Yolanda Lillo Mata ISAM

**Práctica 4.**

**1. Comunicación cliente-servidor HTTP.**

Abre la captura http1.cap y responde a las siguientes preguntas:

1. Indica qué dirección IP es la de la máquina cliente HTTP y cuál la del servidor.

Cliente: 13.0.0.13

Servidor: 21.0.0.21

2. Indica que version HTTP están utilizando cliente/servidor.

Está usando HTTP versión 1.1.

3. Indica si están utilizando conexiones persistentes o no persistentes y el número de conexiones que se ven en el fichero de captura.

Son conexiones persistentes puesto que, solo se está llevando a cabo la conexión entre dos máquinas.

4. ¿Cuántas peticiones GET observas desde el cliente?

Se observan dos peticiones GET desde el cliente, una en el paquete 8 y otra en el paquete 16.

5. ¿Cuántas URLs crees que ha escrito el usuario en el navegador para obtener dicha captura?¿Cual/es? ¿Por qué?

La vemos en el paquete 8 que es el mensaje get.

[Full request URI: http://www1/index.html]

6. Fijate en el contenido de la página index.html que se ha descargado el cliente. ¿Qué crees que ocurrirá cuando el navegador se haya descargado index.html?

Como vemos en el paquete 16 se descargara una foto y después acaba la conexión.

Abre la captura http2.cap y responde a las siguientes preguntas:

7. Indica que version HTTP están utilizando cliente/servidor.

Esta utilizando la versión de HTTP 1.0 el cliente y la versión 1.1 el servidor.

8. Indica si están utilizando conexiones persistentes o no persistentes y el número de conexiones que se ven en el fichero de captura. ¿Por qué?

Es persistente, si fuese no persistente iniciaría un mensaje y lo terminaría y así sucesivamente pero vemos como en esta captura eso no pasa. Aunque por defecto la que envía el cliente sea no persistente pero al tener keep-alive es persistente.

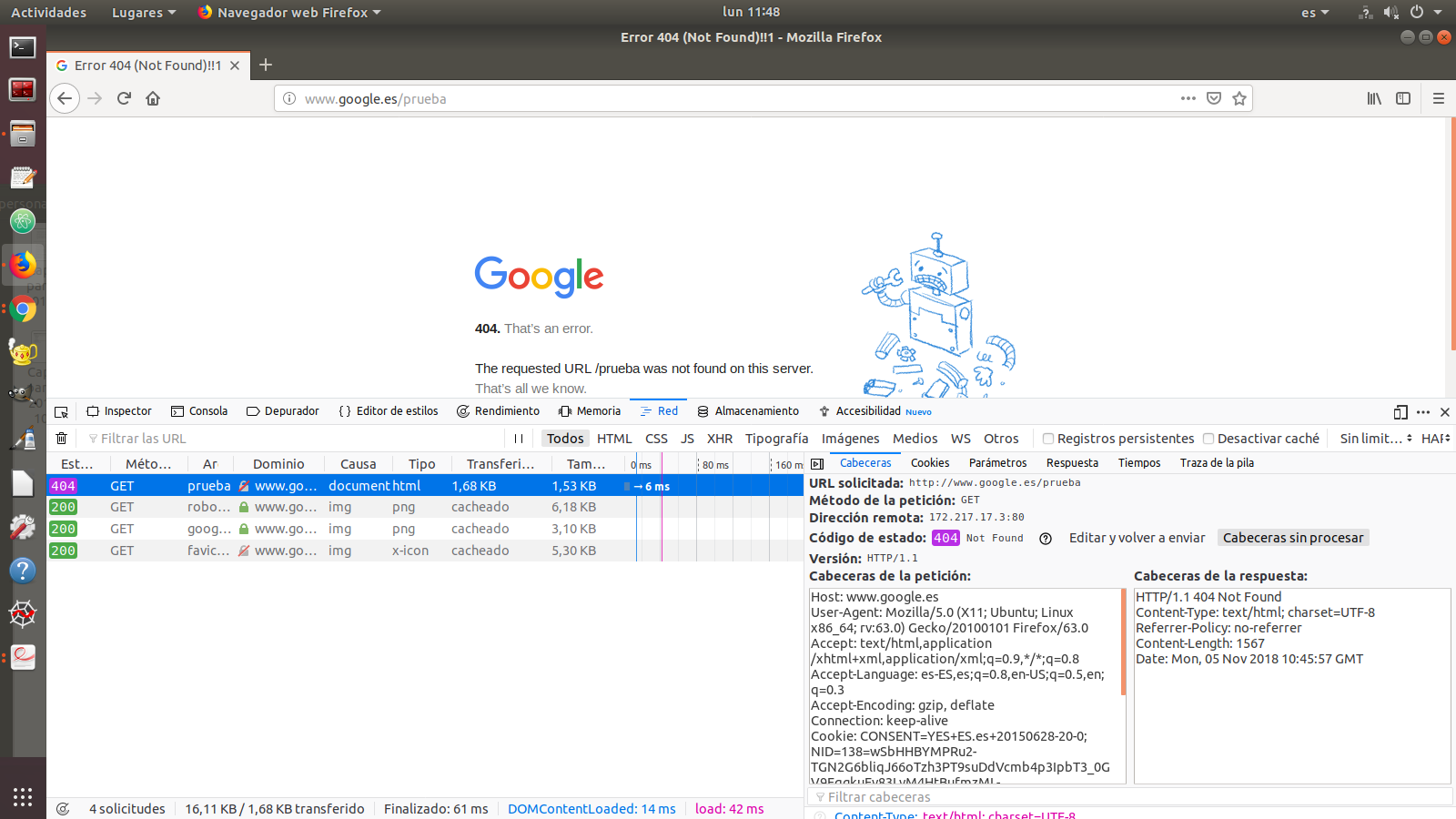
**2.Diferentes tipos de respuestas de un servidor.**

Abre una pestaña en el navegador y selecciona en el menú Herramientas → Desarrollador web → Alternar herramientas. Selecciona la pestaña Red y HTML:

Esta vista del navegador te permitirá observar todos los mensajes HTTP que se están intercambiando al cargar una página.

Explica lo que ocurre al cargar las siguientes URLs:

1. Dentro de esa pestaña carga la página http://www.google.es/prueba. Selecciona dentro de las herramientas del desarrollador la petición GET y la pestaña Cabeceras. Fı́jate en el campo Código de estado que indica el tipo de respuesta recibida y explica su contenido.

