

Universidad de Valladolid  
E.T.S Ingeniería Informática  
Grado en Ingeniería Informática [Tecnologías de la información]

**Curso 2017/2018**  
**Evaluación de Sistemas Informaticos**  
**Práctica de Laboratorio 1**  
**Evaluación del rendimiento de un servidor web.**

Grupo de Laboratorio 07  
Esteban Pellejero, Sergio  
González Bravo, Miguel  
González Pérez, Daniel  
Rabadán Martín, Borja

# Índice

<b>1. Introducción y objetivos</b>	<b>3</b>
<b>2. Información disponible</b>	<b>3</b>
<b>3. Herramientas utilizadas</b>	<b>4</b>
3.1. Configuración de AWSTATS . . . . .	4
<b>4. Análisis de utilización</b>	<b>5</b>
4.1. Respuestas del servidor . . . . .	5
4.2. Análisis sobre los tipos y tamaños de los documentos . . . . .	7
4.3. Análisis sobre la popularidad de los documentos y transferencias . . . . .	9
<b>5. Análisis general de los datos</b>	<b>10</b>
5.1. Estadísticas globales de uso . . . . .	10
5.2. Popularidad de ficheros y páginas, descargas más frecuentes . . . . .	11
5.3. Visitantes . . . . .	12
5.4. Patrones temporales de tráfico soportado por el servidor . . . . .	12
5.5. Caracterización de las respuestas del servidor . . . . .	12
<b>6. Caracterización de la carga. Partición y modelo.</b>	<b>12</b>
6.1. Filtrado inicial . . . . .	13
6.2. Preparación de los datos para el análisis estadístico . . . . .	14
6.3. Análisis estadístico . . . . .	16
6.3.1. Resumen de la aglomeración . . . . .	16
6.3.2. Gráficos resultantes . . . . .	17
<b>7. Conclusiones</b>	<b>19</b>
<b>8. Bibliografía</b>	<b>21</b>
<b>Appendices</b>	<b>22</b>
<b>ANEXO ANEXO I.</b>	<b>22</b>
<b>ANEXO ANEXO II.</b>	<b>26</b>
<b>ANEXO ANEXO III.</b>	<b>31</b>

## 1. Introducción y objetivos

En la práctica anterior determinábamos el funcionamiento de un sistema informático en base a las pruebas y cargas que nosotros realizábamos contra él. A partir de esas pruebas extraíamos unos datos y unas conclusiones que nos permitían saber donde estaban los cuellos de botella, los recursos más demandados, etc. Sin embargo el objetivo de esta práctica es caracterizar y analizar el comportamiento de un servidor web en base a los ficheros de registro que han quedado almacenados en él. Con estos datos podremos saber a que recursos se accede más frecuentemente, a que horas tenemos que evitar por ejemplo hacer el mantenimiento del sistema (ya que hay mucha demanda de usuarios), etc.

En resumen, vamos a caracterizar la carga de nuestro servidor web, pero siguiendo un método diferente. También haremos un modelo teórico de la carga que ha soportado ese servidor web durante el periodo de tiempo suministrado por los archivos de log.

## 2. Información disponible

Para conseguir evaluar nuestro sistema informático, el servidor de la escuela de Ingeniería Informática, y hacer el modelo de carga, únicamente vamos a disponer de un fichero de log generado por el propio servidor Apache, durante las fechas 18 de Octubre de 2017 y 3 de Diciembre de 2017. En total vamos a analizar 47 días de funcionamiento  $\simeq$  1 mes y medio.

Este fichero log de apache tiene dos características diferenciadas de los ficheros autogenerados por los servidores convencionales. las diferencias son:

- **Direcciones ip:** como veremos más adelante, en el log se reflejan las direcciones desde las que se ha accedido al servidor. ya que esto es un estudio teórico para una asignatura, se va a mantener la confidencialidad y las direcciones están anonimizadas.

La estructura que tienen los logs de apache la podemos encontrar en el enunciado de la práctica, por lo tanto no vamos a definir aquí lo que es cada campo. Pero para entenderlo mejor sí que vamos a poner un pequeño ejemplo mostrando en el fichero de log, los diferentes campos, separados por colores. Los campos de los ficheros de log, y la estructura del fichero que vamos a analizar se puede ver en la **Figura 1**.



```
ip1601 - [18/Oct/2017:09:35:40 +0200] "GET / HTTP/1.1" 200 27974 39393
```

Figura 1: Formato del fichero de log de Apache2

Cada uno de los colores corresponde a un campo diferente:

- **Cliente remoto** → verde claro.
- **Autenticación** → azul claro.
- **Autenticación** → rosa claro.
- **Fecha y hora de la petición** → amarillo.

- **Petición** → gris.
- **Status** → verde.
- **Volumen transferido** → azul.
- **Elapsed time** → rosa.

Como hemos dicho el significado de los campos lo podemos encontrar en el enunciado de la práctica o sino en el manual de referencia oficial proporcionado por Apache<sup>1</sup>

### 3. Herramientas utilizadas

Para la parte de análisis de los ficheros de log de apache hemos decidido utilizar un programa llamado AWSTATS, que nos proporciona de manera automatizada la mayoría de la información requerida en la práctica. La parte de información que quizá no nos proporciona será sacada por programas propios o por algún otro programa como web log analyzer.

#### 3.1. Configuración de awstats

AWSTATS es un software que nos permite ver de manera gráfica en una página HTML los resultados de un análisis automatizado de un fichero de log. Para poder utilizar de manera correcta Awstats hay que instalarlo en un servidor web ya configurado. Nosotros en la asignatura disponemos uno, pero en el momento de instalarlo, por mayor comodidad de acceso, el programa se ha instalado en un servidor propio, siendo. Es decir, podemos acceder a la información desde cualquier punto sin depender de las máquinas de la universidad. La URL es: *http://serverandroid.ddns.net:47000/cgi-bin/awstats.pl?config=uva.es*.

La configuración e instalación de este software es bastante simple. Tenemos que instalar los paquetes en el servidor y adaptar la configuración de AWSTATS para que coja los ficheros de log que nosotros le pasamos. Una vez hecho eso, configuramos el servidor web para que muestre los resultados en Internet y ya lo tenemos. La configuración exacta que se ha seguido, se puede ver en Internet<sup>2</sup>.

Como podemos ver en la **Figura 2**, AWSTATS nos muestra la información de una manera muy amigable.

Algo a tener en cuenta durante el desarrollo de esta primera parte de la práctica son los tráfico de datos generados por los bots, web spiders o crawlers. Estos tráfico no se deberían de tener en cuenta para hacer el análisis del rendimiento del servidor ni para el modelado de la carga, ya que no son, por así decirlo, uso real del sistema por parte de usuarios finales. Por ejemplo los bots o los web spiders se caracterizan por conexión al servidor de muy poco periodo de tiempo, pero de gran frecuencia.

Empezamos con la primera parte de esta práctica que es el análisis de la utilización del servidor.

---

<sup>1</sup>[http://httpd.apache.org/docs/current/mod/mod\\_log\\_config.html](http://httpd.apache.org/docs/current/mod/mod_log_config.html)

<sup>2</sup><https://www.maketecheasier.com/set-up-awstats-ubuntu/>

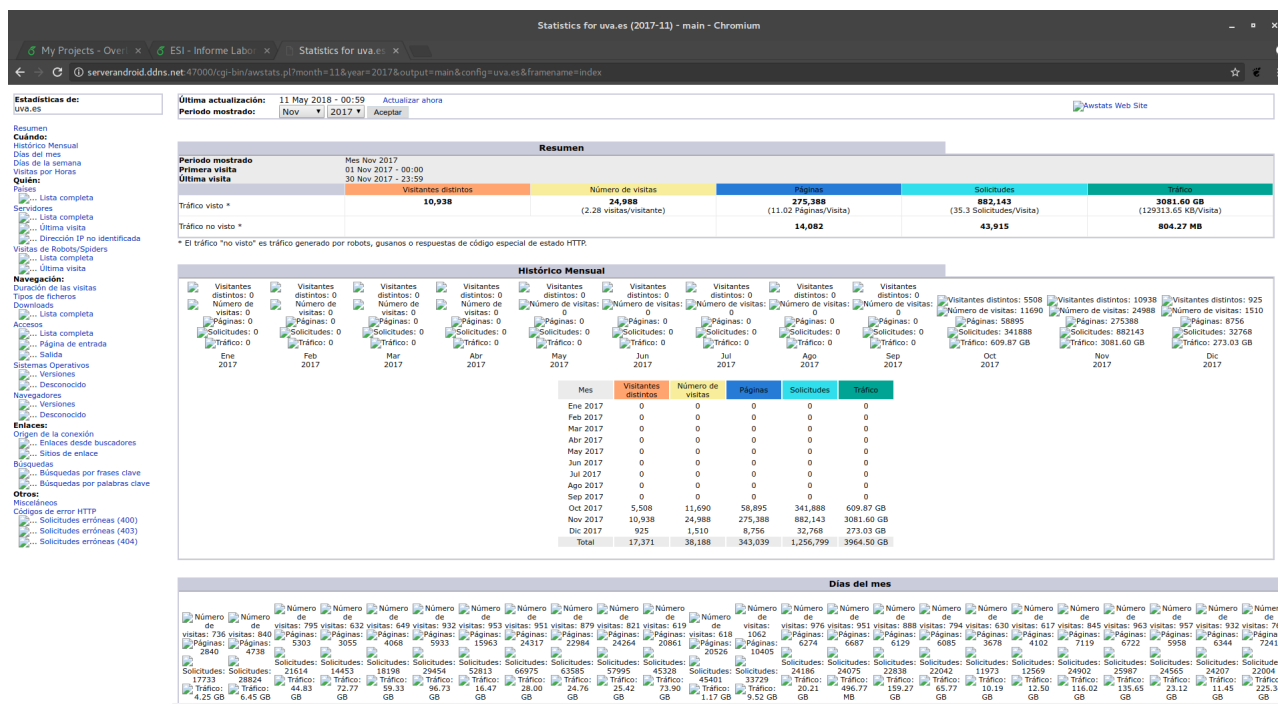


Figura 2: Visualización de la información con AWSTATS

## 4. Análisis de utilización

El análisis de utilización tiene implicaciones en la detección de problemas y ajuste del rendimiento del servidor. En este sentido el análisis de los datos de registro de accesos puede servir para abordar estudios relacionados con la decisión de qué documentos poner en caché, ajuste de sistemas relacionados con la gestión de la memoria caché, etc.

### 4.1. Respuestas del servidor

El análisis de las respuestas del servidor nos permite, por una parte, la identificación de las métricas a elegir en el estudio del rendimiento, y por parte, evaluar la calidad de servicio (QoS) ofrecida por nuestro sistema informático. Tenemos que saber que las posibles respuestas que se obtiene de un servidor web cuando se hace una petición http son las siguientes:

- **Éxito:** el usuario obtiene el documento, por lo que se consideran métricas de productividad, velocidad y utilización. No modificado, el usuario ya tiene una copia del documento y no se transfieren bytes.
- **Error:** documento encontrado pro en otro sitio.
- **Error:** documento no encontrado o no se dispone de permisos o privilegios suficientes.

Primero tenemos que especificar que la información ha sido proporcionada y extraída con AWSTATS y se ha hecho un resumen de las figuras que se encuentran en el ANEXO II (**Figura 22** y **Figura 23**). Tenemos que especificar que la **Figura 22** nos muestra la cantidad de peticiones que se han hecho al servidor y la cantidad de páginas. Hemos interpretado que la cantidad de páginas es la cantidad de respuestas con código 200, ya que aunque se han hecho muchas más

peticiones, algunas pueden haber devuelto códigos de error. Sin embargo las páginas servidas son las que respuestas correctas. A partir de estos datos hemos elaborado la siguiente figura (**Figura 3**) de resultados.

	<b>Éxito (200)</b>	<b>Error (301+302)</b>	<b>Error (404)</b>	<b>Error (403)</b>
<b>Octubre</b>	58.845	9.514	2.093	1.967
<b>Noviembre</b>	275.388	23.349	4.017	4.441
<b>Diciembre</b>	8.756	1.318	320	180

Figura 3: Tabla con las diferentes respuestas del servidor

Primero vamos a especificar los diferentes códigos de error:

- **200**: como hemos dicho anteriormente son las respuestas correctas, es decir que se ha servido la página correctamente.
- **301**: son las páginas movidas permanentemente, es decir, se han encontrado pero el sitio ha cambiado de manera permanente.
- **302**: son las páginas movidas temporalmente, es decir, se han encontrado pero el sitio ha cambiado de manera temporal.
- **404**: páginas no servidas porque no se ha encontrado la dirección.
- **403**: páginas no servidas porque no se disponía de los permisos necesarios para acceder a esa página.

