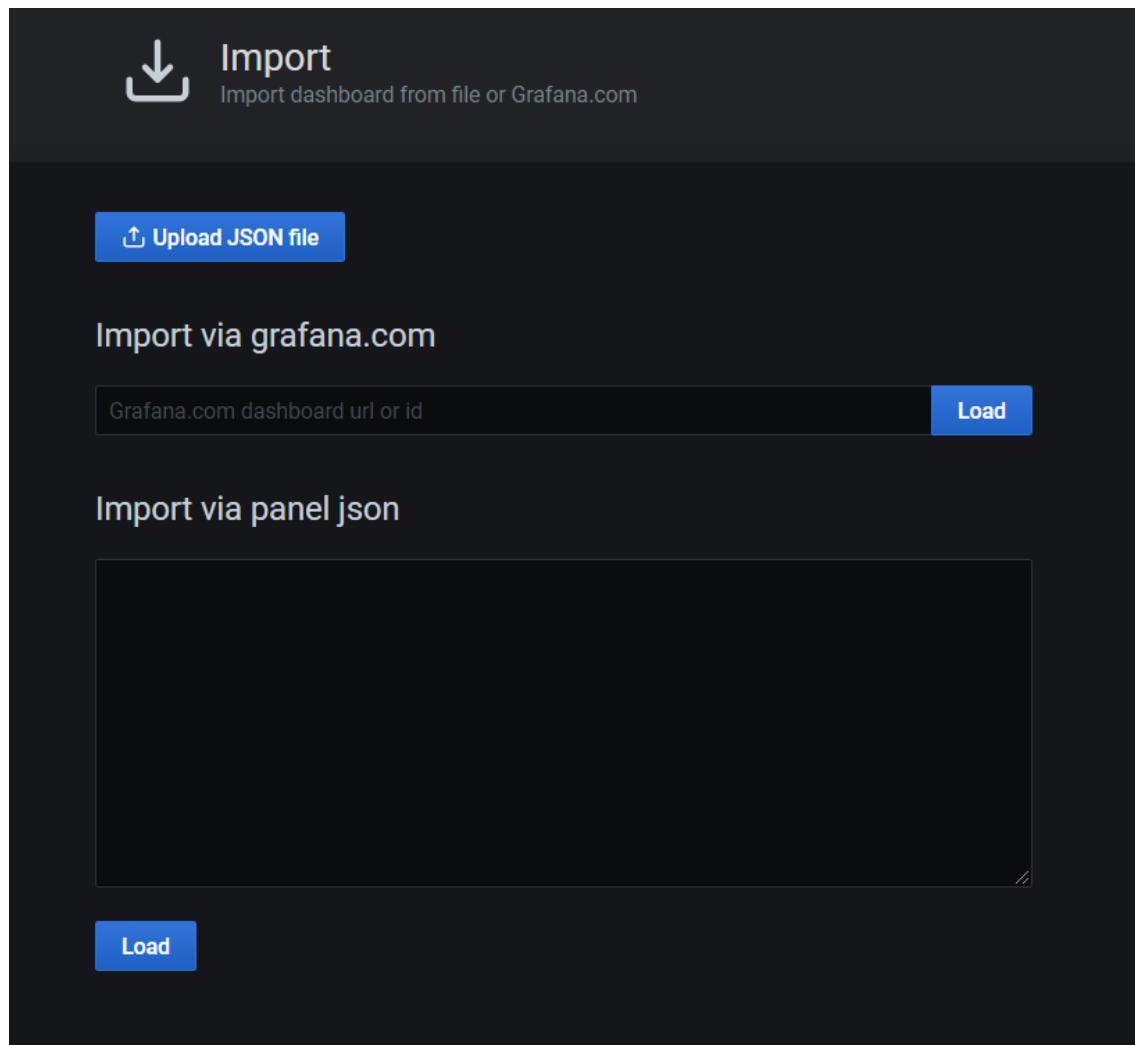


Ilustración 78: Importar Tableros Grafana

En este momento daríamos por finalizada toda la configuración necesaria para que nuestro proyecto funcione correctamente.

21 LIBRERÍAS Y CÓDIGOS RELEVANTES DEL SISTEMA

En este apartado especificaremos cada uno de los códigos relevantes que se encargan de recolectar la información de la red. Los dividiremos en tres apartados según tecnología.