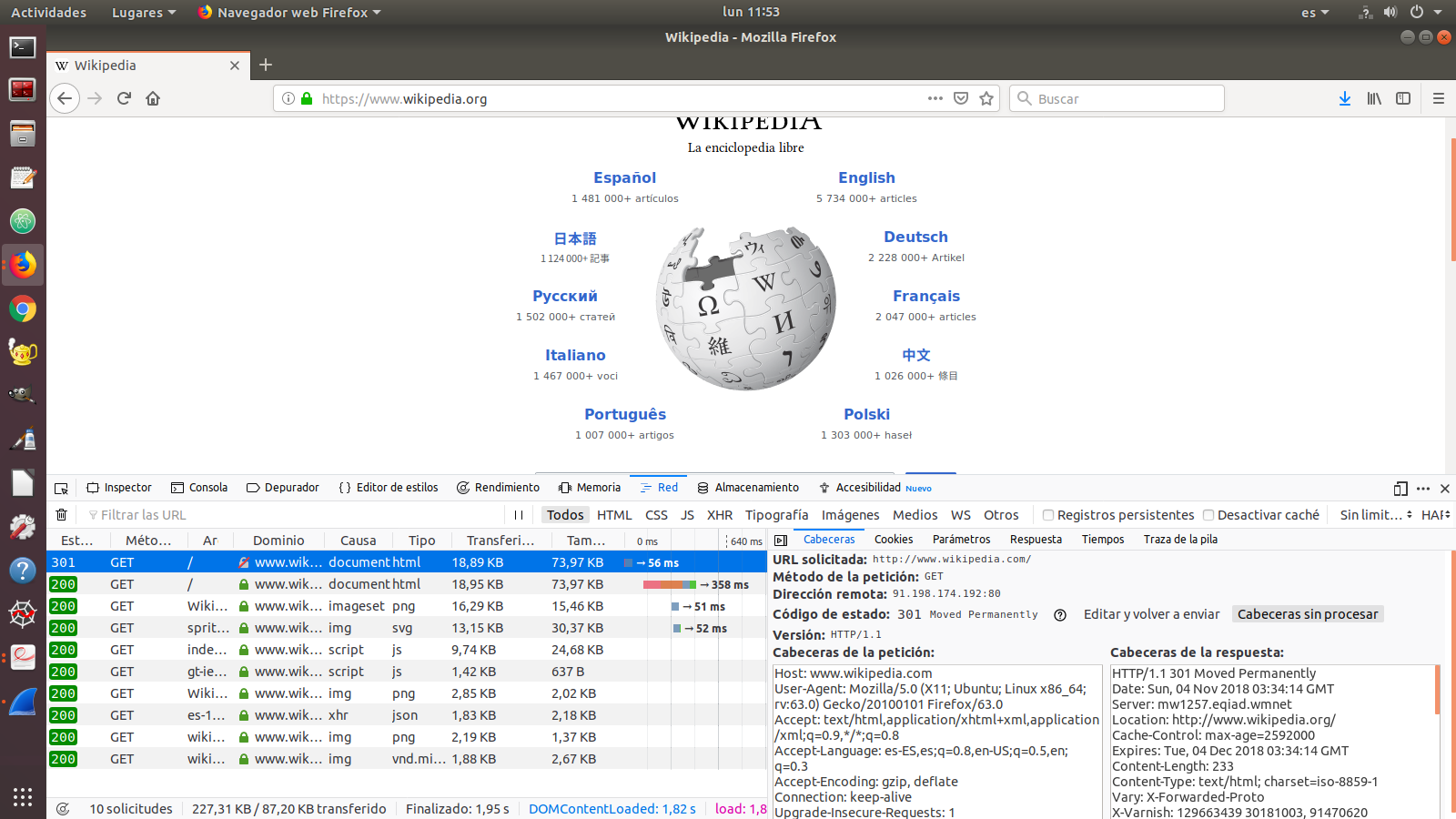


2. En esa misma pestaña carga la página http://www.wikipedia.com. Explı́ca qué ocurre en

la primera petición GET y a partir de las lı́neas de cabecera que ves en la respuesta, explica

la segunda petición GET. A la derecha del “Código de Estado”, pulsa en “Cabeceras sin

procesar” para ver la solicitud y la respuesta tal y como han sido transmitida



En la primera petición GET me aparece error 301 que indica que el host ha sido

capaz de comunicarse con el servidor pero que el recurso solicitado ha sido

movido a otra dirección permanentemente.

En el segundo GET ya se ha redirigido y el servidor va enviando los recursos de

wikipedia.

**3.Formularios en HTTP.**

Abre la captura http3.cap y responde a las siguientes preguntas:

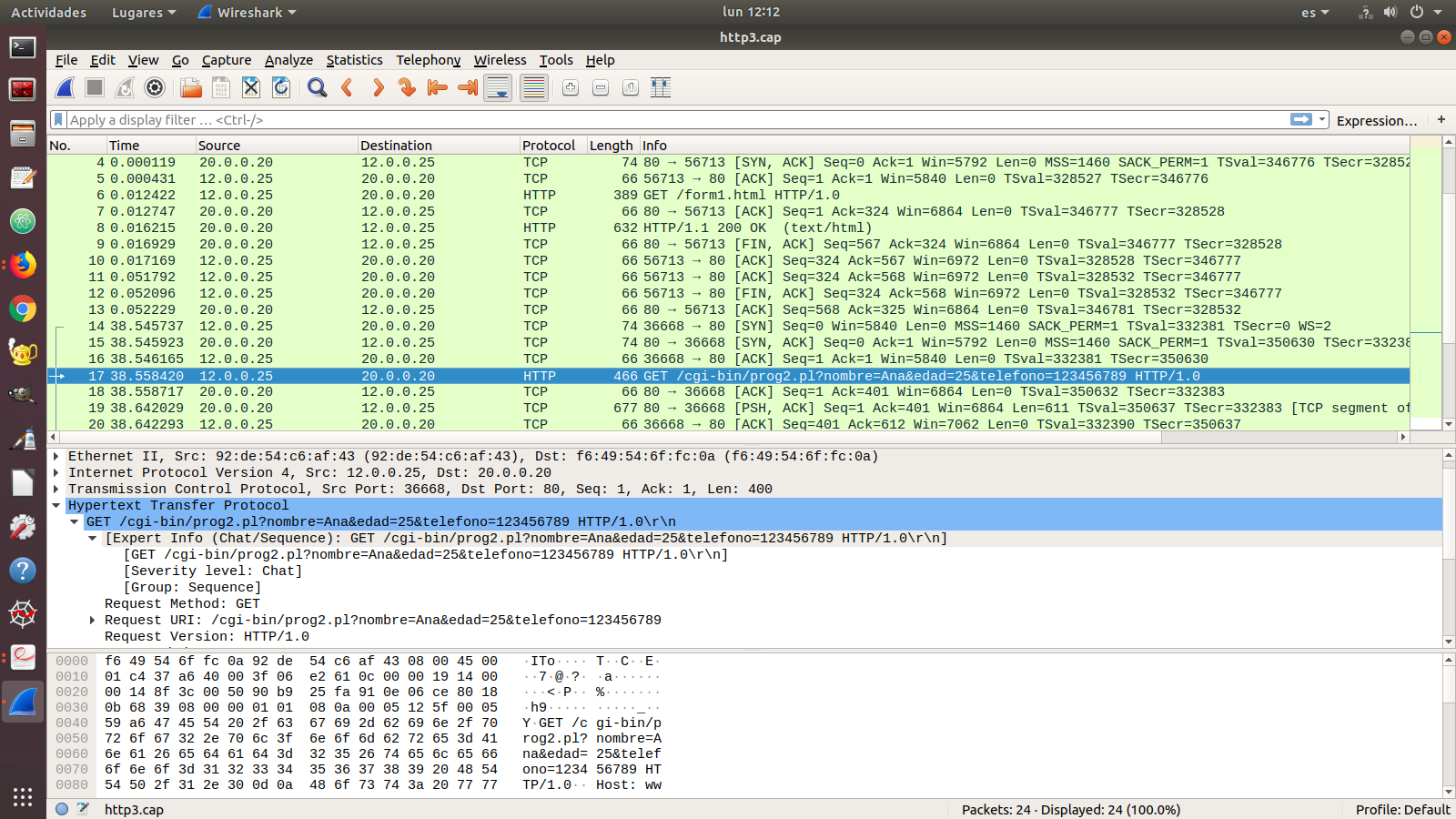
1. Indica qué tipo de conexiones HTTP utilizan el cliente y servidor e indica el número de

conexiones entre cliente y servidor que aparecen en la captura.

Cliente: 12.0.0.25 con versión 1.0

Servidor: 20.0.0.20 con versión 1.1

2. Busca en la captura el segmento donde el servidor le envı́a al cliente un formulario. Indica los nombres de los campos del formulario que rellenará el usuario.



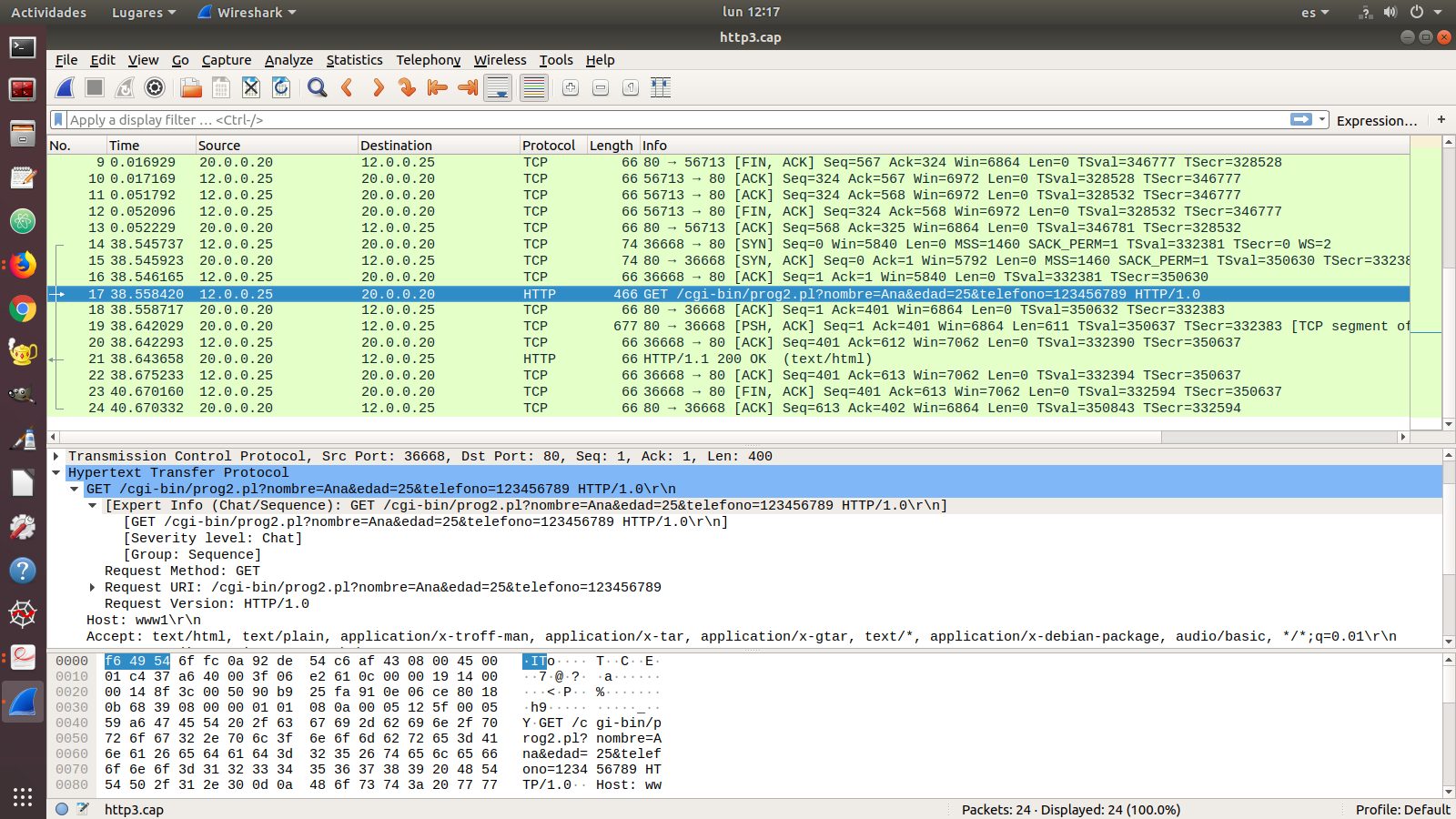
Vemos como que a lo que rellenan es el nombre, edad y teléfono.

