

INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

**PRÁCTICA 3: MONITOREO DE SERVICIOS DE
RED**

ADMINISTRACIÓN DE SERVICIOS EN RED

EQUIPO 1 (Evaluación 4), EQUIPO 4 (Evaluación 5):

+ HERNÁNDEZ PINEDA MIGUEL ANGEL

+ MONROY MARTOS ELIOTH

+ ORTA CISNEROS SABRINA

+ RAMÍREZ CENTENO HUGO ENRIQUE

+ SALDAÑA AGUILAR ANDRÉS ARNULFO

+ ZÚÑIGA HERNÁNDEZ CARLOS

PROFRA. TANIBET PÉREZ DE LOS SANTOS MONDRAGÓN

18 de Mayo 2019

Índice general

1. Introducción	1
1.1. Servicios	1
1.1.1. HTTP	1
1.1.2. SSH	1
1.1.3. SMTP	1
1.1.4. FTP	2
1.1.5. DNS	2
2. Desarrollo y Resultados	3
2.1. Evaluación 4.- Configuración de Servicios	3
2.1.1. Servidor de información HTTP	3
2.1.2. Servicio de Acceso Seguro SSH (Secure Shell)	4
2.1.3. Servicio de Nombres de Dominio DNS	5
2.1.4. Servicio de Transferencia de Archivos FTP	9
2.1.5. Servicio de correo electrónico SMTP	13
2.2. Evaluación 5.- Monitoreo de Rendimiento de Servicios	28
2.2.1. Monitoreo de redimiento SSH	28
2.2.2. Monitoreo de redimiento SMTP	28
2.2.3. Monitoreo de redimiento HTTP	28
2.2.4. Monitoreo de redimiento FTP	29
2.2.5. Monitoreo de rendimiento DNS	29
2.2.6. Resultados Obtenidos	29
3. Códigos	31
3.1. Evaluación 4	31
3.1.1. Servidor de información HTTP	31
3.2. Evaluación 5	32
4. Conclusiones	59
Referencias Bibliográficas	60

Capítulo 1

Introducción

1.1. Servicios

1.1.1. HTTP

Hypertext Transfer Protocol (HTTP) (o Protocolo de Transferencia de Hipertexto en español) es un protocolo de la capa de aplicación para la transmisión de documentos hipermedia, como HTML. Fue diseñado para la comunicación entre los navegadores y servidores web, aunque puede ser utilizado para otros propósitos también. Sigue el clásico modelo cliente-servidor, en el que un cliente establece una conexión, realizando una petición a un servidor y espera una respuesta del mismo. Se trata de un protocolo sin estado, lo que significa que el servidor no guarda ningún dato (estado) entre dos peticiones. Aunque en la mayoría de casos se basa en una conexión del tipo TCP/IP, puede ser usado sobre cualquier capa de transporte segura o de confianza, es decir, sobre cualquier protocolo que no pierda mensajes silenciosamente, tal como UDP.

1.1.2. SSH

SSH (o Secure SHell) es un protocolo que facilita las comunicaciones seguras entre dos sistemas usando una arquitectura cliente/servidor y que permite a los usuarios conectarse a un host remotamente. A diferencia de otros protocolos de comunicación remota tales como FTP o Telnet, SSH encripta la sesión de conexión, haciendo imposible que alguien pueda obtener contraseñas no encriptadas. SSH está diseñado para reemplazar los métodos más viejos y menos seguros para registrarse remotamente en otro sistema a través de la shell de comando, tales como telnet o rsh. El uso de métodos seguros para registrarse remotamente a otros sistemas reduce los riesgos de seguridad tanto para el sistema cliente como para el sistema remoto.

1.1.3. SMTP

El SMTP (Simple Mail Transfer Protocol, o protocolo simple de transferencia de correo) nació en 1982 y sigue siendo el estándar de Internet más utilizado a día de

hoy. SMTP es un protocolo de mensajería empleado para mandar un email de un servidor de origen a un servidor de destino, ambos servidores SMTP.

El servidor SMTP es un ordenador encargado de llevar a cabo el servicio SMTP, permitiendo el transporte de correo electrónico por Internet. La retransmisión SMTP funciona de la siguiente manera: si el servidor SMTP confirma las identidades del remitente y del destinatario, entonces el envío se realiza.

1.1.4. FTP

FTP (File Transfer Protocol o Protocolo de transferencia de archivos) es un protocolo de red para la transferencia de archivos entre sistemas conectados a una red TCP (Transmission Control Protocol), basado en la arquitectura cliente-servidor. Desde un equipo cliente se puede conectar a un servidor para descargar archivos desde él o para enviarle archivos.

El servicio FTP es ofrecido por la capa de aplicación del modelo de capas de red TCP/IP al usuario, utilizando normalmente el puerto de red 20 y el 21. Un problema básico de FTP es que está pensado para ofrecer la máxima velocidad en la conexión, pero no la máxima seguridad, ya que todo el intercambio de información, desde el login y password del usuario en el servidor hasta la transferencia de cualquier archivo, se realiza en texto plano sin ningún tipo de cifrado, con lo que un posible atacante puede capturar este tráfico, acceder al servidor y/o apropiarse de los archivos transferidos.

1.1.5. DNS

El DNS, o sistema de nombres de dominio, traduce los nombres de dominios a direcciones IP. El sistema DNS de Internet administra el mapeo entre los nombres y las direcciones IP. Los servidores de DNS convierten las solicitudes de nombres en direcciones IP y controlan a qué servidor se dirigirá un usuario final cuando escriba un nombre de dominio. Existen distintos tipos de servicios DNS:

DNS autoritativo: un servicio de DNS autoritativo proporciona un mecanismo de actualización que los desarrolladores utilizan para administrar sus nombres DNS públicos.

DNS recurrente: los clientes normalmente no realizan consultas directamente a los servicios de DNS autoritativo. En su lugar, generalmente se conectan con otro tipo de servicio de DNS conocido como solucionador o un servicio de DNS recurrente.

Capítulo 2

Desarrollo y Resultados

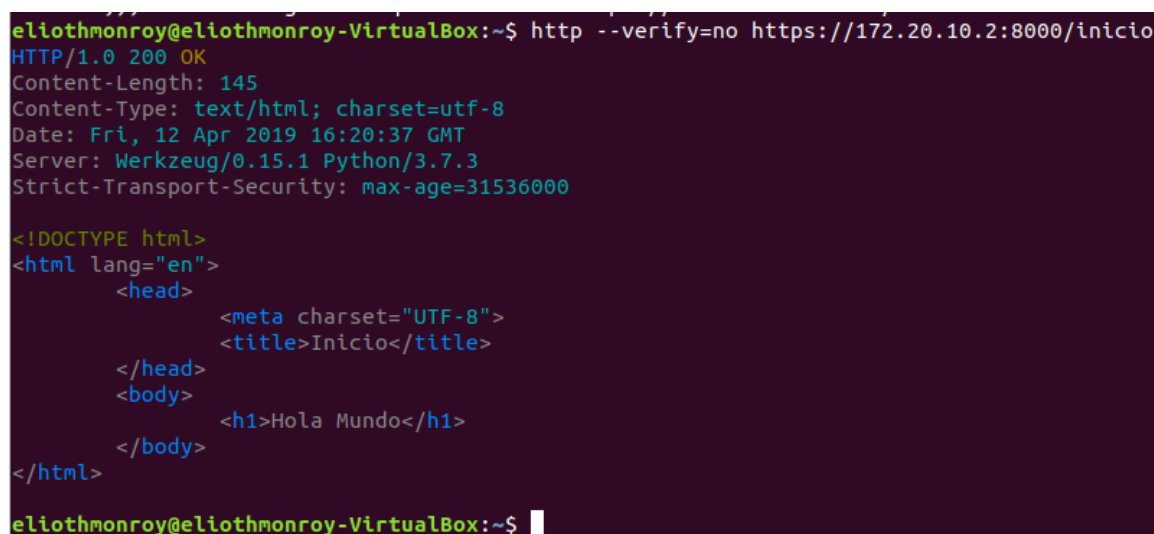
2.1. Evaluación 4.- Configuración de Servicios

2.1.1. Servidor de información HTTP

El servidor HTTP, fue implementado usando Python a través de una librería para el desarrollo web y de apis llamada Flask, esto nos permite tener un servidor multiplataforma, por lo cual no está limitado a correr bajo ciertos entornos. Esto es bueno al momento de realizar pruebas, como es nuestro caso.

Para ejecutar el servidor, basta con tener python 3 y flask instalados. Además de que Flask permite trabajar con certificados ssl en caso de ser necesario, además de que permite recibir conexiones desde otros equipos de cómputo conectados a la red.

Para probar la funcionalidad del server, se ejecutó en un equipo, y posteriormente usando la utilidad httpie, se realizó una petición get al mismo desde otro equipo, el cual en este caso era una máquina virtual. El resultado de la petición se puede observar en la Figura 2.46.



```
eliommonroy@eliommonroy-VirtualBox:~$ http --verify=no https://172.20.10.2:8000/inicio
HTTP/1.0 200 OK
Content-Length: 145
Content-Type: text/html; charset=utf-8
Date: Fri, 12 Apr 2019 16:20:37 GMT
Server: Werkzeug/0.15.1 Python/3.7.3
Strict-Transport-Security: max-age=31536000

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Inicio</title>
  </head>
  <body>
    <h1>Hola Mundo</h1>
  </body>
</html>

eliommonroy@eliommonroy-VirtualBox:~$
```

Figura 2.1: Funcionamiento correcto de servidor de información HTTP

2.1.2. Servicio de Acceso Seguro SSH (Secure Shell)

Para el funcionamiento del servicio de SSH se necesita un cliente y un servidor; el Servidor SSH fue implementado en Ubuntu.

Implementación del Servidor SSH

1. Ejecutar los siguientes comandos en una Terminal de Linux:

```
sudo apt-get update  
sudo apt install openssh-server
```

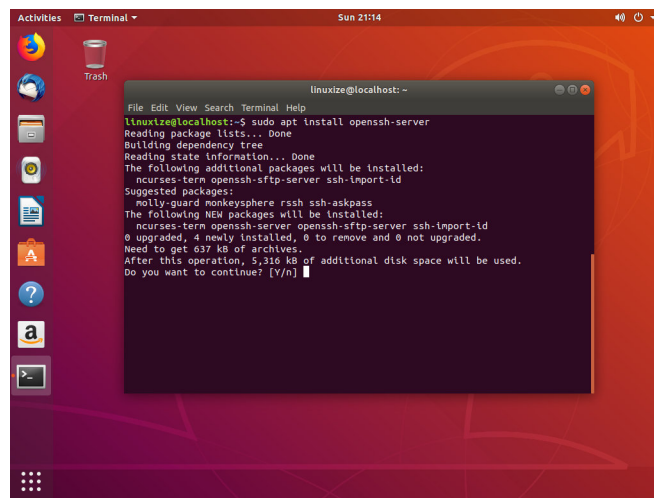


Figura 2.2: Instalación del Servidor de SSH

2. para revisar el status del servicio se ejecuta la linea de comando
sudo systemctl status ssh

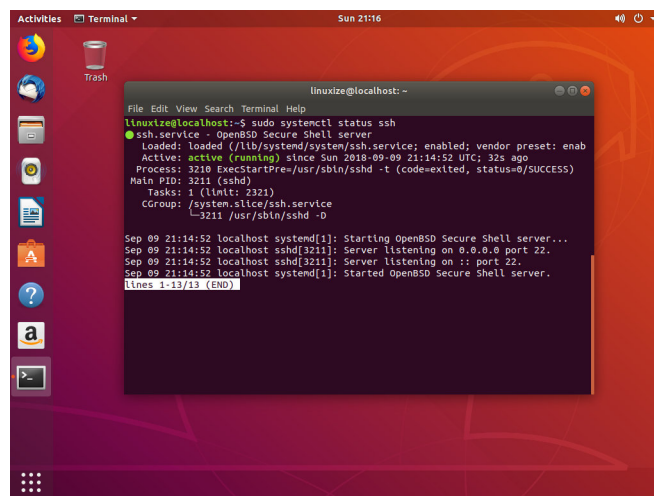


Figura 2.3: Servicio de SSH correctamente instalado

Implementación del cliente SSH

1. Se ejecuta el siguiente comando en la terminal

```
sudo apt-get install openssh-client
```

Seguir con los pasos de la instalación

2. Para conectarse a un servidor SSH se ejecuta la siguiente instrucción

```
ssh -p 22 [username]@[ip]
```

por ejemplo

```
ssh -p 22 user_redes@192.168.0.43
```

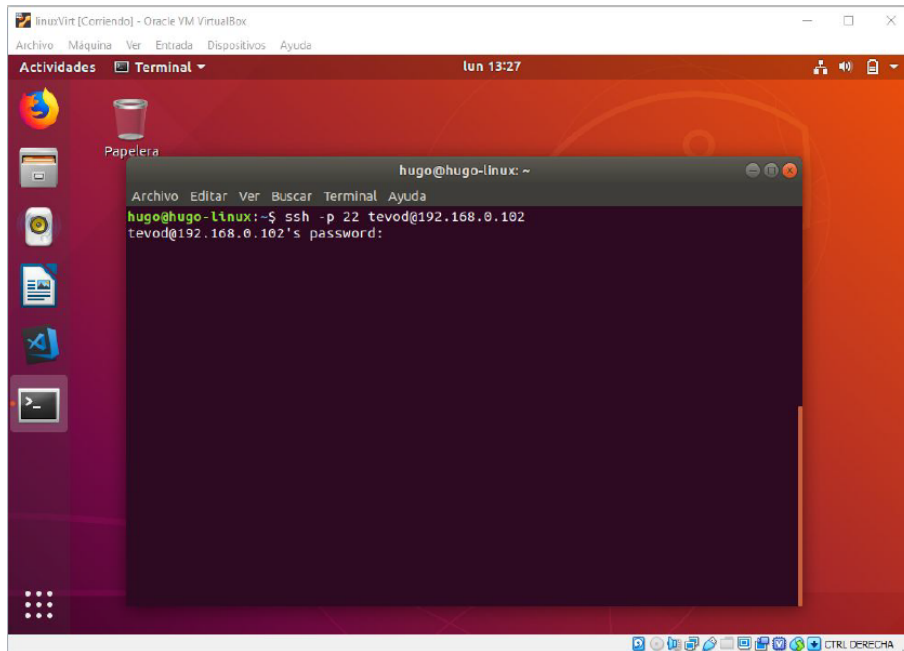


Figura 2.4: Conexión a servidor SSH

2.1.3. Servicio de Nombres de Dominio DNS

Los servidores de DNS traducen los nombres de dominio en una dirección IP y también pueden suceder de manera inversa

Para configurar un servicio de DNS en ubuntu se ejecutan los siguientes pasos:

1. Se instala el repositorio de bind9

```
sudo apt-get install bind9
```

2. Nos cambiamos de directorio

```
cd /etc/bind/
```

3. Editamos el archivo «named.conf.local»

agregamos una nueva «zona» en la cual va a estar nuestro nombre de dominio
agregamos las siguientes líneas al archivo

```
zone "midominio.com"{
    type master;
    file "/etc/bind/db.conf_midominio"
};
```

El archivo «db.conf_midominio» será donde se guarde la ip a resolver

4. Para crear el archivo «db.conf_midominio» tomamos como base el archivo db.local y realizamos una copia

```
sudo cp db.local db.conf_midominio
```

Remplazamos todos los «localhost» por el nombre de nuestro dominio y remplazamos la IP «127.0.0.1» por la ip deseada

```
BIND      DATA file for local loopback interface;
$TTL      604800
@IN       SOA      midominio.com.root.midominio.com(
                                2          ;Serial
                                604800     ;Refresh
                                86400      ;Retry
                                2419200    ;Expire
                                604800 )   ;Negative Cache TTL
;
@         IN       NS      midominio.com.
@         A        A       192.168.0.43
```

5. Reiniciamos el servicio con la siguiente instruccion en la terminal

```
sudo /etc/init.d/bind9 restart
```

6. Editamos el archivo «resolv.conf» para que la computadora utilice el DNS que hemos configurado

```
sudo nano /etc/resolv.conf
```

Dejamos la siguiente linea en el archivo de texto

```
nameserver 127.0.0.1
```


7. Probamos el servicio utilizando el siguiente comando

```
host midominio.com
```

8. Para configurar el DNS reverso, es decir dada una ip obtener el nombre de dominio
añadimos otra «zona» al archivo de configuracion «named.conf.local»

```
sudo nano named.conf.local
```

```

;
    BIND        DATA file for local loopback interface
;
$TTL          604800
@IN           SOA      midominio.com.root.midominio.com(
                                2          ;Serial
                                604800     ;Refresh
                                86400      ;Retry
                                2419200    ;Expire
                                604800 )   ;Negative Cache TTL
;
@             IN       NS          midominio.com.
34.0.168      IN       PTR         midominio.com.
```

El archivo «db.192» contendrá el nombre del dominio el cual va a resolver la ip

9. Para crear el archivo «db.192» nos hacemos una copia del archivo db.127

```
sudo cp db.127 db.192
```

10. Remplazamos los «localhost» por nuestro dominio y en la ultima linea ingresamos la ip

```

;
    BIND        DATA file for local loopback interface
;
$TTL          604800
@IN           SOA      midominio.com.root.midominio.com(
                                2          ;Serial
                                604800     ;Refresh
                                86400      ;Retry
                                2419200    ;Expire
                                604800 )   ;Negative Cache TTL
;
@             IN       NS          midominio.com.
34.0.168      IN       PTR         midominio.com.
```

11. Reiniciamos el servicio

```
sudo /etc/init.d/bind9 restart
```

12. Comprobamos que funcione el servicio con el siguiente comando *se ingresa la IP*

```
host 192.168.0.34
```

2.1.4. Servicio de Transferencia de Archivos FTP

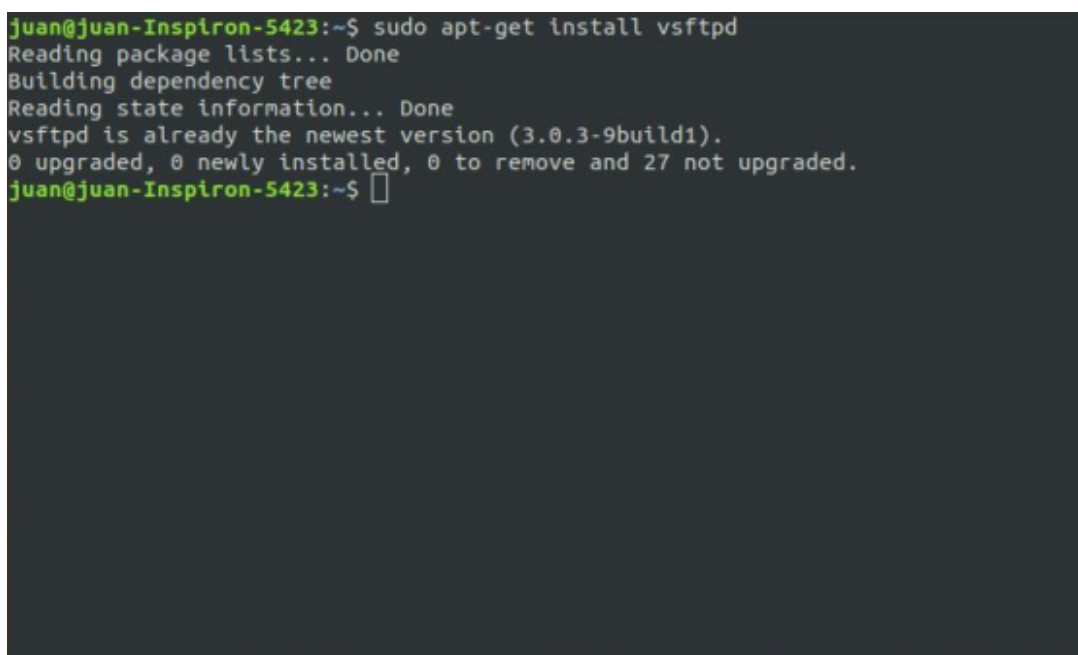
El servidor FTP que se configuró fue el de VSFTPD que se encuentra disponible para sistemas UNIX, incluyendo Linux. La configuración de este servidor es simple y ofrece desde los requerimientos mínimos como seguridad, desempeño y estabilidad, hasta requerimientos específicos como la configuración de IPs virtuales, usuarios virtuales, etc.

Los pasos que se llevaron a cabo para instalar el servidor FTP y configurarlo fueron los siguientes:

1. Ejecutar el siguiente comando en una Terminal de Linux:

```
sudo apt-get install vsftpd
```

En la Figura 2.5 se puede apreciar la ejecución del comando anterior y su salida.