21.1 ICMP

21.1.1 Vigilante.py

```
1.#!/usr/bin/env python3
2.import time
3.import subprocess
4.import re
5.import json
```

```
6. import os
7. import requests
8. import socket
9. import math
10. import sys
11. import mysql.connector
12. from multiprocessing import Process, Manager
13.
14.
15. class Vigilante():
16.     def __init__(self):
17.         self.dir = os.path.dirname(os.path.abspath(__file__))
18.         self.filename = os.path.join(self.dir, 'vigilante.json') #ruta del archivo json donde vuelca los datos
19.         self.read_keys()
20.         self.local_ip = self.get_local_ip() #coge ip privada de la raspberry
21.         self.manager = Manager() #Manager() crea procesos (hilos) concurrentemente
22.
23.         if os.path.exists(self.filename):
24.             self.read()
25.         else:
26.             self.ips = []
27.             self.macs = {}
28.             self.save()
29.
30.     def get_mac_address(self, ip):
31.         try:
32.             proc = subprocess.Popen(['arp', '-n', ip], stdout=subprocess.PIPE)
33.             out, err = proc.communicate()
34.             ans = re.search(r'([0-9A-F]{2}[:-])\{5}([0-9A-F]{2})', out.decode(), re.I).group()
35.         except:
36.             ans = 'Unknown'
37.         return ans
38.
39.     def get_local_ip(self):
40.         proc = subprocess.Popen(['hostname', '-I'], stdout=subprocess.PIPE)
41.         out, err = proc.communicate()
42.         return out.decode().split(' ')[0]
43.
44.     def read_keys(self):
45.         filename = os.path.join('../Monitoring', 'keys.json')
46.         f = open(filename, 'r')
47.         content = f.read()
48.         f.close()
49.         data = json.loads(content)
50.         self.token = data['token']
51.         self.channel_id = data['channel_id']
52.
53.     def send_warning(self, message):
```

```
54.         url = 'https://api.telegram.org/bot{0}/sendMessage'.format(self.token)
55.         data = {'chat_id': self.channel_id, 'text': message}
56.         r = requests.post(url, data)
57.
58.     def insert_db(self, values):
59.         cn = mysql.connector.connect(host="127.0.0.1",
60.             database="xx", user="xx", password="xx",
61.             auth_plugin="mysql_native_password")
62.         sql_executer = cn.cursor()
63.
64.         aql1 = "INSERT INTO hosts (hostname, ip, mac)
65.             VALUES (""
66.             aql2 = ""
67.             for i in range (len(values)):
68.                 values[i] = "\"" + str(values[i]) + "\""
69.                 if (i!=2):
70.                     values[i] += ","
71.                 aql2 = aql2 + values[i]
72.             aql = aql1 + aql2 + ");"
73.
74.         print (aql)
75.         sql_executer.execute(aql)
76.         cn.commit()
77.
78.     def read(self):
79.         f = open(self.filename, 'r')
80.         content = f.read()
81.         f.close()
82.         data = json.loads(content)
83.         self.macs = data['macs']
84.         self.ips = data['ips']
85.
86.     def save(self):
87.         f = open(self.filename, 'w')
88.         data = {'macs': self.macs, 'ips': self.ips}
89.         f.write(json.dumps(data))
90.
91.     def exists(self, item):
92.         if item.find(':') > -1:
93.             return mac in self.macs
94.         else:
95.             return ip in self.ips
96.
97.     def add(self, item):
98.         if type(item) == dict:
99.             self.macs.append(item)
100.            else:
101.                self.ips.append(item)
102.
103.    def check_host(self, ip):
104.        ret_code = subprocess.call(['ping', '-c1', '-W20', ip],
105.                                stdout=subprocess.PIPE)
```

```
103.             if ret_code == 0:
104.                 self.data[ip] = self.get_mac_address(ip)
105.
106.         def vigila(self):
107.             self.data = self.manager.dict() #self.data es
108.                 un diccionario compartido entre todos los procesos que
109.                 cree manager.
110.                 jobs = []
111.                 range_ip = '.'.join(self.local_ip.split('.')[
112.                     :-1]) #range_ip = 192.168.1
113.                     for i in range(1, 254): #254
114.                         ip = '{0}.{1}'.format(range_ip,
115.                             i) #ip = 192.168.1.1 - 192.168.1.255
116.                         if self.local_ip != ip:
117.                             job = Process(target=self.check_host,
118.                               args=(ip,))
119.                             jobs.append(job)
120.                             job.start()
121.                         for job in jobs:
122.                             job.join()
123.                         for ip in self.data.keys():
124.
125.                             try:
126.                                 self.hostname = socket.getfqdn(ip)
127.                             except (socket.error, socket.herror,
128.                                 socket.gaierror) as e:
129.                                 self.hostname = "Unknown"
130.
131.                             if (self.hostname == ip):
132.                                 self.hostname = "Unknown"
133.
134.                             if self.data[ip] == 'Unknown':
135.                                 if ip not in self.ips:
136.                                     self.ips.append(ip)
137.                                     message = "New device called {0}
138.                                         with IP {1} and MAC {2}".format(self.hostname, ip,
139.                                           str(self.data[ip]))
140.                                         #message = "New device with IP
141.                                         {0} and MAC {1}".format(ip, self.data[ip])
142.                                         self.send_warning(message)
143.                                         values = [self.hostname, ip, str(se
144.                                           lf.data[ip])]
145.                                         self.insert_db(values)
146.                                         else:
147.                                             if self.data[ip] not in
148.                                               self.macs.keys(): #Si es una
149.                                                   self.macs[self.data[ip]] = {'last
150.                                                     viewed': math.trunc(time.time()), 'in': True }
151.                                                   message = "New device called {0}
152.                                                       with IP {1} and MAC {2}".format(self.hostname, ip,
153.                                                         str(self.data[ip]))
154.                                                       #message = "New device with IP
155.                                                       {0} and MAC {1}".format(ip, self.data[ip])
156.                                                       self.send_warning(message)
157.                                                       values = [self.hostname, ip, str(se
158.                                                         lf.data[ip])]
```

```
143.                     self.insert_db(values)
144.                 else:
145.                     if self.data[ip] in
146.                         if
147.                             self.macs[self.data[ip]]['in'] is False:
148.                                 message = self.hostname +
149.                                     '[' + str(self.data[ip]) + "] is back"
150.                                     #message = "MAC
151.                                     " + str(self.data[ip]) + " is back"
152.                                     self.send_warning(message
153. )
154.                     self.macs[self.data[ip]] = {'last
155. _viewed': math.trunc(time.time()), 'in': True}
156.                 for mac in self.macs.keys():
157.                     #if
158.                     self.macs[mac]['last_viewed'] - time.time() > 30 * 60 and
159.                     self.macs[mac]['in'] is True:
160.                         if
161.                             math.trunc(time.time()) - self.macs[mac]['last_viewed'] >
162.                                 30 * 60 and self.macs[mac]['in'] is True:
163.                                     self.macs[mac]['in'] = False
164.                                     message = self.hostname + '[' + str(s
165. self.data[ip]) + "] is gone"
166.                                     #message = "MAC " + str(mac) + " is
167.                                     gone"
168.                                     self.send_warning(message)
169.             if __name__ == '__main__':
170.                 start = time.time()
171.                 vigilante = Vigilante()
172.                 vigilante.vigila()
173.                 vigilante.save()
174.                 print('Total time:
175. {0}'.format(time.time() - start))
```

21.2 SNMP

21.2.1 Get-Response

21.2.1.1 config.py

```
1. import os
2.
3. base_path = os.path.join(os.path.dirname(__file__))
4.
5. community = '<<community>>'
6. linux_target_addresses = ['<<ip>>']
7. windows_target_addresses = ['<<ip>>']
8.
```

21.2.1.2 SnmpCPULinux.py

```
1. #!/usr/bin/env python
2.
3. import sys
4. from pysnmp.entity.rfc3413.oneliner import cmdgen
5. import config
6.
7. class SnmpCpuLinux():
8.     def __init__(self):
9.         self.snmp_ois = {
10.             'userCpu': '1.3.6.1.4.1.2021.11.9.0',
11.             'systemCpu': '1.3.6.1.4.1.2021.11.10.0',
12.             'idleCpu': '1.3.6.1.4.1.2021.11.11.0',
13.             'load1': '1.3.6.1.4.1.2021.10.1.3.1',
14.             'load5': '1.3.6.1.4.1.2021.10.1.3.2',
15.             'load15': '1.3.6.1.4.1.2021.10.1.3.3',
16.         }
17.         self.port=161
18.         self.cpu = []
19.
20.     def get_value(self, list, value):
21.         for name, val in list:
22.             if str(name) == value:
23.                 return float(val)
24.         return None
25.
26.     def monitor_cpu(self, host, community):
27.         cmdGen = cmdgen.CommandGenerator()
28.         errorIndication, errorStatus, errorIndex,
29.         varBinds = cmdGen.getCmd(
30.             cmdgen.CommunityData(community),
31.             cmdgen.UdpTransportTarget((host,
32.             self.port)),
33.             self.snmp_ois.get('userCpu'),
34.             self.snmp_ois.get('systemCpu'),
35.             self.snmp_ois.get('idleCpu'),
36.             self.snmp_ois.get('load1'),
37.             self.snmp_ois.get('load5'),
38.             self.snmp_ois.get('load15'),
39.         )
40.         if errorIndication:
41.             print("ERROR1: %s" % errorIndication)
42.             sys.exit(1)
43.         else:
44.             if errorStatus:
45.                 print('ERROR2: %s at %s' % (
46.                     errorStatus.prettyPrint(),
47.                     errorIndex and varBinds[int(error
48.                     Index)-1] or '?'
49.                 ))
50.                 sys.exit(1)
51.             else:
```

```
50.          load1 = self.get_value(varBinds,
51.             self.snmp_ois.get('load1'))*100/4
52.             load5 = self.get_value(varBinds,
53.               self.snmp_ois.get('load5'))*100/4
54.               load15 = self.get_value(varBinds,
55.                 self.snmp_ois.get('load15'))*100/4
56.                 userCpu = self.get_value(varBinds,
57.                   self.snmp_ois.get('userCpu'))
58.                   systemCpu = self.get_value(varBinds,
59.                     self.snmp_ois.get('systemCpu'))
60.                     idleCpu = self.get_value(varBinds,
61.                       self.snmp_ois.get('idleCpu'))
62.                       print ('Cpu Load over the last
63.   minutes: 1 minute last: {0}% - 5 minutes last: {1}% - 15
64.   minutes last: {2}%'.format(
65.     load1,
66.     load5,
67.     load15,
68.   ));
69.   print('Cpu Usage over last minute;
70. User Usage: {0:.2}% - System Usage: {1:.2}% - Idle Time:
71. {2:.2f}%'.format(
72.   userCpu,
73.   systemCpu,
74.   idleCpu,
75. ));
76.   self.cpu = [load1, load5, load15, userCpu,
77.   systemCpu, idleCpu]
78.   return self.cpu
79.
80.   #On a multi-processor system, the 'ssCpuRaw*'#
81.   #counters are cumulative over all CPUs, so their#
82.   #sum will typically be N*100 (for N processors). -
83.   > x ESO EL /4
84.   #
85.   #
86.   #if __name__ == '__main__':
87.   #    monitor = SnmpCpuLinux()
88.   #    monitor.monitor_cpu(config.target_address,
89.   config.community)
```