3. Indica si es el cliente o el servidor el que decide cómo debe enviar el cliente los datos del

formulario (GET/POST) . ¿Qué método están usando en este caso? ¿Cómo lo sabes?

Está usando el método GET. Vemos que en la cabecera observamos “method\_GET”.

4. Busca en la captura el segmento donde el cliente le envı́a los datos del formulario al servidor y comprueba que se está realizando con el método GET.



Rellena el nombre, la edad y el teléfono.

5. Fı́jate cómo se llama el programa del servidor que va a recibir esos datos.

Se llama prog2.pl

6. ¿Dónde viajan los datos que el cliente le envı́a al servidor? ¿Cuáles son esos datos?

En la cabecera GET del paquete 25 vemos que el nombre es Ana, la edad es 25 y el teléfono es 123456789.

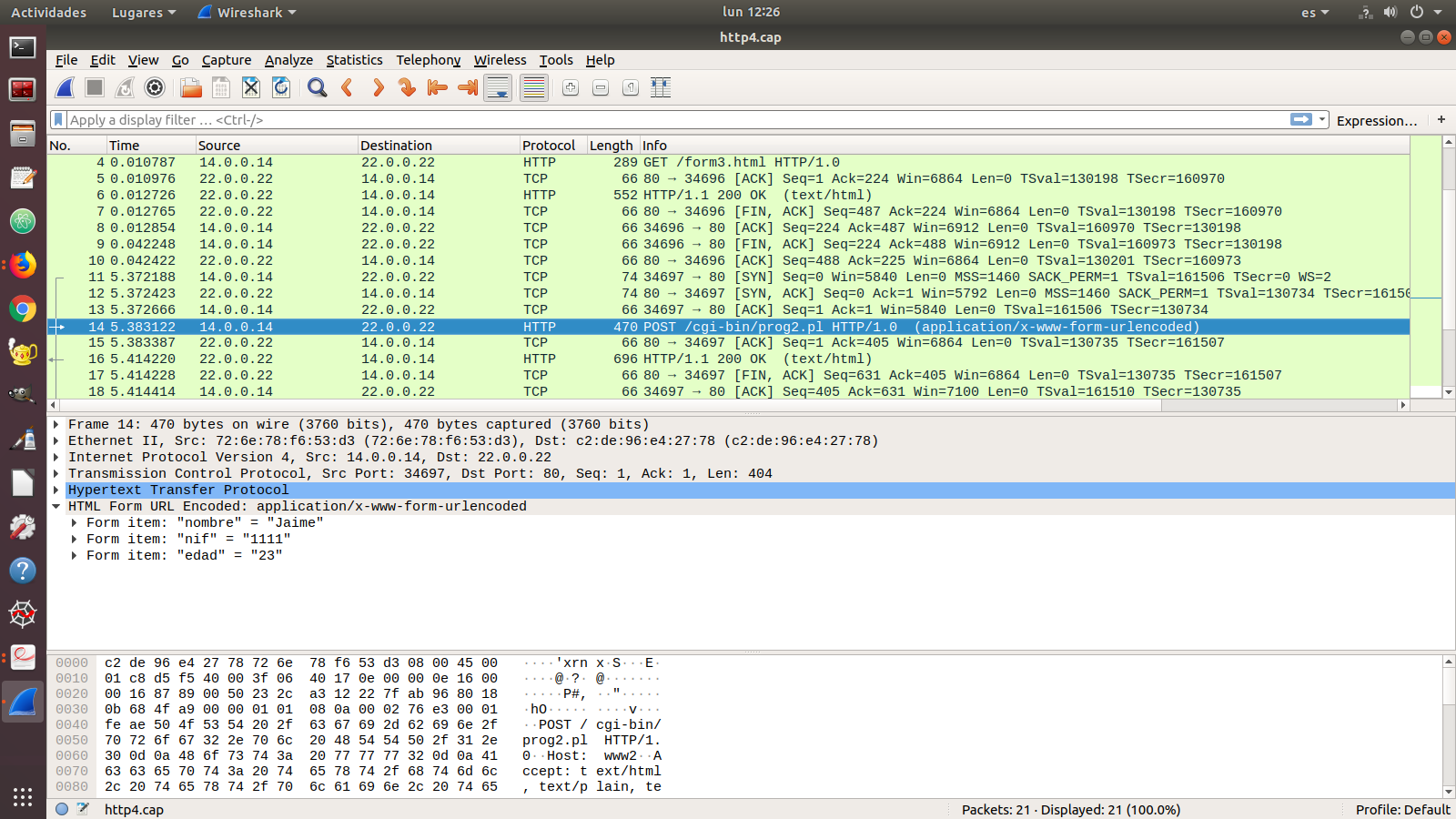
7. Indica qué cabecera es la que representa el tipo de contenido del mensaje que el cliente envı́a al servidor con los datos del formulario.

En el content-type del servidor vemos como es txt/html.

Abre la captura http4.cap y responde a las siguientes preguntas:

9. Busca en la captura el segmento donde el servidor le envı́a al cliente un formulario. Indica los nombres de los campos del formulario que rellenará el usuario.

Envía un formulario en el paquete 4, y rellenará los siguientes datos nombre, nif y edad.



10. Indica si es el cliente o el servidor el que decide cómo debe enviar el cliente los datos del formulario (GET/POST) . ¿Qué método están usando en este caso? ¿Cómo lo sabes?

El servidor en el paquete 14 pone que tiene que ser con POST en la primera línea del formulario.

11. Busca en la captura el segmento donde el cliente le envı́a los datos del formulario al servidor y comprueba que se está realizando con el método POST.

Es correcto se está realizando con el método POST.

12. Fı́jate cómo se llama el programa del servidor que va a recibir esos datos.

El programa se llama prog2.pl

13. Indica qué cabecera es la que representa el tipo de contenido que el cliente envı́a al servidor y cuál es su valor.

Cabecera: POST

Nombre: Jaime

NIF: 1111

Edad : 23

14. Indica en qué parte del mensaje van los datos del formulario que el cliente le envı́a al servidor.

Al ser de tipo POST, los datos del formulario se encuentran en el cuerpo del mensaje. En Wireshark lo podemos ver en HTML Form URL Encoded.

15. Explica si en este caso es necesario la cabecera Content-Length en el mensaje HTTP que el cliente envı́a al servidor con los datos del formulario. ¿Por qué?