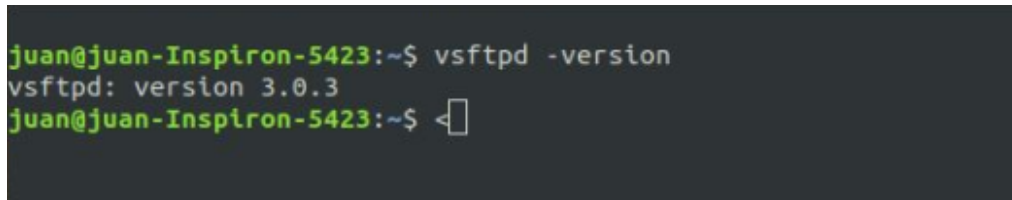
```
juan@juan-Inspiron-5423:~$ sudo apt-get install vsftpd
Reading package lists... Done
Building dependency tree
Reading state information... Done
vsftpd is already the newest version (3.0.3-9build1).
0 upgraded, 0 newly installed, 0 to remove and 27 not upgraded.
juan@juan-Inspiron-5423:~$
```

Figura 2.5: Instalación del servidor VSFTPD

2. Verificar la instalación del servidor con el siguiente comando, el cual retorna la versión instalada.

```
vsftpd -version
```

En la Figura 2.6, se muestra que se ha instalado correctamente, debido a que el comando retornó la versión correspondiente.



```
juan@juan-Inspiron-5423:~$ vsftpd -version
vsftpd: version 3.0.3
juan@juan-Inspiron-5423:~$
```

Figura 2.6: Verificación de la instalación del servidor VSFTPD

3. Administrar la ejecución del servidor mediante los siguientes comandos:

`sudo systemctl restart vsftpd` - Reinicia el servidor FTP

`sudo systemctl start vsftpd` - Inicia el servidor FTP

`sudo systemctl stop vsftpd` - Detiene el servidor FTP

`sudo systemctl stop vsftpd` - Detiene el servidor FTP

`sudo systemctl status vsftpd` - Consulta el status del servidor FTP

En la Figura 2.7, se ejecutan los comandos en un determinado orden para que al ejecutar el que retorna el status del servidor, éste aparezca con el estado de **active (running)**.

```

juan@juan-Inspiron-5423:~$ sudo systemctl stop vsftpd
juan@juan-Inspiron-5423:~$ sudo systemctl start vsftpd
juan@juan-Inspiron-5423:~$ sudo systemctl restart vsftpd
juan@juan-Inspiron-5423:~$ sudo systemctl restart vsftpd
juan@juan-Inspiron-5423:~$ sudo systemctl status vsftpd
● vsftpd.service - vsftpd FTP server
   Loaded: loaded (/lib/systemd/system/vsftpd.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2019-04-10 22:58:58 CDT; 3s ago
     Process: 27948 ExecStartPre=/bin/mkdir -p /var/run/vsftpd/empty (code=exited, status=0/S
   Main PID: 27949 (vsftpd)
       Tasks: 1 (limit: 4915)
      CGroup: /system.slice/vsftpd.service
              └─27949 /usr/sbin/vsftpd /etc/vsftpd.conf

abr 10 22:58:58 juan-Inspiron-5423 systemd[1]: Starting vsftpd FTP server...
abr 10 22:58:58 juan-Inspiron-5423 systemd[1]: Started vsftpd FTP server.
lines 1-11/11 (END)

```

Figura 2.7: Administración de la ejecución

4. Verificar el estado del firewall. Es necesario permitir las conexiones a los puertos 20 y 21. En algunas distribuciones UNIX el firewall bloquea estos puertos. Así, lo anterior se puede llevar a cabo con los siguientes comandos:

```
sudo ufw status
```

```
sudo ufw allow 21
```

```
sudo ufw allow 21
```

En la Figura 2.8, se puede apreciar la ejecución de los comandos anteriores.

```

juan@juan-Inspiron-5423:~$ sudo ufw status
Status: inactive
juan@juan-Inspiron-5423:~$ sudo ufw allow 21
Rules updated
Rules updated (v6)
juan@juan-Inspiron-5423:~$ sudo ufw allow 20
Rules updated
Rules updated (v6)
juan@juan-Inspiron-5423:~$ 

```

Figura 2.8: Verificar estado de firewall y permitir el uso de los puertos 20 y 21

Como cliente FTP, es posible conectarse al servidor mediante un navegador o usando Filezilla, el cual es un cliente que soporta los protocolos FTP, SFTP y FTPS.

En este caso se utilizó un navegador web. Para esto es necesario ingresar la siguiente línea en la barra de direcciones del navegador:

ftp://url

Donde url es la IP del servidor o bien puede ser localhost en el caso en el que se estén realizando pruebas de manera local.

En la Figura 2.9, se puede apreciar la ejecución de los comandos anteriores.

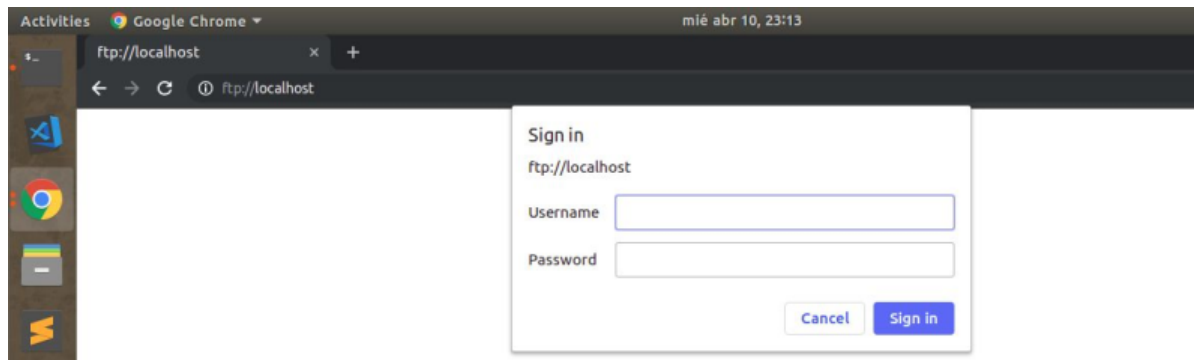


Figura 2.9: Accediendo a los recursos de VSFTPD mediante un navegador web

Por default, el usuario y la contraseña son **root**, por lo cual se recomienda crear un nuevo usuario que tenga únicamente los permisos necesarios.

La Figura 2.10, muestra el árbol de directorios a los que se pueden acceder.

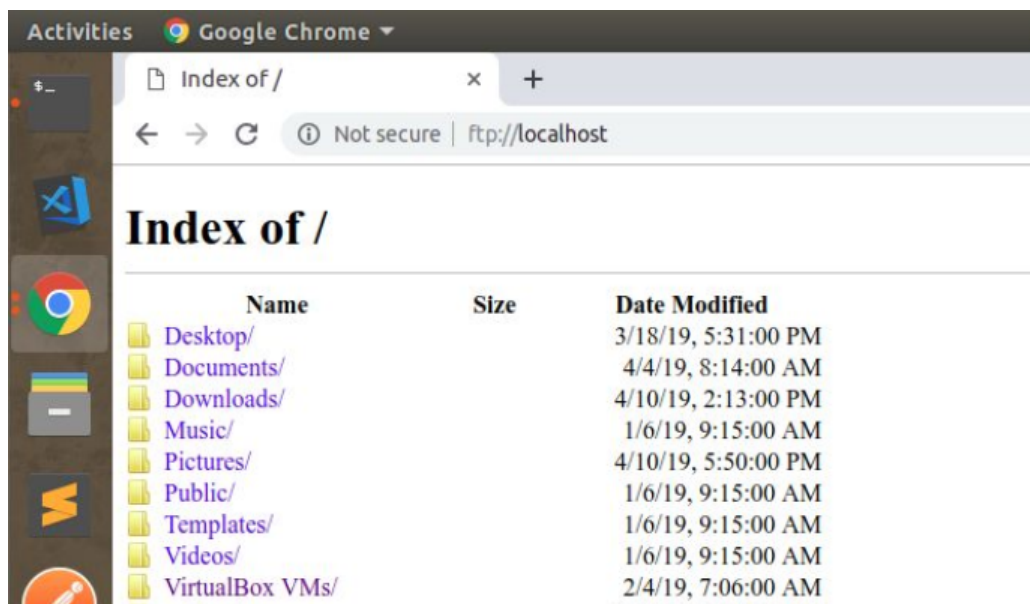


Figura 2.10: Directorios disponibles

Si se desea descargar algún recurso, basta con dar click encima del mismo para que se comience la descarga.

La Figura 2.11, corresponde a la evidencia que fue adjuntada para la evaluación para validar que el servidor de FTP que se configuró funciona correctamente.

```

carlos@carlos-VirtualBox:~$ sudo systemctl status vsftpd
[sudo] contraseña para carlos:
● vsftpd.service - vsftpd FTP server
   Loaded: loaded (/lib/systemd/system/vsftpd.service; enabled; vendor preset: e
   Active: active (running) since Fri 2019-04-12 15:47:02 CDT; 4h 18min left
     Main PID: 659 (vsftpd)
       Tasks: 1 (limit: 4915)
    CGroup: /system.slice/vsftpd.service
            └─659 /usr/sbin/vsftpd /etc/vsftpd.conf

abr 12 15:47:02 carlos-VirtualBox systemd[1]: Starting vsftpd FTP server...
abr 12 15:47:02 carlos-VirtualBox systemd[1]: Started vsftpd FTP server.
abr 12 10:51:15 carlos-VirtualBox vsftpd[1701]: pam_unix(vsftpd:auth): check pas
abr 12 10:51:15 carlos-VirtualBox vsftpd[1701]: pam_unix(vsftpd:auth): authentic
abr 12 11:15:18 carlos-VirtualBox vsftpd[1967]: pam_unix(vsftpd:auth): check pas
abr 12 11:15:18 carlos-VirtualBox vsftpd[1967]: pam_unix(vsftpd:auth): authentic
lines 1-14/14 (END)

```

Figura 2.11: Comando `sudo systemctl status vsftpd` para visualizar el status del servidor FTP

2.1.5. Servicio de correo electrónico SMTP

Para la transferencia de datos que componen a los correos electrónicos se utiliza el protocolo SMTP, que define una serie de normas para el envío. Para ello este servicio utiliza una estructura cliente-servidor.

Para instalar y configurar el servicio de correo electrónico usando SMTP en Ubuntu es necesario Postfix, el cual, es un agente de transporte de correo de manera que nos permite enrutar y transferir correos electrónicos. Así, se llevaron a cabo los siguientes pasos:

1. Ejecutar el siguiente comando para instalar Postfix:

```
sudo apt-get install postfix
```

2. En la configuración de Postfix se elige “Sitio de internet”

En la Figura 2.12, se muestran las opciones disponibles para la configuración de Postfix.

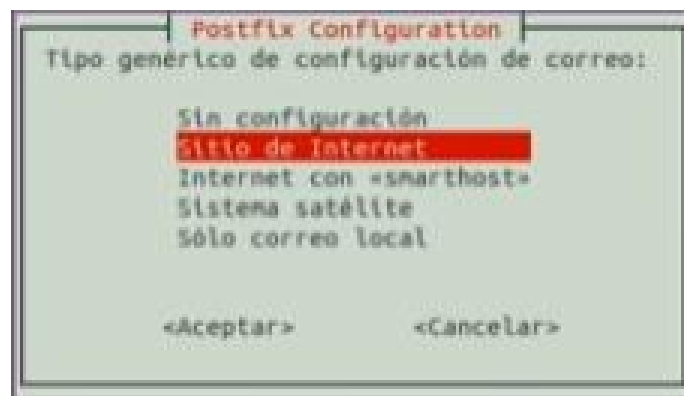


Figura 2.12: Configuración de Postfix

3. Se escribe el dominio “redes.com”, el cual será el nombre del host

En la Figura 2.13, se puede apreciar que la configuración solicita que se ingrese el nombre del sistema de correo; es decir, del host.

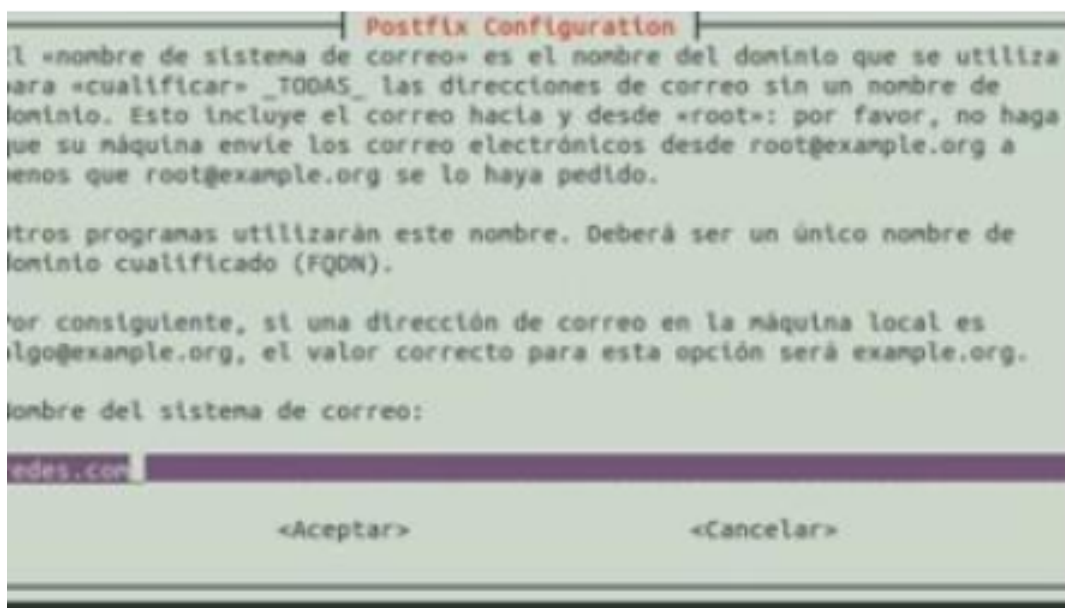


Figura 2.13: Asignando nombre al sistema de correo

4. Para hacer cambios en la configuración de Postfix tenemos que editar el archivo “etc/postfix/main.cf”. Después de modificar dicho archivo, debemos asegurarnos de ejecutar “etc/init.d/postfix reload” o el comando “service postfix restart” para que los cambios sean aplicados. Para modificar el archivo, usamos “nano /etc/postfix/main.cf” con la finalidad de poder configurar la dirección IP

En la Figura 2.14, se puede observar el parte del contenido del archivo main.cf. Para configurar la dirección IP, en la Figura 2.15, se muestra cómo debe de quedar configurada.


```

GNU nano 2.2.6      Archivo: /etc/postfix/main.cf
# See /usr/share/postfix/main.cf.dist for a commented, more complete version

# Debian specific: Specifying a file name will cause the first
# line of that file to be used as the name. The Debian default
# is /etc/mailname.
#myorigin = /etc/mailname

smtpd_banner = $myhostname ESMTTP $mail_name (Ubuntu)
biff = no

# appending .domain is the MUA's job.
append_dot_mydomain = no

# Uncomment the next line to generate "delayed mail" warnings
#delay_warning_time = 4h

readme_directory = no

[ 41 líneas leídas ]
^G Ver ayuda  ^O Guardar   ^R Leer Fich ^Y RePág.    ^K Cortar Tex ^C Pos actual
^X Salir      ^J Justificar ^W Buscar    ^V Pág. Sig. ^U PegarTxt   ^T Ortografia

```

Figura 2.14: Parte del contenido del archivo main.cf

```

mynetworks = 192.168.1.69/24, 127.0.0.0/8 [::ffff:127.0.0.0]/104 [::1]/128
mailbox_size_limit = 0
recipient_delimiter = +

```

Figura 2.15: Configuración de la dirección IP

5. Se reinicia el servicio de Postfix para que los cambios surtan efecto.

En la Figura 2.16, se tiene la salida del comando "sudo service postfix restart".

```

emerson@emerson-VirtualBox:~$ sudo service postfix restart
* Stopping Postfix Mail Transport Agent postfix [ OK ]
* Starting Postfix Mail Transport Agent postfix [ OK ]

```

Figura 2.16: Reinicio del servicio Postfix

6. Es necesario instalar el paquete "mailutils" para enviar correos desde la línea de comandos. Nota: se puede usar mailx. Por lo tanto, se ejecuta el comando "sudo aptget install mailutils"

En la Figura 2.17, se tiene la salida del comando "sudo aptget install mailutils".

```

emerson@gemerson-VirtualBox:~$ sudo apt-get install mailutils
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
El paquete indicado a continuación se instaló de forma automática y ya no es ne
esarios.
  brsero-cdrkit
Use 'apt-get autoremove' to remove it.
Se instalarán los siguientes paquetes extras:
  libgsasl7 libkyotocabinet16 libmailutils4 libntlm0 mailutils-common
Paquetes sugeridos:
  mailutils-mh mailutils-doc
Se instalarán los siguientes paquetes NUEVOS:
  libgsasl7 libkyotocabinet16 libmailutils4 libntlm0 mailutils
  mailutils-common
0 actualizados, 6 se instalarán, 0 para eliminar y 625 no actualizados.
Necesito descargar 1 287 kB de archivos.
Se utilizarán 6 922 kB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] S

```

Figura 2.17: Instalación de mailutils

7. Creación de usuarios para realizar pruebas del correcto funcionamiento del servidor. Para esto, es necesario utilizar los comandos que ofrece mailutils.

Por ejemplo, para enviar un correo a user@example.com, se ejecuta la siguiente línea en una Terminal:

```
mail user@example.com
```

La cual va a devolver como salida lo siguiente:

```
Cc:
Subject:
```

En “Cc:” se pone un destinatario que recibirá una copia del correo. En “Subject:” el asunto del mensaje. Después, se podrá escribir libremente el contenido del mensaje y para terminar de redactar el contenido se pulsa CTRL-D para finalizar y enviar el mensaje.

8. Ejecución del comando “adduser nombre_usuario” para agregar un usuario al servidor de correo. Nota: para ello es necesario agregarlo como super usuario con sudo su antecediendo el comando anterior.

En la Figura 2.18, se tiene la salida del comando “adduser nombre_usuario”.

```

emerson@emerson-VirtualBox:~$ adduser user1
adduser: Sólo root puede añadir un usuario o un grupo al sistema.
emerson@emerson-VirtualBox:~$ sudo su
[sudo] password for emerson:
root@emerson-VirtualBox:/home/emerson# adduser user1
Añadiendo el usuario `user1' ...
Añadiendo el nuevo grupo `user1' (1001) ...
Añadiendo el nuevo usuario `user1' (1001) con grupo `user1' ...
Creando el directorio personal `/home/user1' ...
Copiando los ficheros desde `/etc/skel' ...
Introduzca la nueva contraseña de UNIX:
Vuelva a escribir la nueva contraseña de UNIX:
passwd: contraseña actualizada correctamente
Cambiando la información de usuario para user1
Introduzca el nuevo valor, o presione INTRO para el predeterminado
    Nombre completo []: usuario redes 1
    Número de habitación []:
    Teléfono del trabajo []:
    Teléfono de casa []:
    Otro []:
¿Es correcta la información? [S/n] S

```

Figura 2.18: Creación de usuarios del servidor de correo

9. Inicio de sesión de alguno de los usuarios creados. Esta acción se puede apreciar en la Figura 2.19

```

root@emerson-VirtualBox:/home/emerson# login user1
Contraseña:
Welcome to Ubuntu 14.04.2 LTS (GNU/Linux 3.16.0-30-generic i686)

 * Documentation:  https://help.ubuntu.com/

515 packages can be updated.
433 updates are security updates.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

user1@emerson-VirtualBox:~$

```

Figura 2.19: Inicio de sesión

10. Envío de correo a un usuario diferente usando “mail nombre_usuario@host”. Se termina de redactar con Ctrl+D. Véase la Figura 2.20

```

user1@emerson-VirtualBox:~$ mail user2@redes.com
Cc: user3
Subject: prueba de correo
Estimado esta es una prueba de correo con copia al usuario3.
user1@emerson-VirtualBox:~$ █

```

Figura 2.20: Envío de correo

11. Cierre de sesión utilizando “logout” para iniciar sesión con el usuario a quien se le mando el correo. Se puede observar en la Figura 2.21 que en primera instancia aparece el mensaje “tiene correo”

```

user1@emerson-VirtualBox:~$ mail user2@redes.com
Cc: user3
Subject: prueba de correo
Estimado esta es una prueba de correo con copia al usuario3.
user1@emerson-VirtualBox:~$ logout
root@emerson-VirtualBox:/home/emerson# login user2
Contraseña:
Welcome to Ubuntu 14.04.2 LTS (GNU/Linux 3.16.0-30-generic i686)

 * Documentation:  https://help.ubuntu.com/

New release '16.04.6 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

Tiene correo.
user2@emerson-VirtualBox:~$ █

```

Figura 2.21: Consultando bandeja de entrada

12. Con el comando “mail” se ve el correo que está en bandeja de entrada y se selecciona el correo con el número que le corresponde. Véase la Figura 2.22

```

user2@emerson-VirtualBox:~$ mail
"/var/mail/user2": 1 mensaje 1 nuevo
>N 1 usuario redes 1   mié abr 10 07:3  14/625  prueba de correo
? 1
Return-Path: <user1@emerson-VirtualBox>
X-Original-To: user2@redes.com
Delivered-To: user2@redes.com
Received: by emerson-VirtualBox (Postfix, from userid 1001)
        id 1468724F74; Wed, 10 Apr 2019 07:38:03 -0500 (CDT)
To: <user2@redes.com>
Cc: <user3@emerson-VirtualBox>
Subject: prueba de correo
X-Mailer: mail (GNU Mailutils 2.99.98)
Message-Id: <20190410123805.1468724F74@emerson-VirtualBox>
Date: Wed, 10 Apr 2019 07:38:03 -0500 (CDT)
From: user1@emerson-VirtualBox (usuario redes 1)

Estimado usuario2, esta es un prueba de correo. Asi mismo se copia en el correo
Estimado esta es una prueba de correo con copia al usuario3.