21.2.1.3 SnmpCPUWindows.py

```
1.#!/usr/bin/env python
2.
3.import sys
4.import config
5.from pysnmp.entity.rfc3413.oneliner import cmdgen
6.import mysql.connector
7.
8.class SnmpCpu():
9.    def __init__(self):
```

```
10.          self.snmp_ois = {'cpu': '1.3.6.1.2.1.25.3.3.1
11.          .2.'}
12.          self.port=161
13.          self.cpu = 0
14.      def get_value(self, list, value):
15.          for name, val in list:
16.              if str(name) == value:
17.                  return float(val)
18.          return None
19.
20.      def count_cores(self, list, value):
21.          for name, val in list:
22.              if (format(val) != None):
23.                  return True
24.              else:
25.                  return False
26.
27.      def monitor_cpu(self, host, community):
28.          i=8
29.          cmdGen = cmdgen.CommandGenerator()
30.          errorIndication, errorStatus, errorIndex,
31.          varBinds = cmdGen.getCmd(
32.              cmdgen.CommunityData(community),
33.              cmdgen.UdpTransportTarget((host,
34.                  self.port)),
35.              self.snmp_ois.get('cpu') + str(i),
36.              )
37.          while(self.count_cores(varBinds,
38.              self.snmp_ois.get('cpu') + str(i))):
39.              errorIndication, errorStatus, errorIndex,
40.              varBinds = cmdGen.getCmd(
41.                  cmdgen.CommunityData(community),
42.                  cmdgen.UdpTransportTarget((host,
43.                      self.port)),
44.                  self.snmp_ois.get('cpu') + str(i),
45.                  )
46.              if errorIndication:
47.                  #print("ERROR1:
48.                  %s" % errorIndication)
49.                  break
50.              else:
51.                  if errorStatus:
52.                      #print('ERROR2: %s at %s' % (
53.                          #    errorStatus.prettyPrint(),
54.                          #    errorIndex and varBinds[int(
55.                              errorIndex)-1] or '?'
56.                          #
57.                          #
58.                          break
59.                  else:
60.                      cpu_usage = self.get_value(varBin
61.                      ds, self.snmp_ois.get('cpu')+str(i))
62.                      self.cpu += cpu_usage
```

```
55.                     print("The percentage of work
   time of Core{0} in the last minute was {1}%".format(i-7,
   cpu_usage))
56.                     i=i+1
57.             return self.cpu/(i-8)
```

21.2.1.4 SnmpRamLinux.py

```
1. #!/usr/bin/env python
2.
3. import sys
4. import config
5. from pysnmp.entity.rfc3413.oneliner import cmdgen
6.
7. class SnmpRamLinux():
8.     def __init__(self):
9.         self.snmp_ois = {
10.             'total': '1.3.6.1.4.1.2021.4.5.0',
11.             'free': '1.3.6.1.4.1.2021.4.6.0',
12.             'buffer': '1.3.6.1.4.1.2021.4.14.0',
13.             'cache': '1.3.6.1.4.1.2021.4.15.0',
14.         }
15.         self.port=161
16.         self.ram = []
17.
18.     def get_value(self, list, value):
19.         for name, val in list:
20.             if str(name) == value:
21.                 return float(val)
22.         return None
23.
24.     def monitor_ram(self, host, community):
25.         cmdGen = cmdgen.CommandGenerator()
26.         errorIndication, errorStatus, errorIndex,
27.         varBinds = cmdGen.getCmd(
28.             cmdgen.CommunityData(community),
29.             cmdgen.UdpTransportTarget((host,
30.                                         self.port)),
31.             self.snmp_ois.get('total'),
32.             self.snmp_ois.get('free'),
33.             self.snmp_ois.get('buffer'),
34.             self.snmp_ois.get('cache')
35.         )
36.         if errorIndication:
37.             print("ERROR1: %s" % errorIndication)
38.             sys.exit(1)
39.         else:
40.             if errorStatus:
41.                 print('ERROR2: %s at %s' %
42.                     errorStatus.prettyPrint(),
43.                     errorIndex and varBinds[int(error
44.             Index)-1] or '?'
45.                 )
46.             sys.exit(1)
```

```
45.             else:
46.                 total_ram = float(self.get_value(varB
    inds, self.snmp_ois.get('total')) / pow(1024,2) * 1.074)
47.                 free_ram = float(self.get_value(varBi
    nds, self.snmp_ois.get('free')) / pow(1024,2) * 1.074)
48.                 cache_ram = float(self.get_value(varB
    inds, self.snmp_ois.get('cache')) / pow(1024,2) * 1.074)
49.                 used_ram = total_ram - free_ram - cac
    he_ram
50.                 percent_ram = (used_ram / total_ram ) *
        100
51.
52.                 print('Space Ram of the Host: Total:
    {0:.2f}GB - Free: {1:.2f}GB - Cache: {2:.2f}GB - Percent
    of Used Ram: {3:.2f}%'.format(
53.                     total_ram,
54.                     free_ram,
55.                     cache_ram,
56.                     percent_ram
57.                     )
58.                 );
59.                 self.ram = [total_ram, free_ram, cache_ram,
    used_ram, percent_ram]
60.             return self.ram
61.
62.     #if __name__ == '__main__':
63.     #     monitor = SnmpRamLinux()
64.     #     monitor.monitor_ram(config.target_address,
    config.community)
```

21.2.1.5 SnmpRamWindows.py

```
1.#!/usr/bin/env python
2.
3. import config
4. import sys
5. from pysnmp.entity.rfc3413.oneliner import cmdgen
6.
7. class SnmpRam():
8.     def __init__(self):
9.         self.snmp_ois = {
10.             'ram': '1.3.6.1.2.1.25.2.2.0',
11.             'ram_used' : '1.3.6.1.2.1.25.2.3.1.6.',
12.             'allocation' : '1.3.6.1.2.1.25.2.3.1.4.'
13.         }
14.         self.port=161
15.         self.ram = 0
16.         self.cont=0
17.
18.     def get_value(self, list, value):
19.         for name, val in list:
20.             if str(name) == value:
21.                 return float(val)
22.         return None
23.
```

```
24.         def is_partition(self, list, value):
25.             for name, val in list:
26.                 if (format(val) is not ''):
27.                     return True
28.                 else:
29.                     return False
30.
31.         def count_partition(self, host, community):
32.             i=1
33.             cmdGen = cmdgen.CommandGenerator()
34.             errorIndication, errorStatus, errorIndex,
35.             varBinds = cmdGen.getCmd(
36.                 cmdgen.CommunityData(community),
37.                 cmdgen.UdpTransportTarget((host,
38.                     self.port)),
39.                 self.snmp_ois.get('ram_used')+str(i),
40.                 self.snmp_ois.get('allocation') + str(i)
41.             )
42.             while (self.is_partition(varBinds,
43.                 self.snmp_ois.get('ram_used') + str(i))):
44.                 errorIndication, errorStatus, errorIndex,
45.                 varBinds = cmdGen.getCmd(
46.                     cmdgen.CommunityData(community),
47.                     cmdgen.UdpTransportTarget((host,
48.                         self.port)),
49.                     self.snmp_ois.get('ram_used')+str(i),
50.                     self.snmp_ois.get('allocation') + str
51.                     (i)
52.                 )
53.                 i=i+1
54.                 self.cont=self.cont+1
55.
56.         def monitor_ram(self, host, community):
57.             self.count_partition(host, community)
58.             cmdGen = cmdgen.CommandGenerator()
59.             errorIndication, errorStatus, errorIndex,
60.             varBinds = cmdGen.getCmd(
61.                 cmdgen.CommunityData(community),
62.                 cmdgen.UdpTransportTarget((host,
63.                     self.port)),
64.                 self.snmp_ois.get('ram'),
65.                 self.snmp_ois.get('ram_used')+str(sel
66.                     f.cont-1),
67.                 self.snmp_ois.get('allocation')+str(s
68.                     elf.cont-1)
69.             )
70.             if errorIndication:
71.                 print("ERROR1: %s" % errorIndication)
72.             else:
73.                 if errorStatus:
74.                     print('ERROR2: %s at %s' %
75.                         errorStatus.prettyPrint(),
76.                         errorIndex and varBinds[int(error
77.                             Index)-1] or '?'
78.                     )
79.             )
```

```
69.         else:
70.             total_ram = self.get_value(varBinds,
    self.snmp_ois.get('ram'))
71.             used_ram = self.get_value(varBinds,
    self.snmp_ois.get('ram_used')+str(self.cont-
1)) * self.get_value(varBinds,
    self.snmp_ois.get('allocation')+str(self.cont-1))
72.             print("The total RAM of the host is:
{0:.3f}GB".format(total_ram/pow(1024,2)))
73.             print("The used RAM of the host is:
{0:.3f}GB".format(used_ram/pow(1024,3)))
74.             self.ram = [(total_ram/pow(1024,2)),
    (used_ram/pow(1024,3))]
75.             return self.ram
76.
77.     if __name__ == '__main__':
78.         monitor = SnmpRam()
79.         monitor.monitor_ram('192.168.1.50',
config.community)
```

