



LABORATORIO DE PROCESAMIENTO DE INFORMACIÓN EN APLICACIONES TELEMÁTICAS

Práctica 2.

Curso 2022/2023

Contenido

1	Objetivos.....	1
2	Descripción	1
2.1	Estadísticos generales	1
2.2	Estadísticos agregados por tipo de servidor y día.....	1
2.3	Estadísticos agregados por cuenta emisora.....	2
3	Realización	2
3.1	Restricciones de implementación.....	3
3.2	Ejemplo de ejecución.....	4
3.2.1	Ejemplo de salida de estadísticos generales.....	4
3.2.2	Ejemplo de salida de estadísticos agregados por tipo de servidor y día.....	4
3.2.3	Ejemplo de salida de estadísticos agregados por cuenta emisora.....	5
3.3	Estrategia de realización	5
4	Material disponible.....	6
5	Entrega.....	7
	ANEXO.....	8
	Arquitectura	8
	Ejemplos de trazas.....	9
1.	Mensaje entrante válido.....	10
2.	Mensaje entrante clasificado como spam	11
3.	Mensaje entrante con virus.....	12
4.	Mensaje entrante para cuenta inexistente	12
5.	Mensaje saliente válido.....	13
6.	Mensaje saliente clasificado como spam y/o con virus.....	13
7.	Mensaje saliente que no se puede entregar a la estafeta de destino.....	14
8.	Sobrecarga en alguno de los servidores de la cadena	14

1 Objetivos

El objetivo general de la práctica es aplicar los conocimientos adquiridos en las sesiones de teoría sobre expresiones regulares y su uso en la realización de aplicaciones programadas en lenguaje *java*.

El objetivo concreto es aplicar expresiones regulares para la extracción de información agregada e historizada a partir de los ficheros de *log* de un sistema de correo electrónico.

2 Descripción

Desarrollar una aplicación *java* que extraiga información agregada e historizada a partir de los ficheros de *log* registrados por un sistema de correo electrónico. La arquitectura del sistema de correo electrónico y el formato de los ficheros de *log* se encuentra descrita en el [Anexo](#).

La aplicación deberá obtener los estadísticos descritos a continuación y presentarlos por el dispositivo de salida estándar del ordenador donde se ejecute esta.

2.1 Estadísticos generales

Obtener los siguientes estadísticos:

- Para cada tipo de servidor, número de servidores de los que se han procesado trazas.
- Número de ficheros procesados.
- Número de trazas procesadas.
- Número de trazas con errores de formato.

En el apartado [3.2.1](#) puede ver un ejemplo de cómo se deben mostrar estos datos.

2.2 Estadísticos agregados por tipo de servidor y día

Por cada tipo de servidor se deberán obtener los estadísticos que se especifican a continuación:

- **msgIn**: número de mensajes entrantes en el sistema. Se considerarán todos los intentos de conexión desde el exterior a los servidores de entrada, independientemente de que estos puedan ser entregados a los destinatarios.
- **msgOut**: número de mensajes que se intentan entregar desde el sistema a destinatarios externos a la estafeta¹.
- **msgINFECTED**: número de mensajes clasificados como `INFECTED` (con virus).

¹ Si la cuenta de correo del destinatario es inválida o tiene algún problema, el sistema no lo puede entregar y reporta el fallo al remitente. Pero como ha sido un intento de entrega, este mensaje se considerará.

- **msgSPAM**: número de mensajes clasificados como SPAM que no han sido bloqueados.
- **code 4.3.2**: número de trazas con información de intentos de entrega con código de estado 4.3.2 (overload).
- **code 5.1.1**: número de trazas con información de intentos de entrega de mensajes entrantes al sistema con código de estado 5.1.1 (bad destination mailbox address).

Estos estadísticos estarán agrupados por tipo servidor, y de cada tipo, agrupados por días y mostrados ordenados por día.

Hay que tener en cuenta que hay estadísticos que no son aplicables a todos los tipos de servidores. Por ejemplo, el estadístico **msgIn** únicamente se puede obtener de los servidores **smtp-in**, mientras que es estadístico **code 4.3.2** se puede obtener de todos los servidores. En el apartado [3.2.2](#) puede ver un ejemplo de cómo se deben mostrar estos datos.

2.3 Estadísticos agregados por cuenta emisora

Obtener las cuentas de correo internas del sistema desde las que se haya intentado² enviar más de 500 mensajes. Para cada una de ellas se deberá indicar el **nombre del usuario** y el **número de mensajes** enviados. El listado deberá estar ordenado alfabéticamente por el nombre de usuario.

En el apartado [3.2.3](#) puede ver un ejemplo de cómo se deben mostrar estos datos.

3 Realización

La aplicación recibirá, a través de la línea de órdenes, un argumento (`pathEntrada`) indicando la ruta del directorio en el que se encuentren los ficheros de *log* y deberá realizar la siguiente secuencia de acciones:

- **Verificación y validación de los argumentos de entrada.** Si no recibe exactamente un argumento, o el argumento no se corresponde con el `path` de un directorio existente o no tienen permisos de lectura, la aplicación deberá finalizar de forma ordenada indicando la causa.
- **Procesamiento de los ficheros pertinentes.** El procesamiento deberá aplicarse únicamente a los ficheros del directorio `pathEntrada` que tengan extensión *log*.
- **Presentación de resultados.** Deberá presentar los estadísticos computados por el dispositivo de salida estándar del ordenador que la ejecuta.