```

Figura 2.22: Visualizando correo recibido

13. Instalación de dovecot, el cual permitirá ver los correos en un MDA, para ello dovecot usa los protocolos pop e imap. Para realizar esto, se debe ejecutar el

siguiente comando:

```
sudo apt-get install dovecot-po3d dovecotimapd
```

En la Figura 2.23, se muestra la salida del comando anterior.

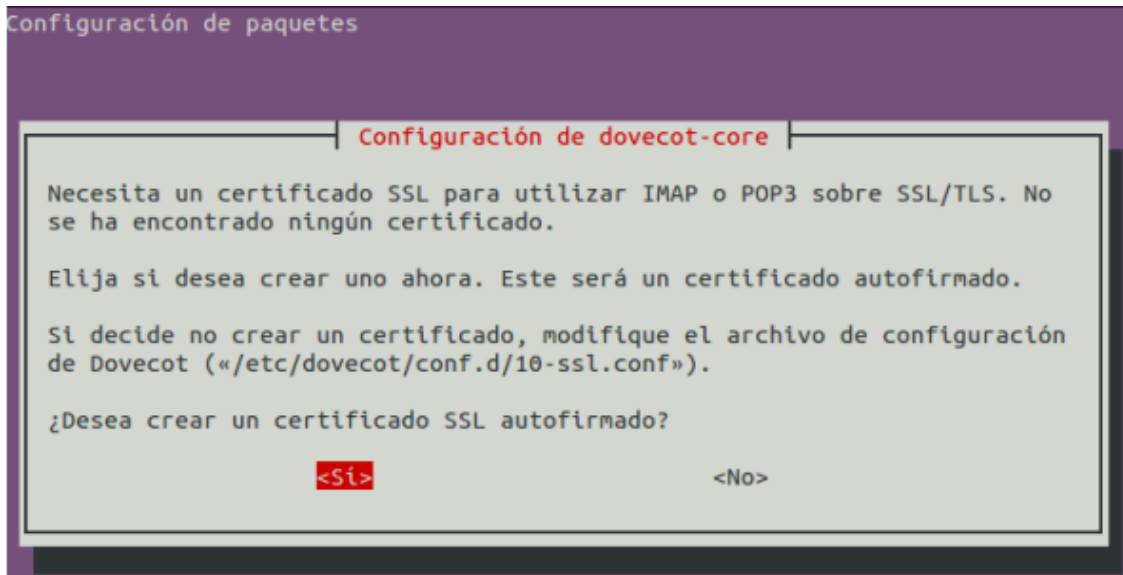


Figura 2.23: Instalación de Dovecot

En el nombre del equipo, para este caso, se escribirá “redes.com”.

14. Modificación del archivo de configuración con “sudo nano /etc/dovecot/dovecot.conf” para quitar el comentario “listen”. Esta acción se muestra en la Figura 2.24

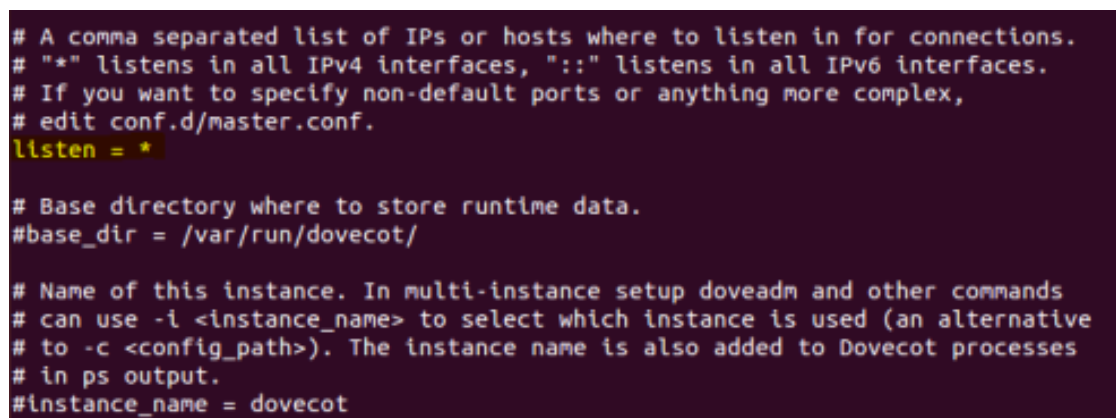


Figura 2.24: Modificación del archivo dovecot.conf

15. Se ubica el fichero de configuración donde se asegura que sea una conexión segura. Para ello, es necesario ejecutar el siguiente comando: “sudo nano /etc/dovecot/conf.d/auth.conf”. Véase la Figura 2.25


```

GNU nano 2.2.6 Archivo: /etc/dovecot/conf.d/10-auth.conf

##
## Authentication processes
##

# Disable LOGIN command and all other plaintext authentications unless
# SSL/TLS is used (LOGINDISABLED capability). Note that if the remote IP
# matches the local IP (ie. you're connecting from the same computer), the
# connection is considered secure and plaintext authentication is allowed.
# See also ssl=required setting.
#disable_plaintext_auth = yes

# Authentication cache size (e.g. 10M). 0 means it's disabled. Note that
# bsdauth, PAM and vpopmail require cache_key to be set for caching to be use
#auth_cache_size = 0
# Time to live for cached data. After TTL expires the cached record is no
# longer used, *except* if the main database lookup returns internal failure.
# We also try to handle password changes automatically: If user's previous
# authentication was successful, but this one wasn't, the cache isn't used.
# For now this works only with plaintext authentication.

```

Figura 2.25: Modificación del archivo 10-auth.conf

Posteriormente, se elimina el comentario a la directriz “disable_plaintext_auth” y se agrega el valor “no”. De tal forma que quede “disable_plaintext_auth=no”.

16. Se abre el archivo “sudo nano /etc/dovecot/conf.d/10-mail.conf” para la ubicación del mail. Esta acción se puede visualizar en la Figura 2.26

```

#
mail_location = mbox:~/mail:INBOX=/var/mail/%u

```

Figura 2.26: Ubicación del mail

17. Se reinicia el servicio con “service dovecot restart” (Figura 2.27)

```

root@emerson-VirtualBox:/home/emerson# service dovecot restart
dovecot stop/waiting
dovecot start/running, process 11428
root@emerson-VirtualBox:/home/emerson#

```

Figura 2.27: Reinicio del servicio

18. Verificar que el servidor tenga montado imap y pop3. Véase la Figura 2.28

```

root@emerson-VirtualBox:/home/emerson# telnet 192.168.1.69 imap
Trying 192.168.1.69...
Connected to 192.168.1.69.
Escape character is '^]'.
* OK [CAPABILITY IMAP4rev1 LITERAL+ SASL-IR LOGIN-REFERRALS ID ENABLE IDLE START
TLS AUTH=PLAIN] Dovecot (Ubuntu) ready.

root@emerson-VirtualBox:/home/emerson# telnet 192.168.1.69 smtp
Trying 192.168.1.69...
Connected to 192.168.1.69.
Escape character is '^]'.
220 emerson-VirtualBox ESMTP Postfix (Ubuntu)

```

Figura 2.28: Verificación de componentes instalados

19. Agregamos en los hosts la dirección y dominio del servidor (/etc/hosts) con el comando “sudo nano /etc/hosts” (Figura 2.29)

```

GNU nano 2.2.6      Archivo: /etc/hosts      Modificado
127.0.0.1          localhost
127.0.1.1          emerson-VirtualBox

# The following lines are desirable for IPv6 capable hosts
::1                ip6-localhost ip6-loopback
fe00::0            ip6-localnet
ff00::0            ip6-mcastprefix
ff02::1            ip6-allnodes
ff02::2            ip6-allrouters
192.168.1.69       imap.redes.com
192.168.1.69       pop3.redes.com
192.168.1.69       smtp.redes.com

```

Figura 2.29: Modificación del archivo hosts

20. Se eliminan los comentarios para las líneas “ssl=yes , ssl_cert_file y ssl_key” (Figura 2.30)

```

##
## SSL settings
##
# SSL/TLS support: yes, no, required. <doc/wiki/SSL.txt>
ssl = yes

# PEM encoded X.509 SSL/TLS certificate and private key. They're opened before
# dropping root privileges, so keep the key file unreadable by anyone but
# root. Included doc/mkcert.sh can be used to easily generate self-signed
# certificate, just make sure to update the domains in dovecot-openssl.cnf
ssl_cert = </etc/dovecot/dovecot.pem
ssl_key = </etc/dovecot/private/dovecot.pem

```

Figura 2.30: Eliminación de comentarios en las líneas de ssl

21. Se abre fichero de configuración de Postfix con “sudo nano /etc/postfix/master.cf” (Figura 2.31)

```

GNU nano 2.2.6      Archivo: /etc/postfix/master.cf

#
# Postfix master process configuration file.  For details on the format
# of the file, see the master(5) manual page (command: "man 5 master" or
# on-line: http://www.postfix.org/master.5.html).
#
# Do not forget to execute "postfix reload" after editing this file.
#
# =====
# service type private unpriv chroot wakeup maxproc command + args
#          (yes)   (yes)   (yes)   (never) (100)
# =====
smtp      inet  n       -       -       -       -       smtpd
#smtp     inet  n       -       -       -       1       postscreen
#smtpd    pass  -       -       -       -       -       smtpd
#dnsblog  unix  -       -       -       -       0       dnsblog
#tlsproxy unix  -       -       -       -       0       tlsproxy
#submission inet n       -       -       -       -       smtpd
#  -o syslog_name=postfix/submission
#  -o smtpd_tls_security_level=encrypt

```

Figura 2.31: Visualización de archivo master.cf

Se eliminan los comentarios a las líneas sombreadas en la Figura 2.32

```

GNU nano 2.2.6      Archivo: /etc/postfix/master.cf      Modificado

#  -o smtpd_reject_unlisted_recipient=no
#  -o smtpd_client_restrictions=$mua_client_restrictions
#  -o smtpd_helo_restrictions=$mua_helo_restrictions
#  -o smtpd_sender_restrictions=$mua_sender_restrictions
#  -o smtpd_recipient_restrictions=
#  -o smtpd_relay_restrictions=permit_sasl_authenticated,reject
#  -o milter_macro_daemon_name=ORIGINATING
#smtps    inet  n       -       -       -       -       smtpd
#  -o syslog_name=postfix/smtps
#  -o smtpd_tls_wrappermode=yes
#  -o smtpd_sasl_auth_enable=yes
#  -o smtpd_reject_unlisted_recipient=no
#  -o smtpd_client_restrictions=$mua_client_restrictions
#  -o smtpd_helo_restrictions=$mua_helo_restrictions
#  -o smtpd_sender_restrictions=$mua_sender_restrictions
#  -o smtpd_recipient_restrictions=
#  -o smtpd_relay_restrictions=permit_sasl_authenticated,reject
#  -o milter_macro_daemon_name=ORIGINATING
#628      inet  n       -       -       -       -       qmqpd

^G Ver ayuda  ^O Guardar   ^R Leer Fich ^Y RePág.    ^K Cortar Tex^C Pos actual
^X Salir      ^J Justificar^W Buscar    ^V Pág. Sig. ^U PegarTxt  ^T Ortografía

```

Figura 2.32: Eliminación de comentarios

22. Se reinicia el servicio de Postfix (Figura 2.33)

```

root@emerson-VirtualBox:/home/emerson# service postfix restart
* Stopping Postfix Mail Transport Agent postfix      [ OK ]
* Starting Postfix Mail Transport Agent postfix      [ OK ]
root@emerson-VirtualBox:/home/emerson#

```

Figura 2.33: Reinicio del servicio Postfix

23. Hasta este punto se configuró e instaló correctamente postfix, y dovecot. Además,

se probó el funcionamiento correcto del servidor y apartir de este punto se puede usar Thunderbird el cual es un cliente de correo electrónico open/source.

Las Figuras que se muestran a continuación, sirvieron de ayuda para utilizar Thunderbird.

