

**INFORME**  
**Protocolos de Comunicación**  
**Trabajo Práctico Especial**

**Proxy POP3 Filter**

Grupo 2: Matías Heimann(57503), Lóránt Mikolás (57347),  
Diego Bruno Cilla (57028) y Johnathan Katan(56653).

Vencimiento de la entrega: 06/11/2018 (16:00).

<b>1. Índice .....</b>	<b>1</b>
<b>2. Descripción detallada de los protocolos y aplicaciones desarrolladas.....</b>	<b>2</b>
<b>2.1. Aplicación pop3filter .....</b>	<b>2</b>
<b>2.2. Aplicación stripmime .....</b>	<b>3</b>
<b>2.3. Aplicación de management .....</b>	<b>3</b>
<b>2.4. Protocolo POP3FMP .....</b>	<b>4</b>
<b>2.5. Protocolo POP3 .....</b>	<b>11</b>
<b>2.6. Logging .....</b>	<b>11</b>
<b>3. Problemas encontrados durante el diseño y la implementación .....</b>	<b>12</b>
<b>4. Limitaciones de la aplicación .....</b>	<b>12</b>
<b>5. Posibles extensiones .....</b>	<b>13</b>
<b>6. Conclusiones .....</b>	<b>13</b>
<b>7. Ejemplos de prueba .....</b>	<b>14</b>
<b>7.1 Mensaje informativo de login al conectarse a través del proxy.....</b>	<b>14</b>
<b>7.2 Mensaje informativo de logout al desconectarse del proxy ....</b>	<b>14</b>
<b>7.3 Mensaje de error cuando se fuerza la terminación de un cliente.....</b>	<b>14</b>
<b>7.4 Filtro de mensajes con proceso externo .....</b>	<b>15</b>
<b>7.5 Ejemplo de pipelining .....</b>	<b>15</b>
<b>8. Guía de instalación .....</b>	<b>17</b>
<b>9. Instrucciones para la configuración .....</b>	<b>17</b>
<b>10. Ejemplos de configuración y monitoreo .....</b>	<b>17</b>
<b>11. Documento de diseño del proyecto .....</b>	<b>20</b>

## 2. Descripción detallada de los protocolos y aplicaciones desarrolladas.

En este apartado se explicarán los protocolos, aplicaciones desarrolladas y cómo interactúan entre sí .

### 2.1. Aplicación pop3filter

La aplicación pop3filter es un proxy que se encarga de transmitir las requests de un cliente hacia a un servidor pop, y de llevar las responses del servidor al cliente. La ventaja de conectarse al servidor mediante el proxy es que se pueden filtrar el contenido de los mails que se leen mediante el comando RETR, y además el proxy cuenta con la capacidad de pipelining para el caso en el que el servidor origen no lo soporte. El proxy permite la concurrencia de múltiples clientes utilizando sockets no bloqueantes. Se utiliza un socket pasivo que espera conexiones, y las acepta creando sockets activos. Los file descriptor de estos sockets son agregados al selector anexando un puntero a una estructura (pop3filter) que contiene toda la información de esa conexión (file descriptors, buffers, maquina de estados, parsers, etc.).

Los primeros estados se encargan de la resolución de nombres del servidor origen (en un thread aparte para evitar el bloqueo con getaddrinfo), y la conexión en sí misma con el servidor. El estado siguiente, HELLO, se encarga de leer el saludo del origen y escribirlo en el cliente. Después, el estado CAPA se encarga de escribir en el origen el comando CAPA, y de ver en la response si soporta pipelining. A continuación el flujo pasa constantemente, por los estados REQUEST (para leer la request del cliente y escribirla en el origen), RESPONSE (para leer la response del origen y escribirla en el cliente) y FILTER (para filtrar los mails). El estado FILTER es un paso intermedio dentro del estado RESPONSE, para el caso en el que el request fue un RETR válido), ya que se llega a ese estado después de leer la response del origen y se sale antes de escribir la response (ya transformada) en el cliente.

Al mismo tiempo se parsean los request para saber si la respuesta puede ser multilínea, y luego en caso de terminar siéndolo se parsean las responses para saber cuando se llegó al final de la respuesta. El hecho de saber cuando una request o una response término de leerse por completo sirve para controlar el flujo de la aplicación, y poder soportar el caso en el que se manden los bytes de uno en uno.

Para realizar el pipelining, en caso de que el origen no lo soporte, lo que se hace es mandar una sola request, y solo después de terminar de escribir la response en cliente, se escribe la próxima request en el servidor.

Para el caso en el que el origen soporte pipelining, se parsean todas las request y se guardan sus tipos en una lista encadenada, para que después de mandarlas todas juntas, pueda saberse cuántas y qué tipo de respuestas se esperan leer.

## 2.2. Aplicación stripmime

La aplicación stripmime tiene como propósito censurar algunos headers y el cuerpo de determinados media-types. El cuerpo de los media-types seleccionados a censurar los censura con un mensaje a elección el cual está configurado en la variable de entorno FILTER\_MSG. Los media-types a censurar están en otra variable de entorno llamada FILTER\_MEDIAS, cada media-type está separado con una coma del anterior.

Esta aplicación lee de entrada estándar (STDIN) el mensaje a censurar. Se lee en un buffer, y por cada byte de este se realiza una operación en función del contexto. Este contexto se delimita en una máquina de estados, en función del estado que pertenezca qué operación se hará.

El contexto consiste en una máquina de estados de headers, reconoce los headers, una máquina de estados de content-type, reconoce los content-type censurable, los multipart/ y los message. Una máquina de estados y un comparador de boundaries, la diferencia entre estos es que la máquina de estados reconoce los boundaries del header y el comparador encuentra los boundaries en el body. Además cuenta con flags indicadores de si el content-type es multipart, message o si debe ser censurado. También para los headers que causan problemas, se utilizan flags para saber que estos headers ya se pasaron, estos headers son, content-type, content-transfer-encoding, content-md5 y content-length, los últimos dos se suprimen ya que es recomendable no usarlo y el segundo se modifica en caso que hay que censurar el mensaje. El contexto cuenta con un buffer que busca reducir la cantidad de system calls realizadas, esto lo que hace es recopilar los bytes y luego los muestra todos juntos o los guarda en el contexto. Por último el contexto cuenta con 2 pilas, una de boundaries y otra de estados. La de boundaries se ocupa de guardar el orden en que se deben buscar los boundaries en el cuerpo y la pila de estados se ocupa de marcar la acción a realizar, se eligió una pila para saber qué acciones se realizaron previamente y así modificar el comportamiento dentro de un estado. En base a esto, la máquina de estados principal, es decir, la que maneja el flujo de la operación maneja cada estado y las acciones que se realizan en él.