21.2.1.6 SnmpDiskLinux.py

```
1. #!/usr/bin/env python
2.
3. import sys
4. import config
5. from pysnmp.entity.rfc3413.oneliner import cmdgen
6.
7. class SnmpStorageLinux():
8.     def __init__(self):
9.         self.snmp_ois = {
10.             'total': '1.3.6.1.4.1.2021.9.1.6.1',
11.             'free': '1.3.6.1.4.1.2021.9.1.7.1',
12.             'used': '1.3.6.1.4.1.2021.9.1.8.1',
13.             'percentused': '1.3.6.1.4.1.2021.9.1.9.1'
14.         }
15.         self.port=161
16.         self.disk = []
17.
18.     def get_value(self, list, value):
19.         for name, val in list:
20.             if str(name) == value:
21.                 return float(val)
22.         return None
23.
24.     def monitor_storage(self, host, community):
25.         cmdGen = cmdgen.CommandGenerator()
26.         errorIndication, errorStatus, errorIndex,
varBinds = cmdGen.getCmd(
27.             cmdgen.CommunityData(community),
28.             cmdgen.UdpTransportTarget((host,
    self.port)),
29.             self.snmp_ois.get('total'),
30.             self.snmp_ois.get('free'),
```

```
31.                     self.snmp_ois.get('used'),
32.                     self.snmp_ois.get('percentused')
33.                 )
34.             if errorIndication:
35.                 print("ERROR1: %s" % errorIndication)
36.                 sys.exit(1)
37.             else:
38.                 if errorStatus:
39.                     print('ERROR2: %s at %s' % (
40.                         errorStatus.prettyPrint(),
41.                         errorIndex and varBinds[int(error
   Index)-1] or '?'
42.                     )
43.                 )
44.                 sys.exit(1)
45.             else:
46.                 disk_total = float(self.getValue(var
   Binds, self.snmp_ois.get('total'))) / pow(1024,2) * 1.074
47.                 disk_free = float(self.getValue(varB
   inds, self.snmp_ois.get('free'))) / pow(1024,2) * 1.074
48.                 disk_used = float(self.getValue(varB
   inds, self.snmp_ois.get('used'))) / pow(1024,2) * 1.074
49.                 disk_percentused = float(self.getValue(
   varBinds, self.snmp_ois.get('percentused')))
50.
51.                 print('Space for Linux System on the
   host; Total: {0:.2f}GB - Free: {1:.2f}GB - Used: {2:.2f}GB
   - Percent of Used Disk: {3:.2f}%'.format(
52.                     disk_total,
53.                     disk_free,
54.                     disk_used,
55.                     disk_percentused
56.                 )
57. );
58.                 self.disk = [disk_total, disk_free,
   disk_used, disk_percentused]
59.                 return self.disk;
60.
61.
62.     if __name__ == '__main__':
63.         monitor = SnmpStorageLinux()
64.         monitor.monitor_storage(config.target_address,
   config.community)
```

21.2.1.7 SnmpDiskWindows.py

```
1.#!/usr/bin/env python
2.
3.import sys
4.import config
5.from pysnmp.entity.rfc3413.oneliner import cmdgen
6.
7.class SnmpStorage():
8.    def __init__(self):
9.        self.snmp_ois = {
```

```
10.                 'total': '1.3.6.1.2.1.25.2.3.1.5.',
11.                 'used': '1.3.6.1.2.1.25.2.3.1.6.',
12.                 'allocation': '1.3.6.1.2.1.25.2.3.1.4.',
13.             }
14.             self.port=161
15.             self.total=0
16.             self.used=0
17.
18.     def get_value(self, list, value):
19.         for name, val in list:
20.             if str(name) == value:
21.                 return float(val)
22.         return None
23.
24.     def count_partition(self, list, value):
25.         for name, val in list:
26.             if (format(val) != None):
27.                 return True
28.             else:
29.                 return False
30.
31.     def monitor_storage(self, host, community):
32.         i=1
33.         cmdGen = cmdgen.CommandGenerator()
34.         errorIndication, errorStatus, errorIndex,
35.         varBinds = cmdGen.getCmd(
36.             cmdgen.CommunityData(community),
37.             cmdgen.UdpTransportTarget((host,
38.             self.port)),
39.             self.snmp_ois.get('total') + str(i),
40.             self.snmp_ois.get('used') + str(i),
41.             self.snmp_ois.get('allocation') + str
42.             (i)
43.         )
44.         while(self.count_partition(varBinds,
45.             self.snmp_ois.get('total') + str(i))):
46.             errorIndication, errorStatus, errorIndex,
47.             varBinds = cmdGen.getCmd(
48.                 cmdgen.CommunityData(community),
49.                 cmdgen.UdpTransportTarget((host,
50.                 self.port)),
51.                 self.snmp_ois.get('total') + str(i),
52.                 self.snmp_ois.get('used') + str(i),
53.                 self.snmp_ois.get('allocation') + str
54.                 (i)
55.             )
56.             if errorIndication:
57.                 #print("ERROR1:
58.                 %s" % errorIndication)
59.                 break
60.             else:
61.                 if errorStatus:
62.                     #print('ERROR2: %s at %s' % (
63.                         #    errorStatus.prettyPrint(),
64.                         #    errorIndex and varBinds[int(
65.                             errorIndex)-1] or '?'
```

```
57.             #      )
58.             #)
59.             break
60.         else:
61.             self.total = self.total + sel
62.             f.get_value(varBinds,
63.                         self.snmp_ois.get('total')+str(i)) * self.get_value(varBind
64.             ds, self.snmp_ois.get('allocation')+str(i))
65.             self.used = self.used + self.
66.             get_value(varBinds,
67.                         self.snmp_ois.get('used')+str(i)) * self.get_value(varBind
68.             s, self.snmp_ois.get('allocation')+str(i))
69.             i=i+1
70.             free = self.total - self.used
71.             print("Space of the host disk is:")
72.             print("\tTotal:
73.                 {0:.4f}GB".format(self.total / pow(1024,3)))
74.             print("\tUsed:
75.                 {0:.4f}GB".format(self.used / pow(1024,3)))
76.             print("\tFree:
77.                 {0:.4f}GB".format(free / pow(1024,3)))
78.             print("\tPercentage of used Disk:
79.                 {0:.2f}%".format(((self.used*100)/self.total)))
80.             disk = [self.total / pow(1024,3),
81.                     self.used / pow(1024,3),
82.                     free / pow(1024,3), ((self.used*100)/self.total)]
83.             return disk
84.
85.     if __name__ == '__main__':
86.         #    monitor = SnmpStorage()
87.         #
```