² Si el mensaje tiene virus y/o es spam posiblemente no llegue a salir del sistema, pero el usuario lo ha intentado, por tanto este mensaje debe ser considerado.

3.1 Restricciones de implementación

- La aplicación debe ser multihilo. El hilo principal (el que se instancia al arrancar la aplicación y ejecuta el método `main()`) creará hilos, denominados trabajadores, que serán los encargados de procesar los ficheros de *log*. Cada hilo trabajador se debe encargar de procesar un único fichero de *log*.
- El número máximo de hilos trabajadores que pueden estar en ejecución se debe corresponder con el número de núcleos que disponga el ordenador donde se ejecuta la aplicación.
- La aplicación deberá implementarse mediante las siguientes clases `java` (se le proporcionará un esqueleto de estas clases) pertenecientes al paquete `piat.regExp`:
 - o **EstadisticasLog**. Clase inicial de la aplicación que contendrá el método estático `main()`. Esta clase no se debe instanciar, por lo que todos los métodos y atributos deben definirse como *static*. Realizará las siguientes acciones:
 1. Instanciar los recursos (por ejemplo las colecciones) que compartirá con los hilos trabajadores.
 2. Crear un hilo trabajador por cada fichero de *log* a procesar, pasándole como argumentos al instanciarlo el fichero de *log* y los recursos compartidos.
 3. Esperar a que todos los hilos trabajadores terminen.
 4. Realizar el post-procesamiento que sea necesario con los datos obtenidos por los hilos trabajadores.
 5. Mostrar la información descrita en el apartado 2.
 - o **Trabajador**. Clase que implementa el comportamiento del hilo trabajador. Se encargará de procesar el fichero de *log* que recibe como parámetro en el constructor, dejando en los recursos compartidos los resultados.
- La aplicación deberá satisfacer las siguientes restricciones para el cálculo de los estadísticos:
 - o Únicamente se podrá tomar como evidencias el contenido de los ficheros de *log*, salvo para el estadístico general **Número de ficheros procesados**.
 - o Cada fichero deberá procesarse una única vez y por un único hilo.
 - o En los ficheros de *log* que se le da como ejemplo, todas las trazas de un mismo fichero indican el servidor que las ha generado, por tanto podría obtener el primer estadístico general a partir del nombre del fichero, o bien leyendo únicamente la primera traza de cada fichero. Pero ambos métodos no se considerarán correctos, pues la aplicación deberá estar preparado para leer ficheros de *log* cuyo nombre no contenga el nombre del servidor, o bien ficheros de *log* con trazas de distintos servidores.
 - o No se podrán definir clases auxiliares ni utilizar ficheros temporales para el almacenamiento de información, solo se podrá hacer uso de colecciones

(List, Map, Set, ...) que almacenen objetos de clases *Wrapper* como String, Integer, ...

- En el código de la aplicación no puede haber ninguna referencia al sistema de ficheros local donde se encuentran los ficheros de *log*. Debe por tanto poderse ejecutar en cualquier ordenador pasando el argumento oportuno que indica el camino donde están los ficheros de *log*.
- La codificación de los ficheros *java* debe ser UTF-8.
- En los dos ficheros *java* debe estar escrito, antes de la definición de la clase, el nombre del alumno en formato Javadoc (dentro del *tag* @author).

3.2 Ejemplo de ejecución

A continuación, se muestra un ejemplo de lo que se obtendría por la salida estándar al ejecutar la aplicación. Para poder realizar una explicación del contenido se ha dividido la salida en varios apartados. En moodle podrá encontrar un ejemplo de ejecución con la salida completa. Vea el apartado [Material disponible](#) donde se le explica dónde está este fichero de ejemplo y su nombre.

3.2.1 Ejemplo de salida de estadísticos generales

```
Servidores: security-in=3, smtp-out=2, msa=2, security-out=2, user-mailbox=2, smtp-in=2
Número de ficheros: 102
Número de líneas: 35775524
Líneas erróneas: 4
```

3.2.2 Ejemplo de salida de estadísticos agregados por tipo de servidor y día

A continuación, se muestran algunas líneas de la salida del programa para los estadísticos agregados, pues salvo en el servidor *msa*, que muestra los datos completos, del resto de servidores solo se muestran los datos de un día, por lo que los puntos suspensivos que aparecen indican que falta mostrar los datos del resto de los días para ese servidor. En Moodle encontrará el ejemplo completo.

```
Estadísticas agregadas:
msa:
    2020-02-20 - code 4.3.2:      356
    2020-02-21 - code 4.3.2:      377
    2020-02-22 - code 4.3.2:      361
security-in:
    2020-02-20 - code 4.3.2:      476
    2020-02-20 - code 5.1.1:      536
    2020-02-20 - msgINFECTED:    4293
    2020-02-20 - msgSPAM:       29080
...
security-out:
    2020-02-20 - code 4.3.2:      372
    2020-02-20 - msgINFECTED:    771
...
smtp-in:
    2020-02-20 - code 4.3.2:      401
    2020-02-20 - code 5.1.1:      412
    2020-02-20 - msgInput:       40203
...
smtp-out:
    2020-02-20 - code 4.3.2:      379
```

```
2020-02-20 - msgOut:          35797
...
user-mailbox:
2020-02-20 - code 4.3.2:      167
...
```

Este es un ejemplo de salida, la aplicación que tiene que realizar no debe obligatoriamente mostrar la información exactamente igual. Lo que si tienen que salir igual son los valores de cada estadístico.