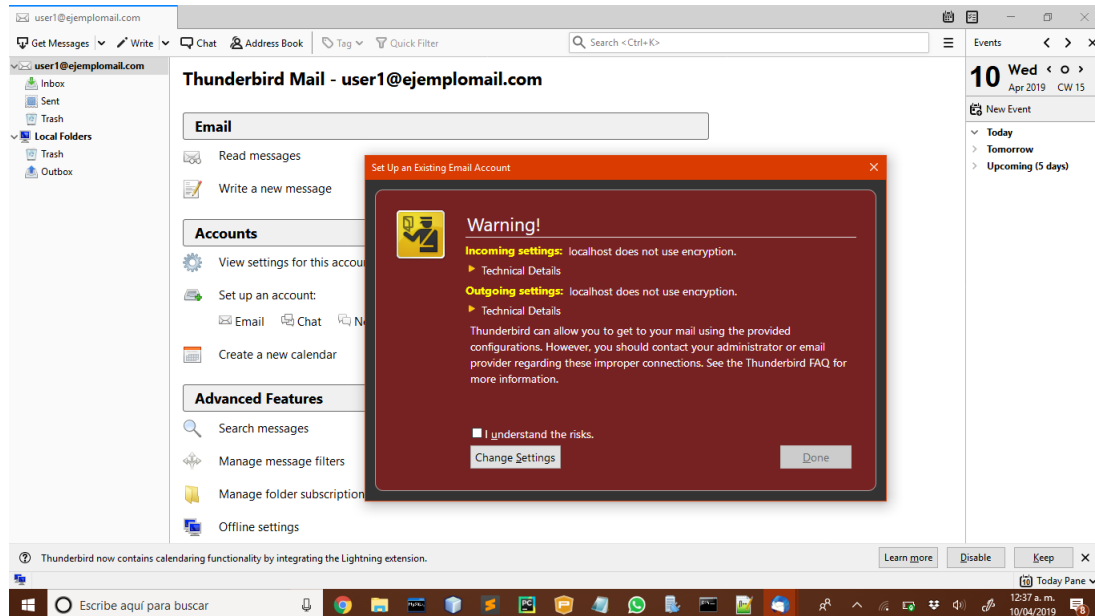


Figura 2.34: Se acepta el riesgo

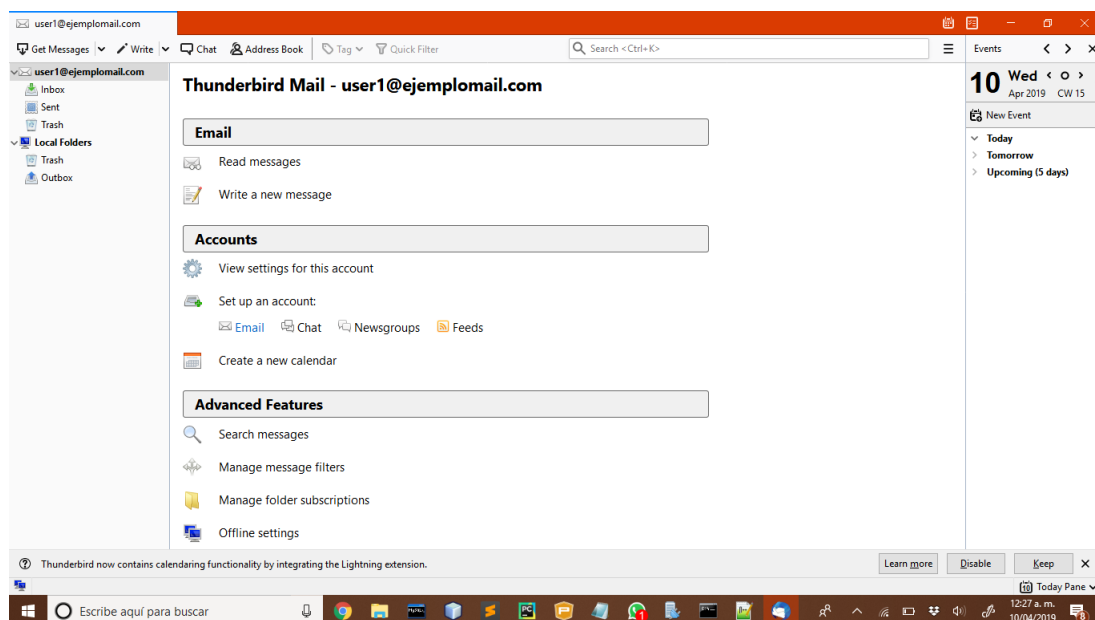


Figura 2.35: Inicio de Thunderbird

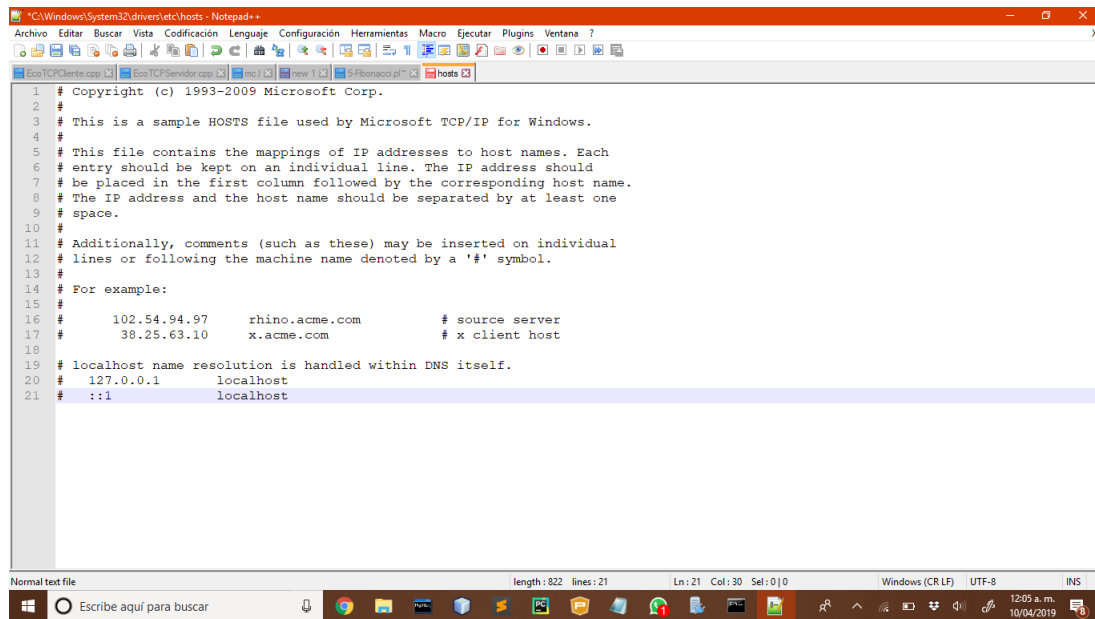


Figura 2.36: Se agrega la IP en el archivo Hosts

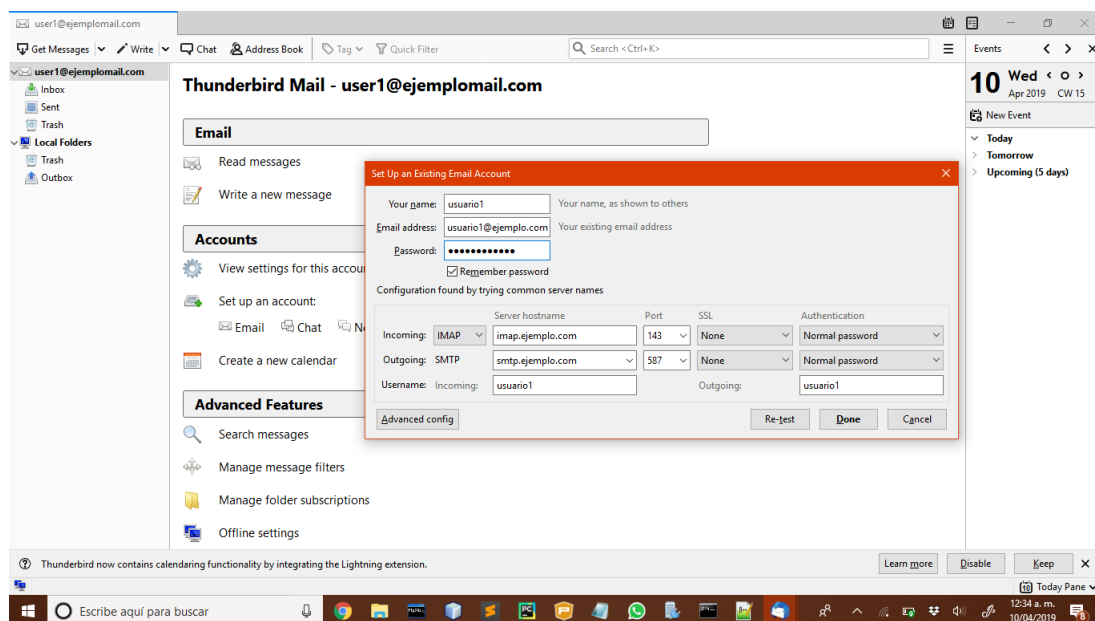


Figura 2.37: Configuración del inicio

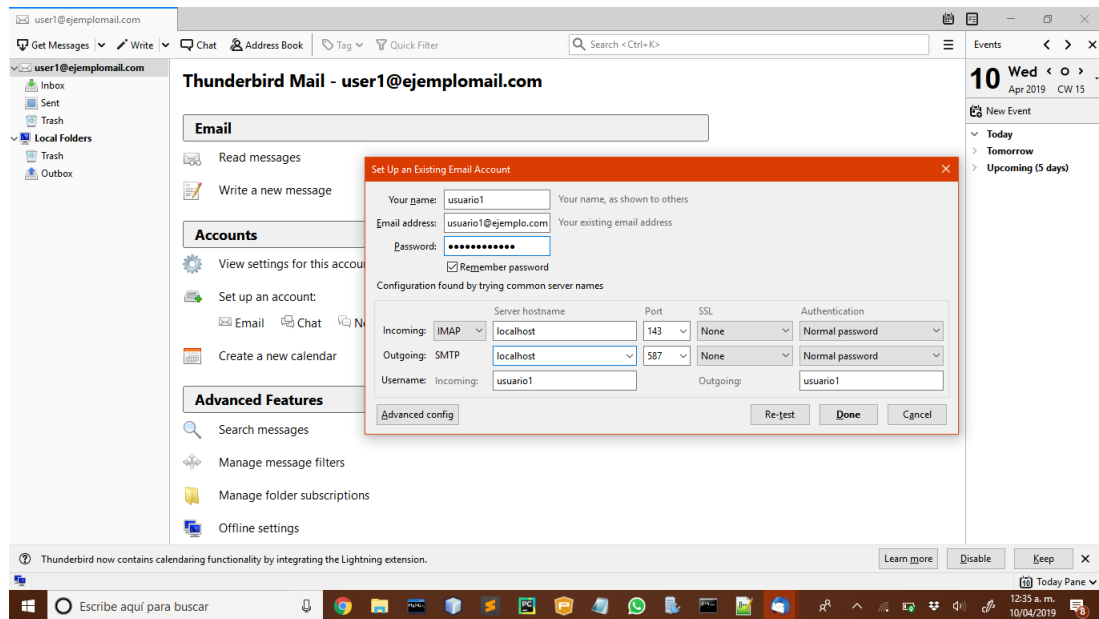


Figura 2.38: Registro

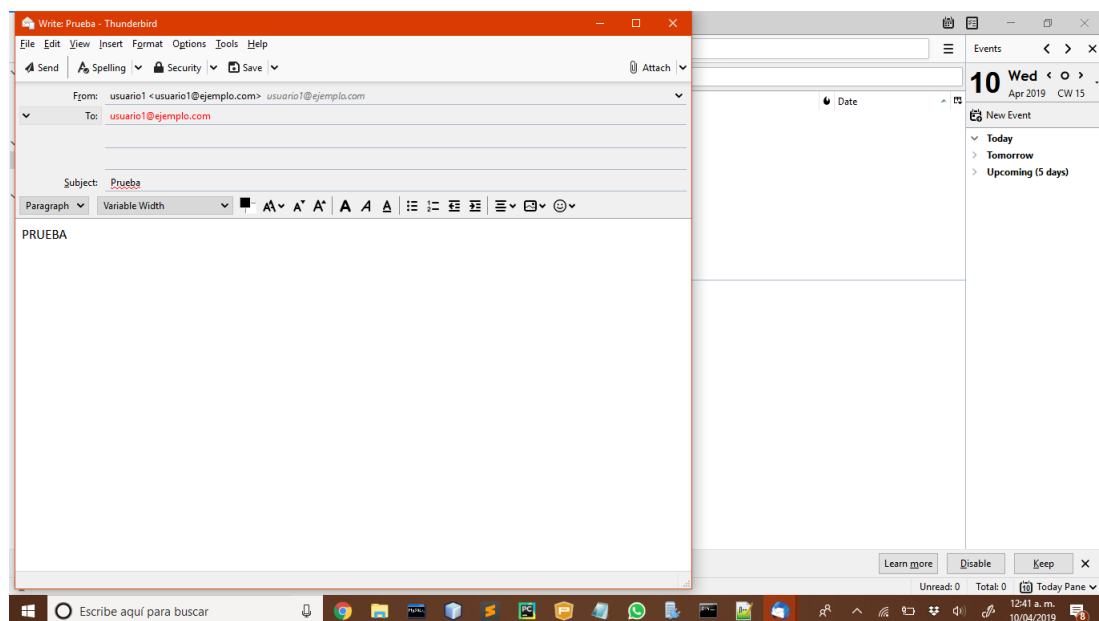


Figura 2.39: Enviar correo

Las Figuras siguientes, corresponden a la evidencia que fue adjuntada para la evaluación para validar que el servidor de correo que se configuró funciona correctamente.

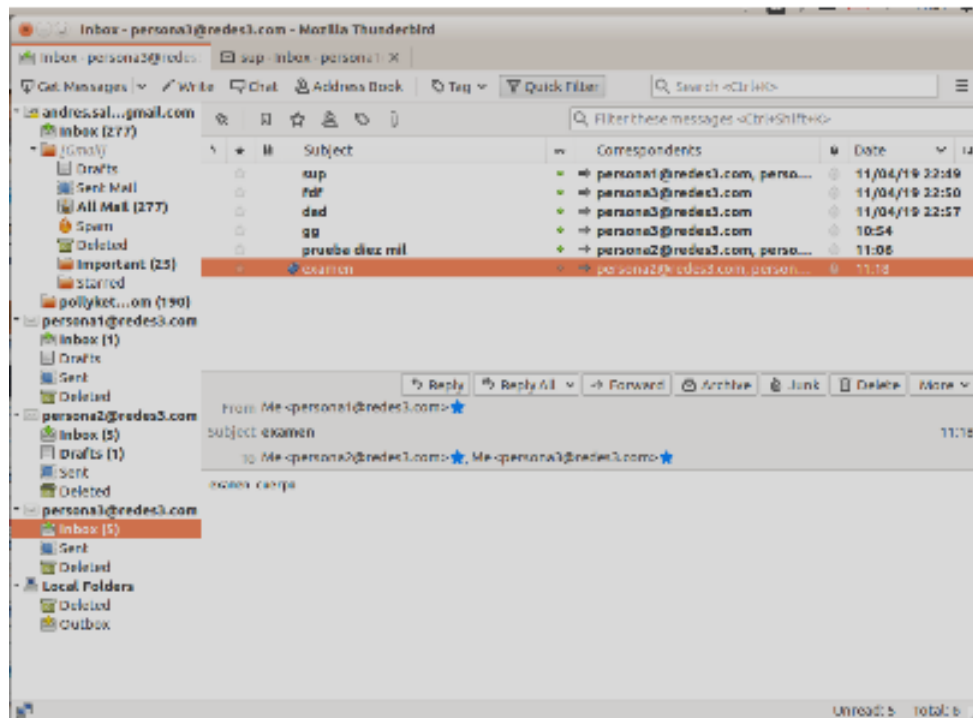


Figura 2.40: Mensaje enviado a persona 3 desde persona 1

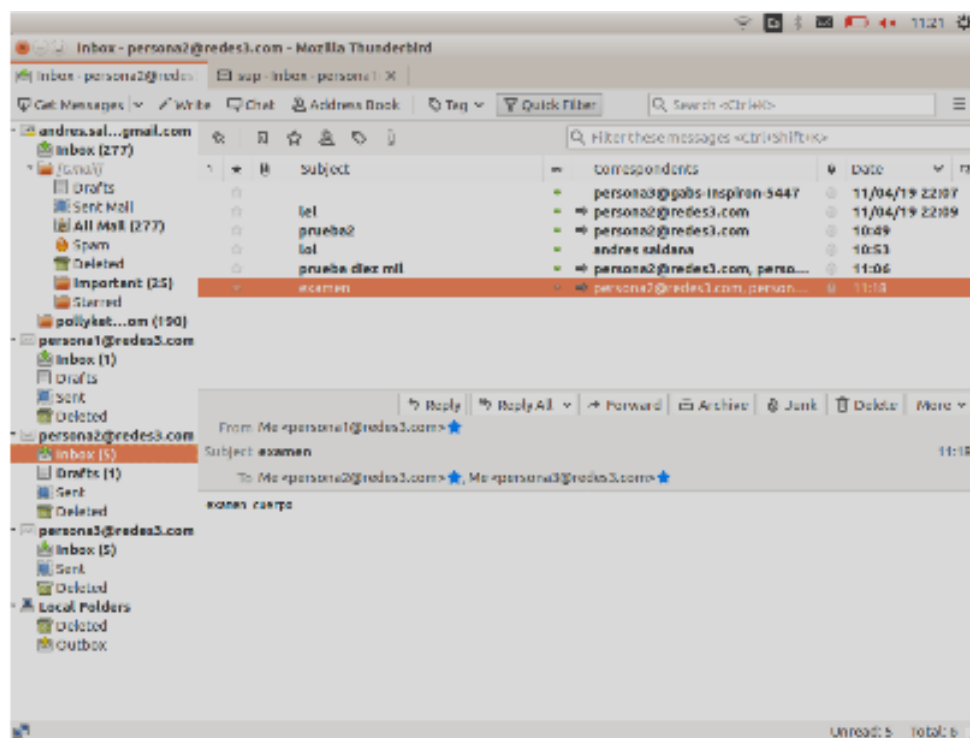


Figura 2.41: Mensaje enviado a persona 2 desde persona 1

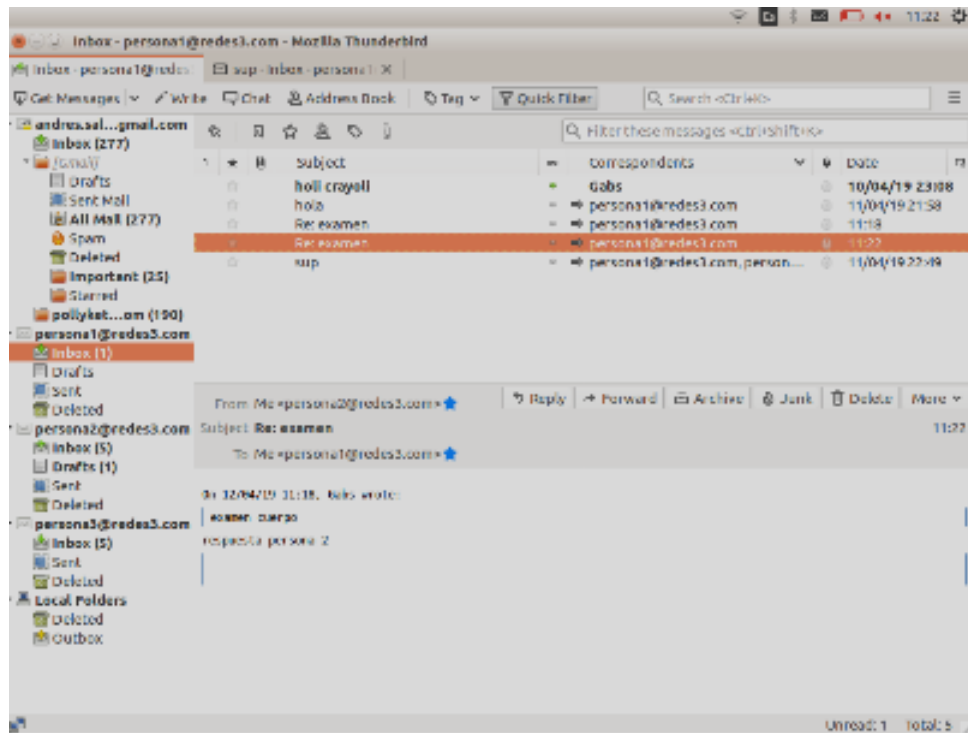


Figura 2.42: Mensaje respondido a persona 1 desde persona 2

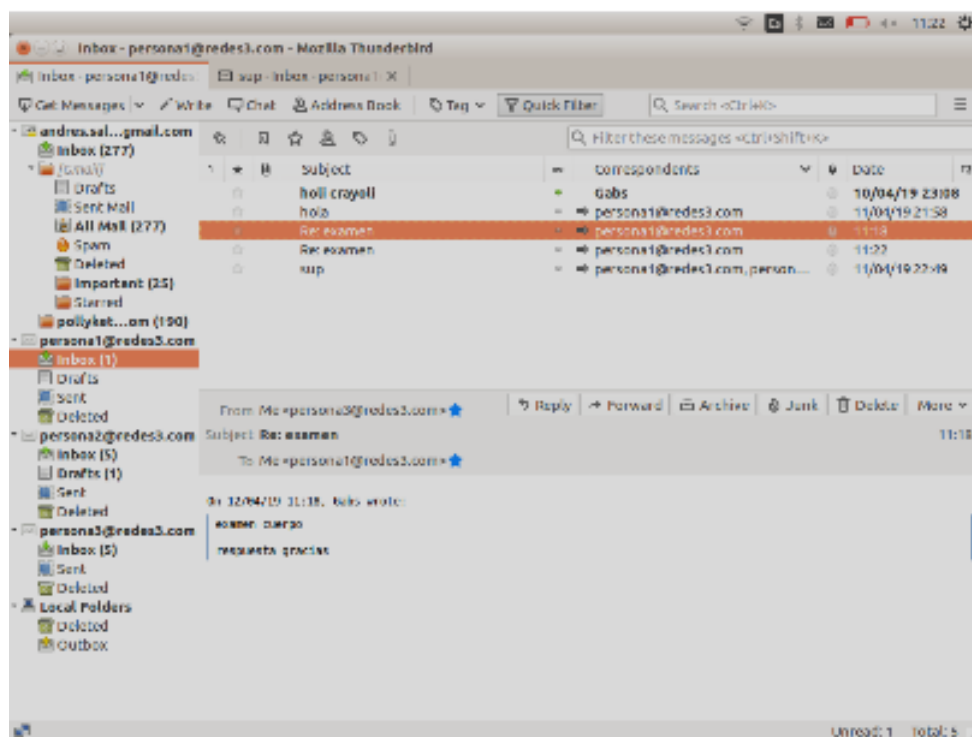


Figura 2.43: Mensaje respondido a persona 1 desde persona 3

2.2. Evaluación 5.- Monitoreo de Rendimiento de Servicios

2.2.1. Monitoreo de rendimiento SSH

La evaluación de rendimiento SSH lo hacemos por medio de un programa escrito en Python y que utiliza una biblioteca llamada **paramiko**, que nos brinda herramientas para trabajar con un servidor SSH.

Iniciando sesión remotamente con SSH y ejecutando un comando de SSH por medio de la biblioteca podemos conocer el numero de conexiones realizadas al servidor, también podemos conocer el tiempo de conexión una vez que el cliente conectado cierra su conexión, el trafico de entrada y salida, por ultimo, restando el tiempo desde el inicio de sesión menos el tiempo de cierre.

2.2.2. Monitoreo de rendimiento SMTP

Para evaluar el rendimiento del servicio SMTP, ocupamos las bibliotecas de Python **smtplib**, **poplib** y **imaplib**, adicionalmente debemos tener una cuenta de correo existente que tenga acceso a su correo por medio de IMAP4 y POP3 en el servidor SMTP que habilitamos, ya que por medio de un programa en Python, enviaremos un correo con el servidor SMTP, midiendo el tiempo de envío.

En el momento que se hace el envío exitoso de SMTP, comenzamos a verificar la bandeja de entrada del correo por medio de POP para conocer el tiempo que el servicio tarda en recibir el correo, para por ultimo paso borrar los correos creados por esta prueba, haremos el mismo procedimiento con IMAP, de esta manera, conseguimos tanto el tiempo de respuesta de SMTP, IMAP y POP para conocer su rendimiento, si alguno de estos pasos falla el estado del servicio cambiara a "DOWN".

2.2.3. Monitoreo de rendimiento HTTP

Para conocer el rendimiento de nuestro servidor HTTP, hicimos uso de la biblioteca de Python **http.client**:

El tiempo de respuesta es calculado enviando una petición GET por un recurso del servidor, en este caso conseguiremos un archivo contenido en el servidor, con el propósito de conocer el tamaño en bytes de la respuesta y cual es el ancho de banda, como el ancho de banda es la cantidad de información o de datos que se puede enviar a través de una conexión de red en un período de tiempo dado, esa cifra la podemos conocer dividiendo el tamaño del archivo conseguido entre el tiempo de respuesta.

2.2.4. Monitoreo de redimiento FTP

Para llevar a cabo el monitoreo del servicio FTP se utilizó el código *ftp-cliente.py* donde se hizo uso de las bibliotecas **urllib.request** y **ftplib** que nos permite trabajar con archivos de acuerdo con el servidor FTP.

Una vez que está iniciado el servidor FTP, el cliente hace una petición de un archivo, solicitando un documento que se encuentra en una ruta especificada y mostrando las credenciales de acceso, el servidor envía el archivo como respuesta, el cliente lo almacena y se termina la conexión, en caso de error, la conexión se cierra de manera automática.

2.2.5. Monitoreo de rendimiento DNS

En cuanto al servidor DNS, se monitoraron los valores haciendo uso de la biblioteca **dns.resolver** que nos permite hacer una conexión al servidor DNS proporcionándole el puerto y la dirección a la que deseamos conectarnos.

El cliente inicia la conexión al momento en que se le es asignado una dirección IP o dominio y se define el puerto de conexión, si dichos parámetros son correctos, la conexión al servidor se hace de manera correcta, en caso contrario se muestra un mensaje de error y los parámetros devueltos por el servidor indican que la conexión no se llevó a cabo definiendo el status del servidor como *down*.

2.2.6. Resultados Obtenidos

Una vez que se obtuvieron los datos necesarios de cada servidor, se generó un archivo PDF donde se muestran las gráficas generadas y los parámetros de tiempo de respuesta de cada servicio configurado. dicho PDF se muestra a continuación.