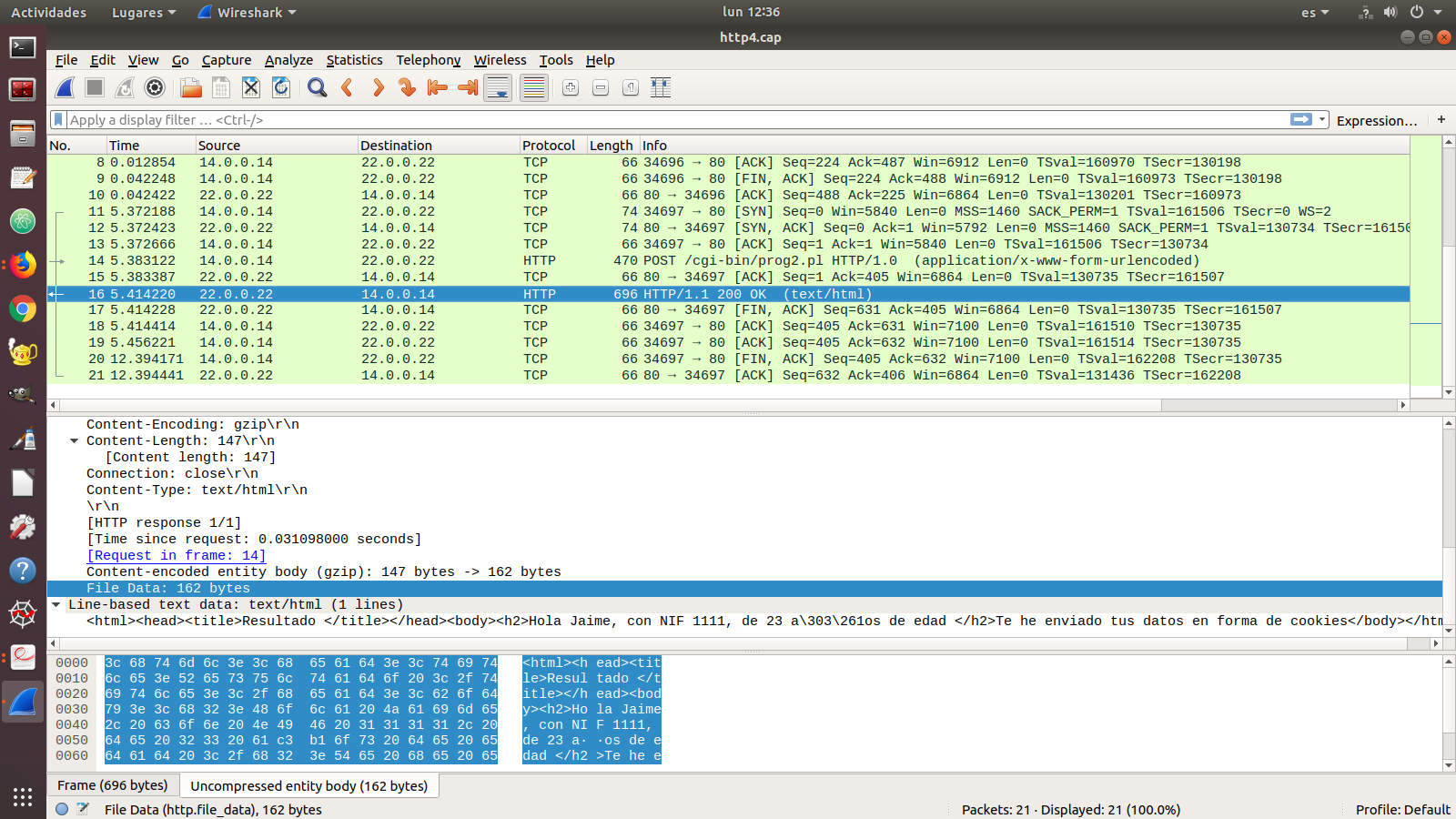
Si es necesario porque dicha cabecera nos indica el número de bytes que tiene el cuerpo del mensaje.

16. Observa si el servidor le manda alguna respuesta cuando recibe los datos del formulario del cliente. En caso afirmativo localiza el número de segmento y observa en las cabeceras HTTP: tipo de contenido, longitud y cuerpo del mensaje

Observando el paquete 16 vemos el texto que ha enviado el cliente en el formulario. La longitud es de 162 bytes. Su content-Type es text/html\r\n.El cuerpo del mensaje sería:

Hola Jaime, con NIF 1111 de 23 años de edad

Te he enviado tus datos en forma de cookies



**4.Cookies.**

**4.1.Almacén de cookies en el navegador Firefox.**

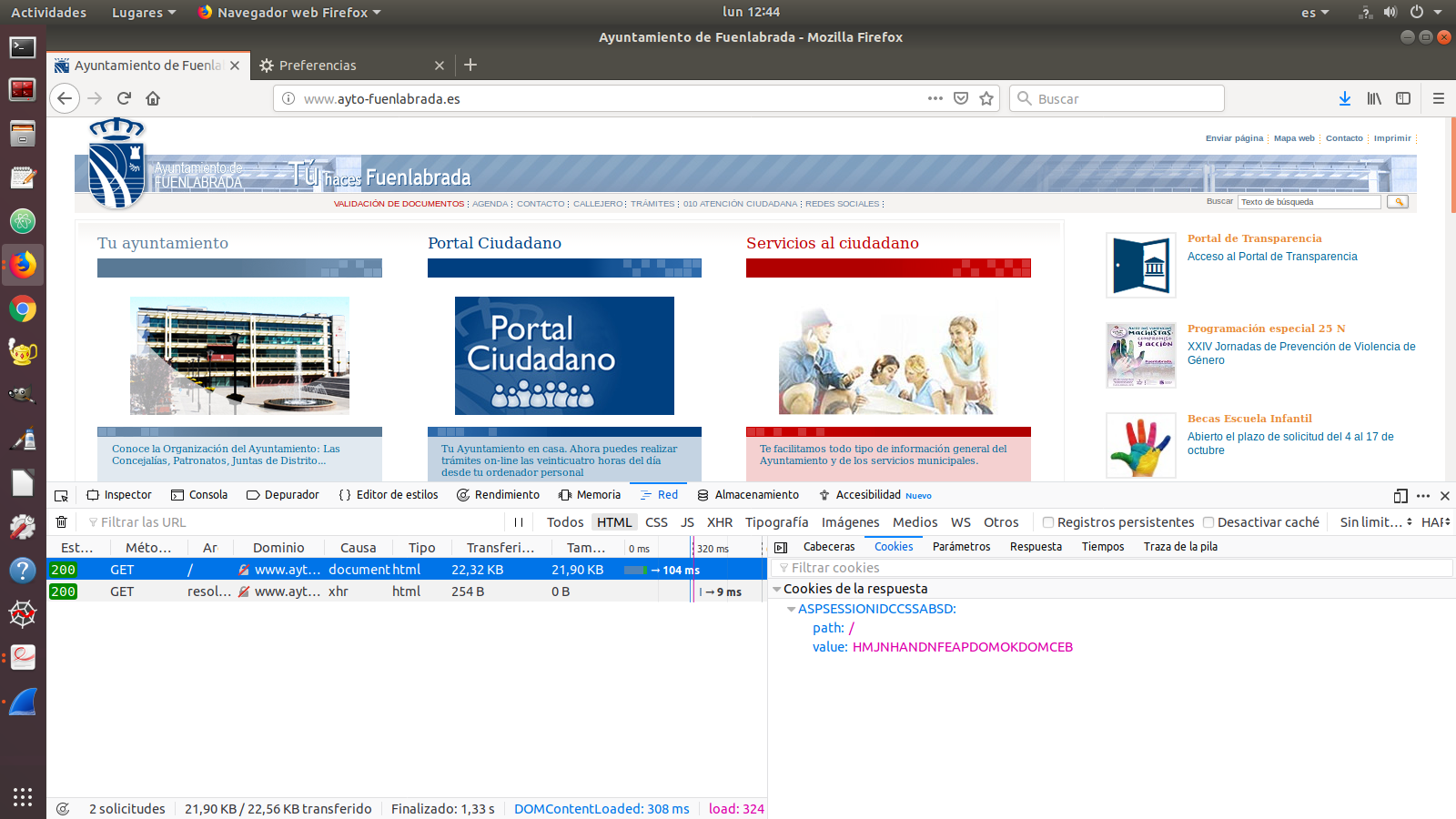
Para ver las Cookies en el navegador Firefox, selecciona la opción de menú: Editar → Pre-

ferencias. En la zona de la izquierda selecciona la pestaña Privacidad y seguridad. Dentro de la sección Historial, en Cookies y Datos pulsa en “Administrar Datos”. Podrás consultar la lista de cookies que tienes almacenadas actualmente. Mira si tienes cookies del sitio web del Ayuntamiento de Fuenlabrada ayto-fuenlabrada.es/ (si las tienes, elimina sólo esas cookies).

Abre otra pestaña en el navegador y selecciona en el menú Herramientas → Desarrollador web → Alternar herramientas. Selecciona la pestaña Red y HTML:

Dentro de esa pestaña carga la página http://www.ayto-fuenlabrada.es/. Selecciona dentro

de las herramientas del desarrollador la petición GET y la pestaña Cabeceras.