3.2.3 Ejemplo de salida de estadísticos agregados por cuenta emisora

A continuación, se muestra un resumen de la salida del programa de los 7 primeros y 7 últimos usuarios que han enviado más de 500 mensajes, por lo que los puntos suspensivos que aparecen indican que faltan el resto de los usuarios. En moodle podrá encontrar la salida completa.

```
Estadísticas de usuarios que han enviado más de 500 mensajes:
alberto.gallego:      1335
alfonso.navarro:      875
alfonso.santana:      605
amalia.izquierdo:     633
amalia.padilla:       872
amalia.pons:          1291
ana.mateos:           1092
...
sergio.lozano:        1042
sonia.palacios:       1306
susana.aranda:        1056
susana.bravo:         1371
susana.gracia:        855
yolanda.torres:       658
yolanda.vila:         710
```

3.3 Estrategia de realización

A continuación, se le proponen unos pasos que podría realizar para llegar a la aplicación completa. No es necesario seguirlos, pero le puede ayudar a abordarla:

1. Empiece usando una herramienta en línea, como [regex101](https://regex101.com/), con la que probar las expresiones regulares que necesita usar en esta práctica, tomando como ejemplo de texto origen una línea sacada de un fichero de *log*. Una vez que haya obtenido todas las expresiones regulares que necesita, anótelas. Tenga en cuenta que no todos los ficheros de *log* le sirven para los estadísticos que se solicitan en la práctica, por ejemplo el estadístico **msgIn** únicamente es aplicable a los servidores **smtp-in**, por tanto solo lo podrá encontrar en los ficheros de *log* que han generado esos servidores.
2. Realice una aplicación en java que aplique cada expresión regular, obtenida en el paso anterior, sobre algún ejemplo de línea de un fichero de *log*, y verifique que la aplicación procesa correctamente la expresión regular pudiendo obtener la información de grupos en el resultado.
3. Amplie la aplicación anterior para que recoja como argumento un fichero de *log* y procese todas las líneas con una expresión regular concreta, mostrando la cuenta de líneas que casan con la expresión regular.

4. Una vez que tenga claro cómo hacer expresiones regulares y como realizarlas mediante un programa en java, empiece por hacer pruebas con la aplicación usando un directorio que contenga un solo fichero de *log*, teniendo en cuenta que el estadístico a obtener se encuentre en ese fichero de *log*.
5. A continuación, pruebe con un directorio que contenga un fichero de *log* de cada tipo de servidor, o dependiendo de la prueba, con varios ficheros de *log* de un solo tipo de servidor.
6. Complete el código que se le indica en los comentarios *//TODO* de los ficheros que se le proporcionan como esqueletos de las clases *EstadisticasLog.java* y *Trabajador.java*. Si quiere ejecutar la aplicación con un solo hilo trabajador, imponga el valor 1 en la variable *numDeNucleos*.
7. Use el depurador en el hilo trabajador para poder ver el contenido de cada mapa según se va ejecutando.

Todos los recursos compartidos entre el hilo principal y los hilos trabajadores tienen que ser *Thread Safe*, por ello en la aplicación de ejemplo se usa *ConcurrentHashMap* y *AtomicInteger*. Si hace buen uso de estos recursos no debería necesitar establecer regiones críticas.

4 Material disponible

En Moodle encontrará el siguiente material para la realización de la práctica:

- *FicherosDeLog.zip*. Los ficheros de *log* que deberá usar la aplicación.
- Ejemplo de aplicación, compuesta por las clases java *EstadisticasLog* y *Trabajador*. Tenga en cuenta:
 - Esta aplicación es totalmente funcional y al ejecutarse muestra una pequeña parte de la información que se pide en la práctica.
 - Su diseño se basa en el uso de las colecciones del documento *Practica 2. Clase de apoyo* disponible en Moodle.
 - Como ejemplo de estadísticas agregada muestra los mensajes bloqueados y los mensajes pasados al siguiente servidor, si bien estos no son los que se piden en esta práctica. Además muestra estos datos por tipo de servidor, cuando en la práctica se deben mostrar por tipo de servidor y día.
 - Todo el código que falta por desarrollar se indica en comentarios *//TODO*.
 - No es obligatorio usar esta aplicación ni seguir las recomendaciones del documento citado siempre y cuando haga una aplicación que cumpla con las restricciones especificadas en el apartado [3.1](#) y su salida sea correcta.
- *Salida.txt*. Ejemplo de lo que se debería mostrar por pantalla al ejecutar la aplicación. No es imprescindible que tenga este formato, pero sí que los datos sean los mismos.

5 Entrega

Con anterioridad al 13 de abril de 2023 a las 10:30 horas, deberá entregarse en Moodle, dentro de `Espacio para la entrega de la práctica 2`, los siguientes ficheros (sin comprimir):

- Los dos ficheros Java desarrollados con el nombre del alumno en formato Javadoc (dentro del *tag* `@author`) antes de la definición de la clase.
- Un fichero `txt` con la salida de la aplicación al ejecutarla sobre el conjunto de ficheros de *log* proporcionados.