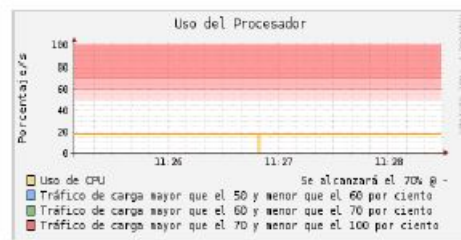
	agregar_agente	14/05/2019 08:33 ...	Archivo PY	3 KB
	archivo_ftp	14/05/2019 08:33 ...	Archivo TXT	0 KB
	detector_umbrales	14/05/2019 08:33 ...	Archivo PY	14 KB
	dispersion	14/05/2019 08:33 ...	Hoja de cálculo d...	43 KB
	Evaluacion5_sinEstres	14/05/2019 08:33 ...	Adobe Acrobat D...	37 KB
	ftp_cliente	14/05/2019 08:33 ...	Archivo PY	3 KB
	gestor	14/05/2019 08:33 ...	Archivo PY	3 KB
	http_cliente	14/05/2019 08:33 ...	Archivo PY	2 KB
	linux	14/05/2019 08:33 ...	Archivo PNG	540 KB

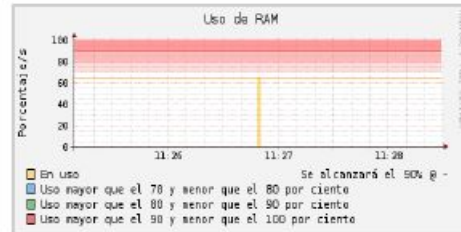
Figura 2.44: PDF Generado

Rendimiento de Servidor

Uso de CPU



Uso de RAM



Uso de HDD

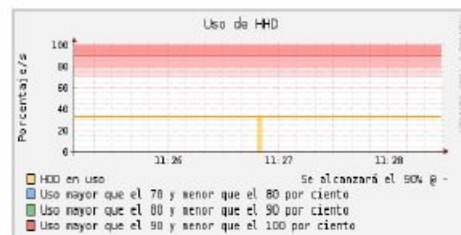


Figura 2.45: Gráficas de Rendimiento del Servidor

Supervisión de Sensores

Sensor SMTP & IMAP/POP3	Tiempo de Respuesta SMTP-IMAP: 0.02222442626953125 Tiempo de Respuesta IMAP: 0.11541318893432617 Tiempo de Respuesta SUMA: 0.12731409072875977 Tiempo de Respuesta SMTP-POP3: 0.006242275238037109 Tiempo de Respuesta POP3: 0.0175936222076416 Tiempo de Respuesta SUMA: 0.02383589744567871 Status: Up
Sensor HTTP	Tiempo de Respuesta: 0.026473522186279297 Bytes Recibidos: 102400 Ancho de Banda de descarga: 3868.01572074425 Kb/s Status: Up
Sensor FTP	Tiempo de Respuesta: 0.0012526512145996094 Respuesta del Servidor: 226 Transfer complete. Status: Up
Sensor FTP Server File Count	No. de Archivos: 2
Sensor DNS	Tiempo de Respuesta: 0.002599477767944336 Status: up

Figura 2.46: Parámetros monitoreados de cada servicio

Capítulo 3

Códigos

3.1. Evaluación 4

3.1.1. Servidor de información HTTP

Para el servidor de información http, se configuró un pequeño servidor implementado en python usando la librería de Flask. A continuación se muestra el código. servidor-flask.py:

```

1 #pip3 install flask
2 #openssl req -x509 -newkey rsa:4096 -keyout key.pem -out cert.pem -days 365
3 #pip3 install Flask-SSLify
4 #http --verify=no POST https://example.org
5 from flask import Flask, request, render_template
6 from flask_sslify import SSLify
7
8 app=Flask(__name__)
9 sslify = SSLify(app)
10
11 @app.route('/', methods=[ 'GET', 'POST', 'HEAD', 'PUT', 'DELETE', 'CONNECT', 'OPTIONS', '
    TRACE', 'PATCH' ])
12 def hola():
13     if request.method=='POST':
14         return "Hola, mundo POST"
15     elif request.method=='GET':
16         return "Hola, mundo GET"
17     elif request.method=='HEAD':
18         return "Hola, mundo HEAD"
19     elif request.method=='PUT':
20         return "Hola, mundo PUT"
21     elif request.method=='DELETE':
22         return "Hola, mundo DELETE"
23     elif request.method=='CONNECT':
24         return "Hola, mundo CONNECT"
25     elif request.method=='OPTIONS':
26         return "Hola, mundo OPTIONS"
27     elif request.method=='TRACE':
28         return "Hola, mundo TRACE"
29     elif request.method=='PATCH':
30         return "Hola, mundo PATCH"
31     else:
32         return "Hola, mundo método que no debería de existir"
33
34 @app.route('/inicio', methods=[ 'GET' ])
35 def inicio():

```

```

36     return render_template('inicio.html')
37
38 if __name__ == '__main__':
39     # app.run(debug=True, port=5000, host='0.0.0.0', ssl_context=('cert.pem', 'key.
    pem'))
40     app.debug=False
41     app.run(debug=False, port=5000, host='0.0.0.0', ssl_context=('cert.pem', 'key.pem
    '))

```

3.2. Evaluación 5

Para el desarrollo del monitor de los servidores, se continuó trabajando en Python 3 usando como base los programas anteriormente desarrollados. El programa principal que ejecuta todo el monitoreador es el siguiente.

main.py:

```

1 from gestor import Gestor
2
3 gestor=Gestor()

```

El siguiente archivo, funciona para permitir al usuario agregar agentes, y a su vez monitorearlos, creando un hilo por cada agente al momento de que se da click.

gestor.py:

```

1 #https://www.tutorialspoint.com/python/python_gui_programming.htm
2 from tkinter import *
3 from agregar_agente import AgregarAgente
4 from agente import obtenerAgentes, obtenerInfoPrincipalAgente, eliminar
5 from monitor import Monitor
6 from functools import partial
7
8 class Gestor():
9     """Clase principal del programa"""
10     def __init__(self):
11         #Creamos la ventana y sus medidas
12         self.top=Tk()
13         self.top.geometry("800x600")
14         self.top.resizable(0,0)
15         #Definimos los elementos de la misma y a cada uno le hacemos un pack
16         self.b = Button(self.top, text="Agregar agente", command=self.agregarAgente)
17         self.b.grid(row=0, column=3, sticky=W+E+N+S)
18         self.obtenerAgentes()
19         #Final
20         self.top.mainloop()
21     def agregarAgente(self):
22         #Destruimos la ventana actual y traemos a AgregarAgente
23         self.top.destroy()
24         self.agente=AgregarAgente()
25     def monitorearAgente(self, id_agente):
26         monitor=Monitor(id_agente)
27         monitor.start()
28
29     def eliminarAgente(self, id_agente):
30         self.top.destroy()
31         eliminar(id_agente)
32
33     def obtenerAgentes(self):

```

```

34     ids_agentes=obtenerAgentes()
35     if len(ids_agentes)>0:
36         Label(self.top, text="Total de agentes registrados: "+ str(len(ids_agentes
37         ))) .grid(row=0, sticky=W)
38     else:
39         Label(self.top, text="No hay agentes registrados").grid(row=0, sticky=W)
40     for id_agente in ids_agentes:
41         info=obtenerInfoPrincipalAgente(id_agente)
42         frame=Frame(self.top, width = 600, height = 200, relief = 'raised',
43         borderwidth=2)
44         frame.grid(row = int(id_agente), column = 0, columnspan=6, sticky=W+E+N+S
45         )
46         Label(frame, text="ID agente: "+id_agente).grid(row=1, column=0, sticky=W)
47         Label(frame, text="Hostname: "+info[0]).grid(row=2, column=0, sticky=W)
48         Label(frame, text="IP: "+info[1]).grid(row=3, column=0, sticky=W)
49         if info[2]==0:
50             Label(frame, text="Estado de conexión: Desconectado").grid(row=4, column
51             =0, sticky=W)
52             Button(frame, text="Monitorear", state=DISABLED, command=partial(self.
53             monitorearAgente, id_agente)).grid(row=2, column=3, sticky=E)
54         else:
55             Label(frame, text="Estado de conexión: Conectado").grid(row=4, column=0,
56             sticky=W)
57             Button(frame, text="Monitorear", command=partial(self.monitorearAgente,
58             id_agente)).grid(row=2, column=3, sticky=E)
59             #Label(frame, text="").grid(row=1, column=0, sticky=W)
60             Button(frame, text="Eliminar", command=partial(self.eliminarAgente,
61             id_agente)).grid(row=4, column=3, sticky=E)

```

El archivo que contiene la clase con al que se modela a un agente es la siguiente.
agente.py:

```

1 import json
2 import os.path
3 from SNMP import getInfo, consultav2SNMP, consultav3SNMP
4 def obtenerAgentes():
5     print("Voy a obtener los agentes")
6     with open('agentes.json','r') as f:
7         if os.path.getsize('agentes.json') > 0:
8             #print("Existe al menos un agente")
9             data=json.load(f)
10            return list(data.keys())
11        else:
12            pass
13            #print("No hay agentes registrados")
14    return []
15 def obtenerInfoPrincipalAgente(id):
16     data={}
17     info=[]
18     with open('agentes.json','r') as f:
19         if os.path.getsize('agentes.json') > 0:
20             data=json.load(f)
21             #0->hostname
22             info.append(data[id]["hostname"])
23             #1->ip
24             info.append(data[id]["ip"])
25             #2-> estado conexión
26             res=consultav2SNMP(data[id]["comunidad"],data[id]["ip"],int(data[id]["version"]
27             ),0,'SNMPv2-MIB','sysName',int(data[id]["puerto"]))
28             if res=="":
29                 info.append(0)
30             else:

```

```

30     info.append(1)
31     #info.append(consultav2SNMP(data[id]["comunidad"], data[id]["ip"], int(data[id]
32     #info.append(consultav3SNMP(data[id]["comunidad"], data[id]["ip"], int(data[id]
33     info.append(data[id]["comunidad"])
34     info.append(data[id]["version"])
35     info.append(data[id]["puerto"])
36     return info
37
38 def obtenerInfoAgente(id):
39     data={}
40     info=[]
41     with open('agentes.json','r') as f:
42         if os.path.getsize('agentes.json') > 0:
43             data=json.load(f)
44
45     aux=getInfo(data[id]["comunidad"], data[id]["ip"], int(data[id]["version"]), int(
46     #obtener nombre versión y logo SO, número de interfaces de red, tiempo de
47     #ubicación física e información de contacto del administrador
48     #0->nombre so
49     if "Ubuntu" in aux[1]:
50         info.append("Ubuntu")
51         info.append("ubuntu.png")
52     elif "Windows" in aux[1]:
53         info.append("Windows")
54         info.append("windows.png")
55     elif "Darwin" in aux[1]:
56         info.append("MacOs")
57         info.append("macos.png")
58     else:
59         info.append("Linux")
60         info.append("linux.png")
61     continuar=0
62     info.append("Sin especificar")
63     for a in aux[1].split():
64         if continuar==1:
65             info[2]=a
66             break
67         if "Version" in a:
68             continuar=1
69     info.append(aux[2])
70     #info.append(int(aux[3])/100)
71     info.append(aux[3])
72     info.append(aux[4])
73     #print (info)
74     return info
75
76 def eliminar(id_agente):
77     data={}
78     with open('agentes.json','r+') as f:
79         if os.path.getsize('agentes.json') > 0:
80             data=json.load(f)
81             data.pop(id_agente, None)
82             #print(data)
83             f.seek(0)
84     with open('agentes.json','w') as f:
85         json.dump(data, f, sort_keys="True", indent=4)
86     from gestor import Gestor

```

```

87 gestor=Gestor()
88
89 class Agente():
90     def __init__(self,id_agente):
91         self.id_agente=id_agente
92         self.hostname, self.ip, self.conexion, self.comunidad, self.version, self.
puerto = obtenerInfoPrincipalAgente(id_agente)
93         self.nombre_so, self.logo_so, self.version_so, self.num_interfaces, self.
ubicacion, self.contacto=obtenerInfoAgente(id_agente)
94         #Falta obtener tiempo de reinicio

```

Para agregar un agente, es usado el siguiente programa.
agregar-agente.py:

```

1 from tkinter import *
2 from tkinter import messagebox
3 import json
4 import os.path
5 #from main import Main
6
7 class AgregarAgente():
8     """docstring for Agente"""
9     def __init__(self):
10         #Creamos la ventana y sus medidas
11         self.data={}
12         self.top=Tk()
13         self.top.geometry("500x500")
14         self.top.resizable(0,0)
15         #Definimos los elementos de la misma y a cada uno los acomodamos en el grid
16         self.b = Button(self.top, text="Regresar", command=self.cancelar)
17         self.b.grid(row=10, column=2, sticky=W+E+N+S)
18         self.b = Button(self.top, text="Agregar agente", command=self.crearAgente)
19         self.b.grid(row=10, column=1, sticky=W+E+N+S)
20
21         Label(self.top, text="Hostname").grid(row=1, sticky=W)
22         self.hostname=Entry(self.top)
23         self.hostname.grid(row=1, column=1)
24         Label(self.top, text="IP").grid(row=2, sticky=W)
25         self.ip=Entry(self.top)
26         self.ip.grid(row=2, column=1)
27         Label(self.top, text="Versión SNMP").grid(row=3, sticky=W)
28         self.version=Entry(self.top)
29         self.version.grid(row=3, column=1)
30         Label(self.top, text="Puerto").grid(row=4, sticky=W)
31         self.puerto=Entry(self.top)
32         self.puerto.insert(0, '161')
33         self.puerto.grid(row=4, column=1)
34         Label(self.top, text="Comunidad").grid(row=5, sticky=W)
35         self.comunidad=Entry(self.top)
36         self.comunidad.grid(row=5, column=1)
37         #Final
38         self.top.mainloop()
39         #Regresar a la pantalla anterior
40         def cancelar(self):
41             self.top.destroy()
42             #El import lo puse aquí por que si no marca un error extraño
43             from gestor import Gestor
44             self.gestor=Gestor()
45         #Crear un agente nuevo
46         def crearAgente(self):
47             agente_id=1
48             with open('agentes.json','r+') as f:

```

```

49     if os.path.getsize('agentes.json') > 0:
50         #print("No estoy vacío")
51         self.data = json.load(f)
52         f.seek(0)
53         llaves=list(self.data.keys())
54         lista = [int(x) for x in llaves]
55         lista.sort()
56         agente_id=int(llaves[len(lista)-1])+1
57     nuevo_agente={
58         'hostname': self.hostname.get(),
59         'ip': self.ip.get(),
60         'version': self.version.get(),
61         'puerto': self.puerto.get(),
62         'comunidad': self.comunidad.get()
63     }
64     self.data[str(agente_id)]=nuevo_agente
65     json.dump(self.data,f,sort_keys="True",indent=4)
66     messagebox.showinfo("Éxito", "Agente registrado exitosamente")

```

Esta es la clase hilo encargada de monitorear tanto los objetos mib del agente, como a los servidores configurados anteriormente.

monitor.py:

```

1  from tkinter import *
2  from PIL import Image, ImageTk
3  import threading, time, calendar
4  from agente import Agente
5  from SNMP import *
6  from notificador import Notificador
7  from logger import Logger
8  import rrdtool
9  from http_cliente import HTTPMonitor
10 from PDFM import PDF
11 from smtp_cliente import SENSOR
12 from Sensor_SSH import SSH
13 from ftp_cliente import FTPMonitor
14 from sensDNS import sensDNS
15
16 class Monitor(threading.Thread):
17     """docstring for Agente"""
18     def __init__(self, id_agente):
19         threading.Thread.__init__(self)
20         self.agente=Agente(id_agente)
21         self.data={}
22         self.top=Toplevel()
23         self.top.geometry("2100x800")
24         self.top.resizable(0,0)
25         self.b = Button(self.top, text="Salir", command=self.salir)
26         self.b.grid(row=1, column=2, sticky=W+E+N+S)
27         self.boton_pdf = Button(self.top, text="Generar reporte", command=self.
generar_pdf)
28         self.boton_pdf.grid(row=1, column=3, sticky=W+E+N+S)
29         self.frame=Frame(self.top, width = 550, height = 150, relief = 'raised',
borderwidth=3)
30         self.frame.grid(row = 1, column = 0, columnspan=1, sticky=W+E+N+S)
31         self.continuar=True
32         self.image=[""]*8
33         self.photo=[""]*8
34         self.label=[""]*8
35         self.umbralCPU, self.umbralRAM, self.umbralHDD=self.obtenerUmbrales()
36         self.notificacionCPU=self.notificacionRAM=self.notificacionHDD=False

```

```

37 self.noti=Notificador(self.agente.hostname)
38 self.log=Logger(self.agente.hostname)
39 self.mostrarInfo()
40 self.crearRRDs()
41 #sensores
42 self.monitorHTTP=HTTPMonitor(self.agente.ip,5000)
43 self.monitorSMTP=SENSOR()
44 #self.monitorSSH=SSH(self.agente.ip,"gabs","0001")
45 self.monitorFTP=FTPMonitor(self.agente.ip,"gabs","0001")
46 self.monitorDNS=sensDNS()
47 def run(self):
48     while self.continuar:
49
50         #Monitorear HTTP
51         self.monitorHTTP.actualizar()
52         #self.monitorHTTP.imprimir()
53         #Monitorear SMTP
54         self.monitorSMTP.scan_smtp()
55
56         #self.monitorSSH.SSHConnection()
57         self.monitorFTP.actualizar_SensorFTP()
58         self.monitorFTP.actualizar_SensorFTP_SFC()
59         self.monitorDNS.dnsServer('lostani.ex.net')
60
61
62         #Interfaces
63         self.graficarRRD('IF-MIB','ifInOctets','ifOutOctets',"interfaz","Tráfico
de red en interfaces ('ifInOctets','ifOutOctets')",2)
64         #ICMP ->Deprecated
65         self.graficarRRD('IP-MIB','icmpInMsgs','icmpOutMsgs',"icmp","Tráfico ICMP
('icmpInMsgs','icmpOutMsgs')",0)
66         #TCP
67         self.graficarRRD('TCP-MIB','tcpInSegs','tcpOutSegs',"tcp","Tráfico
segmentos TCP ('tcpInSegs','tcpOutSegs')",0)
68         #SNMP
69         self.graficarRRD('SNMPv2-MIB','snmpInPkts','snmpOutPkts',"snmp","Tráfico
SNMP ('snmpInPkts','snmpOutPkts')",0)
70         #####
71         #LOS ULTIMOS VALORES PARA CPU, RAM Y HDD, VARIAN PARA WINDOWS Y LINUX
72         #PARA WINDOWS LOS VALORES SON: 6,3,1
73         #PARA LINUX SON: 196608,1,36
74         #####
75         if self.agente.nombre_so=="Ubuntu":
76             #UDP
77             self.graficarRRD('UDP-MIB','udpInDatagrams','udpOutDatagrams',"udp","
Tráfico UDP Datagramas ('udpInDatagrams','udpOutDatagrams')",0)
78             #CPU
79             self.graficarCPU("HOST-RESOURCES-MIB","hrProcessorLoad","cpu","Uso del
Procesador",196608)
80             #RAM
81             self.graficarRAM("HOST-RESOURCES-MIB","hrStorageSize","hrStorageUsed","
hrStorageAllocationUnits","ram","Uso de RAM",1)
82             #HDD
83             self.graficarHDD("HOST-RESOURCES-MIB","hrStorageSize","hrStorageUsed","
hrStorageAllocationUnits","hdd","Uso de HDD",36)
84             elif self.agente.nombre_so=="Windows":
85                 #UDP
86                 self.graficarRRD('UDP-MIB','udpInDatagrams','udpOutDatagrams',"udp","
Tráfico UDP Datagramas ('udpInDatagrams','udpOutDatagrams')",0)
87                 #CPU
88                 self.graficarCPU("HOST-RESOURCES-MIB","hrProcessorLoad","cpu","Uso del

```

```

Procesador",6)
89     #RAM
90     self.graficarRAM("HOST-RESOURCES-MIB","hrStorageSize","hrStorageUsed","
hrStorageAllocationUnits","ram", "Uso de RAM",3)
91     #HDD
92     self.graficarHDD("HOST-RESOURCES-MIB","hrStorageSize","hrStorageUsed","
hrStorageAllocationUnits","hdd", "Uso de HDD",1)
93     elif self.agente.nombre_so=="MacOs":
94         #UDP
95         self.graficarRRD('UDP-MIB','udpInErrors','udpOutDatagrams',"udp","
Tráfico UDP Datagramas ('udpInDatagrams','udpOutDatagrams')",0)
96         #CPU
97         self.graficarCPU("HOST-RESOURCES-MIB","hrProcessorLoad","cpu","Uso del
Procesador",196608)
98         #RAM
99         self.graficarRAM("HOST-RESOURCES-MIB","hrStorageSize","hrStorageUsed","
hrStorageAllocationUnits","ram", "Uso de RAM",1)
100        #HDD->el 31 es tentativo
101        self.graficarHDD("HOST-RESOURCES-MIB","hrStorageSize","hrStorageUsed","
hrStorageAllocationUnits","hdd", "Uso de HDD",31)
102        elif self.agente.nombre_so=="Linux":
103            #UDP
104            self.graficarRRD('UDP-MIB','udpInDatagrams','udpOutDatagrams',"udp","
Tráfico UDP Datagramas ('udpInDatagrams','udpOutDatagrams')",0)
105            #CPU
106            self.graficarCPU("HOST-RESOURCES-MIB","hrProcessorLoad","cpu","Uso del
Procesador",1281)
107            #RAM
108            self.graficarRAM("HOST-RESOURCES-MIB","hrStorageSize","hrStorageUsed","
hrStorageAllocationUnits","ram", "Uso de RAM",1)
109            #HDD
110            self.graficarHDD("HOST-RESOURCES-MIB","hrStorageSize","hrStorageUsed","
hrStorageAllocationUnits","hdd", "Uso de HDD",31)
111
112            #Ahora actualizamos las imágenes
113            self.actualizarImagen(0,self.agente.hostname+"_"+self.agente.id_agente+"
_interfaz.png",2,0)
114
115            self.actualizarImagen(1,self.agente.hostname+"_"+self.agente.id_agente+"
_icmp.png",3,0)
116
117            self.actualizarImagen(2,self.agente.hostname+"_"+self.agente.id_agente+"
_tcp.png",2,1)
118
119            self.actualizarImagen(3,self.agente.hostname+"_"+self.agente.id_agente+"
_udp.png",3,1)
120
121            self.actualizarImagen(4,self.agente.hostname+"_"+self.agente.id_agente+"
_snmp.png",2,2)
122
123            self.actualizarImagen(5,self.agente.hostname+"_"+self.agente.id_agente+"
_cpu.png",2,3)
124
125            self.actualizarImagen(6,self.agente.hostname+"_"+self.agente.id_agente+"
_ram.png",3,3)
126
127            self.actualizarImagen(7,self.agente.hostname+"_"+self.agente.id_agente+"
_hdd.png",2,4)
128
129            time.sleep(5)
130            self.top.destroy()