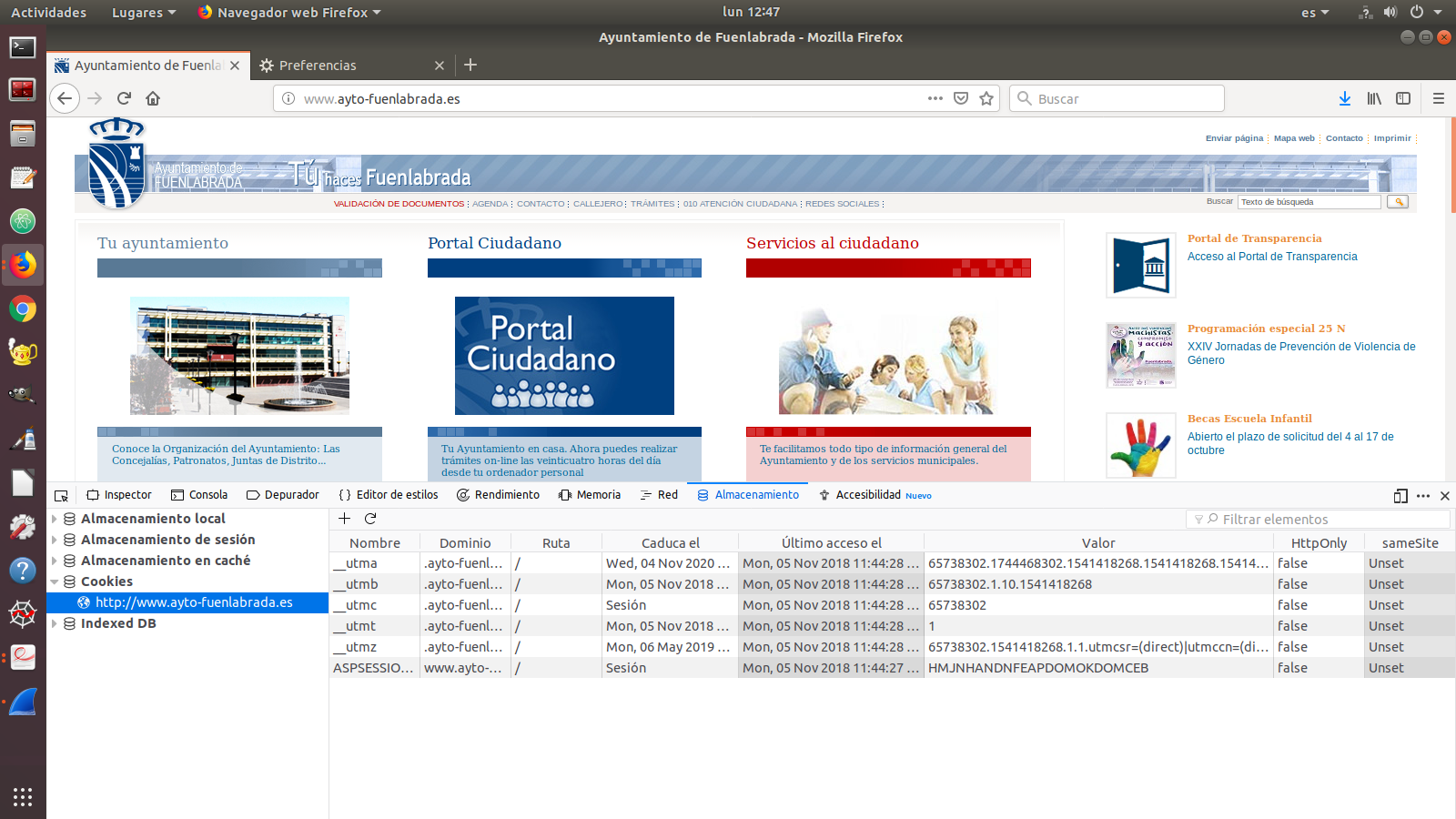


1. Fı́jate en la lı́nea de cabecera que el servidor le envı́a al cliente con el contenido de las cookies.

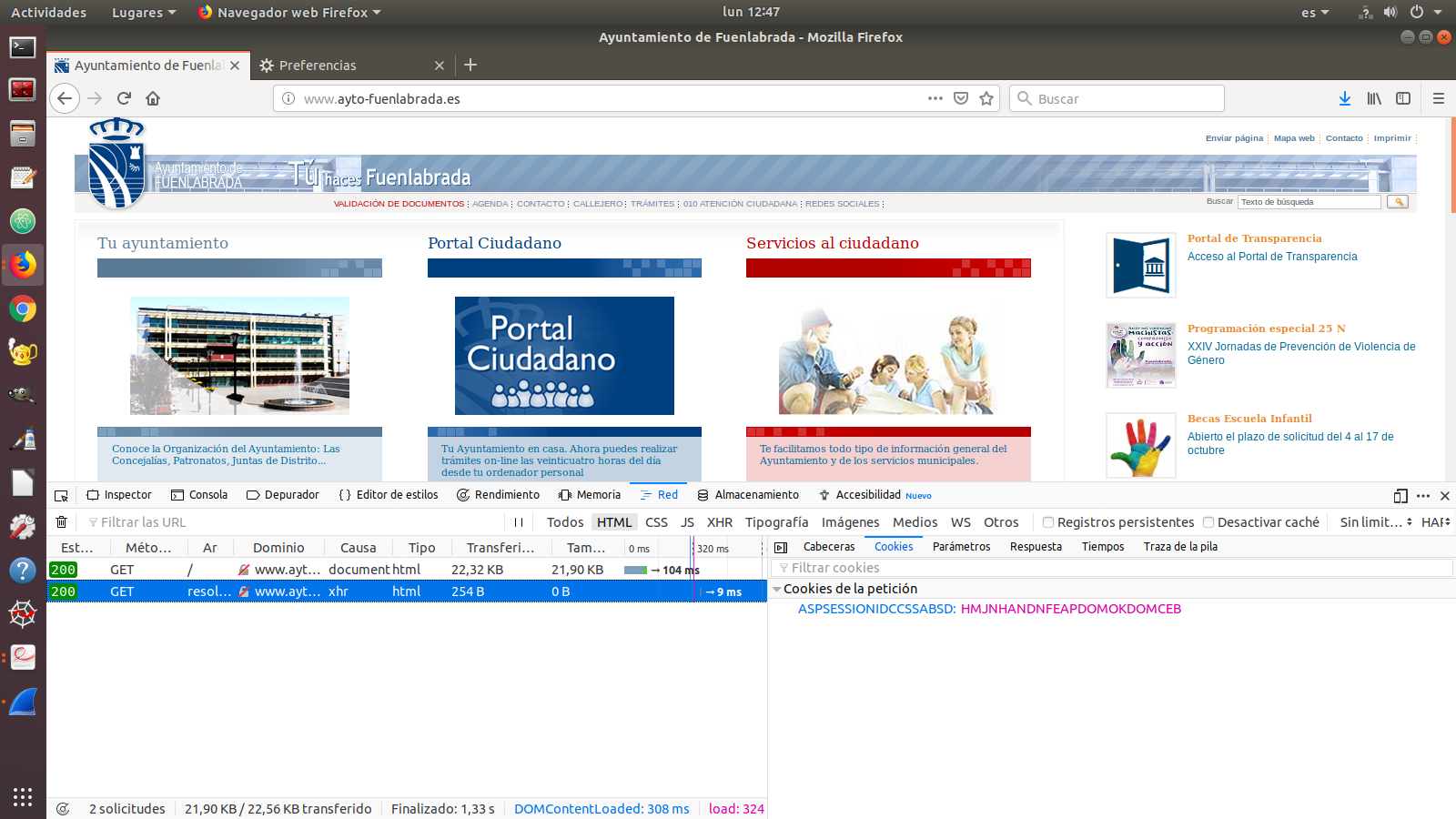
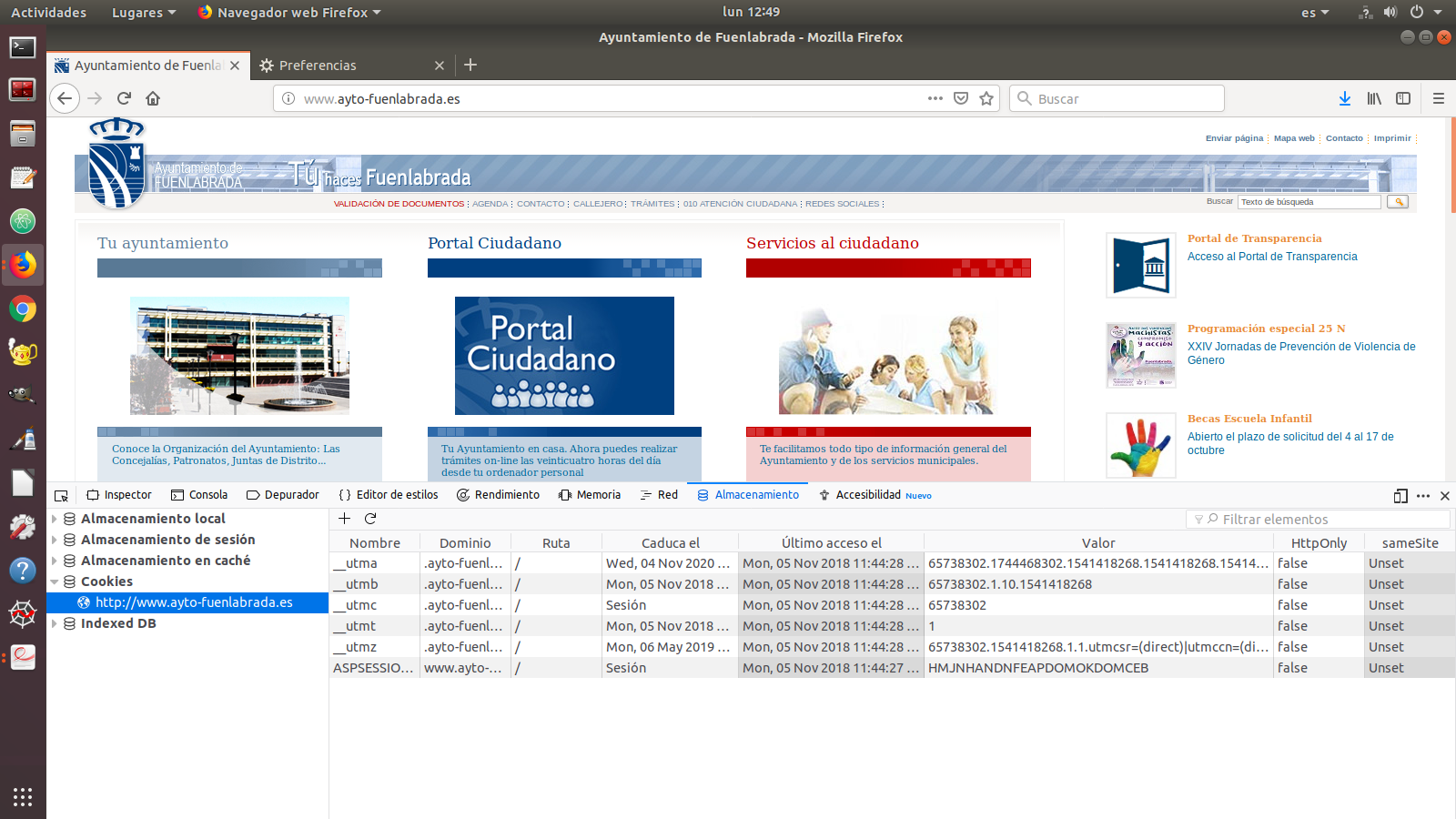
2. Pulsa sobre la pestaña “Cookies” para poder ver de forma más clara el contenido de las cookies. Copia los campos importantes. Fı́jate que no hay fecha de expiración, eso quiere decir que la Cookie se eliminará cuando se cierre el navegador.