Hemos decidido agrupar las métricas de los códigos 301 y 302 porque en el fondo son lo mismo y además las redirecciones temporales son minúsculas en comparación a las permanentes.

En nuestra opinión, lo importante es saber la distribución de como se realizan las peticiones correctas, el resto de peticiones, las erróneas suelen ser en momentos puntuales, que habrá que estudiar, pero creemos de mayor importancia saber las distribuciones más concretas de los códigos 200. Hemos elaborado la siguiente tabla, representada en la **Figura 15**. Esta tabla cuando se producen más visitas y menos visitas, filtrado por día de la semana y por horas del día.

	<b>Día semana + pags</b>	<b>Día semana - pags</b>	<b>Hora día + pags</b>	<b>Hora día - pags</b>
<b>Octubre</b>	X-V	S-D	9-11	4-6
<b>Noviembre</b>	J-V	S-D	8-19	1-6
<b>Diciembre</b>	V	D	10-16	2-8

Figura 4: Tabla con diferentes distribuciones de páginas servidas por el servidor

Como podemos observar, en general los días que más accesos hay al servidor son de mitad de semana hasta el viernes. Sin embargo los días que claramente hay menos accesos es durante el finde semana. Principalmente se debe al alma fiesterera de los estudiantes. También comentar las horas del día son lógicas, empiezan a servirse las peticiones entre las 8-9 de la mañana y su volumen es más o menos constante hasta las 18-19 de la tarde. A partir de esa hora parece que no hay demasiados accesos en comparación al resto de horas del día.

La tabla anterior ha sido confeccionada a partir de las figuras obtenidas por AWSTATS, que se encuentran en el ANEXO II (**Figura 24, Figura 25, Figura 26, Figura 27, Figura 28 y Figura 29**).

## 4.2. Análisis sobre los tipos y tamaños de los documentos

Como se dice en el enunciado de la práctica no es lo mismo servir páginas web HTML que documentos de vídeo que son de mucho mayor tamaño. Si sabemos la distribución y el porcentaje de veces que se solicita cada tipo de página sabemos como distribuir los recursos de nuestro servidor de una manera más eficiente, como por ejemplo dando una mayor prioridad al almacenamiento rápido en vez de a la memoria, etc.

nuestra partición en tipos d páginas se ha hecho en base a las extensiones de las páginas servidas por el servidor. Dividimos los tipos de documentos en 7 clases, cada una de las cuales se asocia con un conjunto de extensiones:

- **HTML**: que se corresponde con los ficheros: .html, .htm y .xml de contenido estático.
- **IMAGEN**: se engloban las extensiones: .jpg, .png, .gif, .ico, jpeg y .svg.
- **VIDEO**: que tiene los archivos .flv y .mp4.
- **ARCHIVOS**: que se corresponde con documentos de tipo word o PDF → .pdf, .doc, .docx, .pptx, .gz, xlsx y .ppsx.
- **FORMATO**: archivos relacionados con los html, que dan el formato a la página web, y que por lo tanto se llaman formateadores: .css.
- **DINAMICAS**: archivos relacionados con formularios o que hacen parte de su trabajo en el servidor: .js y .php.
- **OTRAS**: principalmente se tienen en cuenta los archivos .ova, que son los que tienen una carga significativamente mayor que el resto de documentos que quedan por analizar. Los archivos .ova son archivos correspondientes a configuraciones de máquinas virtuales. Y cada una suele pesar 1 Gb o 2 Gb (por experiencia en las asignaturas). Y como veremos dan mucho tráfico inesperado al servidor.

Para cada uno de estos tipos de páginas, se han recopilado los siguientes datos:

- Total de peticiones.
- Porcentaje de accesos.
- Media peticiones/día.
- Total peticiones diferentes.
- Media peticiones diferentes/día.
- Bytes totales transferidos.
- Bytes totales diferentes transferidos.
- Media de bytes diferentes transferidos.
- Media bytes transferidos.

■ Media tamaño del fichero.

Este estudio se ha hecho para el tráfico de información para cada mes. Es decir tenemos 3 tablas en las que mostramos la información anterior clasificada por mes. Así conseguimos un estudio mas detallado para cada periodo (aunque algunos como Diciembre sean muy cortos). Podemos encontrar la información en la **Figura 5**, **Figura 6** y **Figura 7**. Estas tablas se han realizado a partir de los datos extraídos de AWStats que se encuentran en el ANEXO II en la **Figura 30**, **Figura 31** y **Figura 32**.

Octubre	HTML	IMAGEN	VIDEO	DINAMICAS	ARCHIVOS	FORMATEADORES	OTRAS
Total peticiones	1417,00	149424,00	102,00	132151,00	11449,00	43793,00	2701,00
% accesos	0,30	43,10	0,00	38,60	3,10	12,80	2,10
Media peticiones/día	71,00	8138,00	5,00	4500,00	633,00	1835,00	91,00
Peticiones diferentes	1259,00	138194,00	72,00	61459,00	11200,00	32929,00	143,00
Media peticiones diferentes/día	69,00	7677,00	4,00	3414,00	622,00	1828,00	7,00
B totales transferidos	1265631,00	3758096384,00	338357523578,88	1288490188,80	2147483648,00	53194260,48	17179869184,00
B distintos transferidos	677561,00	221863024,00	10696197259,00	5342400,00	525212690,00	156985,00	96422,00
Media B distintos transferidos/día	37642,00	12325723,00	594233181,00	296800,00	29178482,00	8721,00	5356,00
Media B transferidos	72071,00	213866757,00	2506909779,00	17432717,00	117650463,00	1863084,00	2245672,00
Media tamaño fichero	891289,60	25150,55	3317230623,32	9750,14	187569,54	1214,67	6360558,75

Figura 5: Distribución tipos de páginas web visitadas en Octubre

Noviembre	HTML mb	IMAGEN gb	VIDEO gb	DINAMICAS gb	ARCHIVOS gb	FORMATEADORES mb	OTRAS gb
Total peticiones	94141,00	318230,00	212,00	383639,00	35542,00	43942,00	5317,00
% accesos	10,60	35,40	0,00	42,40	3,80	4,90	2,90
Media peticiones/día	3122,00	10325,00	7,00	5046,00	1181,00	1035,00	115,00
Peticiones diferentes	92775,00	291687,00	166,00	106399,00	33738,00	36623,00	341,00
Media peticiones diferentes/día	3092,00	9722,00	5,00	3546,00	1124,00	1220,00	11,00
B totales transferidos	90,08	8,03	1574,81	4,82	7,20	70,00	24,00
B distintos transferidos	1044517,00	251204653,00	21155647051,00	13863299,00	670572054,00	636418,00	1612488918,00
Media B distintos transferidos/día	34817,00	8373488,00	705188025,00	462109,00	22352401,00	21213,00	53749630,00
Media B transferidos	3144609,00	253575641,00	4023337662,00	24718437,00	253575641,00	2759519,00	862753392,00
Media tamaño fichero (b)	1003,37	27094,07	7976129065,35	13490,38	217515,65	1670,39	4846681,17

Figura 6: Distribución tipos de páginas web visitadas en Noviembre

Diciembre	HTML kb	IMAGEN mb	VIDEO gb	DINAMICAS mb	ARCHIVOS mb	FORMATEADORES mb	OTRAS gb
Total peticiones	183,00	16048,00	29,00	12927,00	2147,00	1088,00	271,00
% accesos	0,50	48,60	0,00	39,40	6,30	3,30	1,90
Media peticiones/día	5,00	516,00	1,00	199,00	68,00	37,00	5,00
Peticiones diferentes	167,00	15159,00	28,00	4471,00	2115,00	965,00	21,00
Media peticiones diferentes/día	5,00	489,00	0,00	144,00	68,00	31,00	0,00
B totales transferidos	168,00	409,07	143,65	136,51	390,00	2,17	3,00
B distintos transferidos	136505,00	97255381,00	13663903802,00	926002,00	333797356,00	242719,00	1612470362,00
Media B distintos transferidos/día	4403,00	3137270,00	440771090,00	29871,00	10767656,00	7829,00	52015172,00
Media B transferidos	5443,00	14037602,00	630628118,00	1155930,00	13265799,00	124765,00	104148497,00
Media tamaño fichero	940,07	26728,82	5318724586,81	11073,03	190472,59	2091,37	11886440,86

Figura 7: Distribución tipos de páginas web visitadas en Diciembre

En líneas generales podemos extraer varias conclusiones de esto:



- **Número de peticiones:** si filtramos por número de peticiones observamos que ganan las páginas DINÁMICAS, tanto las de PHP como las de JS son las que más puntos consiguen. Tampoco se quedan atrás las imágenes ya que las páginas modernas tienden a cargar mucho la interfaz de usuario con imágenes para que sea más amigable. Después de esto vienen las páginas de ARCHIVOS es decir, tanto los PDF como los documentos son también los más requeridos.
- **Tamaño de archivos:** si nos fijamos en el tamaño de los archivos tenemos que dejar claro que los ganadores son los vídeos. Claramente no son los tipos de ficheros más demandados en el sistema, pero sí que son los más pesados, y muy por encima de los tipos de archivos anteriores. Siguiendo a los vídeos se encuentran lo que nuestra tabla refleja como OTROS, que no es ni más ni menos que los ficheros .ova correspondientes a máquinas virtuales facilitadas por los profesores.
- **Procentaje de accesos:** aquí es donde se ve claramente los tipos de páginas que son demandados con mayor frecuencia. Los ganadores absolutos de esta categoría, al igual que en número de peticiones, son las páginas dinámicas y las imágenes. En el estudio a lo largo de los días provistos en el log podemos observar que las imágenes ocupan en torno al 35-48 % de los accesos totales mientras que las páginas dinámicas ocupan en torno al 30-40 % de los accesos. Un administrador del sistema ha de tener esto en cuenta para favorecer la descarga de estos tipos de archivos ya que son los más demandados.

#### 4.3. Análisis sobre la popularidad de los documentos y transferencias

Este tipo de análisis sirve para saber que elementos tienen que estar en la cache, cuales son los documentos exactos que más se requieren y cuantos son los documentos a los que se accede una vez, y que son menos importantes a la hora de dar preferencia en el sistema. En líneas generales, las medidas que vamos a obtener en esta fase de la práctica son:

- **Número de Peticiones de documentos diferentes.**
- **Número total de peticiones.**
- **Bytes diferentes.**
- **Bytes totales.**
- **Ficheros diferentes accedidos solamente una vez.**
- **Distribución de los tamaños de los ficheros.**

Hemos tenido en cuenta que las peticiones las hemos considerado solo correctas, es decir, solo hemos tenido en cuenta las que tienen un código 200. Hemos resumido los valores en la siguiente tabla (**Figura 8**). Para elaborar esta tabla se ha tenido en cuenta todo el periodo de observación, es decir, los 48 días que dura la muestra del log.

<b>Peticiones documentos diferentes</b>	43327
<b>Peticiones totales</b>	1236591
<b>Bytes diferentes</b>	35 Gb
<b>Bytes totales</b>	243 Gb
<b>Ficheros diferentes accedidos una sola vez</b>	972

Figura 8: Medidas análisis de popularidad

Recordamos que puede parecer poca la cantidad total de Gb transmitidos, pero solo contamos las respuestas correctas (código 200). Hemos decidido hacerlo así por simplicidad, si quisiéramos hacer un estudio en mayor profundidad habría que mirar las que tienen los códigos 301, 302, etc.

La característica que mide la distribución del tamaño de los accesos hemos cogido los datos proporcionados por AWStats. Estos datos son para las 10 páginas mas demandadas, que en verdad, son las más representativas. No es de mucho interés saber el tamaño medio de los ficheros que se han pedido una vez al año. Por lo menos hemos tomado esa decisión. La información se encuentra recogida en la **Figura 9** , **Figura 10** y **Figura 11**. Nos tenemos que fijar en la columna de color verde. Ahí tenemos los tamaños de los ficheros.