```



```

131 def salir(self):
132     self.continuar=False
133 def generar_pdf(self):
134     PDF(self.agente.id_agente , self.monitorHTPP , self.monitorSMTP , self.monitorFTP ,
        self.monitorDNS , self.agente.hostname+"_"+self.agente.id_agente+"_cpu.png" ,
        self.agente.hostname+"_"+self.agente.id_agente+"_ram.png" , self.agente .
        hostname+"_"+self.agente.id_agente+"_hdd.png")
135 def obtenerUmbrales(self):
136     umbrales=[]
137     with open("umbrales.txt") as f:
138         for i, linea in enumerate(f):
139             umbrales.append(linea.split(":")[1])
140     return umbrales
141 def mostrarInfo(self):
142     imagen1=Image.open(self.agente.logo_so).resize((50,50),Image.ANTIALIAS)
143     photo1=ImageTk.PhotoImage(imagen1)
144     label1=Label(self.frame , image=photo1)
145     label1.image=photo1
146     label1.grid(row=1, column=1, sticky=W+E+N+S)
147     Label(self.frame , text="ID agente: "+self.agente.id_agente).grid(row=2,
        column=1, sticky=W)
148     Label(self.frame , text="Hostname: "+self.agente.hostname).grid(row=3, column
        =1, sticky=W)
149     Label(self.frame , text="IP: "+self.agente.ip).grid(row=4, column=1, sticky=W
        )
150     Label(self.frame , text="Comunidad: "+self.agente.comunidad).grid(row=5,
        column=1, sticky=W)
151     Label(self.frame , text="Versión SNMP: "+str((int(self.agente.version))+1)).
        grid(row=6, column=1, sticky=W)
152     Label(self.frame , text="Puerto: "+self.agente.puerto).grid(row=7, column=1,
        sticky=W)
153     Label(self.frame , text="Nombre SO: "+self.agente.nombre_so).grid(row=2,
        column=2, sticky=W)
154     Label(self.frame , text="Versión SO: "+self.agente.version_so).grid(row=3,
        column=2, sticky=W)
155     Label(self.frame , text="Número de interfaces: "+self.agente.num_interfaces).
        grid(row=4, column=2, sticky=W)
156     Label(self.frame , text="Ubicación física: "+self.agente.ubicacion).grid(row
        =5, column=2, sticky=W)
157     Label(self.frame , text="Contacto: "+self.agente.contacto).grid(row=6, column
        =2, sticky=W)
158 def crearRRDs(self):
159     #Para tr{afico de red:
160     crearRRDDos(self.agente.hostname+"_"+self.agente.id_agente+"_interfaz.rrd","
        N","1","60","1","1","100","100","COUNTER")
161     #Para tr{afico de IP
162     crearRRDDos(self.agente.hostname+"_"+self.agente.id_agente+"_icmp.rrd","N","
        1","60","1","1","100","100","GAUGE")
163     #Para tr{afico de TCP segmentos
164     crearRRDDos(self.agente.hostname+"_"+self.agente.id_agente+"_tcp.rrd","N","1
        ","60","1","1","100","100","GAUGE")
165     #Para tr{afico de Datagramas UDP
166     crearRRDDos(self.agente.hostname+"_"+self.agente.id_agente+"_udp.rrd","N","1
        ","60","1","1","100","100","COUNTER")
167     #Para tr{afico de SNMP
168     crearRRDDos(self.agente.hostname+"_"+self.agente.id_agente+"_snmp.rrd","N","
        1","60","1","1","100","100","COUNTER")
169     #Para CPU
170     crearRRDUno(self.agente.hostname+"_"+self.agente.id_agente+"_cpu.rrd","N","1
        ","60","1","1","100","100","GAUGE")
171     #Para Ram

```

```

172     crearRRDTres(self.agente.hostname+"_"+self.agente.id_agente+"_ram.rrd","N","
173     1","60","1","1","1","100","100","100","GAUGE")
174     #Para HDD
175     crearRRDTres(self.agente.hostname+"_"+self.agente.id_agente+"_hdd.rrd","N","
176     1","60","1","1","1","100","100","100","GAUGE")
177
178 def graficarRRD(self, grupo, oid1, oid2, archivo, header, numero):
179     ultimo=rrdtool.last(self.agente.hostname+"_"+self.agente.id_agente+"_"+
180     archivo+".rrd")
181     consulta=consultaSNMP(self.agente.comunidad, self.agente.ip, int(self.agente.
182     version), numero, grupo, oid1, oid2, int(self.agente.puerto))
183     if consulta[0]!="" or consulta[1]!="":
184         valor = "N:" + str(consulta[0]) + ':' + str(consulta[1])
185         #print (valor)
186         rrdtool.update(self.agente.hostname+"_"+self.agente.id_agente+"_"+archivo+
187         ".rrd", valor)
188         rrdtool.dump(self.agente.hostname+"_"+self.agente.id_agente+"_"+archivo+".
189         rrd", self.agente.hostname+"_"+self.agente.id_agente+"_"+archivo+".xml")
190         ret = rrdtool.graph(self.agente.hostname+"_"+self.agente.id_agente+"_"+
191         archivo+".png",
192             "--start", str(ultimo-100),
193             "--end", str(ultimo+100),
194             "--vertical-label=Bytes/s",
195             "--title="+header,
196             "DEF:in="+self.agente.hostname+"_"+self.agente.
197             id_agente+"_"+archivo+".rrd:in:AVERAGE",
198             "DEF:out="+self.agente.hostname+"_"+self.agente.
199             id_agente+"_"+archivo+".rrd:out:AVERAGE",
200             "LINE1:in#00FF00:In traffic",
201             "LINE1:out#0000FF:Out traffic")
202
203 def graficarCPU(self, grupo, oid1, archivo, header, numero):
204     umbral=int(self.umbralCPU)
205     ultimo=rrdtool.last(self.agente.hostname+"_"+self.agente.id_agente+"_"+
206     archivo+".rrd")
207     primero=rrdtool.first(self.agente.hostname+"_"+self.agente.id_agente+"_"+
208     archivo+".rrd")
209     consulta=consultav2SNMP(self.agente.comunidad, self.agente.ip, int(self.agente
210     .version), numero, grupo, oid1, int(self.agente.puerto))
211     if consulta!="":
212         valor = "N:" + str(consulta)
213         rrdtool.update(self.agente.hostname+"_"+self.agente.id_agente+"_"+archivo+
214         ".rrd", valor)
215         rrdtool.dump(self.agente.hostname+"_"+self.agente.id_agente+"_"+archivo+".
216         rrd", self.agente.hostname+"_"+self.agente.id_agente+"_"+archivo+".xml")
217         ret = rrdtool.graphv(self.agente.hostname+"_"+self.agente.id_agente+"_"+
218         archivo+".png",
219             "--start", str(ultimo-100),
220             "--end", str(ultimo+100),
221             #"--end", str(ultimo+200),
222             "--vertical-label=Porcentaje/s",
223             "--title="+header,
224             "--lower-limit", "0",
225             "--upper-limit", "100",
226             "DEF:usage="+self.agente.hostname+"_"+self.agente.
227             id_agente+"_"+archivo+".rrd:valor1:AVERAGE",
228             "CDEF:umbral"+str(umbral-20)+"_l=usage,"+str(umbral-20)
229             +",GT,usage,"+str(umbral-10)+"_LT,EQ,usage,0,IF",
230             "CDEF:umbral"+str(umbral-10)+"_l=usage,"+str(umbral-10)+"_GT,
231             usage,"+str(umbral)+"_LT,EQ,usage,0,IF",
232             "CDEF:umbral"+str(umbral)+"_l=usage,"+str(umbral)+"_GT,usage,"+
233             str(umbral+(100-umbral))+"_LT,EQ,usage,0,IF",

```

```

214         "AREA: usage#FFBB0077: Uso de CPU",
215         "VDEF: m=usage, LSLSLOPE",
216         "VDEF: b=usage, LSLINT",
217         "CDEF: avg=usage, POP, m, COUNT, *, b, +",
218         "CDEF: umbral"+str(umbral-20)+"=avg, "+str(umbral-20)+" "+str(
umbral-10)+" ,LIMIT",
219         "CDEF: umbral"+str(umbral-10)+"=avg, "+str(umbral-10)+" "+str(
umbral)+" ,LIMIT",
220         "CDEF: umbral"+str(umbral)+"=avg, "+str(umbral)+" "+str(umbral
+(100-umbral))+" ,LIMIT",
221         "VDEF: minUmbral"+str(umbral)+"=umbral"+str(umbral)+" ,FIRST",
222         "VDEF: last=usage, LAST",
223         "PRINT: last:%6.2lf %S",
224         "GPRINT: minUmbral"+str(umbral)+" : Se alcanzará el "+str(umbral)
+"%@ %c : strftime",
225         "LINE1: avg#FF9F00",
226         "LINE2: "+str(umbral-20),
227         "AREA: 10#FF000022::STACK",
228         "AREA: 10#FF000044::STACK",
229         "AREA: "+str(100-umbral)+"#FF000066::STACK",
230         "AREA: umbral"+str(umbral-20)+"#0077FF77: Tráfico de carga
mayor que el "+str(umbral-20)+" y menor que el "+str(umbral-10)+" por ciento
",
231         "AREA: umbral"+str(umbral-10)+"#00880077: Tráfico de carga mayor
que el "+str(umbral-10)+" y menor que el "+str(umbral)+" por ciento",
232         "AREA: umbral"+str(umbral)+"#FF000088: Tráfico de carga mayor que
el "+str(umbral)+" y menor que el "+str(umbral+(100-umbral))+" por ciento",
233         "AREA: umbral"+str(umbral-20)+"_1#0077FF77",
234         "AREA: umbral"+str(umbral-10)+"_1#00880077",
235         "AREA: umbral"+str(umbral)+"_1#FF000088")
236     valor=float(ret["print[0]"])
237     if valor>float(umbral):
238         if not self.notificacionCPU:
239             self.notificacionCPU=True
240             #self.noti.enviarCorreo(0, self.agente.hostname+"_"+self.agente.
id_agente+"_"+archivo+".png")
241             self.log.escribirLog(0)
242         else:
243             self.notificacionCPU=False
244
245     def graficarHDD(self, grupo, oid1, oid2, oid3, archivo, header, numero):
246         umbral=int(self.umbralHDD)
247         ultimo=rrdtool.last(self.agente.hostname+"_"+self.agente.id_agente+"_"+
archivo+".rrd")
248         primero=rrdtool.first(self.agente.hostname+"_"+self.agente.id_agente+"_"+
archivo+".rrd")
249         consulta=consultav5SNMP(self.agente.comunidad, self.agente.ip, int(self.agente
.version), numero, grupo, oid1, oid2, oid3, int(self.agente.puerto))
250         if consulta[0]!="" or consulta[1]!="" or consulta[2]!="":
251             valor = "N:" + str(consulta[0]) + ':' + str(consulta[1]) + ':' + str(
consulta[2])
252             rrdtool.update(self.agente.hostname+"_"+self.agente.id_agente+"_"+archivo+
".rrd", valor)
253             rrdtool.dump(self.agente.hostname+"_"+self.agente.id_agente+"_"+archivo+".
rrd", self.agente.hostname+"_"+self.agente.id_agente+"_"+archivo+".xml")
254             ret = rrdtool.graphv(self.agente.hostname+"_"+self.agente.id_agente+"_"+
archivo+".png",
255                                 "--start", str(ultimo-100),
256                                 "--end", str(ultimo+100),
257                                 "--vertical-label=Porcentaje/s",
258                                 "--title="+header,

```

```

259         "—lower-limit", "0",
260         "—upper-limit", "100",
261         "DEF: val1="+self.agente.hostname+"_"+self.agente.
id_agente+"_"+archivo+".rrd:size:AVERAGE",
262         "DEF: val2="+self.agente.hostname+"_"+self.agente.
id_agente+"_"+archivo+".rrd:used:AVERAGE",
263         "DEF: val3="+self.agente.hostname+"_"+self.agente.
id_agente+"_"+archivo+".rrd:all:AVERAGE",
264         "CDEF: totalHDD=val1, val3, *",
265         "CDEF: usoHDDporcentaje=val2, val3, *, 100, *, totalHDD, /",
266         "CDEF: usoHDD=val2, val3, *",
267         "CDEF: umbral"+str(umbral-20)+"_1=usoHDDporcentaje,"+str(umbral
-20)+" ,GT, usoHDDporcentaje,"+str(umbral-10)+" ,LT,EQ, usoHDDporcentaje ,0 ,IF",
268         "CDEF: umbral"+str(umbral-10)+"_1=usoHDDporcentaje,"+str(umbral
-10)+" ,GT, usoHDDporcentaje,"+str(umbral)+" ,LT,EQ, usoHDDporcentaje ,0 ,IF",
269         "CDEF: umbral"+str(umbral)+"_1=usoHDDporcentaje,"+str(umbral)+" ,
GT, usoHDDporcentaje,"+str(umbral+(100-umbral))+" ,LT,EQ, usoHDDporcentaje ,0 ,IF
",
270         "AREA: usoHDDporcentaje#FFBB0077:HDD en uso",
271         "VDEF:m=usoHDDporcentaje ,LSLSLOPE",
272         "VDEF:b=usoHDDporcentaje ,LSLINT",
273         "CDEF: avg=usoHDDporcentaje ,POP,m,COUNT,* ,b,+ ",
274         "CDEF: umbral"+str(umbral-20)+"=avg,"+str(umbral-20)+" ,",
+str(umbral-10)+" ,LIMIT",
275         "CDEF: umbral"+str(umbral-10)+"=avg,"+str(umbral-10)+" ,"+str(
umbral)+" ,LIMIT",
276         "CDEF: umbral"+str(umbral)+"=avg,"+str(umbral)+" ,"+str(umbral
+(100-umbral))+" ,LIMIT",
277         "VDEF: minUmbral"+str(umbral)+"=umbral"+str(umbral)+" ,FIRST",
278         "VDEF: last=usoHDDporcentaje ,LAST",
279         "PRINT: last:%6.2lf %S",
280         "GPRINT: minUmbral"+str(umbral)+" : Se alcanzará el "+str(umbral)
+"% @ %c : strftime",
281         "LINE1: avg#FF9F00",
282         "LINE3: "+str(umbral-20),
283         "AREA: 10#FF000022::STACK",
284         "AREA: 10#FF000044::STACK",
285         "AREA: "+str(100-umbral)+"#FF000066::STACK",
286         "AREA: umbral"+str(umbral-20)+"#0077FF77: Uso mayor que el "+str(
umbral-20)+" y menor que el "+str(umbral-10)+" por ciento",
287         "AREA: umbral"+str(umbral-10)+"#00880077: Uso mayor que el "+str(
umbral-10)+" y menor que el "+str(umbral)+" por ciento",
288         "AREA: umbral"+str(umbral)+"#FF000088: Uso mayor que el "+str(
umbral)+" y menor que el "+str(umbral+(100-umbral))+" por ciento",
289         "AREA: umbral"+str(umbral-20)+"_1#0077FF77",
290         "AREA: umbral"+str(umbral-10)+"_1#00880077",
291         "AREA: umbral"+str(umbral)+"_1#FF000088")
292     valor=float(ret["print[0]"])
293     if valor>float(umbral):
294         if not self.notificacionHDD:
295             self.notificacionHDD=True
296             #self.noti.enviarCorreo(2, self.agente.hostname+"_"+self.agente.
id_agente+"_"+archivo+".png")
297             self.log.escribirLog(2)
298         else:
299             self.notificacionHDD=False
300
301 def graficarRAM(self, grupo, oid1, oid2, oid3, archivo, header, numero):
302     umbral=int(self.umbralRAM)
303     ultimo=rrdtool.last(self.agente.hostname+"_"+self.agente.id_agente+"_"+
archivo+".rrd")

```

```

304 primero=rrdtool.first(self.agente.hostname+"_"+self.agente.id_agente+"_"+
archivo+".rrd")
305 consulta=consultav5SNMP(self.agente.comunidad,self.agente.ip,int(self.agente
.version),numero,grupo,oid1,oid2,oid3,int(self.agente.puerto))
306 if consulta[0]!="" or consulta[1]!="" or consulta[2]!="":
307     valor = "N:" + str(consulta[0]) + ':' + str(consulta[1]) + ':' + str(
consulta[2])
308     rrdtool.update(self.agente.hostname+"_"+self.agente.id_agente+"_"+archivo+
".rrd", valor)
309     rrdtool.dump(self.agente.hostname+"_"+self.agente.id_agente+"_"+archivo+
".rrd",self.agente.hostname+"_"+self.agente.id_agente+"_"+archivo+".xml")
310     ret = rrdtool.graphv(self.agente.hostname+"_"+self.agente.id_agente+"_"+
archivo+".png",
311         "--start",str(ultimo-100),
312         "--end",str(ultimo+100),
313         "--vertical-label=Porcentaje/s",
314         "--title="+header,
315         "--lower-limit","0",
316         "--upper-limit","100",
317         "DEF:val1="+self.agente.hostname+"_"+self.agente.
id_agente+"_"+archivo+".rrd:size:AVERAGE",
318         "DEF:val2="+self.agente.hostname+"_"+self.agente.
id_agente+"_"+archivo+".rrd:used:AVERAGE",
319         "DEF:val3="+self.agente.hostname+"_"+self.agente.
id_agente+"_"+archivo+".rrd:all:AVERAGE",
320         "CDEF:totalRAM=val1,val3,*",
321         "CDEF:usoRAMporcentaje=val2,val3,*,100,*,totalRAM,/ ",
322         "CDEF:usoRAM=val2,val3,*",
323         "CDEF:umbral"+str(umbral-20)+"_l=usoRAMporcentaje,"+str(umbral
-20)+" ,GT,usoRAMporcentaje,"+str(umbral-10)+" ,LT,EQ,usoRAMporcentaje,0,IF",
324         "CDEF:umbral"+str(umbral-10)+"_l=usoRAMporcentaje,"+str(umbral
-10)+" ,GT,usoRAMporcentaje,"+str(umbral)+" ,LT,EQ,usoRAMporcentaje,0,IF",
325         "CDEF:umbral"+str(umbral)+"_l=usoRAMporcentaje,"+str(umbral)+" ,
GT,usoRAMporcentaje,"+str(umbral+(100-umbral))+" ,LT,EQ,usoRAMporcentaje,0,IF
",
326         "AREA:usoRAMporcentaje#FFBB0077:En uso",
327         "VDEF:m=usoRAMporcentaje,LSLSLOPE",
328         "VDEF:b=usoRAMporcentaje,LSLINT",
329         "CDEF:avg=usoRAMporcentaje,POP,m,COUNT,*,b,+",
330         "CDEF:umbral"+str(umbral-20)+"=avg,"+str(umbral-20)+" ,"+str(
umbral-10)+" ,LIMIT",
331         "CDEF:umbral"+str(umbral-10)+"=avg,"+str(umbral-10)+" ,"+str(
umbral)+" ,LIMIT",
332         "CDEF:umbral"+str(umbral)+"=avg,"+str(umbral)+" ,"+str(umbral
+(100-umbral))+" ,LIMIT",
333         "VDEF:minUmbral"+str(umbral)+"=umbral"+str(umbral)+" ,FIRST",
334         "VDEF:last=usoRAMporcentaje,LAST",
335         "PRINT:last:%6.2lf %S",
336         "GPRINT:minUmbral"+str(umbral)+" : Se alcanzará el "+str(umbral)
+"%@ %c :strftime",
337         "LINE1:avg#FF9F00",
338         "LINE3:"+str(umbral-20),
339         "AREA:10#FF000022::STACK",
340         "AREA:10#FF000044::STACK",
341         "AREA:"+str(100-umbral)+"#FF000066::STACK",
342         "AREA:umbral"+str(umbral-20)+"#0077FF77:Uso mayor que el "+str(
umbral-20)+" y menor que el "+str(umbral-10)+" por ciento",
343         "AREA:umbral"+str(umbral-10)+"#00880077:Uso mayor que el "+str(
umbral-10)+" y menor que el "+str(umbral)+" por ciento",
344         "AREA:umbral"+str(umbral)+"#FF000088:Uso mayor que el "+str(
umbral)+" y menor que el "+str(umbral+(100-umbral))+" por ciento",

```

```

345         "AREA:umbral"+str(umbral-20)+"_1#0077FF77",
346         "AREA:umbral"+str(umbral-10)+"_1#00880077",
347         "AREA:umbral"+str(umbral)+"_1#FF000088")
348     valor=float(ret["print[0]"])
349     if valor>float(umbral):
350         if not self.notificacionRAM:
351             self.notificacionRAM=True
352             #self.noti.enviarCorreo(1,self.agente.hostname+"_"+self.agente.
353             id_agente+"_"+archivo+".png")
354             self.log.escribirLog(1)
355         else:
356             self.notificacionRAM=False
357
358     def actualizarImagen(self,index,archivo,fila,columna):
359         self.image[index]=Image.open(archivo).resize((400,200),Image.ANTIALIAS)
360         self.photo[index]=ImageTk.PhotoImage(self.image[index])
361         self.label[index]=Label(self.top,image=self.photo[index])
362         self.label[index].image=self.photo[index]
363         self.label[index].grid(row=fila,column=columna,sticky=W)

```

Este archivo contiene las funciones snmp necesarias para que se puedan monitorear los objetos de la mib.