21.2.1.8 SnmpLinux.py

```
1. import sys
2. import mysql.connector
3. import config
4. import datetime
5. import socket
6. import SnmpCpuLinux
7. import SnmpRamLinux
8. import SnmpStorageLinux
9.
10.
11.     def insert_db(values):
12.         cn = mysql.connector.connect(host="127.0.0.1"
13.             , database="xx", user="xx", password="xx",
14.             auth_plugin="mysql_native_password")
15.         sql_executer = cn.cursor()
16.
17.         aql1 = "INSERT INTO snmp_linux (hour,
18.             hostname, load1, load5, load15, user_cpu, system_cpu,
```

```
    idle_cpu, ram_total, ram_free, ram_buffer, ram_cache,
    disk_total, disk_free, disk_used, disk_percent) VALUES (
16.          aql2 = ""
17.          for i in range (len(values)):
18.              values[i] = "\"" + str(values[i]) + "\""
19.              if (i!=15):
20.                  values[i] += ","
21.                  aql2 = aql2 + values[i]
22.          aql = aql1 + aql2 + ");"
23.          print (aql)
24.          sql_executer.execute(aql)
25.          cn.commit()
26.
27.      def gethostname(ip):
28.          try:
29.              hostname = socket.getfqdn(ip)
30.          except (socket.error, socket.herror,
31.          socket.gaierror) as e:
32.              hostname = "Unknown"
33.
34.          if (hostname == ip):
35.              return "Unknown"
36.          else:
37.              return hostname
38.
39.      if __name__ == '__main__':
40.          for i in
        range (len(config.linux_target_addresses)):
41.              cpu = SnmpCpuLinux.SnmpCpuLinux()
42.              ram = SnmpRamLinux.SnmpRamLinux()
43.              disk = SnmpStorageLinux.SnmpStorageLinux()
44.
45.              cpu_value = cpu.monitor_cpu(config.linux_targ
        et_addresses[i], config.community)
46.              ram_value = ram.monitor_ram(config.linux_targ
        et_addresses[i], config.community)
47.              disk_value = disk.monitor_storage(config.linu
        x_target_addresses[i], config.community)
48.              time = datetime.datetime.now().strftime("%Y-
        %m-%d %H:%M:%S")
49.              hostname = gethostname(config.linux_target_ad
        dresses[i])
50.
51.              values = [time, hostname, str(cpu_value[0]),
        str(cpu_value[1]), str(cpu_value[2]), str(cpu_value[3]),
        str(cpu_value[4]), str(cpu_value[5]), str(ram_value[0]),
        str(ram_value[1]), str(ram_value[2]), str(ram_value[3]),
        str(disk_value[0]), str(disk_value[1]),
        str(disk_value[2]), str(disk_value[3])]
52.              insert_db(values)
```

21.2.1.9 SnmpWindows.py

```
1. import sys
```

```
2. import mysql.connector
3. import config
4. import datetime
5. import socket
6. import SnmpCpu
7. import SnmpRam
8. import SnmpStorage
9.
10.
11.     def insert_db(values):
12.         cn = mysql.connector.connect(host="127.0.0.1",
13.             database="xx", user="xx", password="xx",
14.             auth_plugin="mysql_native_password")
15.             sql_executer = cn.cursor()
16.
17.             aql1 = "INSERT INTO snmp_windows (hour,
18.                 hostname, cpu_usage, ram_total, ram_used, disk_total,
19.                 disk_used, disk_free, disk_percent) VALUES (""
20.                 aql2 = ""
21.                 for i in range (len(values)):
22.                     values[i] = "\"" + str(values[i]) + "\""
23.                     if (i!=8):
24.                         values[i] += ","
25.                         aql2 = aql2 + values[i]
26.                         aql = aql1 + aql2 + ");"
27.                         print (aql)
28.                         sql_executer.execute(aql)
29.                         cn.commit()
30.
31.     def gethostname(ip):
32.         try:
33.             hostname = socket.getfqdn(ip)
34.         except (socket.error, socket.herror,
35.             socket.gaierror) as e:
36.             hostname = "Unknown"
37.
38.         if (hostname == ip):
39.             return "Unknown"
40.         else:
41.             return hostname
42.
43.     if __name__ == '__main__':
44.         for i in
45.             range (len(config.windows_target_addresses)):
46.                 cpu = SnmpCpu.SnmpCpu()
47.                 ram = SnmpRam.SnmpRam()
48.                 disk = SnmpStorage.SnmpStorage()
49.                 ram_value = ram.monitor_ram(config.windows_ta
50.                   rget_addresses[i], config.community)
51.                 disk_value = disk.monitor_storage(config.wind
52.                   ows_target_addresses[i], config.community)
53.                 cpu_value = cpu.monitor_cpu(config.windows_
54.                   rget_addresses[i], config.community)
55.                 time = datetime.datetime.now().strftime("%Y-
56.                   %m-%d %H:%M:%S")
```

```
47.         hostname = gethostname(config.windows_target_
    addresses[i])
48.         print (ram_value[1])
49.
50.         values = [time, hostname, str(cpu_value),
    str(ram_value[0]), str(ram_value[1]), str(disk_value[0]),
    str(disk_value[1]), str(disk_value[2]),
    str(disk_value[3])]
51.
52.         insert_db(values)
```

21.2.2 TRAPS

21.2.2.1 trap_handler.py

```
1. #!/usr/bin/env python
2. import os
3. import sys
4. import json
5. import requests
6. import pprint
7.
8. class TrapHandler():
9.     def __init__(self):
10.         self.read_keys()
11.
12.     def read_trap(self):
13.         with open("/var/log/traps.log") as myFile:
14.             text = myFile.read()
15.             myFile.close()
16.             result = text.split('/////')
17.             self.format_message(result)
18.
19.     def insert_split_character(self):
20.         with open("/var/log/traps.log", 'a+') as
    myFile:
21.             myFile.write('/////')
22.             myFile.close()
23.             self.read_trap()
24.
25.     def read_keys(self):
26.         filename = os.path.join('../Desktop/Monitorin
    g', 'keys.json')
27.         f = open(filename, 'r')
28.         content = f.read()
29.         f.close()
30.         data = json.loads(content)
31.         self.token = data['token']
32.         self.channel_id = data['channel_id']
33.
34.     def send_warning(self, message):
35.         url = 'https://api.telegram.org/bot{0}/sendMe
    ssage'.format(self.token)
36.         data = {'chat_id': self.channel_id, 'text':
    message}
```

```
37.             r = requests.post(url, data)
38.
39.     def format_message(self, result):
40.         line1 = result[-4].split(' ')
41.         message = "Received new trap message: \n"
42.         message += "-----\n"
43.         message += "Date: "
44.         " + line1[0] + ' ' + line1[1] + "\n"
45.         message += "HostName: " + line1[2] + "\n\n"
46.         message += "Trap Info: \n" + str(result[-4])
47.         self.send_warning(message)
48.
49.     if __name__ == '__main__':
50.         handler = TrapHandler()
51.         handler.insert_split_character()
```

21.3 ISP

21.3.1 Config.py

```
1. import os
2.
3. base_path = os.path.join(os.path.dirname(__file__))
4.
5. # Internet Speeds
6. down_speed = 100.0
7. up_speed = 100.0
8.
9. provider = '@provider'
10.
11. telegram_message = """
12.     Actualmente la calidad de la conexión es:
13.         - Velocidad de bajada [↓]: {down:.2f} Mbps
14.         - Velocidad de subida [↑]: {up:.2f} Mbps
15.         - Latencia: {ping:.2f} ms
16.         - Conectado durante: {uptime}
17.         - Ratio de lo contratado: {ratio:.2f}%
18.         - Comentario: {reaction}
19. """
20.
21.
22. messages = dict(
23.     awesome='Increíble',
24.     great='Genial',
25.     fair='Buena',
26.     mediocre='Regular',
27.     bad='Mala',
28.     terrible='Terrible',
29.     illegal='Illegal'
30. )
31.
32. movistar = {
```

```
33.         'down_ratio': 0.4,
34.         'up_ratio': 0.4,
35.         'ping': 80,
36.         'online': 0.995,
37.     }
```