Páginas-URLs (Top 10) - Lista completa - Página de entrada - Salida				
2,302 páginas diferentes				
	Accesos	Tamaño medio	Página de entrada	Salida
/wp-admin/admin-ajax.php	21,997	132 Bytes	496	5,256
/	5,876	41.97 KB	3,688	1,203
/feed/	4,678	4.71 KB	1,629	1,683
/wp-cron.php	2,635		10	14
/instalacion-del-sistema-operativo-linux/	1,238	19.31 KB	973	194
/instalacion-del-sistema-operativo-windows/	834	18.62 KB	556	192
/wp-content/plugins/wp-ui/css/css.php	801	17.83 KB	17	91
/wp-json/oembed/1.0/embed	750	1.44 KB	24	33
/charlas-de-los-mercados/feed/	740	765 Bytes	451	374
/wp-login.php	420	5.00 KB	256	112
Otros	18,926	857.39 KB	3,590	2,628

Figura 9: Tamaño medio ficheros Octubre

Páginas-URLs (Top 10) - Lista completa - Página de entrada - Salida				
2,491 páginas diferentes				
	Accesos	Tamaño medio	Página de entrada	Salida
/wp-admin/admin-ajax.php	54,406	164 Bytes	1,026	11,054
/wp-content/plugins/bwp-minify/min/	28,711	13.55 KB	345	1,321
/	11,804	39.67 KB	7,371	2,489
/feed/	10,757	3.76 KB	3,424	2,559
/wp-cron.php	7,803	0	21	20
/instalacion-del-sistema-operativo-linux/	2,966	19.26 KB	2,333	203
/instalacion-del-sistema-operativo-windows/	1,800	19.11 KB	1,215	115
/charlas-de-los-mercados/feed/	1,544	764 Bytes	1,000	827
/wp-content/plugins/wp-ui/css/css.php	1,619	18.57 KB	30	106
/wp-login.php	1,262	5.12 KB	771	266
Otros	152,616	188.08 KB	7,452	5,028

Figura 10: Tamaño medio ficheros Noviembre

Páginas-URLs (Top 10) - Lista completa - Página de entrada - Salida				
1,268 páginas diferentes				
	Accesos	Tamaño medio	Página de entrada	Salida
/wp-content/plugins/bwp-minify/min/	2,019	14.09 KB	60	139
/wp-admin/admin-ajax.php	1,744	59 Bytes	37	352
/feed/	801	4.31 KB	280	278
/	503	44.17 KB	363	136
/wp-cron.php	404		2	
/instalacion-del-sistema-operativo-linux/	151	19.35 KB	127	5
/charlas-de-los-mercados/feed/	120	765 Bytes	66	59
/instalacion-del-sistema-operativo-windows/	83	17.63 KB	53	4
/wp-content/plugins/wp-ui/css/css.php	66	17.00 KB	4	3
/office365/	64	15.95 KB	54	4
Otros	2,801	1.12 MB	464	302

Figura 11: Tamaño medio ficheros Diciembre

## 5. Análisis general de los datos

En esta parte se va a incluir la parte de análisis un poco más global que se pide en el enunciado de la práctica.

### 5.1. Estadísticas globales de uso

Algunos datos importantes que se pueden extraer de la información que nos proporciona el log de Apache son:

- **Periodo de observación:** 47 días  $\Rightarrow$  3 días de Diciembre + 31 días de Noviembre + 13 días de Octubre.

- **HITS:** durante esos 47 días la cantidad de HITS, fue de 1.256.799  $\Rightarrow$  32.768 (Dic) + 882.143 (Nov) + 341.888 (Oct).
- **Visitantes:** en el periodo de 47 días hubo 38.188 visitantes  $\Rightarrow$  1.510 (Dic) + 24.988 (Nov) + 11.690 (Oct).
- **Ficheros descargados:** los ficheros descargados los tenemos ordenados por popularidad de descargas y filtrados por mes en las figuras: **Figura 12**, **Figura 13** y **Figura 14**. A grandes rasgos decir que el tipo de fichero más descargado son los PDF y normalmente temas relacionados con: horarios de clases, horarios de exámenes, calendarios de clases o charlas de los miércoles.

Downloads (Top 10) - Full list				
Downloads: 1582				
	Hits	206 Hits	Bandwidth	Average size
/wp-content/uploads/2017/06/Hor1718Grado1C_Segundo_T1.pdf	141	0	36.52 MB	265.26 KB
/wp-content/uploads/2017/06/Hor1718Grado1C_Primer_T3.pdf	103	0	28.74 MB	285.75 KB
/wp-content/uploads/2017/06/Hor1718Grado1C_Primer_T2.pdf	92	0	25.85 MB	287.70 KB
/wp-content/uploads/2017/06/CalendarioExámenes-1718_Grado_Segund...	69	0	15.60 MB	231.55 KB
/wp-content/uploads/2017/06/CalendarioExámenes-1718_Grado_Primer...	61	0	13.50 MB	226.68 KB
/wp-content/uploads/2017/06/Hor1718Grado1C_Segundo_T2.pdf	61	0	15.00 MB	251.74 KB
/wp-content/uploads/2017/06/ExámenesFC2017.pdf	54	0	5.04 MB	95.57 KB
/wp-content/uploads/2014/11/Docker.pdf	52	0	17.07 MB	336.06 KB
/wp-content/uploads/2017/06/Hor1718Grado1C_Primer_T1.pdf	52	0	13.86 MB	272.93 KB
/wp-content/uploads/2017/09/OfertaEmpleoIndra.pdf	46	1	7.75 MB	168.94 KB

Figura 12: Ficheros descargados durante el mes de Octubre

Downloads (Top 10) - Full list				
Downloads: 1617				
	Hits	206 Hits	Bandwidth	Average size
/wp-content/uploads/2017/06/ExámenesFC2017.pdf	343	0	31.35 MB	93.60 KB
/wp-content/uploads/2017/09/OfertaEmpleoIndra.pdf	268	0	48.64 MB	185.84 KB
/wp-content/uploads/2013/01/00-GuiaAlumnoTFG_2017.pdf	253	0	34.74 MB	140.60 KB
/wp-content/uploads/2017/05/charla-20170503_best.pdf	231	0	78.41 MB	347.60 KB
/wp-content/uploads/2017/09/everis_carter_taller_IE_Mindfulness....	229	0	130.99 MB	585.74 KB
/wp-content/uploads/2016/06/ExaGradoExtr16.pdf	228	0	77.84 MB	349.58 KB
/wp-content/uploads/2017/10/20171002_CalendarioTareas.pdf	227	0	45.57 MB	205.55 KB
/wp-content/uploads/2016/11/Manifiesto-25N-2016.pdf	224	0	15.81 MB	72.29 KB
/wp-content/uploads/2017/09/ActividadesPrimerCuatrimestre_1718.p...	222	0	26.73 MB	123.28 KB
/wp-content/uploads/2017/06/CalendarioExámenes-1718_Grado_Primer...	221	1	50.79 MB	234.30 KB

Figura 13: Ficheros descargados durante el mes de Octubre

Downloads (Top 10) - Full list				
Downloads: 1316				
	Hits	206 Hits	Bandwidth	Average size
/wp-content/uploads/2017/07/InformacionGeneral_UVa_EscuelaIngInf...	11	0	666.40 KB	60.58 KB
/wp-content/uploads/2017/06/CalendarioExámenes-1718_Grado_Segund...	8	0	1.69 MB	216.02 KB
/wp-content/uploads/2017/06/Hor1718Grado1C_Segundo_T1.pdf	7	0	1.63 MB	238.65 KB
/wp-content/uploads/2017/06/CalendarioExámenes-1718_Grado_Primer...	7	0	1.46 MB	213.36 KB
/wp-content/uploads/2017/06/Hor1718Grado1C_Tercero_IS.pdf	6	0	2.82 MB	481.91 KB
/wp-content/uploads/2017/06/CalendarioExámenes-DT-1718-Tercero.p...	6	0	1.20 MB	205.54 KB
/wp-content/uploads/2017/06/Hor1718Grado1C_Primer_T1.pdf	6	0	1.44 MB	246.40 KB
/wp-content/uploads/2017/06/AsignacionGrupos201718.pdf	6	0	2.57 MB	438.12 KB
/wp-content/uploads/2017/06/Hor1718Grado1C_Primer_T3.pdf	6	0	1.46 MB	249.06 KB
/wp-content/uploads/2017/09/OfertaEmpleoIndra.pdf	5	0	865.04 KB	173.01 KB

Figura 14: Ficheros descargados durante el mes de Octubre

## 5.2. Popularidad de ficheros y páginas, descargas más frecuentes

Las descargas más frecuentes las acabamos de dejar cubiertas en el punto anterior. Y también el estudio detallado de la popularidad de los ficheros, páginas junto con sus tamaños y descargas los tenemos cubiertos en el **Apartado 4.3**.

### 5.3. Visitantes

Aquí vamos a comentar un poco los visitantes y la duración de las visitas. Cabe destacar que como las direcciones IP han sido anonimizadas, no podemos sacar un buen perfil de los accesos al servidor. Ya que por ejemplo dependiendo de como empieza la IP sabemos que puede pertenecer a un proveedor de Internet o a otro, también gracias a la IP podríamos conocer los países de acceso más frecuentes. Teniendo en cuenta que es la facultad de Ingeniería Informática de la Universidad de Valladolid, podemos suponer que los accesos más frecuentes se realizan desde España. Aunque podríamos ver si algún país está atacando nuestros servidores viendo esas direcciones IP diferentes.

la duración media de las visitas la podemos obtener de la siguiente tabla, generada a partir de la información obtenida de AWStats de las figuras (**Figura 33**, **Figura 34** y **Figura 35**) que se encuentran en el ANEXO II.

Mes	Visitas totales	Duración media (s)
Octubre	11.690	449
Noviembre	24.988	436
Diciembre	1.510	491
Total	38.188	458.67 $\simeq$ 8 min

Figura 15: Tabla con diferentes distribuciones de páginas servidas por el servidor

### 5.4. Patrones temporales de tráfico soportado por el servidor

Los patrones de acceso al servidor, en las diferentes horas del día ya los hemos tratado con anterioridad, los podemos encontrar en el **Apartado 4.1** en la **Figura 15**.

En lo respectivo a los patrones de tamaño y archivos por bytes, los hemos tratado con anterioridad en el **Apartado 4.2** y en las figuras: **Figura 5**, **Figura 6** y **Figura 7**.

### 5.5. Caracterización de las respuestas del servidor

Tratado un poco en el primer apartado del análisis de utilización, **Apartado 4.1**. Ahí vemos cuales son los patrones de acceso al servidor en función de las horas del día y de los días de la semana, basándonos en las peticiones realizadas correctamente a nuestro servidor (código 200).

## 6. Caracterización de la carga. partición y modelo.

Normalmente los analistas de rendimiento desea como se van a comportar uno o varios sistemas informáticos ante una carga de trabajo determinada. Lo usual es que esta carga de trabajo se desconozca. normalmente se utilizan modelos de carga, que representan de alguna manera una aproximación e la carga real o futura de un cierto sistema, en unas determinadas condiciones.

Además los modelos de carga permiten la portabilidad y reproducción de las necesidades en diferentes entornos.

Estos modelos de carga se extraen a partir de una caracterización de cargas anteriores o de especulaciones de las mismas. Una vez que tenemos determinado un modelo de carga para un sistema, podemos ver como varía el comportamiento de un sistema dependiendo de los cambios que se produzcan en la carga, esto lo hacemos mediante los modelos de carga. Tenemos que tener presentes algunas definiciones antes de empezar a explicar el proceso de caracterización.

- **Carga** (workload): todas las demandas que realizan los usuarios a un sistema informático durante un periodo de tiempo. En nuestro caso son peticiones a un servidor web durante unos 48 días.
- **Caracterización de la carga**: proceso por el cual se define un modelo de carga que reproduce lo mejor posible las características de una carga real. El modelo de carga se establece en función de los parámetros principales que afecten al sistema a nivel de rendimiento y también de los objetivos que se persiguen. Este proceso se hace por medio de parámetros cuantitativos y funciones.

Ahora que ya sabemos lo que es un modelo de carga y para que se utilizan, vamos a describir el proceso seguido para extraer nuestro modelo de carga.