SNMP.py:

```

1 from pysnmp.hlapi import *
2 import rrdtool
3 import time
4 #Funcion para obtener las interfaces
5 def getInterfaces(comunidad,host,version,puerto):
6     resultado=""
7     errorIndication,errorStatus,errorIndex,varBinds = next(getCmd(SnmpEngine(),
8     CommunityData(comunidad,mpModel=version),UdpTransportTarget((host,puerto)),
9     ContextData(),
10     ObjectType(ObjectIdentity('IF-MIB','ifNumber',0).addAsn1MibSource('file:///usr/share/snmp','http://mibs.snmplabs.com/asn1/@mib@'))))
11     if errorIndication:
12         print(errorIndication)
13     elif errorStatus:
14         print('%s at %s' %(errorStatus.prettyPrint(),
15         errorIndex and varBinds[int(errorIndex)-1][0] or '?'))
16     else:
17         for varBind in varBinds:
18             VarB=('='.join([x.prettyPrint() for x in varBind]))
19             resultado=VarB.partition('=')[2]
20     return resultado
21
22 #Funcion para obtener el estatus de una interfaz
23 def getStatus(comunidad,host,version,interfaz,puerto):
24     errorIndication,errorStatus,errorIndex,varBinds = next(getCmd(SnmpEngine(),
25     CommunityData(comunidad,mpModel=version),UdpTransportTarget((host,puerto)),
26     ContextData(),
27     ObjectType(ObjectIdentity('IF-MIB','ifAdminStatus',interfaz).
28     addAsn1MibSource('file:///usr/share/snmp','http://mibs.snmplabs.com/asn1/@mib@'))))
29     if errorIndication:
30         print(errorIndication)
31     elif errorStatus:
32         print('%s at %s' %(errorStatus.prettyPrint(),

```



```

30         errorIndex and varBinds[int(errorIndex) - 1][0] or '?'
31     ))
32     else:
33         for varBind in varBinds:
34             VarB=(' = '.join([x.prettyPrint() for x in varBind]))
35             resultado=VarB.partition(' = ')[2]
36         return resultado
37 #Funcion para obtener informacion del estado del dispositivo
38 def getInfo(comunidad, host, version, puerto):
39     resultado=[]
40     errorIndication, errorStatus, errorIndex, varBinds = next(getCmd(SnmpEngine(),
41         CommunityData(comunidad, mpModel=version), UdpTransportTarget((host, puerto)),
42         ContextData(),
43         ObjectType(ObjectIdentity('SNMPv2-MIB', 'sysName', 0).addAsn1MibSource('file:///usr/share/snmp', 'http://mibs.snmplabs.com/asn1/@mib@')),
44         ObjectType(ObjectIdentity('SNMPv2-MIB', 'sysDescr', 0).addAsn1MibSource('file:///usr/share/snmp', 'http://mibs.snmplabs.com/asn1/@mib@')),
45         ObjectType(ObjectIdentity('IF-MIB', 'ifNumber', 0).addAsn1MibSource('file:///usr/share/snmp', 'http://mibs.snmplabs.com/asn1/@mib@')),
46         #ObjectType(ObjectIdentity('SNMPv2-MIB', 'sysUpTime', 0).addAsn1MibSource('file:///usr/share/snmp', 'http://mibs.snmplabs.com/asn1/@mib@')),
47         ObjectType(ObjectIdentity('SNMPv2-MIB', 'sysLocation', 0).addAsn1MibSource('file:///usr/share/snmp', 'http://mibs.snmplabs.com/asn1/@mib@')),
48         ObjectType(ObjectIdentity('SNMPv2-MIB', 'sysContact', 0).addAsn1MibSource('file:///usr/share/snmp', 'http://mibs.snmplabs.com/asn1/@mib@'))))
49     if errorIndication:
50         print(errorIndication)
51     elif errorStatus:
52         print('%s at %s' % (errorStatus.prettyPrint(),
53             errorIndex and varBinds[int(errorIndex) - 1][0] or '?'
54         ))
55     else:
56         for varBind in varBinds:
57             VarB=(' = '.join([x.prettyPrint() for x in varBind]))
58             resultado.append(VarB.partition(' = ')[2])
59     return resultado
60 def getInfo2(comunidad, host, version, puerto, oids):
61     resultado=[]
62     for element in oids:
63         errorIndication, errorStatus, errorIndex, varBinds = next(getCmd(SnmpEngine(),
64             CommunityData(comunidad, mpModel=version), UdpTransportTarget((host, puerto)),
65             ContextData(), ObjectType(ObjectIdentity(element))))
66         if errorIndication:
67             print(errorIndication)
68         elif errorStatus:
69             print('%s at %s' % (errorStatus.prettyPrint(),
70                 errorIndex and varBinds[int(errorIndex) - 1][0] or '?'
71             ))
72         else:
73             for varBind in varBinds:
74                 VarB=(' = '.join([x.prettyPrint() for x in varBind]))
75                 resultado.append(VarB.partition(' = ')[2])
76     return resultado
77 #Funcion para hacer consultas version facil (entrada y salida)
78 def consultaSNMP(comunidad, host, version, interfaz, grupo, objeto1, objeto2, puerto):
79     resultado=[]

```

```

78 errorIndication , errorStatus , errorIndex , varBinds = next(getCmd(SnmpEngine() ,
    CommunityData(comunidad , mpModel=version) , UdpTransportTarget((host , puerto)) ,
    ContextData() ,
79     ObjectType( ObjectIdentity(grupo , objeto1 , interfaz) . addAsn1MibSource(' file :///
        usr / share / snmp' , ' http :// mibs . snmplabs . com / asn1 / @mib@' ) ) ,
80     ObjectType( ObjectIdentity(grupo , objeto2 , interfaz) . addAsn1MibSource(' file :///
        usr / share / snmp' , ' http :// mibs . snmplabs . com / asn1 / @mib@' ) ) ) )
81
82 if errorIndication :
83     print(errorIndication)
84     resultado.append("")
85     resultado.append("")
86 elif errorStatus :
87     print('%s at %s' % (errorStatus.prettyPrint() ,
88         errorIndex and varBinds[int(errorIndex) - 1][0] or '?')
89 ))
90 else :
91     for varBind in varBinds :
92         VarB=(' = '.join([x.prettyPrint() for x in varBind]))
93         resultado.append(VarB.partition(' = ')[2])
94     return resultado
95
96 #Funcion para hacer consultas a un solo objeto con referencias al nombre
97 def consultav2SNMP(comunidad , host , version , interfaz , grupo , objeto , puerto) :
98     resultado=""
99     errorIndication , errorStatus , errorIndex , varBinds = next(getCmd(SnmpEngine() ,
    CommunityData(comunidad , mpModel=version) , UdpTransportTarget((host , puerto)) ,
    ContextData() ,
100     ObjectType( ObjectIdentity(grupo , objeto , interfaz) . addAsn1MibSource(' file :///
        usr / share / snmp' , ' http :// mibs . snmplabs . com / asn1 / @mib@' ) ) ) )
101
102 if errorIndication :
103     print(errorIndication)
104 elif errorStatus :
105     print('%s at %s' % (errorStatus.prettyPrint() ,
106         errorIndex and varBinds[int(errorIndex) - 1][0] or '?')
107 ))
108 else :
109     for varBind in varBinds :
110         VarB=(' = '.join([x.prettyPrint() for x in varBind]))
111         resultado=VarB.partition(' = ')[2]
112     return resultado
113
114 #Funcion para hacer consultas de un objeto con el OID
115 def consultav3SNMP(comunidad , host , version , oid , puerto) :
116     resultado=""
117     errorIndication , errorStatus , errorIndex , varBinds = next(getCmd(SnmpEngine() ,
    CommunityData(comunidad , mpModel=version) , UdpTransportTarget((host , puerto)) ,
    ContextData() ,
118     ObjectType( ObjectIdentity(oid) ) ) )
119
120 if errorIndication :
121     print(errorIndication)
122 elif errorStatus :
123     print('%s at %s' % (errorStatus.prettyPrint() ,
124         errorIndex and varBinds[int(errorIndex) - 1][0] or '?')
125 ))
126 else :
127     for varBind in varBinds :
128         VarB=(' = '.join([x.prettyPrint() for x in varBind]))
129         resultado=VarB.partition(' = ')[2]

```



```

127     return resultado
128
129 #Funcion para hacer consultas de dos objetos con OID
130 def consultav4SNMP(comunidad, host, version, oid1, oid2, puerto):
131     resultado=[]
132     errorIndication, errorStatus, errorIndex, varBinds = next(getCmd(SnmpEngine(),
133         CommunityData(comunidad, mpModel=version), UdpTransportTarget((host, puerto)),
134         ContextData(),
135         ObjectType(ObjectIdentity(oid1)),
136         ObjectType(ObjectIdentity(oid2))))
137
138     if errorIndication:
139         print(errorIndication)
140     elif errorStatus:
141         print('%s at %s' % (errorStatus.prettyPrint(),
142             errorIndex and varBinds[int(errorIndex) - 1][0] or '?'))
143     else:
144         for varBind in varBinds:
145             VarB=(' = '.join([x.prettyPrint() for x in varBind]))
146             resultado.append(VarB.partition(' = ')[2])
147     return resultado
148
149 #Funcion para ram y HDD
150 def consultav5SNMP(comunidad, host, version, interfaz, grupo, objeto1, objeto2, objeto3,
151     , puerto):
152     resultado=[]
153     errorIndication, errorStatus, errorIndex, varBinds = next(getCmd(SnmpEngine(),
154         CommunityData(comunidad, mpModel=version), UdpTransportTarget((host, puerto)),
155         ContextData(),
156         ObjectType(ObjectIdentity(grupo, objeto1, interfaz).addAsn1MibSource('file:///usr/share/snmp', 'http://mibs.snmplabs.com/asn1/@mib@')),
157         ObjectType(ObjectIdentity(grupo, objeto2, interfaz).addAsn1MibSource('file:///usr/share/snmp', 'http://mibs.snmplabs.com/asn1/@mib@')),
158         ObjectType(ObjectIdentity(grupo, objeto3, interfaz).addAsn1MibSource('file:///usr/share/snmp', 'http://mibs.snmplabs.com/asn1/@mib@'))))
159
160     if errorIndication:
161         print(errorIndication)
162         resultado.append("")
163         resultado.append("")
164     elif errorStatus:
165         print('%s at %s' % (errorStatus.prettyPrint(),
166             errorIndex and varBinds[int(errorIndex) - 1][0] or '?'))
167     else:
168         for varBind in varBinds:
169             VarB=(' = '.join([x.prettyPrint() for x in varBind]))
170             resultado.append(VarB.partition(' = ')[2])
171     return resultado
172
173 #Funcion para crear el archivo .rrd con un valor
174 def crearRRDUno(nombre, inicio, step, tiempo, steps1, steps2, row1, row2, tipo):
175     ret=rrdtool.create(nombre, '--start', inicio, '--step', step,
176         "DS:valor1:" + tipo + ":" + tiempo + "U:U",
177         "RRA:AVERAGE:0.5:" + steps1 + ":" + row1,
178         "RRA:AVERAGE:0.5:" + steps2 + ":" + row2)
179     if ret:
180         print(rrdtool.error())
181
182 #Funcion para crear el archivo .rrd con dos valores

```

```

178 def crearRRDDos(nombre, inicio, step, tiempo, steps1, steps2, row1, row2, tipo):
179     ret=rrdtool.create(nombre, '--start', inicio, '--step', step,
180         "DS: in:"+tipo+": "+tiempo+":U:U",
181         "DS: out:"+tipo+": "+tiempo+":U:U",
182         "RRA:AVERAGE:0.5:"+steps1+": "+row1,
183         "RRA:AVERAGE:0.5:"+steps2+": "+row2)
184     if ret:
185         print(rrdtool.error())
186
187 def crearRRDTres(nombre, inicio, step, tiempo, steps1, steps2, steps3, row1, row2, row3,
188     tipo):
189     ret=rrdtool.create(nombre, '--start', inicio, '--step', step,
190         "DS: size:"+tipo+": "+tiempo+":U:U",
191         "DS: used:"+tipo+": "+tiempo+":U:U",
192         "DS: all:"+tipo+": "+tiempo+":U:U",
193         "RRA:AVERAGE:0.5:"+steps1+": "+row1,
194         "RRA:AVERAGE:0.5:"+steps2+": "+row2,
195         "RRA:AVERAGE:0.5:"+steps3+": "+row3)
196     if ret:
197         print(rrdtool.error())
198
199 """
200 crearRRD("prueba.rrd","N","1","60","1","1","100","100")
201 while 1:
202     consulta=consultav3SNMP("variation/virtualtable","10.100.71.200",1,1,"IF-MIB",
203         "ifOutOctets",1024)
204     valor = "N:" + str(consulta)
205     print(valor)
206     rrdtool.update('prueba.rrd', valor)
207     rrdtool.dump('prueba.rrd','prueba.xml')
208     ultimo=rrdtool.last('prueba.rrd')
209     ret = rrdtool.graph("prueba.png",
210         "--start",str(ultimo-100),
211         "--end","+100",
212         "--vertical-label=Bytes/s",
213         "DEF:valor1=prueba.rrd:valor1:AVERAGE",
214         "AREA:valor1#00FF00:In traffic")
215     if ret:
216         print(rrdtool.error())
217         time.sleep(300)
218 """#
219 """
220
221     ESTE ES UNA PRUEBA QUE HICE PARA VER SI GRAFICABA, Y SI GRAFICA :v
222     consulta3=0
223     crearRRD("prueba.rrd","-700","1","60","1","1","100","100")
224     while 1:
225         consulta=consultaSNMP('grupo4cm1','localhost',0,2,'IF-MIB','ifInOctets','ifOutOctets')
226         valor = "N:" + str(consulta[0]) + ':' + str(consulta[1])
227         print(valor)
228         rrdtool.update('prueba.rrd', valor)
229         rrdtool.dump('prueba.rrd','prueba.xml')
230         ret = rrdtool.graph("prueba.png",
231             "--start",-300,
232             "--end","+100",
233             "--vertical-label=Bytes/s",
234             "DEF:valor1=prueba.rrd:valor1:AVERAGE",
235             "DEF:valor2=prueba.rrd:valor2:AVERAGE",
236             "AREA:valor1#00FF00:In traffic",

```

```

236         "LINE1:valor2#0000FF:Out traffic")
237 if ret:
238     print (rrdtool.error())
239     time.sleep(300)
240 """
241
242 """
243
244     PRUEBA DE LAS FUNCIONES PARA VER SU FUNCIONAMIENTO
245
246
247 interfaces=getInterfaces('grupo4cm1','localhost',0)
248 print(interfaces)
249 estado=getStatus('grupo4cm1','localhost',0,2)
250 print(estado)
251 res=getInfo('grupo4cm1','localhost',1)
252 for element in res:
253     print(element)
254 consulta=consultaSNMP('grupo4cm1','localhost',0,2,'IF-MIB','ifInOctets','ifOutOctets')
255 for element in consulta:
256     print(element)
257 consulta2=consultaSNMP('grupo4cm1','localhost',0,0,'IP-MIB','icmpInMsgs','icmpOutMsgs')
258 for element in consulta2:
259     print(element)
260 consulta3=consultav2SNMP('grupo4cm1','localhost',0,'1.3.6.1.2.1.2.2.1.10.2')
261 print(consulta3)
262
263     LISTA DE LAS 5 OPCIONES CON LAS QUE VAMOS A TRABAJAR, QUIEN SE VAYA A
264     ENCARGAR DE HACER LAS CONSULTAS DEL MODO GRAFICO PUES SOLO HAY QUE MANDAR A
265     LLAMAR LA FUNCION CON LOS PARAMETROS SIGUIENTES
266
267 'IF-MIB','ifInOctets','ifOutOctets'
268 'IP-MIB','icmpInMsgs','icmpOutMsgs'
269 'TCP-MIB','tcpInSegs','tcpOutSegs'
270 'UDP-MIB','udpInDatagrams','udpOutDatagrams'
271 'SNMPv2-MIB','snmpInPkts','snmpOutPkts'
272 """#

```

Este programa, permite monitorear el servidor ssh de un agente.

Sensor-SSH.py:

```

1 #!/usr/bin/env python3
2 import paramiko
3 import sys
4 import keyboard
5 import array
6 import time
7
8 class SSH():
9     def __init__(self, host_address, username, password):
10         self.initTime = float(time.time())
11         self.totalTime = 0.0
12         self.stdout = {}
13
14         self.host_address = host_address
15         self.username = username
16         self.password = password
17         self.status = ""
18
19     def SSHConnection(self):

```

```

20 ssh = paramiko.SSHClient()
21 #ssh.load_system_host_keys()
22 ssh.set_missing_host_key_policy(paramiko.WarningPolicy())
23 try:
24     ssh.connect(self.host_address, 22, self.username, self.password)
25     print ("\n_____")
26     print ("\nConnection established")
27     #getNUmberofConnections(ssh)
28     stdin, self.stdout, stderr=ssh.exec_command('echo $SSH_CONNECTION')
29     if self.stdout:
30         linenum = 1
31         for line in self.stdout:
32             print ("Connection #%i: " % linenum + line)
33             linenum = linenum + 1
34     else:
35         self.status = "Down"
36         print ("Server Down")
37         sys.exit(1)
38     #END#getNUmberofConnections(ssh)
39
40     #getConnectionTime(ssh) CHECAR CUANDO SE DESCONECTE
41
42     #ENDgetConnectionTime(ssh) CHECAR CUANDO SE DESCONECTE
43
44     #getTrafficIN , Out(ssh)
45     stdin, self.stdout, stderr=ssh.exec_command('ifstat -t -i lo 1')
46     if self.stdout or stderr:
47         for line in self.stdout:
48             print (line)
49             if keyboard.is_pressed('q'):
50                 #print ("Key pressed")
51                 ssh.close()
52                 #getConnectionTime()
53                 self.finalTime = float(time.time())
54                 self.totalTime= float(self.finalTime - self.initTime)
55                 print("Time connected: " + str(self.totalTime))
56                 break
57             else:
58                 pass
59
60     else:
61         self.status = "Down"
62         print ("Server Down")
63         sys.exit(1)
64
65
66 except paramiko.ssh_exception as e:
67     print ("Login failed: %s" % e)
68     sys.exit(1)
69
70 SSHClient = SSH("localhost","bri","SWS0810")
71 SSHClient.SSHConnection()

```

Este programa, permite monitorear el servidor smtp de un agente.
smtp-cliente.py:

```

1 import smtplib, ssl, imaplib, poplib, email
2 import time, threading
3
4 class SENSOR():
5     def __init__(self):
6         self.smtp_server = "smtp.redes3.com"

```

```

7         self.sender_email = "smtp@redes3.com"
8         self.message = """"SMTP_SENSOR""""
9         self.smtp_imap_time = 0.0
10        self.smtp_pop_time = 0.0
11        self.imap_time = 0.0
12        self.pop_time = 0.0
13        self.imap_total = 0.0
14        self.pop_total = 0.0
15        self.status = "Up"
16
17    def scan_smtp(self):
18        receiver_email_imap = "persona2@redes3.com"
19        receiver_email_pop = "persona3@redes3.com"
20        # Primero checamos el rendimiento de imap
21        try:
22            smtp = smtplib.SMTP(self.smtp_server)
23            smtp_start = time.time()
24            smtp.sendmail(self.sender_email, receiver_email_imap, self.message)
25            smtp_end = time.time()
26            self.smtp_imap_time =(smtp_end - smtp_start)
27
28            imap_tries = 0
29            imap_start = time.time()
30            while(not popSCAN()):
31                #waiting for email to come to imap client
32                if(imap_tries > 4):
33                    break
34                imap_tries = imap_tries+1
35
36
37            imap_end = time.time()
38            self.imap_time =(imap_end - imap_start)
39            #total
40            self.imap_total = self.smtp_imap_time + self.imap_time
41
42        except:
43            self.smtp_imap_time = "SMTP down"
44            self.status = "SMTP Down"
45            #####
46
47        #Ahora, checamos el rendimiento de pop
48        try:
49            smtp_start = time.time()
50            smtp.sendmail(self.sender_email, receiver_email_pop, self.message)
51            smtp_end = time.time()
52            self.smtp_pop_time =(smtp_end - smtp_start)
53
54            pop_tries = 0
55            pop_start = time.time()
56            while(not imapSCAN()):
57                #waiting for email to come to pop client
58                if(pop_tries > 4):
59                    break
60                pop_tries = pop_tries+1
61
62            if(pop_tries > 4):
63                self.pop_total = "POP down"
64                self.pop_time = "POP down"
65                self.status = "POP Down"
66            else:
67                pop_end = time.time()