ANEXO

Descripción general del sistema de correo electrónico

Arquitectura

La arquitectura del sistema de correo electrónico es la siguiente:

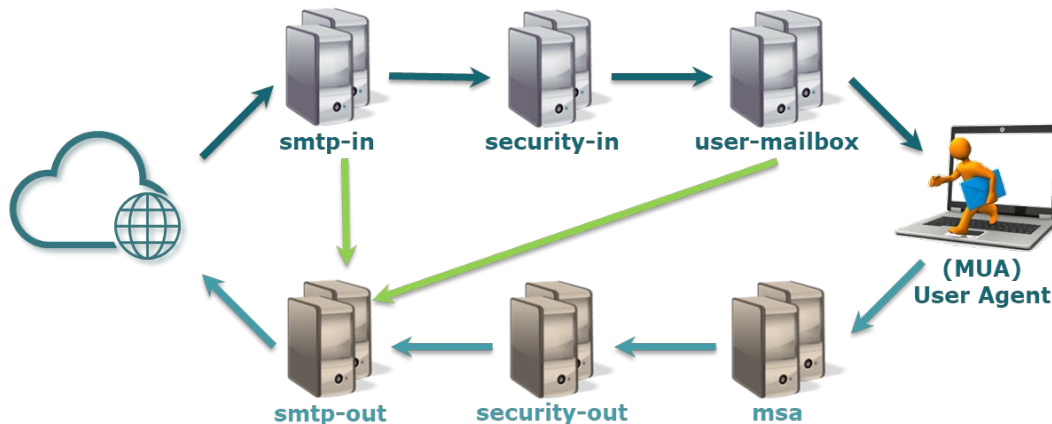


Imagen 1.- Arquitectura del sistema de correo electrónico

Los mensajes entrantes desde Internet son recibidos inicialmente por los servidores **smtp-in**, los cuales se los pasan a los servidores **security-in** que actúan de filtros de antispam y virus entrantes. Si estos filtros detectan un virus, descartan el mensaje y no lo reenvían. En otro caso, se los pasan a los servidores de almacenamiento de correos **user-mailbox**. Los mensajes que se consideren spam se etiquetan con una marca que sirve a los servidores **user-mailbox** para almacenarlos en la carpeta de posible spam de la cuenta del usuario.

Los servidores **security-in** generan mensajes automáticos de notificación de error para correos que no pueden llegar al destino. Esto sucede cuando intentan entregar un mensaje a un **user-mailbox** y éste responde que la dirección de destino es inexistente o bien que la cuenta de destino ya ha alcanzado su cuota máxima de espacio de almacenamiento de mensajes. En ese momento envía un mensaje a **smtp-in** y se notifica este error al remitente través de los servidores **smtp-out**.

Los usuarios del servicio de correo electrónico envían nuevos mensajes, y respuestas a los mensajes recibidos, por medio de los servidores **msa** (mail submission agent). Estos pasan los mensajes a los servidores **security-out** que actúan de filtros de spam y virus salientes. En el caso de que estos filtros detecten algún mensaje de spam o que contenga un virus, lo bloquean con objeto de salvaguardar la reputación del sistema. Finalmente, los mensajes son entregados a los **smtp-out** para que estos los envíen a la estafeta de destino.

Si un **smtp-out** no puede entregar un mensaje saliente (por ejemplo, porque la estafeta de destino dice que la dirección de destino es inexistente), genera una notificación de error automática para el remitente del mensaje.

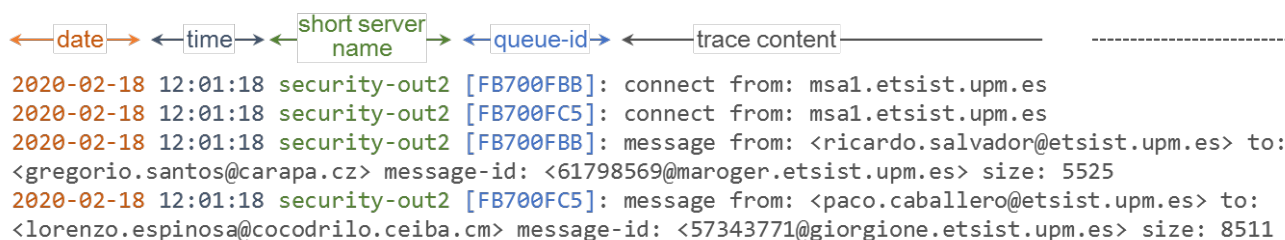
Los usuarios del servicio de correo electrónico (**MUA**) acceden a los mensajes almacenados en los buzones mediante los protocolos POP o IMAP, si bien esto no se contempla en los *logs* de esta práctica.

Cada uno de los bloques (**smtp-in**, **security-in**, **user-mailbox**, **msa**, **security-out**, **smtp-out**) que conforman el sistema, puede estar en una granja de servidores compuesta por una o más máquinas para cada servidor con reparto de carga entre ellos. Es decir, que podrá haber mas de un servidor de tipo **smtp-in**, **security-in**, **user-mailbox**, **msa**, **security-out** o **smtp-out**.

Ejemplos de trazas

A continuación, se muestran algunos ejemplos de trazas que aparecen en los ficheros de *logs* para los diferentes casos representativos de flujo de mensajes entre los servidores implicados.