## 6.1. Filtrado inicial

Primero, acorde con lo que dice el enunciado, tenemos que filtrar el fichero de log original. Este fichero contiene todas las peticiones al servidor web de la escuela durante un cierto periodo de tiempo, si analizamos todas las peticiones, el modelo de carga no va a ser representativo, porque por ejemplo no deberían de interesar para el rendimiento de nuestro sistemas las peticiones que se han completado de manera errónea (códigos 400). Es por ello que primero tenemos que filtrar en base a:

- **GET**: vamos a quedarnos únicamente con las peticiones de tipo GET, que son las que descargan algún tipo de información del servidor.
- **Código 2XX**: quiere decir que nos vamos a quedar con los códigos de respuesta de la serie 200, es decir 200 (OK), 206 (PARTIAL-CONTENT), etc.
- **Código 3XX**: quiere decir que nos quedamos con los códigos de respuesta de la serie 300, estos códigos normalmente son de redirecciones, tanto permanentes como temporales (301, 308, etc).

Este filtrado inicial se ha realizado mediante un pequeño programa python que a grandes rasgos, lo que hace es declarar 3 expresiones regulares y va filtrando los datos por líneas. Leemos el fichero inicial que es el esi.log (proporcionado en la práctica), lee por líneas, primero miramos si es una petición de tipo GET o POST, si es de tipo GET entonces miramos si tiene un código de tipo 200 o de tipo 300, si es de uno de esos tipos se escribe esa línea en un fichero llamado resultado.log, que será el que luego tomemos como entrada para los diferentes programas a la hora de hacer la caracterización de la carga. El programa python es el siguiente (al ser simplemente 30 líneas de programa, lo introducimos aquí ya que es un programa bastante necesario e importante y simple):

---

```
import re
import os
```

```

import os.path as path

# primero compilamos los filtros necesarios para las expresiones
# regulares
get = re.compile('GET')
get200 = re.compile(' 2[0-9][0-9] ')
get300 = re.compile(' 3[0-9][0-9] ')

# abrimos los ficheros necesario, en este caso el log de lectura y el
# resultados en escritura
f = open("esi.log")
# si ya existe nuestro fichero, lo borramos para asegurarnos de que los
# datos que procesamos son los correctos
if path.exists("resultados.log"):
    os.remove("resultados.log")
r = open("resultados.log", "w")

# filtramos el fichero de log
for linea in f:
    # para cada linea comprobamos, que sea una peticion de tipo get y si
    # es 2XX o 3XX
    if (get.search(linea) != None):
        if (get200.search(linea) != None):
            r.write(linea)
        if (get300.search(linea) != None):
            r.write(linea)

```

---

### Filtrado inicial de las peticiones: GET, 200 y 300

## 6.2. Preparación de los datos para el análisis estadístico

Primero cabe destacar que el fichero original contiene 1.324.974 de líneas. Una vez que ha pasado el filtro, el fichero resultante contiene 1.249.701, que aproximadamente son 100.000 líneas menos. Conforme vayamos avanzando en el análisis veremos como se van reduciendo la cantidad de entradas en el fichero. Esto es importante ya que no es lo mismo analizar un millón y medio de datos, que simplemente 36.000, que son los que quedan al final aproximadamente.

Los parámetros que vamos a tener en cuenta para realizar nuestra caracterización de la carga son los siguientes:

- **Número de accesos:** para cada una de los diferentes ficheros servidos se va a calcular su cantidad de accesos (peticiones).
- **Tamaño medio:** para cada uno de los ficheros, se calculará la suma en bytes de todas sus peticiones y se calculará la media, en base al número de accesos. Este parámetro será obtenido de la columna de tamaño transferido del fichero de log.
- **Tiempo medio de ejecución:** para cada uno de los ficheros servidos se calculará el tiempo medio de ejecución. Este parámetro será calculado a partir del atributo *elapsed time* del nuestro fichero de log.

La caracterización de la carga que nosotros vamos a realizar se basa en métodos de agrupamiento aglomerativos jerárquicos, haremos una división en clusters con ayuda de un programa estadístico como StatGraphics o R, más adelante describiremos el proceso. Lo importantes es que la entrada de datos para estos programas tiene que ser de este tipo:

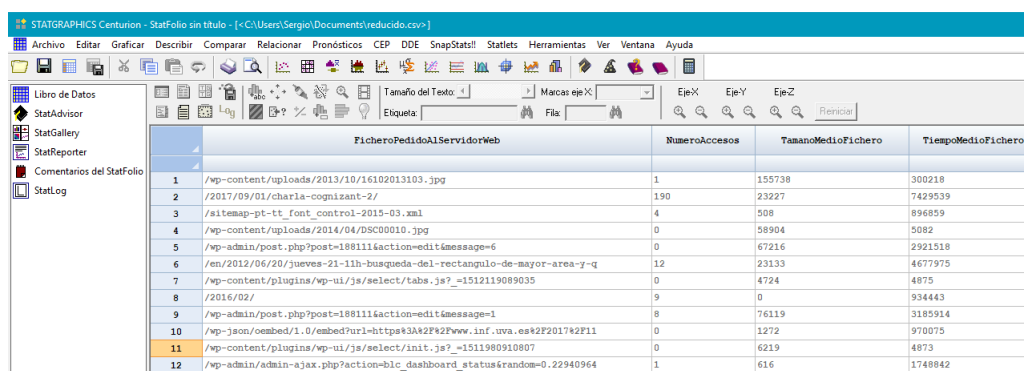
PáginaServida-Parámetros	Número accesos	Tamaño medio fichero	Tiempo medio ejecución
URLpagina1	dato1	dato2	dato3
URLpagina2	dato4	dato5	dato6
URLpagina3	dato7	dato8	dato9
URLpagina4	dato10	dato11	dato12

Figura 16: Tabla con el formato de entrada datos para programa estadístico

Para conseguir estos datos, primero hemos procesado el fichero resultados.log (peticiones filtradas) con un conjunto de scripts en bash y programas java bastante simples (aunque el principal es un poco largo). A grandes rasgos lo que hace nuestro programa en java es crear 3 diccionarios, en cada uno de ellos la clave del diccionario son las diferentes páginas web pedidas al servidor (únicas) y el valor son los parámetros que nosotros queremos. Tenemos un diccionario que tiene como valor los diferentes números de accesos para cada página web, otro diccionario que contiene el tamaño medio de cada una de las páginas servidas y otro diccionario que el valor para cada página servida es el tiempo medio de ejecución. El funcionamiento básico de este programa es ir leyendo los datos de las páginas web, e ir introduciendo en el diccionario los valores, si el fichero ya existe en el diccionario se actualiza su valor y si no, se introduce nuevo.

Al final del procesado de los datos con nuestro programa en java, lo que conseguimos es un fichero de extensión CSV, esta extensión nos permite importar los datos fácilmente como si fueran hojas de cálculo o como entrada de datos de stat grphcis o cualquier otro programa estadístico. El formato CSV simplemente separa los datos con comas o algún otro separador específico. Como nuestros datos también contienen comas (en las URL), nuestro separador es el símbolo , por lo tanto el formato de nuestro fichero, es el mostrado en la **Figura 16** pero en el formato CSV adecuado para StatGrpachis es como el que se muestra en la **Figura 17**.

Podemos encontrar todos los ficheros de procesado, tanto los scripts bash, python como java en el **ANEXO III**. Hemos explicado de manera breve el funcionamiento del programa Java principal, pero el tratamiento y adecuación de los datos requiere varios programas auxiliares simples.



	FicheroPedidoAlServidorWeb	NumeroAccesos	TamanoMedioFichero	TiempoMedioFichero
1	/wp-content/uploads/2013/10/16102013103.jpg	1	155738	300218
2	/2017/09/01/charla-cognizant-2/	190	23227	7429539
3	/sitemap-pt-tt_font_control-2015-03.xml	4	508	896859
4	/wp-content/uploads/2014/04/DSC00010.jpg	0	58904	5082
5	/wp-admin/post.php?post=188111&action=editmessage=6	0	67216	2921518
6	/en/2012/06/20/jueves-21-11h-busqueda-del-rectangulo-de-mayor-area-y-q	12	23133	4677975
7	/wp-content/plugins/wp-ui/js/select/tabs.js?_1512119089035	0	4724	4875
8	/2016/02/	9	0	934443
9	/wp-admin/post.php?post=188111&action=editmessage=1	8	76119	3185914
10	/wp-jacon/oembed/1.0/embed?url=https%3A%2F%2Fwww.inf.uva.es%2F2017%2F11	0	1272	970075
11	/wp-content/plugins/wp-ui/js/select/init.js?_1511980910807	0	6219	4873
12	/wp-admin/admin-ajax.php?action=blc_dashboard_status&random=0.22940964	1	616	1748842

Figura 17: Ejemplo entrada datos Stat Graphics

Cabe destacar que acabado esta preparación de los datos, en los que solo tenemos los ficheros *diferentes* que ha servido el servidor, solo tenemos un total de 34.575 entradas. Esto se debe a que, como podemos observar, muchas de las peticiones de las páginas web se repiten, por eso, con el análisis de conglomerados podemos observar cuales son las que más se repiten, las que menos y hacernos una idea de como es la carga que recibe el sistema.

Cabe destacar que aunque hemos reducido los datos casi un 97% (de 1.500.000 a 35.000) sigue siendo una entrada de datos demasiado grande para StatGpahics y cualquier otro programa de cómputo estadístico que manejemos. Por ejemplo hemos probado a realizar el calculo de los clusters en R, y cuando pedimos que calcule la matriz de distancias, nos informa de un error, y es que R no permite hacer los cálculos que le pedimos sobre matrices de mas de 4.4 Gb. Como la entrada es muy grande, hemos decidido tomar una muestra aleatoria simple de los datos de un tamaño aproximado de 10.000. Esta MAS lo que hace es coger uno de cada 5 filas de datos hasta llegar a 10.000 datos, así recorremos todo el archivo de datos cogiendo solo 10.000.

### 6.3. Análisis estadístico

Para el análisis estadístico, como ya se habrá supuesto, hemos utilizado el programa StatGrapchis. Simplemente hemos importado el CSV generado con nuestro programa java y el resultado, como ya hemos visto anteriormente, queda como se muestra en la **Figura 17**. Ahora simplemente le indicamos al programa los siguientes pasos: Describir - Métodos Multivariados - Análisis de Conglomerados. En la ventana emergente que nos indica las diferentes opciones para hacer el análisis de conglomerados, primero le decimos que los parámetros sobre los que queremos hacer el análisis son las 3 columnas de nuestro folio de datos: numero accesos, tamaño medio del fichero y tiempo medio de ejecución. Y seguidamente, en la ventana emergente para señalar las opciones del análisis introducimos:

- **Método:** vecino más lejano.
- **Métrica de distancia:** euclídeana al cuadrado.
- **Conglomerado:** observaciones.
- **Estandarizar:** si, debido a que los datos tienen mucha variabilidad, es casi obligatorio estandarizarlos para sacar datos coherentes y consistentes al análisis.
- **Número de conglomerados:** 5. Nos ha parecido adecuado, que apriori el número de conglomerados sea 5. Mostraremos si esta decisión ha sido buena o no, una vez hecho el análisis.

Con esos datos introducidos le damos a aceptar y el programa nos pregunta que datos de resultados queremos obtener, le indicamos que nos interesan:

- **Resumen de análisis.**
- **Dendograma.**
- **Gráfico de dispersión 2D.**
- **Gráfico de distancia de aglomeración.**

Con estos datos, le damos a aceptar, y dejamos que el programa realice los cálculos necesarios. El caso en cuestión, con 5 clusters y una entrada de datos de 10.000 filas, con 3 parámetros ha tardado en ser procesado aproximadamente 1 hora. Los resultados obtenidos son los siguientes:

#### 6.3.1. Resumen de la aglomeración

El resumen de la caracterización de la carga, se puede observar en la **Figura 18**.