Cabe aclarar que la estructura del mail se ve afectada, sin embargo, su semántica no sufre modificaciones. Se pueden afectar el orden de los headers y los casos de estos. También se ve afectado en el valor del header, estos pueden no tener el case que tienen en el mensaje original.

## 2.3. Aplicación de management

La aplicación de management permite obtener métricas y cambiar la configuración del *pop3filter* durante su ejecución. Se comunica con el pop3filter mediante el protocolo POP3FMP que corre sobre SCTP (el protocolo pop3fmp es explicado en detalle en la sección siguiente). Esto permitió usar las funciones específicas **sctp\_recvmesg** y **sctp\_sendmsg** con las que ya podíamos asumir que la información era enviada o recibida

en su totalidad. Por otro lado la comunicación se realizó mediante un stream de salida y uno de entrada.

En cuanto al funcionamiento, el cliente lee caracteres por entrada estándar hasta la presencia de un '\n', luego esos caracteres que conforman comandos son transformados a un paquete que cumple con el protocolo POP3FMP y que puede ser interpretado por el pop3filter. Una vez que el comando es recibido e interpretado por el pop3filter, se responde con otro paquete POP3FMP.

La aplicación de management tiene validación de los comandos escritos. En algunos casos más estricta que otros de manera que se pueda ver que cuando el pop3filter recibe un mensaje mal formado no finaliza su ejecución o presenta errores, sino que simplemente responde con un aviso de BAD REQUEST.

La máquina de estados se asegura de que el usuario tiene que estar autenticado a la hora de pedir métricas o exigir cambios de configuración. En este caso el el usuario y contraseña necesarios para ingresar al pop3filter son: 'root' y 'root' respectivamente.

Las métricas y opciones que se pueden solicitar son (se debe tener en cuenta que las métricas no implican a la aplicación de management, es decir, las conexiones concurrentes no contemplan las conexiones a management):

- número de conexiones concurrentes.
- número de accesos históricos.
- número de bytes transferidos.
- número de mails filtrados.
- mediatypes censurados (get y set).
- mensaje de reemplazo (get y set).
- comando de filtrado (get y set).

Otros comando disponibles son:

- user 'username' (para indicar el nombre de usuario en el proceso de autenticación).
- password 'password' (para indicar la contraseña en el proceso de autenticación).
- help (para pedir ayuda)
- exit (para finalizar la conexión).

## 2.4. Protocolo POP3FMP

### 2.4.1.i. Abstracto

El POP3FMP es un protocolo de nivel aplicación encargado de tareas de control, administración y obtención de métricas de un *proxy pop3*.

Fue definido en 2018 para la realización de un trabajo práctico de la materia de Protocolos de Comunicación. La última versión del protocolo es la '1.0'.

También aclaramos que si bien no se trata de un RFC propiamente dicho el objetivo fue explicar el protocolo de manera formal y clara de manera que pueda ser implementado.

### 2.4.2. ii. Propósito

El propósito de este protocolo de nivel aplicación es comunicar una aplicación cliente con un proxy pop3 para configurar las opciones de su funcionamiento y obtener métricas del mismo. El protocolo fue creado con la intención de que en la capa de transporte se utilice SCTP (Stream Control Transmission Protocol).

### **2.4.2. iii. Terminología**

pop3fmp

Las siglas indican *POP3 FILTER MANAGEMENT PROTOCOL*.

conexión

Un circuito virtual de la capa de transporte se establece entre dos programas con el fin de comunicación.

mensaje

La unidad básica de comunicación POP3FMP.

request

Un POP3FMP request como definido en la sección 3.

response

Un POP3FMP response como definido en la sección 4.

### **2.4.1. iv. Versión**

La versión actual del protocolo es la 1.0.

## **2.4.2. Convenciones De Notación y Flujo básico**

### **2.4.2. i. Reglas básicas**

OCTETO = cualquier secuencia de 8-bit de datos. Por ejemplo, 00000001.

STRING = una cadena de caracteres ASCII de longitud variable finalizada en un '\0'.

Se debe aclarar que siempre que se los pone separados por un espacio se lo hace por legibilidad en la documentación. Pero no se encuentran verdaderamente separados en el mensaje sino que los bytes se encuentran contiguos.

Los ejemplos de mensajes finalizados que pueden ser enviados se marcarán con letra **negrita**.

### **2.4.2. ii. Flujo de la Comunicación**

Para que el cliente y el servidor puedan empezar a comunicarse, se debe establecer primero un inicio de conexión. Este inicio de conexión involucra la autenticación del cliente. Una vez autenticado el cliente ante el servidor, se puede proceder a aprovechar los servicios ofrecidos por el protocolo POP3FMP. Los pasos para que un cliente pueda autenticarse ante el servidor son los siguientes, en orden ascendente:

- 1) El cliente debe enviar un request USER (ver sección 3.3) indicando el nombre de usuario.
- 2) El servidor debe responder con un response +OK, en caso de haber procesado el request exitosamente, o con un response -ERR, indicando que hubo un error al procesar el request.
- 3) En caso de haber recibido un response +OK por parte del servidor, el cliente debe enviar un request PASSWORD (ver sección 3.4) indicando la contraseña del usuario ingresado. Si el cliente recibió un response -ERR por parte del servidor, deberá volver a repetir el paso 1).
- 4) El servidor debe responder con un response +OK, en caso de que la autenticación del usuario sea exitosa, o con un response -ERR en caso de que haya fallado la autenticación del cliente.

Una vez completados estos pasos, el cliente puede realizar los requests detallados en la sección 3. Si no se han completado estos 4 pasos al inicio de la comunicación entre el cliente y el servidor, el servidor deberá responder con un response -ERR ante cualquier request que difiera de USER o PASSWORD.

#### **2.4.3. Request**

Un mensaje de request desde el cliente al proxy incluye en primer lugar el método que se desea aplicar, esto se representa con un OCTETO (binario). Estos son los OCTETOS con los que debe comenzar la primera línea:

00000000 indica el pedido de cierre y finalización de la conexión.

00000001 indica el pedido del número de conexiones concurrentes.

00000010 indica el pedido del número de accesos históricos.

00000011 indica el pedido del número de bytes transferidos.

00000100 indica el pedido de la cantidad de mail filtrados.

Desde la secuencia 00000101 hasta el 10000000 (contando sucesivamente) se reservaron los octetos para futuras adiciones al protocolo y no se encuentran usadas en esta versión.