21.3.2 lsp_monitor.py

```
1. #!/usr/bin/env python
2.
3. import json, math, random
4. from datetime import datetime
5. from re import search
6. import os
7. import requests
8. import config
9. import mysql.connector
10.
11.     PING_SERVERS = [
12.         '1.1.1.1',
13.         '1.0.0.1',
14.         '8.8.8.8',
15.         '8.8.4.4',
16.     ]
17.
18.     class ISPMonitor():
19.         def __init__(self):
20.             self.read_keys()
21.
22.
23.         def get_uptime(self):
24.             if self.is_online():
25.                 now = int(datetime.now().strftime('%s'))
26.                 with
27.                     open(config.base_path + '/uptime.log', 'a+') as file:
28.                         file.write('%d\n' % now)
29.                         file.seek(0)
30.                         timestamp = file.readline()
31.                         first_online_timestamp = - now if
32.                         timestamp == '' else int(timestamp)
33.                         uptime = (now - first_online_timestamp) /
34.                             (60 * 60)
35.                         return '%d %s' % (uptime, 'hora' if
36.                           uptime == 1 else 'horas')
37.             else:
38.                 with
39.                     open(config.base_path + '/uptime.log', 'w') as file:
40.                         file.write('')
41.                         raise Exception('There is no Internet
connectivity')
```

```
42.                 ip = random.choice(PING_SERVERS)
43.                 if os.system(f'ping {ip} -c 1 -W
44.                     5') == 0:
45.                         return True
46.                 return False
47.
48.             def run_test(self):
49.                 self.write('Running SpeedTest')
50.                 print(config.base_path + '/results.log')
51.                 os.system('speedtest --json >
52.                   ' + config.base_path + '/results.log')
53.
54.             def parse_results(self):
55.                 self.write('Parsing Results')
56.                 with
57.                     open(config.base_path + '/results.log') as file:
58.                         data = json.loads(file.read())
59.                     return (
60.                         [data['server']['latency'],
61.                          data['download'] / 1048576,
62.                          data['download'] / 1048576 / config.down_
63.                          speed,
64.                          data['upload'] / 1048576,
65.                          data['upload'] / 1048576 / config.up_
66.                          speed,
67.                          data['server']['url'])
68.                     )
69.             def read_keys(self):
70.                 filename = os.path.join('../Desktop/Monitorin
71.                     g', 'keys.json')
72.                 f = open(filename, 'r')
73.                 content = f.read()
74.                 f.close()
75.                 data = json.loads(content)
76.                 self.token = data['token']
77.                 self.channel_id = data['channel_id']
78.
79.             def send_warning(self, text):
80.                 url = 'https://api.telegram.org/bot{0}/sendMe
81.                     ssage'.format(self.token)
82.                 data = {'chat_id': self.channel_id, 'text':
83.                     text}
84.                 r = requests.post(url, data)
85.
86.             def mount_status(self, uptime, results):
87.                 ping, down, down_ratio, up, up_ratio,
88.                 url = results
89.                 reaction=self.get_reaction(down_ratio,
90.                     up_ratio, ping)
91.                 worst_ratio = min(down_ratio, up_ratio)
92.                 text = config.telegram_message.format(
93.                     provider=config.provider,
94.                     down=down,
```

```
88.                 up=up,
89.                 ping=ping,
90.                 ratio=worst_ratio * 100,
91.                 uptime=uptime,
92.                 reaction=reaction,
93.                 url=url,
94.             )
95.             bad_reactions = ['Mala', 'Terrible', 'Illegal']
96.             if (reaction in (bad_reactions)):
97.                 self.send_warning(text)
98.
99.             def get_reaction(self, down_ratio, up_ratio,
100.                             ping):
101.                 if ping > config.movistar['ping'] \
102.                     or down_ratio <= config.movistar['dow
n_ratio'] \
103.                     or up_ratio <= config.movistar['up_ra
tio']:
104.                     return config.movistar['illegal']
105.
106.                     messages = [
107.                         config.messages['terrible'],
108.                         config.messages['bad'],
109.                         config.messages['mediocre'],
110.                         config.messages['fair'],
111.                         config.messages['great'],
112.                         config.messages['awesome'],
113.                     ]
114.                     max_idx = len(messages) - 1
115.                     # convert the worst_ratio within the
range [worst_legal - 1.0] to [0 - number_of_messages]
116.                     worst_ratio = min(down_ratio, up_ratio)
117.                     worst_legal = min(config.movistar['down_ratio
'], config.movistar['up_ratio'])
118.                     best_legal = 1.0
119.                     coefficient = (0 - max_idx) / (worst_legal -
best_legal)
120.                     idx = coefficient * (worst_ratio - worst_lega
l)
121.                     message_idx = min(int(math.ceil(idx)),
max_idx)
122.
123.
124.             def write(self, text, log=False):
125.                 message = '%s - %s' % (self.timestamp(),
text)
126.                 print(message)
127.                 if log:
128.                     with
open(config.base_path + '/error.log', 'a') as file:
129.                         file.write('%s\n' % message)
130.
131.
132.             def timestamp(self):
```

```
133.             return datetime.now().strftime('%Y-%m-%d  
134.               %H:%M:%S')
135.
136.     def insert_db(self, values):
137.         cn = mysql.connector.connect(host="127.0.0.1",
138.           database="xx", user="xx", password="xx",
139.           auth_plugin="mysql_native_password")
140.         sql_executer = cn.cursor()
141.         aql1 = "INSERT INTO isp (hour, latency,
142.           download, download_ratio, upload, upload_ratio) VALUES (" + str(datetime.now().strftime("%H:%M:%S"))
143.           aql2 = ""
144.           for i in range (len(values)):
145.               values[i] = "\"" + str(values[i]) + "\""
146.               if (i!=5):
147.                   values[i] += ","
148.                   aql2 = aql2 + values[i]
149.           aql = aql1 + aql2 + ");"
150.           print (aql)
151.           sql_executer.execute(aql)
152.           cn.commit()
153.           if __name__ == '__main__':
154.               isp = ISPMonitor()
155.               try:
156.                   uptime = isp.get_uptime()
157.                   isp.run_test()
158.                   results = isp.parse_results()
159.                   isp.mount_status(uptime, results)
160.
161.                   results.insert(0,
162.                     datetime.now().strftime("%Y-%m-%d %H:%M:%S"))
163.                   results.pop()
164.                   isp.insert_db(results)
165.
166.                   isp.write('Finished')
167.               except Exception as e:
168.                   isp.write(str(e), True)
```



IMPLEMENTACIÓN DE UN SISTEMA DE MONITORIZACIÓN DE DISPOSITIVOS Y SERVICIOS EN UNA RED DOMÉSTICA

ESPECIFICACIONES DEL SISTEMA

- **DATOS CLIENTE:** ESCUELA SUPERIOR DE INGENIERÍA, UNIVERSIDAD DE CÁDIZ.
AV. UNIVERSIDAD DE CÁDIZ, 10, 11519 PUERTO REAL, CÁDIZ
956 48 32 00
DIRECCION.ESI@UCA.ES
- **DATOS AUTOR:** PABLO MANUEL GARCÍA SANCHEZ
INGENIERO INFORMÁTICO
PABLO.GARCIASANCH@ALUM.UCA.ES

Cádiz, Abril de 2021

ÍNDICE

22	OBJETIVOS DEL SISTEMA	157
23	DESCRIPCIÓN DE ACTORES	157
24	REQUISITOS FUNCIONALES	158
24.1	DIAGRAMAS CASOS DE USO	159
24.1.1	Login	159
24.1.2	Gestión de Dispositivos Administrados	159
24.1.3	Gestión de Alarmas	160
24.2	CASOS DE Uso	160