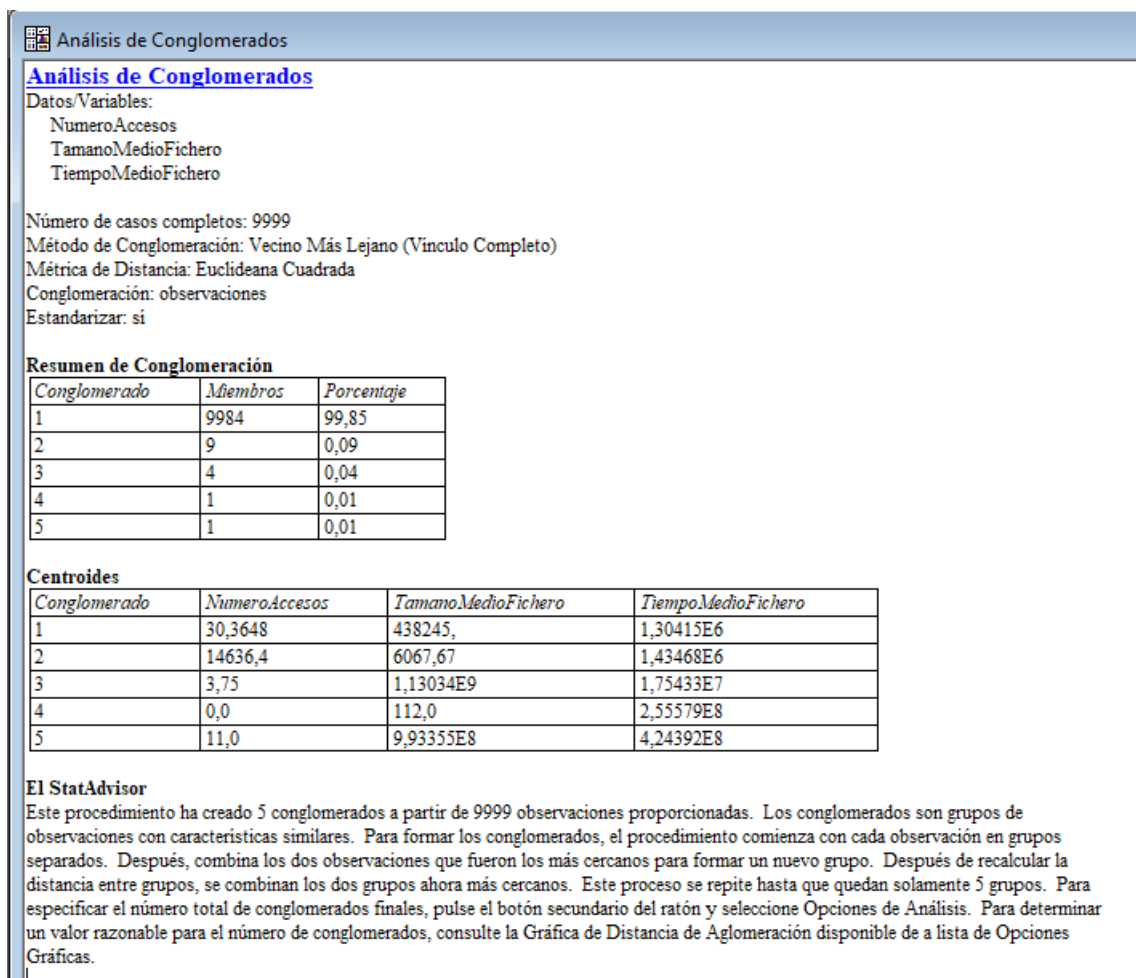


Figura 18: Resumen de la salida de Stat Grpahics al hacer los clusters

### 6.3.2. Gráficos resultantes

Algunos de los gráficos que nos ha parecido importante sacar son:

- **Dendograma:** para ver como se van formando los clusters hasta que se llega al número requerido. Gráfico accesible en la **Figura 19**.

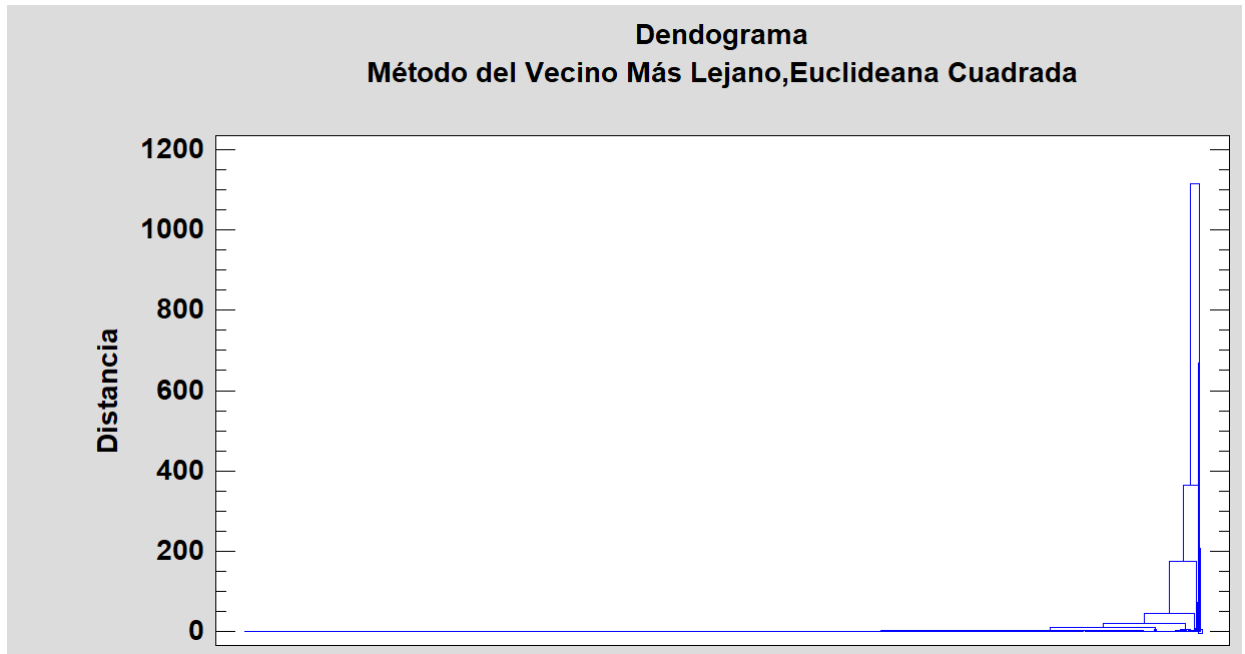


Figura 19: Gráfico Dendrograma - Clusters y formación

- **Gráfico de dispersión 2D:** para ver los datos, y como se agrupan en los diferentes clusters. La verdad es que queda un gráfico bastante poco intuitivo, pero son los resultados obtenidos. En las conclusiones se pasará a describir un poco estos resultados para entenderlos de una manera más amigable. Gráfico visible en **Figura ??**.

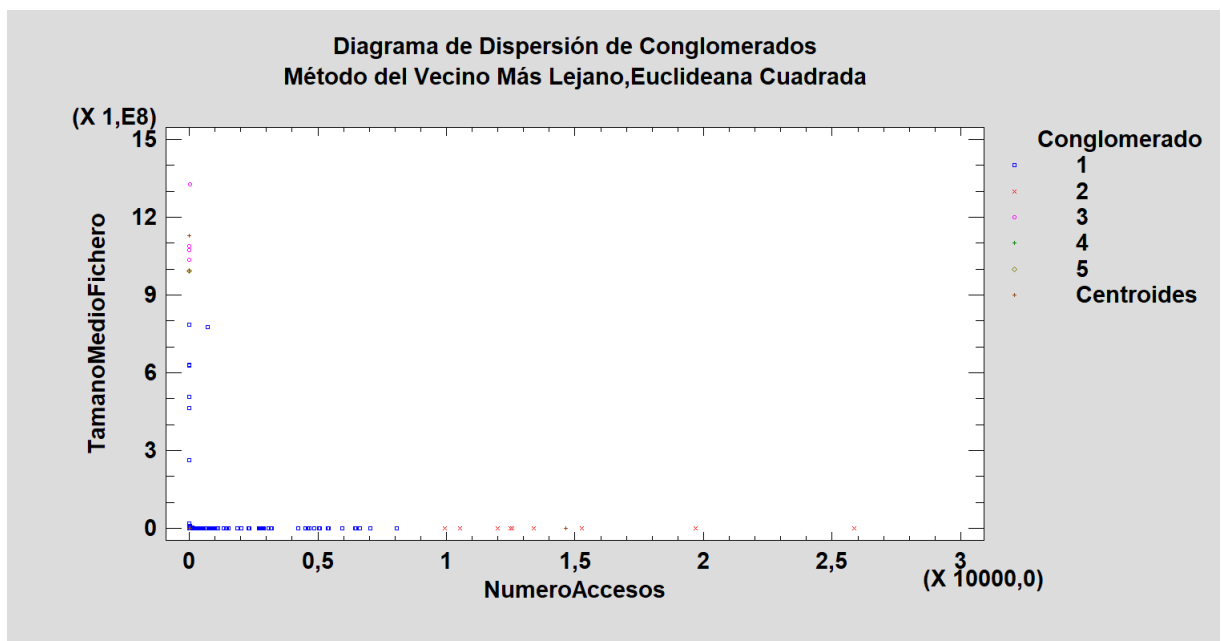


Figura 20: Gráfico de dispersión 2D donde se ven los diferentes clusters y sus centroides

- **Gráfico de distancias entre centroides:** en este gráfico se van mostrando las diferentes distancias entre los centroides de los diferentes clusters. Al igual que el dendrograma, no es

demasiado representativo, por temas de escala, pero se comentará brevemente a continuación. Gráfico accesible en **Figura 21**.

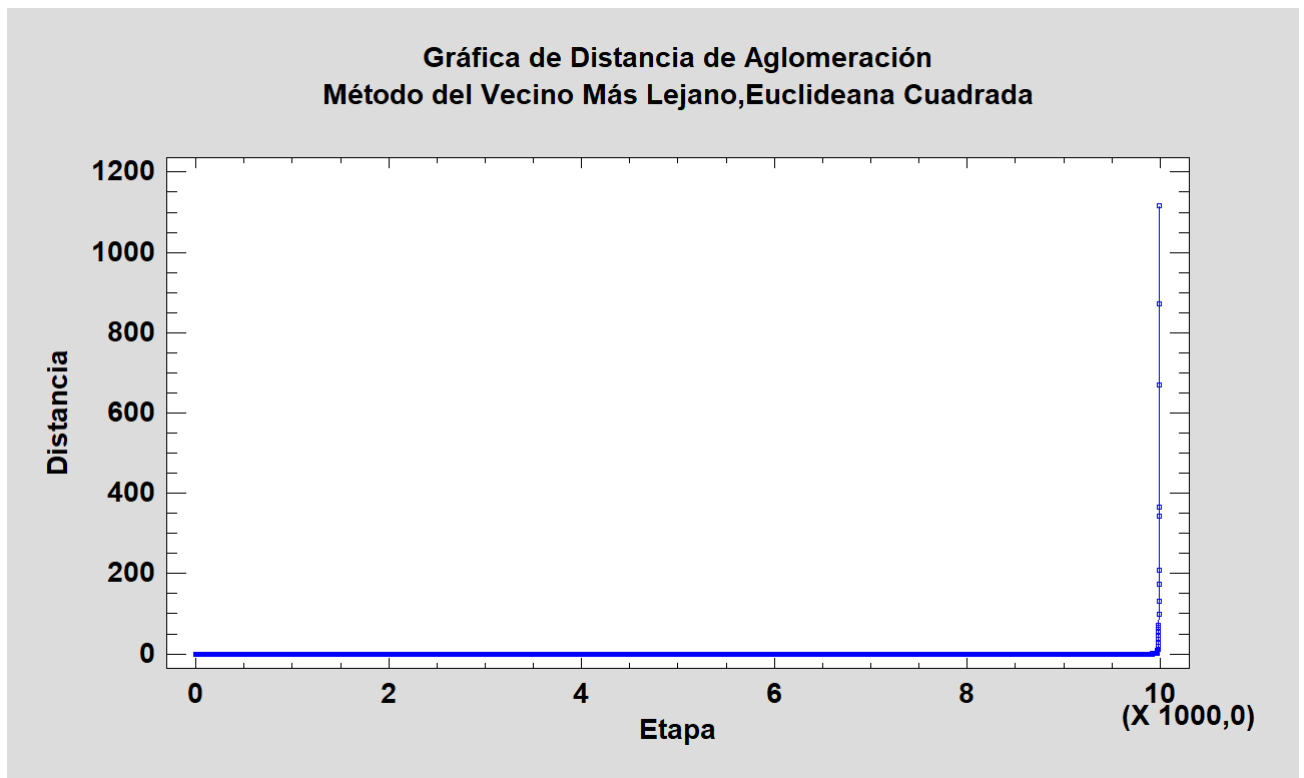


Figura 21: Gráfico de distancias entre centroides a lo largo del cálculo

Hecho esto ya tenemos nuestro modelo de carga aproximado a la realidad, según los datos proporcionados y según el análisis estadístico realizado con stat graphics.