10000000 indica un GET: el pedido de información por parte del cliente acerca de las opciones de configuración del proxy POP3.

10000001 indica un SET: el cambio de configuración de alguna de las opciones que el cliente puede configurar sobre proxy POP3.

10000010 indica USER: señala que a continuación en el mensaje se encuentra el usuario con el que se realizará la autenticación.

10000011 indica el PASSWORD: señala que a continuación en el mensaje se encuentra la contraseña del usuario que busca autenticarse.

Todo primer octeto del request es seguido por otro OCTETO. Este último indicará la versión a ser utilizada del protocolo. En caso de que el servidor proxy POP3 no la soporte y soporte una versión anterior esto será aclarado en el response (sección 4). A partir de allí el

intercambio de mensajes debe ocurrir usando la versión del protocolo que soportan ambos (cliente y servidor). Con lo explicado hasta ahora el mensaje se conforma de la siguiente manera:

OCTETO OCTETO donde el primero OCTETO indica el tipo de request y el segundo la versión del protocolo a utilizar. Para más información sobre las opciones de versión consultar la sección 5.0.

Por un lado para los primeros requests correspondientes al pedido de métricas de funcionamiento algunos ejemplos de mensajes serían:

**00000011 00000001** se está pidiendo el número de bytes transferidos indicando que la versión del protocolo a utilizar es la '1.0'.

**00000100 00000001** se está pidiendo la cantidad de mails filtrados.

Por otro lado, los requests GET, SET, USER y PASSWORD se aclara que son acompañados por más información como parte del mensaje de la forma que explicará en las siguientes subsecciones (3.i., 3.ii., 3.iii. y 3.iv.).

#### **2.4.3. i. GET**

Se adjunta al mensaje que se venía construyendo un OCTETO obteniendo:

OCTETO OCTETO OCTETO

El último indica la información sobre las opciones de configuración del proxy POP3 que se solicita y puede ser:

00000000 señala el pedido de los mediatypes que han sido configurados.

00000001 señala el pedido del mensaje de reemplazo que ha sido configurado.

00000010 señala el pedido del comando utilizado para el filtrado.

Entonces un ejemplo de mensaje sería:

**10000000 00000001 00000000** el primer OCTETO indica un GET, el segundo la versión del protocolo (en este caso '1.0') y por último el pedido de los mediatypes.

#### **2.4.3. ii. SET**

Se adjunta al mensaje que se venía construyendo un OCTETO y un STRING obteniendo:

OCTETO OCTETO OCTETO STRING

El tercer OCTETO indica la opción que se va a configurar. Puede tomar los siguiente valores:

00000000 indica que la opción que se desea afectar son los mediatypes a filtrar.

00000001 indica que la opción que se desea afectar es el mensaje de reemplazo.

00000010 indica que la opción que se desea afectar es el comandos utilizado para el filtrado.

El STRING indica el nuevo valor a guardar. Algunos de estos valores deben ser enviados de una manera específica que será aclarada a continuación.



Los mediatypes son enviados separados por ',' de la siguiente manera:  
"mediatype1,mediatype2,...,mediatypeN".

El mensaje de reemplazo simplemente se envía: "message".

El nombre del comando de filtrado también es simplemente enviado:  
"command".

Entonces un ejemplo de mensaje sería:

**10000001 00000001 00000001 "message"** el primer OCTETO indica un SET, el segundo la versión del protocolo (en este caso '1.0'), el tercero que la opción que se desea afectar es el mensaje de reemplazo y por último el STRING indica el nuevo mensaje de reemplazo.

#### 2.4.3. iii. USER

Se adjunta al mensaje que se venía construyendo el nombre del usuario a autenticarse con un STRING obteniendo:

OCTETO OCTETO STRING donde el STRING indica el nombre del usuario.

Por lo tanto, un ejemplo de mensaje sería:

**10000011 00000001 "username"** el primer OCTETO indica USER, el segundo la versión del protocolo (en este caso '1.0') y por último el STRING indica el nombre de usuario "username".

#### 2.4.3. iv. PASSWORD

Se adjunta al mensaje que se venía construyendo la contraseña del usuario a autenticarse con un STRING obteniendo:

OCTETO OCTETO STRING donde el STRING indica la contraseña del usuario.

Por lo tanto, un ejemplo de mensaje sería:

**10000100 00000001 "password"** el primer OCTETO indica PASSWORD, el segundo la versión del protocolo (en este caso '1.0') y por último el STRING indica la contraseña "password".

#### 2.4.4. Response

Luego de recibir e interpretar el mensaje request, el servidor responde con un mensaje POP3FMP response.

Un mensaje de response desde el servidor al cliente incluye en primer lugar el método al que se responde esto se representa con un OCTETO (binario). Estos son los OCTETOS con los que debe comenzar la primera línea:

00000000 indica respuesta al cierre y finalización de la conexión.

00000001 indica respuesta al pedido de número de conexiones concurrentes.

00000010 indica respuesta al pedido del número de accesos históricos.

00000011 indica respuesta al pedido del número de bytes transferidos.

00000100 indica respuesta al pedido de la cantidad de mail filtrados.

Desde la secuencia 00000101 hasta el 10000000 (contando sucesivamente) se reservaron los octetos para futuras adiciones al protocolo y no se encuentran usadas en esta versión.

10000000 indica una respuesta a GET.

10000001 indica una respuesta a SET.

10000010 indica una respuesta a USER.

10000011 indica una respuesta al PASSWORD.

11111101 indica VERSION NOT SUPPORTED: señala que el servidor no soporta la versión pedida por el cliente.

1000101 indica PERMISSION DENIED: señala que el usuario debe autenticarse para hacer uso de ese comando.

11111111 indica una BAD REQUEST: señala que el mensaje request recibido no cumple con el protocolo.

Todo primer octeto del response es seguido por otro OCTETO. Este último indicará la versión a ser utilizada del protocolo. A partir de allí el intercambio de mensajes debe ocurrir usando la versión del protocolo que soportan ambos (cliente y servidor). Con lo explicado hasta ahora el mensaje se conforma de la siguiente manera:

OCTETO OCTETO donde el primero OCTETO indica el tipo de response y el segundo la versión del protocolo a utilizar. Para más información sobre las opciones de versión consultar la sección 5.0.

En el caso de una PERMISSION DENIED, BAD REQUEST, CLOSE CONNECTION y VERSION NOT SUPPORTED el mensaje queda conformado por los dos OCTETOS. Un ejemplo sería:

**11111111 00000001** donde el primer OCTETO indica una BAD REQUEST y el segundo la versión del protocolo (en este caso '1.0')

Para el resto de los mensajes response se adjunta un STRING (exceptuando los casos mencionados previamente). El mensaje queda así conformado por:

OCTETO OCTETO STRING donde el STRING indica la respuesta propiamente dicha. Se enumeran a continuación en las siguientes secciones las respuestas aceptadas por el protocolo y su formato.