En todas las trazas las dos primeras columnas indican la **fecha** y **hora** a la que se registró la traza, seguidamente aparece el **nombre** corto de la máquina que registró la traza, después entre corchetes el **identificador interno** (**queue_id**) del mensaje en esa máquina y finalmente aparece el **contenido** de la traza. En los ejemplos que se presentan en las siguientes secciones algunas trazas parecen ocupar varias líneas, pero eso es simplemente para que quepan en el cuadro de texto, pues en el fichero de trazas toda la información asociada a una traza se almacena en una única línea.



```

-----
<-->[date]<-->[time]<-->[short server name]<-->[queue-id]<-->[trace content]-----
2020-02-18 12:01:18 security-out2 [FB700FBB]: connect from: msa1.etsist.upm.es
2020-02-18 12:01:18 security-out2 [FB700FC5]: connect from: msa1.etsist.upm.es
2020-02-18 12:01:18 security-out2 [FB700FBB]: message from: <ricardo.salvador@etsist.upm.es> to:
<gregorio.santos@carapa.cz> message-id: <61798569@maroger.etsist.upm.es> size: 5525
2020-02-18 12:01:18 security-out2 [FB700FC5]: message from: <paco.caballero@etsist.upm.es> to:
<lorenzo.espinosa@cocodrilo.ceiba.cm> message-id: <57343771@giorgione.etsist.upm.es> size: 8511

```

Imagen 2.- Formato de las trazas en los ficheros de *log*

Normalmente el paso de un mensaje en el sistema deja varias líneas de traza para dicho mensaje en cada máquina por la que pasa. Además, esas líneas de traza normalmente se intercalan con las trazas de otros mensajes que están siendo procesados concurrentemente por la máquina. Por simplicidad, en ninguno de los siguientes ejemplos se intercalan varias trazas.

Obsérvese que solo se garantiza que un **queue_id** es único dentro de la máquina que lo ha generado y para el conjunto de mensajes en tránsito en ese momento por la máquina. Es decir, el mismo valor de **queue_id** podría aparecer en más de una máquina e incluso podría repetirse en la misma máquina, aunque nunca para dos mensajes en tránsito simultáneamente en la misma máquina.

Los mensajes tienen otro identificador único llamado `message-id` que se supone es único para toda Internet. En esta práctica no existen `message-id` repetidos, aunque en el mundo real no hay garantía de que un `message-id` sea verdaderamente único.

En las trazas relativas a la entrega o intento de entrega de mensajes a otras máquinas aparecen códigos de estado DSN (*Delivery Status Notification*) como 2.0.0 (*success*), 4.3.2 (*overload*) o 5.1.1 (*bad destination mailbox address*). Para más información sobre estos códigos, puede consultarse la [RFC 3463](#).

1. Mensaje entrante válido

La siguiente traza muestra el flujo de un mensaje válido enviado por `lottery@mirabobo.bh` para `diego.blanco@etsist.upm.es`.

En primer lugar, el mensaje es recibido por una de las máquinas de la granja `smtp-in`, en este caso `smtp-in1`, y se generan estas trazas:

```
2020-02-19 20:41:14 smtp-in1 [9A7ABF8E]: connect from: post4.mirabobo.bh
2020-02-19 20:41:14 smtp-in1 [9A7ABF8E]: message from: <lottery@mirabobo.bh> to:
<diego.blanco@etsist.upm.es> message-id: <88835047@haring.mirabobo.bh> size: 2978
2020-02-19 20:41:14 smtp-in1 [9A7ABF8E]: status=2.0.0 (accepted)
2020-02-19 20:41:14 smtp-in1 [9A7ABF8E]: relay to: security-in3.etsist.upm.es
2020-02-19 20:41:14 smtp-in1 [9A7ABF8E]: delivered, dsn: 2.0.0, queued as: DE82243C
```

La máquina `smtp-in1` genera un `queue_id` único en ese momento (9A7ABF8E) con el que etiqueta todas las trazas relativas al mensaje a su paso por `smtp-in1`. Estas trazas indican, en líneas diferentes:

- que el mensaje proviene de la máquina `post4.mirabobo.bh`.
- cuáles son el remitente y destinatario del mensaje, su `message_id`³ y su tamaño en bytes.
- que el mensaje ha sido aceptado por la estafeta `smtp-in1` con el código de estado 2.0.0.
- que el mensaje se va a reenviar a la máquina `security-in3.etsist.upm.es` para continuar su procesamiento.
- que se ha podido entregar el mensaje a la máquina en cuestión, la cual lo aceptó con el código de estado 2.0.0 y le asignó el `queue_id` DE82243C.

Obsérvese que, según las trazas, la máquina `smtp-in1` procesa y envía el mensaje a la siguiente etapa. En el fichero de *log* entre estas trazas habrá otras trazas intercaladas correspondientes a otros mensajes de entrada.

La siguiente etapa del mensaje es el paso por los filtros `antispam` y `antivirus` de entrada (servidores de la granja `security-in`). En el ejemplo analizado el mensaje se

³ Anteriormente se indicó que en esta práctica todos los `'message_id'` son únicos. Además, por simplicidad todos los mensajes tendrán un único destinatario.

ha enviado a la máquina `security-in3`, la cual genera las siguientes trazas sobre este mensaje:

```
2020-02-19 20:41:15 security-in3 [DE82243C]: connect from: smtp-in1.etsist.upm.es
2020-02-19 20:41:15 security-in3 [DE82243C]: message from: <lottery@mirabobo.bh> to:
<diego.blanco@etsist.upm.es> message-id: <88835047@haring.mirabobo.bh> size: 2978
2020-02-19 20:41:15 security-in3 [DE82243C]: status=2.0.0 (accepted)
2020-02-19 20:41:16 security-in3 [DE82243C]: SEC-PASSED: security-antivirus: CLEAN
security-antispam: HAM
2020-02-19 20:41:16 security-in3 [DE82243C]: relay to: user-mailbox1.etsist.upm.es
2020-02-19 20:41:16 security-in3 [DE82243C]: delivered, dsn: 2.0.0, queued as:
801169CD
```