## 7. Conclusiones

En primer lugar para las conclusiones vamos hacer una distinción. Principalmente vamos a basarnos en los tamaños de los ficheros, en el tipo y en el porcentaje de acceso a los mismos.

En base a los parámetros anteriores, por tamaño de ficheros ganan los ficheros de tipo vídeo. Si miramos la cantidad de accesos a estos ficheros no llega a veces ni al 1 % de los accesos totales, pero cuando se accede a uno de ellos el sistema se estresa mucho, sobre todo el disco debido a que la cantidad de Bytes que tiene que transferir es muy alta. En base a estos resultados podríamos potenciar esta transferencia cambiando los discos duros ópticos por discos duros de estado sólido. Es cierto que la relación capacidad de almacenamiento / precio, de estos tipos de discos es menor que la de los discos tradicionales, pero cada vez se aproxima más a ellos y no es tanta la diferencia. También podemos disminuir el tiempo de respuesta en servir estos tipos de ficheros si mejoramos las tarjetas de red y las comunicaciones de nuestro sistema.

En relación al rendimiento del servidor, si nos basamos en el porcentaje de accesos, aproximadamente un 40 % de los accesos a nuestro sistema se basan en petición de contenido dinámico, es

decir, que ejecutan una gran parte del código dentro del servidor. Podemos mejorar el tiempo de respuesta y la productividad de estas páginas aumentando recursos como la memoria o el procesador. Ya que la memoria suele ser una mejora más barata que el procesador, podríamos empezar por ahí. O podríamos poner varias máquinas en forma de sistema distribuido para que la carga se repartiese entre máquinas y no recayese todo sobre un único servidor.

las mejoras anteriores sobre el servidor no solo sirven para reducir el tiempo de respuesta, ni para aumentar la productividad de las páginas. un aumento de recursos en el sistema también nos lleva a que se puedan atender más peticiones por unidad de tiempo y también soportar una cantidad mayor de clientes de manera concurrente en nuestro sistema. Como hemos visto en las pruebas realizadas con los 230 hilos, los recursos del servidor se empiezan a agotar ya que son escasos. En nuestro caso hemos hecho pruebas pequeñas con un sistema reducido y pocos clientes accediendo al sistema. Si nos vamos a las grandes organizaciones, en las que tienen millones de clientes accediendo a los recursos a la vez, tener una gran cantidad de recursos y bien optimizados es primordial.

Vamos a comentar un poco los resultados obtenidos en la parte de la caracterización de la carga. Como hemos dicho inicialmente, hemos escogido 5 clusters, que aproximadamente creíamos que era una división aproximada. La verdad es que según los resultados obtenidos, la mayoría de los datos se agrupan en una clase (99.85 % de los datos). Esta clase tiene 9984 componentes y los datos los podemos ver en la **Figura 18**. El tamaño medio del fichero en esta clase es de 438.245 Bytes, que en términos más amigables son 0.43 Mb, lo que indica un tamaño de fichero pequeño, lo que suelen ser una página web con unas cuantas imágenes, y lo que demanda normalmente la gente por lo que vemos.

El segundo cluster agrupa 9 miembros, representando el 0.09 % de la población total. Con un tamaño medio de fichero bastante pequeño en comparación al primer cluster: 6 Kb.

El tercer y el quinto cluster son también bastante representativos, ya que nos indican la información a cerca de los ficheros de gran tamaño, como pueden ser los ficheros de máquinas virtuales (OVA) o los vídeos, que tienen un tamaño medio bastante superior al cluster 1. El tercer cluster tiene un tamaño medio de 13 Gb y también el tamaño medio del cluster número 5 tiene un tamaño medio de 9 Gb. Evidentemente el tiempo de ejecución medio de los ficheros de gran tamaño es mucho superior que los de tamaño pequeño. Concretamente es del orden de 2 magnitudes superior a los clusters principales.

## **8. Bibliografía**

- [1] J.M. Marqués. Transparencias de la asignatura Evaluación de Sistemas Informáticos.
- [2] AWStats. [http://www.awstats.org/docs/awstats\\_glossary.html](http://www.awstats.org/docs/awstats_glossary.html) .
- [3] Molero. Evaluación y Modelado del rendimiento de los Sistemas Informáticos.
- [4] StatGraphics. <https://www.statgraphics.net/tutoriales/>.

## ANEXO I.

Programas para el análisis de los datos en la parte de análisis de los ficheros de registro del servidor Apache. Simplemente voy a poner un ejemplo de los scripts utilizados, ya que nos caracterizamos por usar nuestros propios programas para el análisis estadístico, sino aquí quedaría excesivamente largo. La base de todos los scripts, por ejemplo en esta parte es: filtrar los documentos dependiendo de la extensión y juntar la información relacionada en un fichero con el tipo. Por ejemplo todos los ficheros con extensiones .html, .htm o .xml añaden la información que nosotros queremos (en unos casos bytes, en otros rutas, en otros tiempo) al fichero HTML, luego se procesa ese fichero y se extrae la información. Normalmente medias. Un ejemplo de esos scripts es:

---

```
#!/bin/bash

mes=/Dec
# primero obtenemos todos los accesos para un TIPO de ficheros y los
  dejamos en el archivo temporal del tipo correspondiente
cat /var/log/apache2/uva.log | grep '.htm ' | grep $mes | grep ' 200 ' |
  cut -d ' ' -f 10 >> HTML
cat /var/log/apache2/uva.log | grep '.html ' | grep $mes | grep ' 200 ' |
  cut -d ' ' -f 10 >> HTML
cat /var/log/apache2/uva.log | grep '.xml ' | grep $mes | grep ' 200 ' |
  cut -d ' ' -f 10 >> HTML

cat /var/log/apache2/uva.log | grep '.jpg ' | grep $mes | grep ' 200 ' |
  cut -d ' ' -f 10 >> IMAGEN
cat /var/log/apache2/uva.log | grep '.png ' | grep $mes | grep ' 200 ' |
  cut -d ' ' -f 10 >> IMAGEN
cat /var/log/apache2/uva.log | grep '.gif ' | grep $mes | grep ' 200 ' |
  cut -d ' ' -f 10 >> IMAGEN
cat /var/log/apache2/uva.log | grep '.ico ' | grep $mes | grep ' 200 ' |
  cut -d ' ' -f 10 >> IMAGEN
cat /var/log/apache2/uva.log | grep '.jpeg ' | grep $mes | grep ' 200 ' |
  cut -d ' ' -f 10 >> IMAGEN
cat /var/log/apache2/uva.log | grep '.svg ' | grep $mes | grep ' 200 ' |
  cut -d ' ' -f 10 >> IMAGEN

cat /var/log/apache2/uva.log | grep '.mp3 ' | grep $mes | grep ' 200 ' |
  cut -d ' ' -f 10 >> SONIDO

cat /var/log/apache2/uva.log | grep '.flv ' | grep $mes | grep ' 200 ' |
  cut -d ' ' -f 10 >> VIDEO
cat /var/log/apache2/uva.log | grep '.mp4 ' | grep $mes | grep ' 200 ' |
  cut -d ' ' -f 10 >> VIDEO

cat /var/log/apache2/uva.log | grep '.pdf ' | grep $mes | grep ' 200 ' |
  cut -d ' ' -f 10 >> ARCHIVO
cat /var/log/apache2/uva.log | grep '.doc ' | grep $mes | grep ' 200 ' |
  cut -d ' ' -f 10 >> ARCHIVO
cat /var/log/apache2/uva.log | grep '.docx ' | grep $mes | grep ' 200 ' |
  cut -d ' ' -f 10 >> ARCHIVO
cat /var/log/apache2/uva.log | grep '.pptx ' | grep $mes | grep ' 200 ' |
  cut -d ' ' -f 10 >> ARCHIVO
cat /var/log/apache2/uva.log | grep '.gz ' | grep $mes | grep ' 200 ' |
  cut -d ' ' -f 10 >> ARCHIVO
cat /var/log/apache2/uva.log | grep '.xlsx ' | grep $mes | grep ' 200 ' |
  cut -d ' ' -f 10 >> ARCHIVO
cat /var/log/apache2/uva.log | grep '.ppsx ' | grep $mes | grep ' 200 ' |
  cut -d ' ' -f 10 >> ARCHIVO
```

```

cat /var/log/apache2/uva.log | grep '.css ' | grep $mes | grep ' 200 ' |
cut -d ' ' -f 10 >> FORMATO

cat /var/log/apache2/uva.log | grep '.woff ' | grep $mes | grep ' 200 ' |
cut -d ' ' -f 10 >> OTRO

cat /var/log/apache2/uva.log | grep '.ova ' | grep $mes | grep ' 200 ' |
cut -d ' ' -f 10 >> OTRO

cat /var/log/apache2/uva.log | grep '.js ' | grep $mes | grep ' 200 ' |
cut -d ' ' -f 10 >> DINAMICA
cat /var/log/apache2/uva.log | grep '.php ' | grep $mes | grep ' 200 ' |
cut -d ' ' -f 10 >> DINAMICA

let sum=0
let dias=31
for i in $(cat HTML | sort | uniq)
do
    if [ "$i" != "-" ]; then
        let sum=$sum+$i
    fi
done
let sum=$sum/$dias
echo Media peticiones dia HTML: $sum
rm HTML

let sum=0
for i in $(cat ARCHIVO | sort | uniq)
do
    if [ "$i" != "-" ]; then
        let sum=$sum+$i
    fi
done
let sum=$sum/$dias
echo Media peticiones dia ARCHIVOS: $sum
rm ARCHIVO

let sum=0
for i in $(cat FORMATO | sort | uniq)
do
    if [ "$i" != "-" ]; then
        let sum=$sum+$i
    fi
done
let sum=$sum/$dias
echo Media peticiones dia FORMATO: $sum
rm FORMATO

let sum=0
for i in $(cat IMAGEN | sort | uniq)
do
    if [ "$i" != "-" ]; then
        let sum=$sum+$i
    fi
done
let sum=$sum/$dias
echo Media peticiones dia IMAGEN: $sum
rm IMAGEN

let sum=0
for i in $(cat OTRO | sort | uniq)

```

```

do
    if [ "$i" != "-" ]; then
        let sum=$sum+$i
    fi
done
let sum=$sum/$dias
echo Media peticiones dia OTROS: $sum
rm OTRO

let sum=0
for i in $(cat SONIDO | sort | uniq)
do
    if [ "$i" != "-" ]; then
        let sum=$sum+$i
    fi
done
let sum=$sum/$dias
echo Media peticiones dia SONDIO: $sum
rm SONIDO

let sum=0
for i in $(cat VIDEO | sort | uniq)
do
    if [ "$i" != "-" ]; then
        let sum=$sum+$i
    fi
done
let sum=$sum/$dias
echo Media peticiones dia VIDEO: $sum
rm VIDEO

let sum=0
for i in $(cat DINAMICA | sort | uniq)
do
    if [ "$i" != "-" ]; then
        let sum=$sum+$i
    fi
done
let sum=$sum/$dias
echo Media peticiones dia DINAMICAS: $sum
rm DINAMICA

```

---

El programa en Java que se ha encargado de sacar la cantidad de accesos para las páginas diferentes es:

---

```

import java.util.Scanner;
import java.util.HashMap;
import java.util.Map;
import java.util.ArrayList;

public class FicherosDif {
    public static void writeMapContent (HashMap<String, Integer> map) {
        // recorremos el mapa para ver que se han anadido correctamente
        for (Map.Entry<String, Integer> entry : map.entrySet()) {
            String key = entry.getKey();
            Integer value = entry.getValue();
            System.out.println(">>" + key + " - " + value);
        }
    }
}

```



```

public static void writeArrayContent (ArrayList<String> array) {
    // recorremos el mapa para ver que se han anadido correctamente
    for (int i=0; i< array.size(); i++) {
        System.out.println("++" + array.get(i));
    }
}

public static void main (String [] args) {
    Scanner in = new Scanner(System.in);
    HashMap<String, Integer> contadorFicherosDiferentes = new HashMap
        <>();
    ArrayList<String> todosFicheros = new ArrayList<>();

    // leemos los datos de los ficheros diferentes y los metemos en
    // el mapa con valor 1
    int i = 0;
    for (i=0; i<1236591; i++) {
        if (i<43327)
            contadorFicherosDiferentes.put(in.nextLine(), 1);
        if (i>=43327)
            todosFicheros.add(in.nextLine());
    }

    // mostramos el contenido para ver si es correcto
    // writeMapContent(contadorFicherosDiferentes);
    // mostramos el contenido para ver si es correcto
    // writeArrayContent(todosFicheros);

    for (i=0; i< todosFicheros.size();i++) {
        if (contadorFicherosDiferentes.containsKey(todosFicheros.get(
            i))) {
            contadorFicherosDiferentes.put(todosFicheros.get(i),
                contadorFicherosDiferentes.get(todosFicheros.get(i))+1)
            ;
        }
    }
    int contador = 0;
    for (Map.Entry<String, Integer> entry :
        contadorFicherosDiferentes.entrySet()) {
        Integer value = entry.getValue();
        if (value == 1)
            contador++;
    }
    System.out.println("Ficheros diferentes accedidos una sola vez: "
        + contador);
}
}