#### **2.4.4. i. Respuesta al Pedido de Número de Conexiones**

El STRING "number" indica el número de conexiones.

Por lo tanto un ejemplo de este tipo de response sería:

**00000001 00000001 "1234"** donde 00000001 indica una respuesta al pedido del número de conexiones, 00000001 la versión del protocolo, "1234" el número de conexiones.

#### **2.4.4. ii. Respuesta al Pedido del Número de Accesos Históricos**

El STRING "number" indica el número de conexiones.

Por lo tanto un ejemplo de este tipo de response sería:

**00000010 00000001 "1234"** donde 00000010 indica una respuesta al pedido del número de accesos históricos, 00000001 la versión del protocolo, "1234" el número de accesos históricos.

#### **2.4.4. iii. Respuesta al Pedido del Número de Bytes Transferidos**

El STRING "number" indica el número de bytes transferidos.

Por lo tanto un ejemplo de este tipo de response sería:

**00000011 00000001 "1240"** donde 00000011 indica una respuesta al pedido del número de bytes transferidos, 00000001 la versión del protocolo, "1240" el número de bytes transferidos.

#### **2.4.4. iv. Respuesta al Pedido de la Cantidad de Mails Filtrados**

El STRING "number" indica la cantidad de mails filtrados.

Por lo tanto un ejemplo de este tipo de response sería:

**00000100 00000001 "113213"** donde 00000100 indica una respuesta al pedido de la cantidad de mails filtrados, 00000001 la versión del protocolo, "113213" la cantidad de mails filtrados.

#### **2.4.4. v. Respuesta GET**

El STRING "respuesta" donde respuesta puede ser los mediatypes solicitados separados por ',', el mensaje de reemplazo o el comando de filtrado

Por lo tanto un ejemplo de este tipo de response sería:

**10000000 00000001 "mediatype"** donde 10000000 indica una respuesta a un GET, 00000001 la versión del protocolo, "+OK: mediatype" indica que el mediatype filtrados es "mediatype".

#### **2.4.4. vi. Respuesta a SET**

El STRING "+OK" indica que las configuraciones fueron correctamente cambiadas.

Por lo tanto un ejemplo de este tipo de response sería:

**10000001 00000001 "+OK"** donde 10000001 indica una respuesta a un SET, 00000001 la versión del protocolo, "+OK" indica que la configuración se realizó con éxito..

#### **2.4.4. vii. Respuesta a USER**

El STRING "+OK" indica que el usuario fue tomado correctamente. El STRING "-ERR" indica que el usuario no fue aceptado.

#### **2.4.4. viii. Respuesta a PASSWORD**

El STRING "+OK" indica que la contraseña fue aceptada correctamente.

El STRING "-ERR" indica que el par usuario y contraseña no son correctos.

Por lo tanto un ejemplo de este tipo de response sería:

**10000100 00000001 “-ERR”** donde 10000100 indica una respuesta al pedido de autenticación de contraseña, 00000001 la versión del protocolo, “-ERR” indica que el par usuario y contraseña no fueron aceptados.

Aclaración si el usuario vuelve a hacer uso de los comandos USER o PASSWORD (una vez que se encuentra logueado) su estado de autenticación será:

1. En caso de que se haya usado USER y fue aceptado: no autenticado pero con el user aceptado.
2. En caso de que se haya usado USER y no fue aceptado: no autenticado pero sin el USER aceptado.
3. En caso de que se haya usado PASSWORD y fue aceptado: autenticado.
4. En caso de que se haya usado PASSWORD y no fue aceptado: no autenticado, sin el USER aceptado.

#### 2.4.5. Versiones

La única versión soportada viene dada por el OCTETO 00000001 que indica la versión ‘1.0’ del protocolo.

Las sucesivas versiones se indicarán sumándole 1 al OCTETO.

La indicación de versión es fundamental para la escalabilidad del protocolo.

Se pueden censurar un número acotado de mediatypes.

### 2.5. Protocolo POP3

El pop3filter es un proxy del protocolo pop3. Para más información acerca del protocolo visitar: <https://www.ietf.org/rfc/rfc1939.txt>.

### 2.6. Logging

Tanto para los mensajes de error como para los mensajes informativos que surgen del servidor proxy se decidió diseñar los logs de una forma similar a como están diseñados en Dovecot, un servidor POP3 tradicional. La decisión de basar los diseños de nuestros logs en los de Dovecot surge de la necesidad de no reinventar la rueda y tomar ventaja de un diseño que ya fue validado y funciona.

Los logs van a ser escritos en la salida estándar en donde se esté ejecutando el servidor proxy. Hay 2 tipos de mensajes distintos, estos son:

- Mensaje informativo: Para informar de algún evento que pueda ser de interés pero que no es crítico. De mensajes informativos, hay dos tipos, de login y de logout.
  - Login: Se imprime a salida estándar cuando un usuario se autentica exitosamente frente al origin server. Este mensaje sirve para estar al tanto de quien está accediendo y usando nuestro proxy. El formato de este mensaje es “INFO: On <date> client logged in: ip=%a, user=%b”

donde <date> es la fecha cuando el cliente hizo log in, %a es el ip del cliente, y %b el nombre de usuario POP3 del cliente.

- Logout: Se imprime cuando el cliente envía un request QUIT al origin server. Este mensaje presenta un resumen de los accesos que realizó el cliente durante la sesión en la que estuvo activo, y el formato es el siguiente: "INFO: On <date> client logged out: ip=%a, user=%b, top=%c, retr=%d, dele=%e", donde <date> es la fecha cuando el cliente hizo log out, %a es el ip del cliente, %b el nombre de usuario POP3 del cliente, %c la cantidad de requests TOP que hizo el cliente, %d la cantidad de requests RETR que hizo el cliente y %e la cantidad de requests DELE que hizo el cliente.
- Mensaje de error: Para informar de algún evento importante que deba ser atendido o del cual deba ser importante estar al tanto. El formato de este tipo de mensajes es: "ERROR: On <date> <message>" donde <date> es la fecha cuando ocurrió el error y <message> el mensaje que describe error.