Obsérvese que todas las trazas están etiquetadas con el `queue_id` `DE82243C`, que es el que registró la máquina anterior (`smtp-in1`) al entregar el mensaje a ésta. Las trazas tienen un significado análogo al visto para `smtp-in1` y únicamente hay una traza nueva que indica el resultado del análisis del filtro y si se ha dejado pasar el mensaje o no. En este caso, el mensaje ha pasado (`SEC-PASSED`), no se ha detectado ningún virus (`CLEAN`) y no se ha considerado que fuera `spam` (`HAM`).

Por último, el mensaje ha llegado a la máquina `user-mailbox1` que se le ha asignado el `queue_id` `801169CD`. Esta máquina ha registrado las siguientes trazas:

```
2020-02-19 20:41:17 user-mailbox1 [801169CD]: connect from: security-in3.etsist.upm.es
2020-02-19 20:41:17 user-mailbox1 [801169CD]: message from: <lottery@mirabobo.bh> to:
<diego.blanco@etsist.upm.es> message-id: <88835047@haring.mirabobo.bh> size: 2978
2020-02-19 20:41:17 user-mailbox1 [801169CD]: status=2.0.0 (accepted)
2020-02-19 20:41:17 user-mailbox1 [801169CD]: stored
```

La única traza novedosa es la última, que indica que se ha almacenado el mensaje en la cuenta del usuario.

2. Mensaje entrante clasificado como spam

Si el mensaje del ejemplo anterior hubiera sido clasificado como `spam`, todas sus trazas habrían sido análogas a las del ejemplo anterior, salvo la siguiente traza del filtro `security-antispam` que se produce en el servidor `security-in3`:

```
2020-02-19 20:41:16 security-in3 [DE82243C]: SEC-PASSED: security-antivirus: CLEAN
security-antispam: SPAM
```

Obsérvese que el servidor `security-in3` deja pasar el mensaje (`SEC-PASSED`), aunque le añade una cabecera o una marca especial para que el `user-mailbox` sepa que debe meterlo en la carpeta de posible `spam`, en lugar de la bandeja de entrada.

3. Mensaje entrante con virus

Si en un mensaje se hubiera detectado un virus, las trazas de la máquina `smtp-in` serían equivalentes a las del ejemplo anterior, pero en la máquina `security-in` se generarían estas trazas:

```
2020-02-19 20:55:42 security-in3 [DE8340BD]: connect from: smtp-in1.etsist.upm.es
2020-02-19 20:55:42 security-in3 [DE8340BD]: message from:
<ramona.lozano@okapi.nogal.so> to: <virginia.rivero@etsist.upm.es> message-id:
<98036692@rodriguez.okapi.nogal.so> size: 641431
2020-02-19 20:55:42 security-in3 [DE8340BD]: status=2.0.0 (accepted)
2020-02-19 20:55:43 security-in3 [DE8340BD]: SEC-BLOCKED: security-antivirus: INFECTED
security-antispam: HAM
```

En este caso el servidor `security-in3` descarta el mensaje (`SEC-BLOCKED`) y no se lo pasa a ninguna máquina `user-mailbox`.

4. Mensaje entrante para cuenta inexistente

Si entra desde Internet un mensaje para una cuenta inexistente, esto se detectará en un `smtp-in` y el `security-in` que intentó entregar el mensaje a un `user-mailbox`. Se generará un mensaje automático de notificación de error para el remitente.

En los servidores implicados se verían unas trazas equivalentes a las siguientes:

```
----- smtp-in2 -----
2020-02-22 04:39:36 smtp-in2 [E21F2C29]: connect from: smtp4.urucu.ec
2020-02-22 04:39:36 smtp-in2 [E21F2C29]: message from: <carlos.izquierdo@urucu.ec> to:
<9clara.reyes@etsist.upm.es> message-id: <13207146@monet.urucu.ec> size: 1883109
2020-02-22 04:39:36 smtp-in2 [E21F2C29]: status=2.0.0 (accepted)
2020-02-22 04:39:36 smtp-in2 [E21F2C29]: relay to: security-in1.etsist.upm.es
2020-02-22 04:39:36 smtp-in2 [E21F2C29]: permanently rejected, dsn: 5.1.1 (bad
destination mailbox address)
2020-02-22 04:39:36 smtp-in2 [E21F2C29]: bounced from: <MAILER-DAEMON@smtp-
in2.etsist.upm.es> to: <carlos.izquierdo@urucu.ec> messageid: <5200793@smtp-
in2.etsist.upm.es> size: 42547 queued as: E21F2C35

2020-02-22 04:39:37 smtp-in2 [E21F2C35]: relay to: smtp-out2.etsist.upm.es
2020-02-22 04:39:37 smtp-in2 [E21F2C35]: delivered, dsn: 2.0.0, queued as: 290581AA

----- security-in1 -----
2020-02-22 04:39:36 security-in1 [200E2E3D]: connect from: smtp-in2.etsist.upm.es
2020-02-22 04:39:36 security-in1 [200E2E3D]: message from: <carlos.izquierdo@urucu.ec>
to: <9clara.reyes@etsist.upm.es> message-id: <13207146@monet.urucu.ec> size: 1883109
2020-02-22 04:39:36 security-in1 [200E2E3D]: status=5.1.1 (bad destination mailbox
address)

----- smtp-out -----
2020-02-22 04:39:38 smtp-out2 [290581AA]: connect from: smtp-in2.etsist.upm.es
2020-02-22 04:39:38 smtp-out2 [290581AA]: message from: <MAILER-DAEMON@smtp-
in2.etsist.upm.es> to: <carlos.izquierdo@urucu.ec> message-id: <5200793@smtp-
in2.etsist.upm.es> size: 42547
2020-02-22 04:39:38 smtp-out2 [290581AA]: status=2.0.0 (accepted)
2020-02-22 04:39:39 smtp-out2 [290581AA]: relay to: post1.urucu.ec
2020-02-22 04:39:39 smtp-out2 [290581AA]: delivered, dsn: 2.0.0, queued as: 9CEF306C
```