22 OBJETIVOS DEL SISTEMA

OBJ-01	Elección del sistema administrador
Descripción	Implementar un sistema de coste reducido, con suficientes recursos y estable.
Importancia	ALTA

Tabla 12: Objetivo 01: Elección del sistema administrador

OBJ-02	Elección de Agente
Descripción	Implementar agentes compatibles en los dispositivos administrados.
Importancia	ALTA

Tabla 13: Objetivo 02: Elección de Agente

OBJ-03	Monitorización variables
Descripción	El sistema debe solicitar a los diferentes dispositivos administrados las variables a monitorizar de manera periódica.
Importancia	ALTA

Tabla 14: Objetivo 03: Monitorización variables

OBJ-04	Almacenamiento datos
Descripción	El sistema debe almacenar los datos recibidos de manera periódica.
Importancia	ALTA

Tabla 15: Objetivo 04: Almacenamiento de datos

OBJ-05	Interfaz usable
Descripción	El sistema debe mostrar los datos almacenados de una forma representativa y útil.
Importancia	MEDIA

Tabla 16: Objetivo 05: Interfaz Usable

OBJ-06	Sistema de Alertas
Descripción	El sistema debe alertar cuando se produzca un evento.
Importancia	ALTA

Tabla 17: Objetivo 06: Sistema de Alertas

23 DESCRIPCIÓN DE ACTORES

ACT-01	Administrador
Descripción	Administrador del sistema.

Tabla 18: Actor 01: Administrador

24 REQUISITOS FUNCIONALES

RF-01	Sistema Administrador de Red
Objetivos asociados	OBJ-01 Elección sistema administrador
Descripción	Se ha de disponer de un equipo para monitorizar.

Tabla 19: Requisito funcional 01: Sistema Administrador de Red

RF-02	Dispositivo Administrado
Objetivos Asociados	<ul style="list-style-type: none"> OBJ-02 Elección de Agente
Descripción	Se ha de disponer de, al menos, un equipo a monitorizar

Tabla 20: Requisito funcional 02: Dispositivo Administrado

RF-03	Disponibilidad
Objetivos Asociados	<ul style="list-style-type: none"> OBJ-03 Monitorización variables OBJ-04 Almacenamiento datos OBJ-06 Sistema de Alertas
Descripción	Es necesario disponer de un sistema 24x7 operativo.

Tabla 21: Requisito funcional 03: Disponibilidad

RF-04	Estado de los dispositivos
Objetivos Asociados	<ul style="list-style-type: none"> OBJ-03 Monitorización variables
Descripción	Es necesario saber el estado de cualquier dispositivo administrado (activo o no) en cada instante.

Tabla 22: Requisito funcional 04: Estado de los dispositivos

RF-05	Espacio en Disco
Objetivos Asociados	<ul style="list-style-type: none"> OBJ-03 Monitorización variables
Descripción	Es necesario saber la capacidad de almacenamiento de cualquier dispositivo administrado en cada instante.

Tabla 23: Requisito funcional 05: Espacio en Disco

RF-06	Memoria Disponible
Objetivos Asociados	<ul style="list-style-type: none"> OBJ-03 Monitorización variables
Descripción	Es necesario saber la memoria de cualquier dispositivo administrado en cada instante.

Tabla 24: Requisito funcional 06: Memoria Disponible

RF-07	Carga del procesador

Objetivos Asociados	• OBJ-03 Monitorización variables
Descripción	Es necesario saber la carga del procesador de cualquier dispositivo administrado en cada instante.

Tabla 25: Requisito funcional 07: Carga del procesador

RF-08	Estado de ISP
Objetivos Asociados	• OBJ-03 Monitorización variables
Descripción	Es necesario saber la calidad de servicio ofrecido por nuestro proveedor en cada instante.

Tabla 26: Requisito funcional 08: Estado ISP

RF-09	Sistema de Alarmas
Objetivos Asociados	• OBJ-06 Sistema de Alertas
Descripción	Es necesario ser advertido cuando se produzca un evento.

Tabla 27: Requisito funcional 09: Sistema de Alarmas

24.1 DIAGRAMAS CASOS DE USO

24.1.1 Login

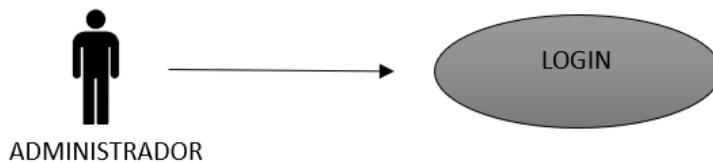


Ilustración 79: Diagrama Caso de Uso Login

24.1.2 Gestión de Dispositivos Administrados

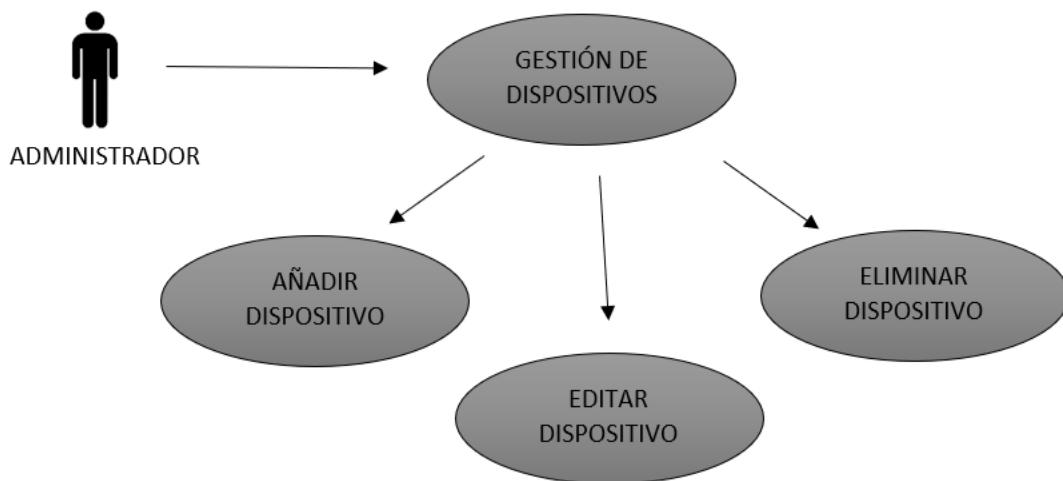


Ilustración 80: Diagrama Caso de Uso Gestión de Dispositivos Administrados

24.1.3 Gestión de Alarmas

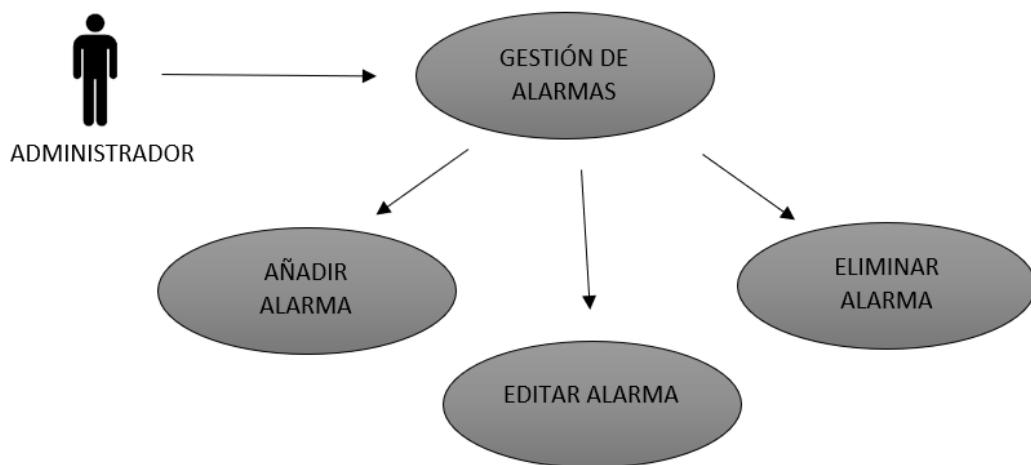


Ilustración 81: Diagrama Caso de Uso Gestión de Alarmas

24.2 CASOS DE USO

UC-01 Login		
Objetivos asociados		OBJ-05 Interfaz usable
Descripción		Se inicia sesión en la interfaz de usuario
Precondición		El actor Administrador (ACT-01) no ha iniciado sesión
Secuencia normal	Paso	Acción
	1	El actor Administrador (ACT-01) escribe en el sistema sus credenciales de usuario
	2	El sistema valida las credenciales e inicia sesión
Postcondición	El actor Administrador (ACT-01) inicia sesión	