### 3. Problemas encontrados durante el diseño y la implementación.

En primer lugar, en cuanto al stripmime encontramos problemas a la hora de diseñar la máquina de estados que le da el flujo a la aplicación. Además tuvimos problemas para encontrar la solución al line folding cuando el contenido del header era importante, en otras palabras, continuar con la ejecución de la máquina de estados luego del \r\n. Por último se nos hizo difícil saber qué hacer con el header de content-transfer-encoding cuando aún no sabíamos cuál iba a ser el content-type.

En segundo lugar, al principio se había optado por un protocolo tipo texto por su fácil lectura pero luego esto se cambió por uno binario. Esto fue debido a que el parseo de los mensajes resultaba más sencillo en el caso binario y no era necesariamente el ser humano el que escribe los mensajes en el protocolo (en cuyo caso hubiera tenido más sentido que sea tipo texto). Otro problema encontrado fue no poder mantener tal cual la estructura del mensaje, es decir, no se mantienen estrictamente los casos de los headers y el orden de estos.

En tercer lugar, en la aplicación de management se encontraron dificultades para la implementación de del protocolo debido a la falta de detalle (al principio) en la descripción de algunos aspectos del mismo. Lo cual llevó al equipo al reescribir las especificaciones.

### 4. Limitaciones de la aplicación.

Algunas de las limitaciones de la aplicación son:

No se pueden atender a más de 1024 usuarios de forma concurrente.

El usuario y la contraseña para la autenticación se encuentra predefinido y no puede ser cambiado (usuario 'root' y contraseña 'root').

La única versión del protocolo **POP3FMP** soportada por management y por el pop3filter es la 1.

Las métricas son volátiles.

No se cuenta con un programa instalador.

## 5. Posibles extensiones.

Una posible extensión sería agregar un programa instalador.

También se podrían agregar métricas más detalladas con gráficas.

Investigación e implementación de otro selector.

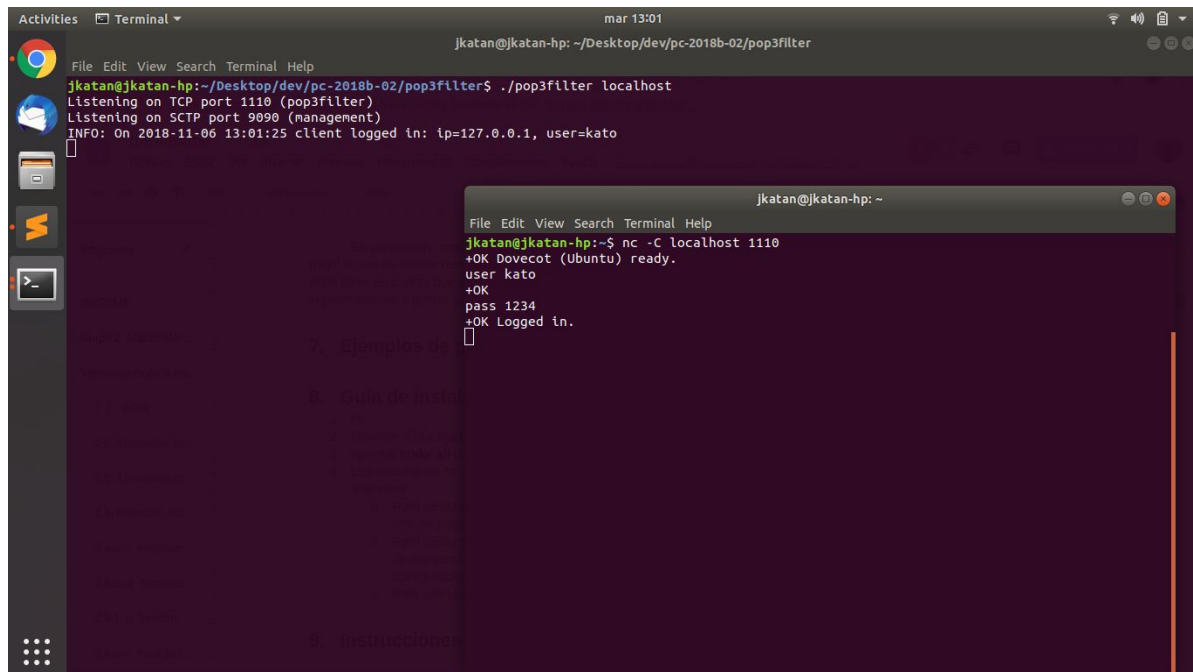
## 6. Conclusiones.

En conclusión, creemos que la realización de un pop3filter nos permitió entender mejor el uso de los sockets, protocolos como pop3, sctp y cómo idear un protocolo en sí. Se debe tener en cuenta que el trabajo realizado cuenta con limitaciones en cuanto su implementación y podría ser expandido y revisado en futuras versiones.

Para finalizar, vale la pena notar la complejidad que puede adquirir una arquitectura de este tipo.

## 7. Ejemplos de prueba.

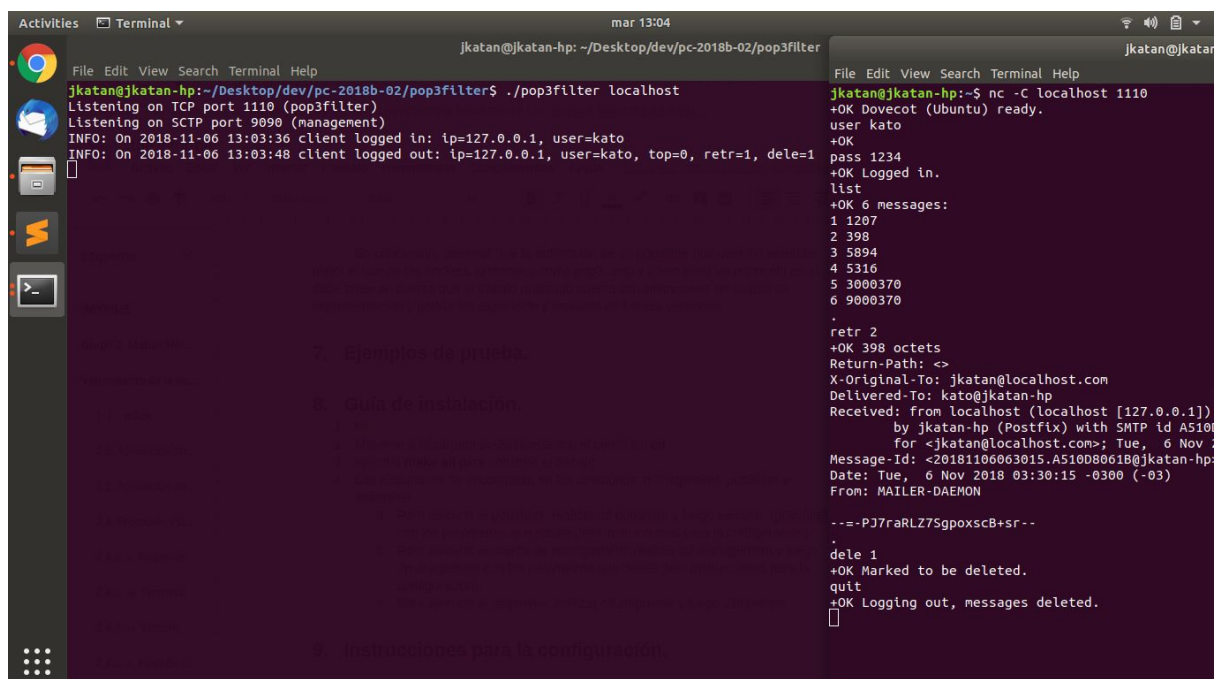
### 7.1 Mensaje informativo de login al conectarse a través del proxy