Las trazas anteriores indican que la máquina `security-in1` no pudo entregar un mensaje por no encontrar un `user-mailbox` asociado y lo rechazó con código de estado

5.1.1 por tratarse de un destinatario desconocido. Dado que un código de estado 5.x.y indica fallo permanente, la máquina `smtp-in1` descarta el mensaje inválido y genera un mensaje automático nuevo de notificación de error para el remitente del mensaje. El mensaje de notificación de error es un mensaje más, con su propio `message_id` y `queue_id` (E21F2C35) y se envía a través de la máquina `smtp-out2`.

5. Mensaje saliente válido

El siguiente ejemplo muestra el recorrido completo de un mensaje escrito por un usuario del sistema para una cuenta externa.

El mensaje sería entregado por el agente de usuario (aplicación cliente de correo) a una de las máquinas `msa`, la cual generaría trazas como estas:

```
2020-02-21 10:24:02 msa1 [F314721F]: connect from: oldenburg.etsist.upm.es
2020-02-21 10:24:02 msa1 [F314721F]: message from: <ramona.rivera@etsist.upm.es> to:
<ramiro.blasco@garrocha.tz> message-id: <38423825@oldenburg.etsist.upm.es> size: 35550
2020-02-21 10:24:02 msa1 [F314721F]: status=2.0.0 (accepted)
2020-02-21 10:24:03 msa1 [F314721F]: relay to: security-out1.etsist.upm.es
2020-02-21 10:24:03 msa1 [F314721F]: delivered, dsn: 2.0.0, queued as: 31CA6785
```

A continuación, pasaría por uno de los `security-out`, que generaría trazas como estas:

```
2020-02-21 10:24:03 security-out1 [31CA6785]: connect from: msa1.etsist.upm.es
2020-02-21 10:24:03 security-out1 [31CA6785]: message from:
<ramona.rivera@etsist.upm.es> to: <ramiro.blasco@garrocha.tz> message-id:
<38423825@oldenburg.etsist.upm.es> size: 35550
2020-02-21 10:24:03 security-out1 [31CA6785]: status=2.0.0 (accepted)
2020-02-21 10:24:05 security-out1 [31CA6785]: SEC-PASSED: security-antivirus: CLEAN
security-antispam: HAM
2020-02-21 10:24:06 security-out1 [31CA6785]: relay to: smtp-out2.etsist.upm.es
2020-02-21 10:24:06 security-out1 [31CA6785]: delivered, dsn: 2.0.0, queued as:
6E300EB3
```

Finalmente llegaría a uno de los `smtp-out`, el cual lo entregaría a la estafeta de la cuenta de destino (en este caso, `mail1.garrocha.tz`):

```
2020-02-21 10:24:06 smtp-out2 [6E300EB3]: connect from: security-out1.etsist.upm.es
2020-02-21 10:24:06 smtp-out2 [6E300EB3]: message from: <ramona.rivera@etsist.upm.es>
to: <ramiro.blasco@garrocha.tz> message-id: <38423825@oldenburg.etsist.upm.es> size:
35550
2020-02-21 10:24:06 smtp-out2 [6E300EB3]: status=2.0.0 (accepted)
2020-02-21 10:24:06 smtp-out2 [6E300EB3]: relay to: mail1.garrocha.tz
2020-02-21 10:24:06 smtp-out2 [6E300EB3]: delivered, dsn: 2.0.0, queued as: 9807AC18
```

La estafeta de destino pertenece a una organización diferente, por lo que no se pueden ver sus trazas, pero sí se sabe que en ellas aparecerá con el `queue_id` 9807AC18.

6. Mensaje saliente clasificado como spam y/o con virus

Si un mensaje saliente es clasificado como `spam` o como `virus`, habría sido descartado por un `security-out` y no habría llegado a los `smtp-out`. En este caso las trazas del `security-out` serían las siguientes:

```

-----si se detecta VIRUS -----
2020-02-20 16:23:41 security-out1 [BCBF911D]: SEC-BLOCKED: security-antivirus:
INFECTED security-antispam: HAM

-----si se detecta SPAM -----
2020-02-20 16:23:45 security-out1 [BCBF92C3]: SEC-BLOCKED: security-antivirus: CLEAN
security-antispam: SPAM

-----si se detecta VIRUS Y SPAM -----
2020-02-20 16:23:38 security-out1 [BCBF901C]: SEC-BLOCKED: security-antivirus:
INFECTED security-antispam: SPAM