```

```

68         self.pop_time =(pop_end - pop_start)
69         #total
70         self.pop_total = self.smtp_pop_time + self.pop_time
71
72     smtp.quit()
73
74     except:
75         self.smtp_pop_time = "SMTP down"
76         self.status = "SMTP Down"
77
78
79
80
81 #Checa por el ultimo correo en el buzón:
82 #False: no ha encontrado el correo
83 #True: encontro el correo
84 def imapSCAN():
85     imap_server = "imap.redes3.com"
86     user = 'persona2'
87     password = '1234'
88
89     M = imaplib.IMAP4(imap_server)
90     M.login(user, password)
91     M.select()
92     typ, data = M.search(None, 'ALL')
93     for num in data[0].split():
94         typ, data = M.fetch(num, '(RFC822)')
95
96     if "SMTP_SENSOR" in str(data[0][1]):
97         M.store(last_mail_index, '+FLAGS', '\\Deleted')
98         M.expunge()
99         M.close()
100        M.logout()
101        return True
102    else:
103        M.close()
104        M.logout()
105        return False
106
107 #Checa por el ultimo correo en el buzón:
108 #False: no ha encontrado el correo
109 #True: encontro el correo
110 def popSCAN():
111     pop_server = "pop3.redes3.com"
112     user = 'persona3'
113     password = '1234'
114     return_value = False
115
116     pop_server = poplib.POP3(pop_server, 110)
117     pop_server.user(user)
118     pop_server.pass_(password)
119
120     numMessages = len(pop_server.list()[1])
121     server_msg, body, octets = pop_server.retr(numMessages)
122     for j in body:
123         try:
124             msg = email.message_from_string(j.decode("utf-8"))
125             msg_text = msg.get_payload()
126             if "SMTP_SENSOR" in msg_text:
127                 return_value = True
128             pop_server.dele(numMessages)

```

```

129         except:
130             pass
131
132     pop_server.quit()
133     return return_value

```

Este programa, permite monitorear el servidor http de un agente.
http-cliente.py:

```

1 #tutorial https://docs.python.org/3.4/library/http.client.html
2 import http.client
3 import ssl, time
4
5 class HTTPMonitor():
6     def __init__(self, url, puerto):
7         self.url=url
8         self.puerto=puerto
9         self.status="Down"
10        self.tiempo_respuesta="No disponible"
11        self.tam="No disponible"
12        self.ancho_banda="No disponible"
13    def actualizar(self):
14        try:
15            #print (self.url)
16            conn = http.client.HTTPSConnection(self.url, self.puerto, context=ssl.
_create_unverified_context())
17            start = time.time()
18            conn.request("GET", "/file")
19            r1 = conn.getresponse()
20            end=time.time()
21            #print(r1.getheaders())
22            #print("Status",r1.status)
23            #print("Reason",r1.reason)
24            data1=r1.read()
25            #print(data1)
26            self.tam=str(len(data1))
27            self.tiempo_respuesta=str(end-start)
28            self.status="Up"
29            self.ancho_banda=str((len(data1)/1000)/(end-start))
30            #print("-----")
31            #print("Tiempo de respuesta: "+self.tiempo_respuesta+" segundos")
32            #print("Tamaño bytes respuesta: "+str(tam)+" bytes")
33            #print("Ancho de banda de descarga: "++" Kb/s")
34            #print("Status: Up")
35        except Exception as e:
36            #print(e)
37            self.status="Down"
38            self.tiempo_respuesta="No disponible"
39            self.tam="No disponible"
40            self.ancho_banda="No disponible"
41    def imprimir(self):
42        print("Tiempo de respuesta: "+self.tiempo_respuesta+" segundos")
43        print("Tamaño bytes respuesta: "+self.tam+" bytes")
44        print("Ancho de banda de descarga: "+self.ancho_banda+" Kb/s")
45        print("Status: "+self.status)

```

Este programa, permite monitorear el servidor ftp de un agente.
ftp-cliente.py:

```

1 from os import system
2 from urllib.request import urlopen
3 from sys import argv

```

```

4 from time import asctime
5 import string
6 from ftplib import FTP
7 from threading import Timer
8 import time
9
10 TIMEOUT = 5;
11
12 class FTPMonitor():
13     def __init__(self, url, usr, pwd, arch_rem="/home/ftp/usuarioftp/archivo.txt",
14                 arch_loc="archivo-ftp.txt", direct="/home/ftp/usuarioftp/"):
15         self.url = url
16         self.usr = usr
17         self.pwd = pwd
18         self.status = "Down"
19         self.tiempo_respuesta = -1
20         self.respuesta = "No disponible"
21         self.arch_rem = arch_rem
22         self.arch_loc = arch_loc
23         self.direct = direct
24         self.conteo = -1
25
26     def actualizar_SensorFTP(self):
27         try:
28             ftp = FTP(self.url, self.usr, self.pwd, None, TIMEOUT)
29
30             inst_arch_l = open(self.arch_loc, 'wb')
31
32             start = time.time()
33             self.respuesta = ftp.retrbinary('RETR %s' % self.arch_rem, inst_arch_l.
34             write)
35             end = time.time()
36
37             self.tiempo_respuesta = (end-start)
38
39             self.status = "Up"
40
41             ftp.quit()
42         except Exception as e:
43             self.status = "Down"
44             self.tiempo_respuesta = -1
45             self.respuesta = "No disponible"
46
47             ftp.quit()
48
49     def actualizar_SensorFTP_SFC(self):
50         try:
51             ftp = FTP(self.url, self.usr, self.pwd, None, TIMEOUT)
52
53             #print(ftp.nlst('/home/ftp/usuarioftp/'))#Prueba del contenido de archivos
54             self.conteo = len(ftp.nlst(self.direct))
55             #print(self.conteo)#Prueba del valor de self.conteo
56
57             ftp.quit()
58         except Exception as e:
59             self.conteo = -1
60             ftp.quit()
61
62     def imprimir(self):
63         print ('Tiempo de respuesta: %.0f' % self.tiempo_respuesta)
64         print ('Respuesta: '+self.respuesta)

```



```

63     print ('Status: '+self.status)
64     #print ('Archivo remoto: '+self.arch_rem)#Descomentar si se desea visualizar
        el nombre del archivo remoto
65     #print ('Archivo local: '+self.arch_loc)#Descomentar si se desea visualizar
        el nombre del archivo local
66     print ('Conteo: %d' % self.conteo)
67
68 #Prueba de la clase#
69 #if __name__ == '__main__':
70 # monitorFTP = FTPMonitor('192.168.100.11','usuarioftp','usuarioftp','archivo.
        txt','hola2.txt','/home/ftp/usuarioftp/')
71 # monitorFTP.actualizar_SensorFTP()
72 # monitorFTP.actualizar_SensorFTP_SFC()
73 # monitorFTP.imprimir()
74 #Prueba de la clase#

```

Este programa, permite monitorear el servidor dns de un agente.

SensDNS.py:

```

1 import dns.resolver
2 import time
3 class sensDNS(object):
4
5     def __init__(self):
6         self.myAnswers = None
7         self.valor = ""
8         self.tiempo = 0.0
9         self.status = ""
10
11     def validate_ip(self,s):
12         a = s.split('.')
13         if len(a) != 4:
14             return False
15         for x in a:
16             if not x.isdigit():
17                 return False
18             i = int(x)
19             if i < 0 or i > 255:
20                 return False
21         return True
22
23     def dnsServer(self, domain, peticiones=2, port=53, address="127.0.0.1"):
24         smtp_start = time.time()
25
26
27
28         myResolver = dns.resolver.Resolver()
29         myResolver.port = int(port)
30         myResolver.nameservers = [str(address)]
31         try:
32             if self.validate_ip(str(domain)):#chechar si el es una ip o dominio
33                 for i in range(0,int(peticiones)):
34                     req = '.'.join(reversed(domain.split("."))) + ".in-addr.arpa"
35
36                     self.myAnswers = myResolver.query(str(req), "PTR")
37
38                     for rdata in self.myAnswers:
39                         self.valor = str(rdata)
40                         #print("Ip: " + str(domain) + "\nDomain: " + str(
41                             rdata))
42             else:
43                 for i in range(0,int(peticiones)):

```

```

42         self.myAnswers = myResolver.query(str(domain), "A")
43         for rdata in self.myAnswers:
44             self.valor = str(rdata)
45             #print("Domain: " + str(domain) + "\nIp: " + str(
rdata))
46         self.status = "up"
47
48     except Exception as e:
49         print(e)
50         print("Query failed")
51         self.valor = "down"
52         self.status = "down"
53
54     smtp_end = time.time()
55
56     self.tiempo = smtp_end-smtp_start
57
58
59     def getAnswer(self):
60         return self.valor
61
62     def getTiempo(self):
63         return self.tiempo
64
65     def getStatus(self):
66         return self.status
67
68
69 #ip = input("Introduce la ip o el dominio\n")
70
71 #obj = sensDNS()
72 #obj.dnsServer(ip)
73 #print(obj.getAnswer())
74 #print(str(obj.getTiempo()))

```

Finalmente, este programa convierte toda la información monitoreada de un agente y sus servidores a un archivo pdf.

PDFM.py:

```

1 from reportlab.platypus import (SimpleDocTemplate, PageBreak, Image, Spacer,
2 Paragraph, Table, TableStyle)
3 from reportlab.lib.styles import getSampleStyleSheet, ParagraphStyle
4 from reportlab.lib.pagesizes import A4
5 from reportlab.lib import colors
6 from agente import Agente
7 from http_cliente import HTTPMonitor
8 #from Sensor_SSH import SSH
9 from ftp_cliente import FTPMonitor
10
11 class PDF():
12     def __init__(self, id_agente, http, smtp, ftp, dns, imagen_cpu, imagen_ram,
imagen_hdd):
13         self.http_monitor = http
14         self.smtp_monitor = smtp
15         #self.ssh_monitor=ssh
16         self.ftp_monitor=ftp
17         self.dns_monitor=dns
18
19         self.agente=Agente(id_agente)
20         doc = SimpleDocTemplate("Evaluacion5.pdf", pagesize = A4)
21         pdf=[]

```

```

22 estiloTabla = TableStyle([
23     ('GRID',(0,0),(-1,-1),0.5,colors.grey),
24     ('ALIGN',(0,0),(-1,-1),'CENTER'),
25     ('ALIGN',(0,0),(1,1),'LEFT'),
26     ('VALIGN',(0,0),(-1,-1),'MIDDLE'),
27     ('BOX',(0,0),(-1,-1),2,colors.black),
28 ])
29
30 estiloGraficas = TableStyle([
31     ('VALIGN',(0,0),(-1,-1),'MIDDLE'),
32 ])
33
34 estiloSensores = TableStyle([
35     ('GRID',(0,0),(-1,-1),0.5,colors.grey),
36     ('VALIGN',(0,0),(-1,-1),'MIDDLE'),
37     ('BOX',(0,0),(-1,-1),2,colors.black),
38 ])
39
40 estiloTexto = ParagraphStyle(name='encabezado',fontSize = 18)
41 Encabezado1 = Paragraph('Rendimiento de Servidor',estiloTexto)
42 Encabezado2 = Paragraph('Supervisión de Sensores',estiloTexto)
43
44 datosGenerales = (
45     ('No. de Equipo: 4', 'Integrantes\nHernández Pineda Miguel Angel\nMonroy
46     Martos Elioth\nOrta Cisneros Sabrina\nRamírez Centeno Hugo Enrique\nSaldaña
47     Aguilar Andrés Arnulfo\nZuñiga Hernández Carlos', 'Fecha\n08/05/2019'),
48     ('Sistema Operativo:'+self.agente.nombre_so, 'Tiempo de Actividad: VALUE',
49     'Numero de Interfaces:'+self.agente.num_interfaces))
50
51 graficaCPU= Image(imagen_cpu,width=200,height=100)
52 graficaRAM= Image(imagen_ram,width=200,height=100)
53 graficaHDD= Image(imagen_hdd,width=200,height=100)
54
55 datosGraficas =(
56     ('Uso de CPU',graficaCPU),
57     ('Uso de RAM',graficaRAM),
58     ('Uso de HDD',graficaHDD))
59
60 datosSensores = (
61     ('Sensor SMTP & IMAP/POP3', 'Tiempo de Respuesta SMTP-IMAP: '+str(self.
62     smtp_monitor.smtp_imap_time)+'\nTiempo de Respuesta IMAP: '+str(self.
63     smtp_monitor.imap_time)+'\nTiempo de Respuesta SUMA: '+str(self.smtp_monitor.
64     imap_total)+'\nTiempo de Respuesta SMTP-POP3: '+str(self.smtp_monitor.
65     smtp_pop_time)+'\nTiempo de Respuesta POP3: '+str(self.smtp_monitor.pop_time)
66     +'\nTiempo de Respuesta SUMA: '+str(self.smtp_monitor.pop_total)+'\nStatus: '+
67     str(self.smtp_monitor.status)),
68     ('Sensor HTTP', 'Tiempo de Respuesta: '+self.http_monitor.tiempo_respuesta
69     +'\nBytes Recibidos: '+self.http_monitor.tam+'\nAncho de Banda de descarga:
70     '+self.http_monitor.ancho_banda+' Kb/s\nStatus: '+self.http_monitor.status),
71     ('Sensor FTP', 'Tiempo de Respuesta: '+str(self.ftp_monitor.
72     tiempo_respuesta)+'\nRespuesta del Servidor: '+self.ftp_monitor.respuesta+'\n
73     Status: '+self.ftp_monitor.status),
74     ('Sensor FTP Server File Count', 'No. de Archivos: '+str(self.ftp_monitor.
75     conteo)),
76     ('Sensor DNS', 'Tiempo de Respuesta: '+str(self.dns_monitor.getTiempo())+'\n
77     Status: '+self.dns_monitor.getStatus()))
78     #('Sensor de Acceso Remoto', 'No. de Conexiones: '+self.ssh_monitor.
79     num_conexiones+'\nTiempo de Actividad: '+self.ssh_monitor.totalTime+'\n
80     Status: '+self.ssh_monitor.status))

```

```
66     TablaPrincipal = Table(data = datosGenerales , style = estiloTabla , colWidths
=180)
67     TablaGraficas = Table(data = datosGraficas , style = estiloGraficas ,
colWidths=210)
68     TablaSensores = Table(data = datosSensores , style = estiloSensores ,
colWidths=270)
69     pdf.append(TablaPrincipal)
70     pdf.append(Spacer(0,30))
71     pdf.append(Encabezado1)
72     pdf.append(Spacer(0,25))
73     pdf.append(TablaGraficas)
74     pdf.append(PageBreak())
75     pdf.append(Encabezado2)
76     pdf.append(Spacer(0,25))
77     pdf.append(TablaSensores)
78     doc.build(pdf)
```

Capítulo 4

Conclusiones

Hernández Pineda Miguel Angel:

Como pudimos observar, el rendimiento de los servicios varía mucho de acuerdo con las peticiones realizadas a cada uno de estos, no importando el tipo de servidor que se ocupe ya sea de información como HHTP o de archivos como FTP. Parte importante de ésta práctica fue el aprender a configurar diferentes tipos de servidores para entender su funcionamiento, además de que se monitoreó el rendimiento del mismo de acuerdo con las peticiones realizadas, ya que durante el ejercicio se montaron todos los servidores en una misma computadora, lo que nos permitió supervizar el rendimiento del CPU, la RAM y el HDD de la misma y las variaciones que se presentaban de acuerdo con el número de peticiones que se realizaban a la misma. Finalmente, uno de los ejercicios era llevar a cabo pruebas de estrés que nos permitieran conocer el límite de rendimiento de nuestros servicios, sin embargo este ejercicio no pudimos realizarlo debido a la falta de conocimientos sobre el control de las peticiones a los servidores mencionados.

Monroy Martos Elioth:

Para el desarrollo de está práctica, fue necesario investigar bastante para poder montar cada uno de los servicios solicitados, siendo el servicio de correo el que más trabajo costó, mientras que http y ssh fueron los dos más sencillos, en la práctica, montar un servidor de http puede ser posible de muchas formas, sin embargo, nos resultó más sencillo implementarlo en python, ya que es un entorno que conocemos. Además, el monitorear cada uno de estos servidores, de igual forma resultó complicado, ya que no resulta tan trivial con en un momento llegamos a pensar, se requiere tener conocimiento de cada uno de los servidores para saber que variables medir y como hacerlo, finalmente. el hacer las pruebas bajo estrés resulto aún más complicado ya que no teníamos conocimiento de que herramientas podíamos usar para poder llevar a cabo esta tarea.

Orta Cisneros Sabrina:

El desarrollo de esta práctica ayudó a conocer el rendimiento en situaciones ideales de diferentes servicios que un servidor puede tener, así como para saber cómo funcionan los protocolos que definen a cada uno de estos servicios sumamente importantes para nuestras actividades diarias al navegar por la web. Es muy conveniente monitorizar estos servicios pues así podemos saber, por ejemplo, cuántas solicitudes de cuántos clientes puede soportar y de esta manera evitar futuras fallas y caídas del servicio pues como se mencionó anteriormente, éstos son indispensables para las diversas acciones que se realizan a través de Internet.

Ramírez Centeno Hugo Enrique:

Los servicios en red son aplicaciones que se encuentran ejecutándose en diferentes equipos y prestan su servicio a través de la red, por lo que es importante poder instalar estos servicios y además monitorizarlos, lo cual fue el objetivo planteado al principio de esta práctica, mismo, el que cumplimos. Cabe mencionar que cada servicio tiene su complejidad para ser instalado, ya que dependiendo de la plataforma sea windows, linux o macOS aumentan la cantidad de pasos para instalar el servicio, otro factor influyente son los usuarios con permisos en los diferentes sistemas operativos, llegamos a la conclusión que es mas sencillo trabajar con un sistema operativo linux. En la parte de monitoreo, fue esencial un trabajo colaborativo para entender cada servicio y poder realizar una herramienta en python que pudiera monitorizar todo al mismo tiempo.

Saldaña Aguilar Andrés Arnulfo:

El monitoreo del rendimiento de los diferentes servicios en el servidor que dimos de alta fue sencillo gracias al uso de las bibliotecas que Python tiene, ya que tienen muchas funcionalidades ya implementadas para darles el propósito que se busca, el único problema que tuvimos fue hacer las pruebas de estrés, que consideramos serian sencillas de realizar y no nos dimos el tiempo necesario para implementarlas, por lo que no pudimos someter los servicios a pruebas para poder tener pruebas mas realistas del rendimiento de nuestros servicios.

Zuñiga Hernández Carlos:

El desarrollo de la práctica implicó familiarizarnos con distintos servicios, con la finalidad de poder medir su rendimiento en la siguiente práctica. Cada servicio contó con mediciones diferentes del desempeño de algunos de sus recursos y eso necesitó de una investigación sobre librerías de Python que nos facilitaran las conexiones a dichos servicios y la obtención de los índices devueltos por sus recursos. También, se notó el impacto de estos servicios en el desempeño del equipo en el que se encontraban alojados, lo cual es bastante útil para notar cuándo es necesario optimizar un servicio o cuándo está sucediendo un ataque o mal funcionamiento. No es trivial definir cuando estos servicios han alcanzado su límite o algún nivel de riesgo, pero la práctica sirve de mucho para adentrarnos en la monitorización de distintos servidores.

Referencias Bibliográficas

- [1] Capítulo 20. Protocolo SSH. (n.d.). Retrieved May 19, 2019, from <https://web.mit.edu/rhel-doc/4/RH-DOCS/rhel-rg-es-4/ch-ssh.html>
- [2] HTTP. (n.d.). Retrieved May 19, 2019, from <https://developer.mozilla.org/es/docs/Web/HTTPReferencias>
- [3] Protocolo de transferencia de archivos. (2019, May 06). Retrieved May 19, 2019, from <https://es.wikipedia.org/wiki/Protocolo-de-transferencia-de-archivos>
- [4] ¿Qué es DNS? – Introducción a DNS - AWS. (n.d.). Retrieved May 19, 2019, from <https://aws.amazon.com/es/route53/what-is-dns/>
- [5] ¿Qué es un Servidor SMTP? Qué significa y cómo usarlo para enviar email. (2019, January 09). Retrieved May 19, 2019, from <https://es.mailjet.com/blog/news/servidor-smtp/>