```

---

Este fichero depende de otro archivo generado por un .bash, que se encarga de sacar a un fichero temporal la información que procesa el JAVA. El java simplemente mantiene un diccionario de las páginas pedidas y el valor del diccionario es la cantidad de accesos, esa información se obtiene de:

---

```

#!/bin/bash

# PETICIONES DIFERENTES: simplemente las ordenamos y quitamos las
# repetidas
echo 'PETICIONES DIFERENTES'
cat /var/log/apache2/uva.log | grep /2017 | grep ' 200 ' | cut -d ' ' -f
7 | sort | uniq | wc -l

```

```

# temporal para el java
cat /var/log/apache2/uva.log | grep /2017 | grep ' 200 ' | cut -d ' ' -f
7 | sort | uniq >> temporalJava

# PETICIONES TOTALES: simplemente es contar las lineas que tiene el
  fichero
echo 'PETICIONES TOTALES'
cat /var/log/apache2/uva.log | grep ' 200 ' | cut -d ' ' -f 7 | wc -l
# temporal para el java
cat /var/log/apache2/uva.log | grep ' 200 ' | cut -d ' ' -f 7 >>
  temporalJava

# BYTES DIFERENTES
echo 'BYTES DIFERENTES'
cat /var/log/apache2/uva.log | grep /2017 | grep ' 200 ' | cut -d ' ' -f
10 | sort | uniq >> bytes
let sum=0
for i in $(cat bytes)
do
    if [ "$i" != "-" ]; then
        let sum=$sum+$i
    fi
done
echo $sum
rm bytes

# BYTES TOTALES
echo 'BYTES TOTALES'
cat /var/log/apache2/uva.log | grep /2017 | grep ' 200 ' | cut -d ' ' -f
10 >> bytes
let sum=0
for i in $(cat bytes)
do
    if [ "$i" != "-" ]; then
        let sum=$sum+$i
    fi
done
echo $sum
rm bytes

```

---

## ANEXO II.

Programas de preparación de datos para la parte de modelado de la carga:

### ■ Filtrado de peticiones GET y códigos 200 y 300 con python:

---

```

import re
import os
import os.path as path

# primero compilamos los filtros necesarios para las expresiones
  regulares
get = re.compile('GET')
get200 = re.compile(' 2[0-9][0-9] ')
get300 = re.compile(' 3[0-9][0-9] ')

# abrimos los ficheros necesario, en este caso el log de lectura y el
  resultados en escritura

```

```

f = open("esi.log")
# si ya existe nuestro fichero, lo borramos para asegurarnos de que los
  datos que procesamos son los correctos
if path.exists("resultados.log"):
    os.remove("resultados.log")
r = open("resultados.log", "w")

print("Analizando el fichero y generando resultados ...")

#filtramos el fichero de log
for linea in f:
    # para cada linea comprobamos, que sea una peticion de tipo get y si
      es 2XX o 3XX
    if (get.search(linea) != None):
        if (get200.search(linea) != None):
            r.write(linea)
        if (get300.search(linea) != None):
            r.write(linea)

print("Fichero analizado, resultados en: resultados.log")

```

---

#### ■ Preparación de datos adicional para el tratamiento con los ficheros java Secundarios:

```

#!/bin/bash

# procesamos para calcular el numero de accesos
echo "Generando fichero para Numero de Accesos"
cat resultados.log | cut -d " " -f 7 >> todasDirecciones.data
echo "Fichero Numero Accesos Generado: todasDirecciones.data"

# procesamos para el tama o medio de los ficheros
echo "Generando fichero para Tama o Medio"
cat resultados.log | cut -d " " -f 10 >> todosTama os.data
echo "Fichero Tama os Medios Generado: todosTamanos.data"

# procesamos para el tiempo de ejecucion de cada fichero
echo "Generando fichero para el tiempo de ejecucion"
cat resultados.log | cut -d " " -f 11 >> todosiTiempos.data
echo "Fichero Tama os Medios Generado: todosTamanos.data"

```

---

#### ■ Preprocesado de datos en Java 1:

```

import java.io.FileReader;
import java.io.FileWriter;
import java.io.PrintWriter;
import java.io.BufferedReader;

public class DirTam {
    public static void main (String [] args) {
        try {
            String lineaDir = null;
            String lineaTam = null;
            FileWriter file2 = new FileWriter("dirTam.data");
            PrintWriter printer = new PrintWriter(file2);
            FileReader file3 = new FileReader("todasDirecciones.data");
            FileReader file4 = new FileReader("todosTama os.data");
            BufferedReader brDir = new BufferedReader(file3);
            BufferedReader brTam = new BufferedReader(file4);

            while ((lineaDir = brDir.readLine()) != null) {
                lineaTam = brTam.readLine();
            }
        }
    }
}

```

```

        printer.println(lineaDir + " " + lineaTam);
    }

    file2.close();
    file3.close();
    file4.close();

} catch (Exception e){
    e.printStackTrace();
}

}

}

```

---

## ■ Preprocesado de datos en Java 2:

---

```

import java.io.FileReader;
import java.io.FileWriter;
import java.io.PrintWriter;
import java.io.BufferedReader;

public class DirTiem {
    public static void main (String [] args) {
        try {
            String lineaDir = null;
            String lineaTam = null;
            FileWriter file2 = new FileWriter("dirTiem.data");
            PrintWriter printer = new PrintWriter(file2);
            FileReader file3 = new FileReader("todasDirecciones.data");
            FileReader file4 = new FileReader("todosiTiempos.data");
            BufferedReader brDir = new BufferedReader(file3);
            BufferedReader brTam = new BufferedReader(file4);

            while ((lineaDir = brDir.readLine()) != null) {
                lineaTam = brTam.readLine();
                printer.println(lineaDir + " " + lineaTam);
            }

            file2.close();
            file3.close();
            file4.close();

        } catch (Exception e){
            e.printStackTrace();
        }
    }
}

```

---

## ■ Procesado final de datos y generación de CSV para StatGraphics:

---

```

import java.util.ArrayList;
import java.util.HashMap;
import java.math.BigInteger;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.PrintWriter;
import java.io.BufferedReader;

public class NumAccesos {
    public static void main (String [] args) {
        HashMap<String, Integer> numAccesos = new HashMap<>();
        HashMap<String, BigInteger> tamMedios = new HashMap<>();
    }
}

```

```

HashMap<String, BigInteger> tiemposMedios = new HashMap<>();

// procesado del n mero de accesos a cada fichero
try {
    String lineDir = null;
    FileReader fileDirecciones = new FileReader("todasDirecciones
        .data");
    BufferedReader buffer = new BufferedReader(fileDirecciones);
    while ((lineDir = buffer.readLine()) != null) {
        if (numAccesos.get(lineDir) == null) {
            numAccesos.put(lineDir, 0);
        } else {
            numAccesos.replace(lineDir, numAccesos.get(lineDir)
                +1);
        }
    }
    fileDirecciones.close();
} catch (Exception e) {
    e.printStackTrace();
}

// procesado del tama o medio de cada fichero
try {
    String lineDirTam = null;
    FileReader fileDirTam = new FileReader("dirTam.data");
    BufferedReader brDirTam = new BufferedReader(fileDirTam);
    while((lineDirTam = brDirTam.readLine()) != null) {
        String[] dirTam = lineDirTam.split(" ");
        if (tamMedios.get(dirTam[0]) == null) {
            if (dirTam[1].equals("-"))
                tamMedios.put(dirTam[0], BigInteger.ZERO);
            else
                tamMedios.put(dirTam[0], new BigInteger(dirTam
                    [1]));
        } else {
            if (dirTam[1].equals("-"))
                tamMedios.replace(dirTam[0], tamMedios.get(dirTam
                    [0]).add(BigInteger.ZERO));
            else
                tamMedios.replace(dirTam[0], tamMedios.get(dirTam
                    [0]).add(new BigInteger(dirTam[1])));
        }
    }

    // calculo del tama o medio
    tamMedios.forEach((k,v) -> {
        if (numAccesos.get(k) != 0) {
            tamMedios.replace(k, v.divide(new BigInteger(String.
                valueOf(numAccesos.get(k)))));
        }
    });
    fileDirTam.close();
} catch (Exception e) {
    e.printStackTrace();
}

// procesado del tiempo de ejecuci n de cada archivo
try {
    String lineDirTime = null;
    FileReader fileDirTime = new FileReader("dirTiem.data");
    BufferedReader brDirTime = new BufferedReader(fileDirTime);

```

```

while((lineDirTime = brDirTime.readLine()) != null) {
    String[] dirTime = lineDirTime.split(" ");
    if (tiemposMedios.get(dirTime[0]) == null) {
        if (dirTime[1].equals("-"))
            tiemposMedios.put(dirTime[0], BigInteger.ZERO);
        else
            tiemposMedios.put(dirTime[0], new BigInteger(
                dirTime[1]));
    } else {
        if (dirTime[1].equals("-"))
            tiemposMedios.replace(dirTime[0], tiemposMedios.
                get(dirTime[0]).add(BigInteger.ZERO));
        else
            tiemposMedios.replace(dirTime[0], tiemposMedios.
                get(dirTime[0]).add(new BigInteger(dirTime[1])
                ));
    }
}

// calculo del tiempo medio
tiemposMedios.forEach((k,v) -> {
    if (numAccesos.get(k) != 0) {
        tiemposMedios.replace(k, v.divide(new BigInteger(
            String.valueOf(numAccesos.get(k)))));
    }
});
fileDirTime.close();
} catch (Exception e) {
    e.printStackTrace();
}

// escribimos los resultados en un fichero en formato CSV el
// formato del archivo de salida que luego exportaremos a Excel
// es el siguiente:
// FicheroRequeridoAlServidor,NumeroAccesos,Tam oMedio,
// TiempoEjecucionMedio
// claveDiccionario,valor1,valor2

try {
    FileWriter ficheroSalida = new FileWriter("
        entradaStatGraphics.csv");
    PrintWriter csvPrinter = new PrintWriter(ficheroSalida);

    csvPrinter.println("FicheroPedidoAlServidorWeb<NumeroAccesos<
        Tama oMedioFichero<TiempoMedioFichero");
    tiemposMedios.forEach((k,v) -> {
        csvPrinter.println(k+"<"+numAccesos.get(k)+"<"+tamMedios.
            get(k)+"<"+tiemposMedios.get(k));
    });
} catch (Exception e) {
    e.printStackTrace();
}
}
}

```

---

# ANEXO III.

Mes	Visitantes distintos	Número de visitas	Páginas	Solicitudes	Tráfico
Ene 2017	0	0	0	0	0
Feb 2017	0	0	0	0	0
Mar 2017	0	0	0	0	0
Abr 2017	0	0	0	0	0
May 2017	0	0	0	0	0
Jun 2017	0	0	0	0	0
Jul 2017	0	0	0	0	0
Ago 2017	0	0	0	0	0
Sep 2017	0	0	0	0	0
Oct 2017	5,508	11,690	58,895	341,888	609.87 GB
Nov 2017	10,938	24,988	275,388	882,143	3081.60 GB
Dic 2017	925	1,510	8,756	32,768	273.03 GB
Total	17,371	38,188	343,039	1,256,799	3964.50 GB

Figura 22: Páginas servidas de manera correcta por el servidor

Códigos de error HTTP			
	Códigos de error HTTP*	Solicitudes	Porcentaje
301	Moved permanently (redirect)	8,939	63.8 %
404	Document Not Found (hits on favicon excluded)	2,093	14.9 %
403	Forbidden	1,967	14 %
302	Moved temporarily (redirect)	575	4.1 %
206	Partial Content	197	1.4 %
500	Internal server Error	196	1.4 %
421	Unknown error	16	0.1 %
405	Method not allowed	10	0 %
503	Server busy	2	0 %