The screenshot shows two terminal windows. The background window, titled 'jkatan@jkatan-hp: ~/Desktop/dev/pc-2018b-02/pop3filter', is running the 'pop3filter localhost' command. It shows the service listening on TCP port 1110 and SCTP port 9090. An informational message is displayed: 'INFO: On 2018-11-06 13:01:25 client logged in: ip=127.0.0.1, user=kato'. The foreground window, titled 'jkatan@jkatan-hp: ~', shows a netcat connection to localhost 1110. The netcat client sends the command 'user kato', receives a '+OK' response, then sends 'pass 1234', and receives another '+OK' response indicating successful login.

En la terminal del fondo está corriendo el proxy, y desde otra terminal se conecta un cliente con netcat. Se puede observar el funcionamiento del mensaje de login, luego de que el usuario realiza exitosamente un request PASS.

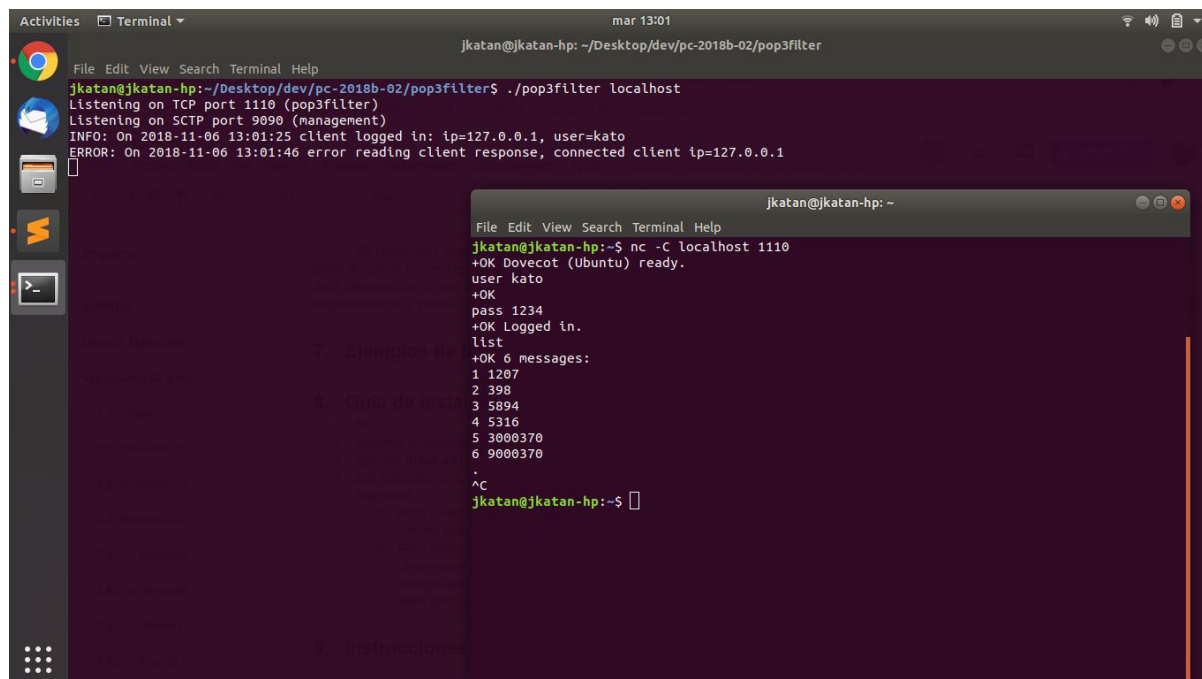
### 7.2 Mensaje informativo de logout al desconectarse del proxy



The screenshot shows two terminal windows. The background window, titled 'jkatan@jkatan-hp: ~/Desktop/dev/pc-2018b-02/pop3filter', shows the proxy service. It displays two informational messages: 'INFO: On 2018-11-06 13:03:36 client logged in: ip=127.0.0.1, user=kato' and 'INFO: On 2018-11-06 13:03:48 client logged out: ip=127.0.0.1, user=kato, top=0, retr=1, dele=1'. The foreground window, titled 'jkatan@jkatan-hp: ~', shows the netcat client's session. After the login sequence, the client sends 'list' and receives a '+OK 6 messages:' response with a list of message IDs. Then, the client sends 'retr 2', receives a '+OK 398 octets' response, and a detailed message header. Finally, the client sends 'quit', and the netcat client receives a '+OK Logging out, messages deleted.' response.

A la derecha se puede ver en la terminal del cliente la actividad a lo largo de la sesión, y a la izquierda el mensaje informativo presentando el resumen de la sesión correspondiente luego de que el cliente haya enviado un QUIT request.