Tabla 28: Caso de Uso 01: Login

UC-02 Añadir dispositivo		
Objetivos asociados	OBJ-02 Elección de Agente OBJ-03 Monitorización variables	
Descripción	Se añade un dispositivo administrado al sistema de monitorización.	
Precondición	El dispositivo no está siendo monitorizado	
Secuencia normal	Paso	Acción
	1	El actor Administrador (ACT-01) instala el agente SNMP en el dispositivo
	2	El actor Administrador (ACT-01) añade la IP al listado de dispositivos monitorizados del sistema
	3	El sistema recopila y almacena los datos de monitorización de forma periódica.
	4	El sistema representa los datos gráficamente en la interfaz de usuario.
Postcondición	El dispositivo comienza a ser monitorizado	

Tabla 29: Caso de Uso 02: Añadir Dispositivo

UC-03 Editar dispositivo		
Objetivos asociados	OBJ-03 Monitorización variables	
Descripción	Se editan los datos de monitorización de un dispositivo administrado.	
Precondición	El dispositivo está siendo monitorizado	
Secuencia normal	Paso	Acción
	1	El actor Administrador (ACT-01) modifica las variables a monitorizar por el sistema
	2	El actor Administrador (ACT-01) modifica la representación de los gráficos en la interfaz de usuario
Postcondición	Se monitorizan nuevos datos del dispositivo	

Tabla 30: Caso de Uso 03: Editar Dispositivo

UC-04 Eliminar dispositivo		
Objetivos asociados	OBJ-02 Elección de Agente OBJ-03 Monitorización variables OBJ-06 Sistema de Alertas	
Descripción	Se elimina un dispositivo administrado del sistema de monitorización.	
Precondición	El dispositivo está siendo monitorizado	
Secuencia normal	Paso	Acción
	1	El actor Administrador (ACT-01) elimina el agente SNMP en el dispositivo
	2	El actor Administrador (ACT-01) elimina la IP al listado de dispositivos monitorizados del sistema
Postcondición	El dispositivo deja de ser monitorizado	

Tabla 31: Caso de Uso 04: Eliminar dispositivo

UC-05 Añadir Alarma		
Objetivos asociados		OBJ-02 Elección de Agente OBJ-03 Monitorización variables OBJ-06 Sistema de Alertas
Descripción		Se alerta al Administrador (ACT-01) cuando se produce un evento en dispositivo administrado.
Precondición		El dispositivo está siendo monitorizado y el parámetro no está configurado en el sistema de Alarmas
Secuencia normal	Paso	Acción
	1	El actor Administrador (ACT-01) modifica el agente SNMP en el dispositivo
	2	El sistema envía un mensaje a Telegram con la información del evento producido
Postcondición	El Administrador (ACT-01) es alertado del evento.	

Tabla 32: Caso de Uso 05: Añadir Alarma

UC-06 Editar Alarma		
Objetivos asociados		OBJ-02 Elección de Agente OBJ-03 Monitorización variables OBJ-06 Sistema de Alertas
Descripción		Se alerta al Administrador (ACT-01) cuando se produce un evento que supera los nuevos umbrales predefinidos en el dispositivo administrado.
Precondición		El dispositivo está siendo monitorizado y el parámetro está configurado en el sistema de Alarmas
Secuencia normal	Paso	Acción
	1	El actor Administrador (ACT-01) modifica los umbrales de alarma en el agente SNMP del dispositivo
	2	El sistema envía un mensaje a Telegram con la información del evento producido cuando se supera el nuevo umbral predefinido.
Postcondición	El Administrador (ACT-01) es alertado del evento que supera el nuevo umbral.	

Tabla 33: Caso de Uso 06: Editar Alarma

UC-07 Eliminar Alarma		
Objetivos asociados		OBJ-02 Elección de Agente OBJ-03 Monitorización variables OBJ-06 Sistema de Alertas
Descripción		Se deja de alertar al Administrador (ACT-01) cuando se produce un evento en dispositivo administrado.
Precondición		El dispositivo está siendo monitorizado y el parámetro está configurado en el sistema de Alarmas
Secuencia normal	Paso	Acción
	1	El actor Administrador (ACT-01) modifica el agente SNMP en el dispositivo

Postcondición	El Administrador (ACT-01) deja de ser alertado del evento.
----------------------	--

Tabla 34: Caso de Uso 07: Eliminar Alarma



IMPLEMENTACIÓN DE UN SISTEMA DE MONITORIZACIÓN DE DISPOSITIVOS Y SERVICIOS EN UNA RED DOMÉSTICA

PRESUPUESTO

- **DATOS CLIENTE:** ESCUELA SUPERIOR DE INGENIERÍA, UNIVERSIDAD DE CÁDIZ.
AV. UNIVERSIDAD DE CÁDIZ, 10, 11519 PUERTO REAL, CÁDIZ
956 48 32 00
DIRECCION.ESI@UCA.ES
- **DATOS AUTOR:** PABLO MANUEL GARCÍA SÁNCHEZ
INGENIERO INFORMÁTICO
PABLO.GARCIASANCH@ALUM.UCA.ES

Cádiz, Abril de 2021

ÍNDICE

25.1	INTRODUCCIÓN	169
25.2	PRESUPUESTO DETALLADO	169
25.2.1	Presupuesto Hardware	169
25.2.2	Presupuesto Software	169
25.2.3	Presupuesto del Personal	169
25.2.4	Resumen del presupuesto	170

25 PRESUPUESTO

25.1 INTRODUCCIÓN

Durante este apartado mostraremos el presupuesto necesario para la consecución del proyecto, descomponiéndolo según componentes y servicios.

25.2 PRESUPUESTO DETALLADO

25.2.1 Presupuesto Hardware

Artículo	Unidades	P.V.P.	Precio Total
Raspberry Pi 4 Model B 2GB ARM-Cortex-A72 4x 1,50GHz, 2GB RAM, WLAN-ac, Bluetooth 5.0, LAN, 4x USB, 2x Micro-HDMI	1	48,08€	48,08€
Router Movistar Fibra ÓPTICA -(HGU) WiFi+ONT+VIDEOBRIDGE 2,4 y 5 GHz	1	40,00€	40,00€
TOTAL			88,08€

Tabla 35: Presupuesto Hardware

25.2.2 Presupuesto Software

Artículo	Unidades	P.V.P.	Precio Total
Raspberry Pi OS 5.4	1	0€	0€
Apache Server 2.4.46	1	0€	0€
MariaDB Server 10.5.8	1	0€	0€
Grafana 5.4.0	1	0€	0€
Python 3.9.0	1	0€	0€
Telegram	1	0€	0€
TOTAL			0€

Tabla 36: Presupuesto Software

25.2.3 Presupuesto del Personal

Artículo	Unidades	Horas	P.V.P./Hora	Precio Total
Operador	1	600	18,5€	11100€
TOTAL				11100€

Tabla 37: Presupuesto del personal

25.2.4 Resumen del presupuesto

El coste final del presupuesto para la realización del proyecto es la suma de todos los presupuestos desglosados, es decir, la suma de los costes Hardware, Software y Personal.

Artículo / Servicio	Precio
Presupuesto Hardware	88,08€
Presupuesto Software	0€
Presupuesto del Personal	11100€
TOTAL	11188,08€

Tabla 38: Resumen del presupuesto