Figura 23: Páginas servidas de manera incorrecta por el servidor

Día	Páginas	Solicitudes	Tráfico
Lun	1,920	11,987	13.96 GB
Mar	1,910	11,490	22.32 GB
Mie	2,210	13,498	9.40 GB
Jue	2,606	15,552	4.28 GB
Vie	2,108	12,090	11.31 GB
Sab	1,360	6,632	68.14 GB
Dom	1,320	6,680	11.19 GB

Figura 24: Distribución páginas por días semana Octubre

Visitas por Horas	Páginas	Solicitudes	Tráfico	Visitas por Horas	Páginas	Solicitudes	Tráfico
00	2,056	12,210	2.80 GB	12	3,595	21,781	2.26 GB
01	1,911	9,651	33.98 GB	13	3,471	19,452	6.32 GB
02	1,657	9,094	25.38 GB	14	3,433	17,925	29.32 GB
03	1,250	7,521	2.81 GB	15	2,621	14,516	11.89 GB
04	1,246	7,036	3.11 GB	16	2,622	18,836	246.48 GB
05	1,111	5,522	17.69 GB	17	2,699	18,728	78.14 GB
06	990	4,201	157.07 MB	18	2,910	20,553	1.79 GB
07	1,255	5,738	173.25 MB	19	2,771	17,856	46.93 GB
08	2,870	13,210	224.44 MB	20	2,594	17,626	20.77 GB
09	4,204	20,165	11.36 GB	21	1,991	12,613	5.74 GB
10	3,884	19,459	8.00 GB	22	1,978	13,977	12.88 GB
11	3,483	20,372	32.29 GB	23	2,293	13,846	9.18 GB

Figura 25: Distribución páginas por horas día Octubre

Día	Páginas	Solicitudes	Tráfico
Lun	8,060	29,528	62.40 GB
Mar	9,199	32,584	91.39 GB
Mie	9,356	31,860	101.76 GB
Jue	10,574	35,235	238.41 GB
Vie	10,723	30,913	90.34 GB
Sab	7,800	21,076	67.99 GB
Dom	8,150	22,564	33.07 GB

Figura 26: Distribución páginas por días semana Noviembre

Visitas por Horas	Páginas	Solicitudes	Tráfico	Visitas por Horas	Páginas	Solicitudes	Tráfico
00	9,918	32,277	867.54 GB	12	15,655	49,284	207.46 GB
01	8,755	27,333	6.53 GB	13	16,267	48,427	177.16 GB
02	8,855	26,589	803.87 MB	14	17,358	48,553	189.97 GB
03	7,123	20,791	5.17 GB	15	12,579	40,331	119.62 GB
04	6,707	18,787	612.26 MB	16	13,860	48,918	54.63 GB
05	6,986	19,197	2.12 GB	17	13,347	46,791	32.59 GB
06	6,580	16,402	5.47 GB	18	13,526	47,688	248.11 GB
07	7,314	19,512	88.95 GB	19	12,391	43,316	131.62 GB
08	12,904	38,211	118.04 GB	20	11,074	39,476	16.92 GB
09	15,305	46,759	90.26 GB	21	10,066	35,302	4.48 GB
10	14,639	49,324	36.70 GB	22	10,199	35,864	61.37 GB
11	13,927	48,901	124.39 GB	23	10,053	34,110	471.11 GB

Figura 27: Distribución páginas por horas día Noviembre

Día	Páginas	Solicitudes	Tráfico
Lun	0	0	0
Mar	0	0	0
Mie	0	0	0
Jue	0	0	0
Vie	1,074	4,037	54.33 GB
Sab	603	2,208	277.77 MB
Dom	72	307	8.67 MB

Figura 28: Distribución páginas por días semana Diciembre

Visitas por Horas	Páginas	Solicitudes	Tráfico	Visitas por Horas	Páginas	Solicitudes	Tráfico
00	425	1,808	68.44 MB	12	396	1,682	5.81 GB
01	429	1,746	650.99 MB	13	434	1,684	14.34 GB
02	312	1,259	43.99 MB	14	434	1,381	41.76 MB
03	225	738	32.67 MB	15	431	1,890	173.22 GB
04	209	865	798.66 MB	16	512	1,465	1.26 GB
05	265	935	796.83 MB	17	450	1,724	41.02 MB
06	137	351	1.49 GB	18	403	1,591	32.00 MB
07	166	327	14.22 MB	19	487	1,565	35.77 MB
08	208	809	18.30 GB	20	345	1,623	13.89 GB
09	375	1,558	2.50 GB	21	349	1,412	877.36 MB
10	513	2,061	48.21 MB	22	289	1,051	483.98 MB
11	655	2,081	38.31 GB	23	307	1,162	43.95 MB

Figura 29: Distribución páginas por horas día Diciembre

File type		Hits	Percent	Bandwidth	Percent
File type					
js	JavaScript file	74,309	21.7 %	358.62 MB	0.1 %
jpg	Image	73,347	21.4 %	1.50 GB	0.4 %
png	Image	65,526	19.1 %	2.04 GB	0.6 %
php	Dynamic PHP Script file	57,842	16.9 %	782.80 MB	0.2 %
css	Cascading Style Sheet file	43,793	12.8 %	50.73 MB	0 %
pdf	Adobe Acrobat file	10,779	3.1 %	1.95 GB	0.5 %
gif	Image	5,526	1.6 %	2.77 MB	0 %
ico	Image	3,570	1 %	1.90 MB	0 %
woff	Font file	2,467	0.7 %	47.14 MB	0 %
xml	HTML or XML static page	1,331	0.3 %	880.53 KB	0 %
jpeg	Image	1,314	0.3 %	45.58 MB	0 %
Unknown		818	0.2 %	2.30 MB	0 %
doc	Document	248	0 %	6.56 MB	0 %
gz	Archive	183	0 %	195.56 KB	0 %
docx	Document	176	0 %	9.20 MB	0 %
svg	Image	141	0 %	459.22 KB	0 %
swf		79	0 %	2.12 MB	0 %
ttf	TrueType scalable font file	72	0 %	1.32 MB	0 %
flv		66	0 %	9.67 GB	2.8 %
htm	HTML or XML static page	56	0 %	282.84 KB	0 %
eot	Font file	40	0 %	777.03 KB	0 %
mp4	Video file	36	0 %	305.45 GB	90.6 %
html	HTML or XML static page	30	0 %	43.64 KB	0 %
map		29	0 %	55.34 KB	0 %
pptx	Document	28	0 %	6.15 MB	0 %
json	JavaScript Object Notation file	26	0 %	513.14 KB	0 %
ppt	Document	16	0 %	105.48 MB	0 %
ova		14	0 %	15.02 GB	4.4 %
xlsx	Document	11	0 %	172.43 KB	0 %
ppsx		8	0 %	3.56 MB	0 %
mp3	Audio file	7	0 %	0	0 %

Figura 30: Distribución tipos de páginas web visitadas en Octubre



File type		Hits	Percent	Bandwidth	Percent
File type					
php	Dynamic PHP Script file	273,909	31 %	4.32 GB	0.2 %
jpg	Image	153,085	17.3 %	3.42 GB	0.2 %
png	Image	141,997	16 %	4.61 GB	0.2 %
js	JavaScript file	109,622	12.4 %	437.95 MB	0 %
xml	HTML or XML static page	93,802	10.6 %	89.27 MB	0 %
css	Cascading Style Sheet file	43,942	4.9 %	68.99 MB	0 %
pdf	Adobe Acrobat file	33,755	3.8 %	7.02 GB	0.4 %
gif	Image	12,146	1.3 %	6.01 MB	0 %
ico	Image	7,237	0.8 %	3.86 MB	0 %
woff	Font file	5,289	0.5 %	99.92 MB	0 %
jpeg	Image	3,286	0.3 %	114.42 MB	0 %
Unknown		829	0 %	3.64 MB	0 %
docx	Document	634	0 %	24.47 MB	0 %
doc	Document	588	0 %	16.60 MB	0 %
gz	Archive	432	0 %	457.44 KB	0 %
svg	Image	330	0 %	1.01 MB	0 %
htm	HTML or XML static page	264	0 %	769.33 KB	0 %
swf		208	0 %	5.66 MB	0 %
ttf	TrueType scalable font file	149	0 %	2.61 MB	0 %
flv		145	0 %	27.08 GB	1.6 %
json	JavaScript Object Notation file	108	0 %	2.08 MB	0 %
html	HTML or XML static page	75	0 %	43.57 KB	0 %
pptx	Document	70	0 %	12.18 MB	0 %
mp4	Video file	67	0 %	1547.73 GB	95.5 %
eot	Font file	60	0 %	1.16 MB	0 %
ova		28	0 %	24.03 GB	1.4 %
ppt	Document	27	0 %	133.96 MB	0 %
xlsx	Document	21	0 %	284.12 KB	0 %
ppsx		15	0 %	10.67 MB	0 %
map		11	0 %	82.63 KB	0 %
mp3	Audio file	11	0 %	0	0 %
xsl	Extensible Stylesheet Language file	1	0 %	1.69 KB	0 %

Figura 31: Distribución tipos de páginas web visitadas en Noviembre

File type		Hits	Percent	Bandwidth	Percent
File type					
php	Dynamic PHP Script file	8,660	26.4 %	123.93 MB	0 %
jpg	Image	7,792	23.7 %	167.57 MB	0.1 %
png	Image	7,107	21.6 %	240.80 MB	0.1 %
js	JavaScript file	4,267	13 %	12.58 MB	0 %
pdf	Adobe Acrobat file	2,029	6.1 %	387.76 MB	0.2 %
css	Cascading Style Sheet file	1,088	3.3 %	2.17 MB	0 %
gif	Image	597	1.8 %	260.38 KB	0 %
ico	Image	348	1 %	189.09 KB	0 %
woff	Font file	271	0.8 %	4.86 MB	0 %
xml	HTML or XML static page	174	0.5 %	143.09 KB	0 %
jpeg	Image	170	0.5 %	5.85 MB	0 %
Unknown		59	0.1 %	254.59 KB	0 %
doc	Document	41	0.1 %	1.47 MB	0 %
docx	Document	33	0.1 %	1.68 MB	0 %
gz	Archive	24	0 %	28.83 KB	0 %
ttf	TrueType scalable font file	19	0 %	377.97 KB	0 %
flv		16	0 %	3.21 GB	2.1 %
swf		16	0 %	447.88 KB	0 %
svg	Image	14	0 %	82.83 KB	0 %
mp4	Video file	13	0 %	140.44 GB	95.1 %
htm	HTML or XML static page	8	0 %	20.04 KB	0 %
pptx	Document	6	0 %	1.22 MB	0 %
html	HTML or XML static page	5	0 %	4.85 KB	0 %
ova		4	0 %	3.00 GB	2 %
xlsx	Document	2	0 %	31.57 KB	0 %
eot	Font file	2	0 %	41.66 KB	0 %
ppt	Document	1	0 %	0	0 %
mp3	Audio file	1	0 %	0	0 %
json	JavaScript Object Notation file	1	0 %	19.74 KB	0 %

Figura 32: Distribución tipos de páginas web visitadas en Diciembre

Visits duration		
Number of visits: 11,690 - Average: 449 s	Number of visits	Percent
0s-30s	8,079	69.1 %
30s-2mn	923	7.8 %
2mn-5mn	440	3.7 %
5mn-15mn	461	3.9 %
15mn-30mn	496	4.2 %
30mn-1h	691	5.9 %
1h+	600	5.1 %

Figura 33: Duración de las visitas en Octubre

Visits duration		
Number of visits: 24,988 - Average: 436 s	Number of visits	Percent
0s-30s	17,371	69.5 %
30s-2mn	2,003	8 %
2mn-5mn	932	3.7 %
5mn-15mn	947	3.7 %
15mn-30mn	1,087	4.3 %
30mn-1h	1,406	5.6 %
1h+	1,242	4.9 %

Figura 34: Duración de las visitas en Noviembre

Visits duration		
Number of visits: 1,510 - Average: 491 s	Number of visits	Percent
0s-30s	1,067	70.6 %
30s-2mn	79	5.2 %
2mn-5mn	39	2.5 %
5mn-15mn	42	2.7 %
15mn-30mn	73	4.8 %
30mn-1h	90	5.9 %
1h+	92	6 %
Unknown	28	1.8 %

Figura 35: Duración de las visitas en Diciembre