## 7.3 Mensaje de error cuando se fuerza la terminación de un cliente



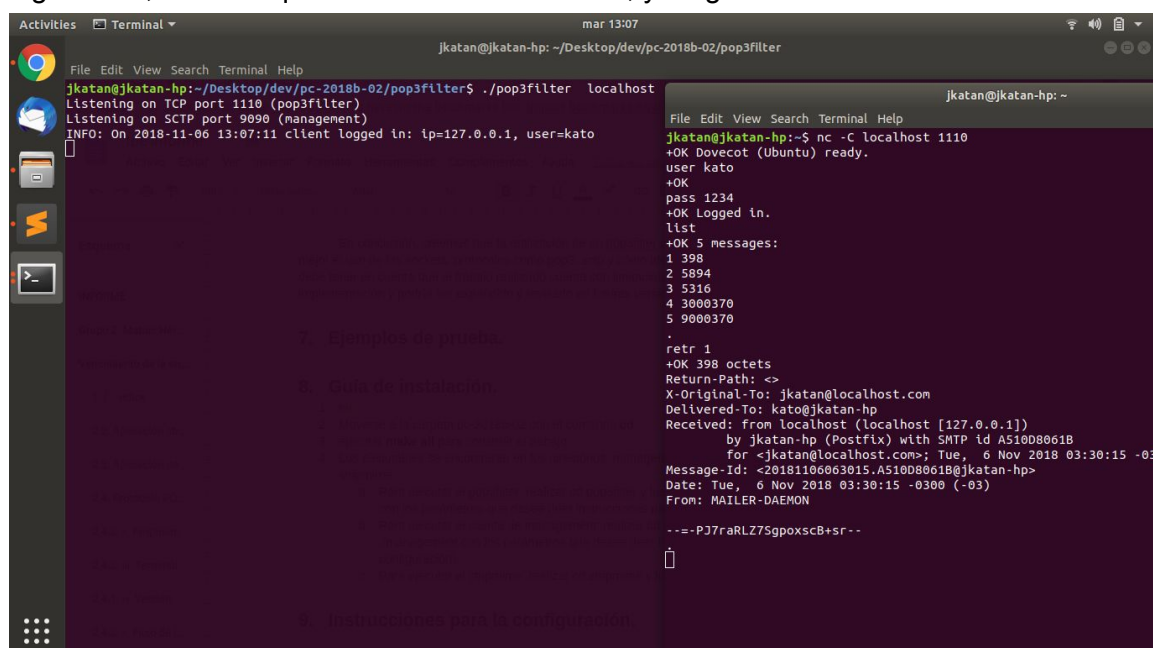
```
jkatan@jkatan-hp: ~/Desktop/dev/pc-2018b-02/pop3filter$ ./pop3filter localhost
Listening on TCP port 1110 (pop3filter)
Listening on SFTP port 9090 (management)
INFO: On 2018-11-06 13:01:25 client logged in: ip=127.0.0.1, user=kato
ERROR: On 2018-11-06 13:01:46 error reading client response, connected client ip=127.0.0.1

jkatan@jkatan-hp: ~$ nc -C localhost 1110
+OK Dovecot (Ubuntu) ready.
user kato
+OK
pass 1234
+OK Logged in.
list
+OK 6 messages:
1 1207
2 398
3 5894
4 5316
5 3000370
6 9000370
.
Ac
jkatan@jkatan-hp: ~$
```

Se puede ver el mensaje de error en el proxy luego de que se haya forzado la terminación de un cliente que estaba conectado. Notar que se provee el ip del cliente en el mensaje de error.

## 7.4 Filtro de mensajes con proceso externo

Primero, se ejecuta el servidor proxy sin pasarle ningún proceso de filtrado como argumento, es decir que usará cat como default, y luego se obtiene un mail con RETR:



```
jkatan@jkatan-hp: ~/Desktop/dev/pc-2018b-02/pop3filter$ ./pop3filter localhost
Listening on TCP port 1110 (pop3filter)
Listening on SFTP port 9090 (management)
INFO: On 2018-11-06 13:07:11 client logged in: ip=127.0.0.1, user=kato

jkatan@jkatan-hp: ~$ nc -C localhost 1110
+OK Dovecot (Ubuntu) ready.
user kato
+OK
pass 1234
+OK Logged in.
list
+OK 5 messages:
1 398
2 5894
3 5316
4 3000370
5 9000370
.
retr 1
+OK 398 octets
Return-Path: <>
X-Original-To: jkatan@localhost.com
Delivered-To: kato@jkatan-hp
Received: from localhost (localhost [127.0.0.1])
  by jkatan-hp (Postfix) with SMTP id A510D8061B
  for <jkatan@localhost.com>; Tue, 6 Nov 2018 03:30:15 -03
Message-Id: <20181106063015.A510D8061B@jkatan-hp>
Date: Tue, 6 Nov 2018 03:30:15 -0300 (-03)
From: MAILER-DAEMON

--P37raRLZ7SgpoXscB+sr--
```



Ahora, se ejecuta el proxy pero esta vez pasándole un proceso de filtrado, que elimina todas las líneas donde aparece “Date” tanto en el body como en el header del mail. Notar que al obtener el mismo mail con RETR, se ha efectuado correctamente el filtro:

```

jkatan@jkatan-hp: ~/Desktop/dev/pc-2018b-02/pop3filter
jkatan@jkatan-hp:~/Desktop/dev/pc-2018b-02/pop3filter$ ./pop3filter -t 'grep -i -v ^Date: ' localhost
Listening on TCP port 1110 (pop3filter)
Listening on SCTP port 9090 (management)
INFO: On 2018-11-06 13:07:53 client logged in: ip=127.0.0.1, user=kato

jkatan@jkatan-hp:~/Desktop/dev/pc-2018b-02/pop3filter$ nc -C localhost 1110
+OK Dovecot (Ubuntu) ready.
user kato
+OK
pass 1234
+OK Logged in.
list
+OK 5 messages:
1 398
2 5894
3 5316
4 3000370
5 9000370
.
retr 1
+OK 398 octets
Return-Path: <>
X-Original-To: jkatan@localhost.com
Delivered-To: kato@jkatan-hp
Received: from localhost (localhost [127.0.0.1])
        by jkatan-hp (Postfix) with SMTP id A510D8061B
        for <jkatan@localhost.com>; Tue, 6 Nov 2018 03:30:
Message-Id: <20181106063015.A510D8061B@jkatan-hp>
From: MAILER-DAEMON

---PJ7raRLZ7SgpoxscB+sr--

```

## 7.5 Ejemplo de pipelining

```

jkatan@jkatan-hp: ~/Desktop/dev/pc-2018b-02/pop3filter
jkatan@jkatan-hp:~/Desktop/dev/pc-2018b-02/pop3filter$ ./pop3filter localhost
Listening on TCP port 1110 (pop3filter)
Listening on SCTP port 9090 (management)
INFO: On 2018-11-06 13:28:58 client logged in: ip=127.0.0.1, user=kato
INFO: On 2018-11-06 13:28:58 client logged out: ip=127.0.0.1, user=kato, top=0, retr=1, dele=0

jkatan@jkatan-hp:~/Desktop/dev/pc-2018b-02/pop3filter$ printf 'USER kato\nPASS 1234\nLIST\nRETR 1\nQUIT\n' | nc localhost 1110
+OK Dovecot (Ubuntu) ready.
+OK
+OK Logged in.
+OK 5 messages:
1 398
2 5894
3 5316
4 3000370
5 9000370
.
+OK 398 octets
Return-Path: <>
X-Original-To: jkatan@localhost.com
Delivered-To: kato@jkatan-hp
Received: from localhost (localhost [127.0.0.1])
        by jkatan-hp (Postfix) with SMTP id A510D8061B
        for <jkatan@localhost.com>; Tue, 6 Nov 2018 03:30:15 -0300 (-03)
Message-Id: <20181106063015.A510D8061B@jkatan-hp>
Date: Tue, 6 Nov 2018 03:30:15 -0300 (-03)
From: MAILER-DAEMON

---PJ7raRLZ7SgpoxscB+sr--
+OK Logging out.
jkatan@jkatan-hp:~$