```

7. Mensaje saliente que no se puede entregar a la estafeta de destino

Si un **smtp-out** no puede entregar a la estafeta de destino un mensaje (porque la dirección de destino es inexistente, por ejemplo), generará un mensaje automático de notificación de error para el remitente y lo enviará directamente al **user-mailbox**, tal como se muestra en las siguientes trazas de un **smtp-out**:

```

2020-02-20 20:08:27 smtp-out1 [DE6E7F24]: connect from: security-out1.etsist.upm.es
2020-02-20 20:08:27 smtp-out1 [DE6E7F24]: message from: <laura.mateo@etsist.upm.es>
to: <javvier.perez@jabillo.pk> messageid: <44033192@pippin.etsist.upm.es> size:
15500404
2020-02-20 20:08:53 smtp-out1 [DE6E7F24]: status=2.0.0 (accepted)
2020-02-20 20:08:55 smtp-out1 [DE6E7F24]: relay to: estafeta4.jabillo.pk
2020-02-20 20:09:13 smtp-out1 [DE6E7F24]: permanently rejected, dsn: 5.1.1 (bad
destination mailbox address)
2020-02-20 20:09:13 smtp-out1 [DE6E7F24]: bounce from: <MAILER-
DAEMON@postman3.etsist.upm.es> to: <laura.mateo@etsist.upm.es> messageid:
<92823734@postman3.etsist.upm.es> size: 30340 queued as: DE6E8013
2020-02-20 20:09:16 smtp-out1 [DE6E8013]: relay to: user-mailbox3.etsist.upm.es
2020-02-20 20:09:16 smtp-out1 [DE6E8013]: delivered, dsn: 2.0.0, queued as: 8CDA55AA

```

8. Sobrecarga en alguno de los servidores de la cadena

En ocasiones puede suceder que alguno de los servidores de la cadena esté sobrecargado, en cuyo caso la máquina rechazará algunos mensajes con un código de estado 4.3.2. Como los códigos de estado 4.x.y indican fallo temporal, el mensaje permanecerá encolado en la máquina anterior, la cual reintentará la entrega al cabo de un rato, posiblemente a otro servidor de la granja. Las siguientes trazas muestran un ejemplo en el que la máquina **smtp-in2** no ha podido entregar la primera vez un mensaje a la máquina **security-in1**:

```

2020-02-22 04:22:25 smtp-in2 [E21F04D7]: connect from: jawlensky.garrofero.tt
2020-02-22 04:22:25 smtp-in2 [E21F04D7]: message from: <horacio.moya@garrofero.tt> to:
<clemente.manzano@etsist.upm.es> message-id: <60359925@jawlensky.garrofero.tt> size:
7313
2020-02-22 04:22:25 smtp-in2 [E21F04D7]: status=2.0.0 (accepted)
2020-02-22 04:22:25 smtp-in2 [E21F04D7]: relay to: security-in1.etsist.upm.es
2020-02-22 04:22:25 smtp-in2 [E21F04D7]: temporarily rejected, dsn: 4.3.2 (overload)
2020-02-22 04:35:05 smtp-in2 [E21F04D7]: relay to: security-in1.etsist.upm.es
2020-02-22 04:35:05 smtp-in2 [E21F04D7]: delivered, dsn: 2.0.0, queued as: 200E2774

```

Como se observa, el intento de entrega a las 04:22:25 no se pudo completar porque **security-in1** rechazó el mensaje con código 4.3.2 por sobrecarga. El mensaje se volvió a intentar entregar a las 04:35:05 y esta vez sí fue aceptado. Entre estos dos

intentos de entrega seguramente habrá muchas otras trazas pertenecientes a otros mensajes que pasan por smtp-in2.

Por su parte, en la máquina security-in1 figurarían estas trazas relacionadas con este mensaje:

```
2020-02-22 04:22:25 security-in1 [200E14B5]: connect from: smtp-in2.etsist.upm.es
2020-02-22 04:22:25 security-in1 [200E14B5]: message from: <horacio.moya@garrofero.tt>
to: <ana.manzano@etsist.upm.es> message-id: <60359925@jaw.garrofero.tt> size: 7313
2020-02-22 04:22:25 security-in1 [200E14B5]: status=4.3.2 (overload)

2020-02-22 04:35:05 security-in1 [200E2774]: connect from: smtp-in2.etsist.upm.es
2020-02-22 04:35:05 security-in1 [200E2774]: message from: <horacio.moya@garrofero.tt>
to: <ana.manzano@etsist.upm.es> message-id: <60359925@jaw.garrofero.tt> size: 7313
2020-02-22 04:35:05 security-in1 [200E2774]: status=2.0.0 (accepted)
2020-02-22 04:35:05 security-in1 [200E2774]: SEC-PASSED: security-antivirus: CLEAN
security-antispam: HAM
2020-02-22 04:35:05 security-in1 [200E2774]: relay to: user-mailbox2.etsist.upm.es
2020-02-22 04:35:05 security-in1 [200E2774]: delivered, dsn: 2.0.0, queued as:
87F4B463
```

Como se puede observar, en el primer intento de entrega (con queue_id 200E14B5), el mensaje fue rechazado por sobrecarga. En el segundo intento (con queue_id 200E2774), el mensaje sí fue aceptado y procesado.

Por supuesto, nada garantiza que en el segundo intento se podrá entregar el mensaje, por lo que podría darse el caso de que un mensaje fuera rechazado más de una vez por sobrecarga.