```

## 8. Guía de instalación.

1. Extraer el tar en la carpeta deseada.
2. Moverse a la carpeta pc-2018b-02 con el comando **cd**.
3. ejecutar **make all** para construir el trabajo.
4. Los ejecutables se encontraran en los directorios: management, pop3filter y stripmime.
  - a. Para ejecutar el pop3filter: realizar cd pop3filter y luego ejecutar ./pop3filter con los parámetros que desee (leer instrucciones para la configuración).
  - b. Para ejecutar el cliente de management: realizar cd management y luego ./management con los parámetros que desee (leer instrucciones para la configuración).
  - c. Para ejecutar el stripmime: realizar cd stripmime y luego ./stripmime.
5. Si se desean limpiar los archivos objeto moverse a la carpeta pc-2018b-02 y ejecutar: **make clean**.

## 9. Instrucciones para la configuración.

En cuanto a los parámetros que pueden ser pasados al pop3filter se pasan por línea de comandos: todos los parámetros posibles se pueden ver ejecutando ./pop3filter -h. Las configuraciones posibles son:

- e: para elegir el archivo de errores.
- l: para la dirección del POP3
- L: para la dirección del management
- m: para el mensaje de reemplazo
- M: para las media types a censurar
- o: para setear el puerto de management
- p: para setear el puerto local
- P: para setear el puerto de origen
- t: comando para transformaciones externas

Todos los comandos son parámetros de ./pop3filter, además se debe especificar la dirección del servidor POP3 al final.

Por último en cuanto a los parámetros de la aplicación de management: todos los parámetros posibles se pueden ver ejecutando ./management -h. Las configuraciones posibles son:

- L: para setear la dirección de management del pop3filter.
- o: para setear el puerto de management del pop3filter.

## 10. Ejemplos de configuración y monitoreo.

Authentication

```
matias@matias-Lenovo-V330-15IKB: ~/TP-Protos/management
matias@matias-Lenovo-V330-15IKB:~/TP-Protos/management$ ./management
Welcome to the management tool.
You can use it to get metrics or change the proxy pop3 filter settings.
Please enter a command and press enter.
help to ask for help
user root
+OK
pass root
Command is not valid.
password root
+OK
```

## Comando help

```
matias@matias-Lenovo-V330-15IKB: ~/TP-Protos/management
matias@matias-Lenovo-V330-15IKB:~/TP-Protos/management$ ./management
Welcome to the management tool.
You can use it to get metrics or change the proxy pop3 filter settings.
Please enter a command and press enter.
help to ask for help
user root
+OK
pass root
Command is not valid.
password root
+OK
help
*These are the available commands:
1 - 'concurrent connections'
2 - 'historical accesses'
3 - 'transferred bytes'
4 - 'filtered mails'
5 - 'get mediatypes'
6 - 'get replacement message'
7 - 'get filter command'
8 - 'set mediatypes <mediatype1,mediatype2,...,mediatypesN>'
9 - 'set replacement message <message>'
10 - 'set filter command <command>'
11 - 'user <username>'
12 - 'password <password>'
13 - 'help'
14 - 'exit'
```

## Varios comandos

```
matias@matias-Lenovo-V330-15IKB: ~/TP-Protos/management
matias@matias-Lenovo-V330-15IKB:~$ cd TP-Protos/management/
matias@matias-Lenovo-V330-15IKB:~/TP-Protos/management$ ./management
Welcome to the management tool.
You can use it to get metrics or change the proxy pop3 filter settings.
Please enter a command and press enter.
help to ask for help
user root
+OK
password root
+OK
help
These are the available commands:
1 - 'concurrent connections'
2 - 'historical accesses'
3 - 'transferred bytes'
4 - 'filtered mails'
5 - 'get mediatypes'
6 - 'get replacement message'
7 - 'get filter command'
8 - 'set mediatypes <mediatype1,mediatype2,...,mediatypesN>'
9 - 'set replacement message <message>'
10 - 'set filter command <command>'
11 - 'user <username>'
12 - 'password <password>'
13 - 'help'
14 - 'exit'
set replacement message eliminated
+OK
get replacement message
eliminated
concurrent connections
0
filtered emails
Command is not valid.
get filter command
cat
get mediatypes
text/plain,image/*,hola/hola,shas/asasd,adsadasuisd/duifasuduiasiu
set mediatypes image/*
+OK
set filter command stripmime/stripmime
+OK

```

## Exit

```
matias@matias-Lenovo-V330-15IKB: ~/TP-Protos/management
7 - 'get filter command'
8 - 'set mediatypes <mediatype1,mediatype2,...,mediatypesN>'
9 - 'set replacement message <message>'
10 - 'set filter command <command>'
11 - 'user <username>'
12 - 'password <password>'
13 - 'help'
14 - 'exit'
set replacement message eliminated
+OK
get replacement message
eliminated
concurrent connections
0
filtered emails
Command is not valid.
get filter command
cat
get mediatypes
text/plain,image/*,hola/hola,shas/asasd,adsadasuisd/duifasuduiasiu
set mediatypes image/*
+OK
set filter command stripmime/stripmime
+OK
help
These are the available commands:
1 - 'concurrent connections'
2 - 'historical accesses'
3 - 'transferred bytes'
4 - 'filtered mails'
5 - 'get mediatypes'
6 - 'get replacement message'
7 - 'get filter command'
8 - 'set mediatypes <mediatype1,mediatype2,...,mediatypesN>'
9 - 'set replacement message <message>'
10 - 'set filter command <command>'
11 - 'user <username>'
12 - 'password <password>'
13 - 'help'
14 - 'exit'
exit
Management client closing safely...
matias@matias-Lenovo-V330-15IKB:~/TP-Protos/management$
```

## **11. Documento de diseño del proyecto (que ayuden a entender la arquitectura de la aplicación).**

1. Archivo README.md en la carpeta pc-2018b-02.
2. Archivo .pdf de la presentación hecha en clase en la carpeta pc-2018b-02.