3. Selecciona ahora la herramienta de desarrollador “Almacenamiento” y en el panel de la izquierda despliega ”Cookies” para ver las cookies almacenadas al descargar esta página:

Indica qué cookies tienes almacenadas para el sitio ayto-fuenlabrada.es. Las cookies \_\_utmc, \_\_utmt son debidas a que el sitio web usa Google Analytics, es decir, al descargar la página del Ayuntamiento de Fuenlabrada se ha descargado también una biblioteca en javascript que ha creado estas cookies para este sitio web dentro de nuestro navegador. Estas cookies permiten medir la interacción de los usuarios con el sitio web. No te fijes en esas cookies de Google Analytics.



4. Vuelve a la herramienta de desarrollador “Red” y pulsa sobre la segunda petición GET que aparece, y oberva las cookies que se envı́an. Usa también la pestaña Cookies para poder ver mejor los valores que se envı́an



**4.2.Envı́o de Cookies en mensajes HTTP**

Abre la captura http5.cap y responde a las siguientes preguntas:

1. Indica qué cookies envı́a el servidor al cliente.

Nombre: Pepe Lozano

Nif: 22034J

Edad:22

2. Indica qué cookies enviará el cliente al servidor cuando acceda a la página con la URL:

<http://elcortebritanico/tienda/index.html>

Nombre, NIF y edad.

3. ¿Y si el cliente accediera en el año 2025 a dicha URL?

No ocurrirá nada porque en ese año todavía estarán los cookies es decir, no habrán caducado.

4. ¿Y si el cliente accediera en el año 2035 a dicha URL?

Como las cookies caducaron en el año 2030, será necesario que las volvamos a poner válidas para que podamos utilizarlas otra vez nuevo.

Abre la captura http6.cap y responde a las siguientes preguntas:

5. Indica qué cookies el cliente está enviando al servidor.

Cuando abrimos el paquete cuatro (HTP)observamos como envía el nombre y el nif.

Cookie2: $Version="1"\r\n

Cookie: Nif=123456789A; Nombre=Andres\r\n

Cookie pair: Nif=123456789A

Cookie pair: Nombre=Andres

6. ¿Por qué crees que le envı́a dichas cookies?

EXPLICACIÓN FORO,

En esta captura, no sabemos qué cookies tenía almacenadas el cliente al hacer la única petición que aparece en el fichero, el paquete 4. En él vemos que envía las cookies Nif y Nombre. Si las envía es que las tenía almacenadas previamente y cumplían las 3 reglas: seguían en vigor (sin caducar), eran para el Domain de esa petición (www2) y eran válidas para el path=/dir1 o para el path=/ (pues la petición es de un recurso /dir1/index.html). Esto te dice que ese servidor debería haber enviado previamente a los paquetes de la captura las siguientes cabeceras, por ejemplo:

Set-Cookie: Nif=0123456789A; expires=30-Jan-2020; domain=www2; path=/

Set-Cookie: Nombre=Andres; expires=30-Jan-2020; domain=www2; path=/dir1

7. Escribe un ejemplo de la posibles cabeceras que le habrá enviado dicho servidor al cliente previamente.

El cliente manda un GET con las cookies. El servidor le podría mandar si ha modificado la información o que no.

8. A partir de la información de la captura ¿crees que si el cliente accede a otra página con la URL: http://www2/dir1/dir2/index.html enviarı́a esas cookies, más o menos?

Debe envía las misma porque tiene el mismo prefijo dir1.

9. A partir de la información de la captura ¿crees que si el cliente accede a otra página con la URL: http://www2/index.html enviarı́a esas cookies, más o menos?

En el caso de que se mandaran desde / si que mandaría las mismas ya que, siguen compartiendo el mismo host pero si fuera desde la carpeta /dir1 no se mandaría porque no está en el path.

10. A partir de la información de la captura ¿crees que si el cliente accede a otra página con la URL: http://www/index.html enviarı́a esas cookies, más o menos?

No se manda nada, puesto que el host ha cambiado.

**5. Comunicación a través de proxy HTTP.**

Abre la captura http7.cap y responde a las siguientes preguntas:

1. Indica qué dirección IP es el cliente, el proxy y el servidor final.

Cliente: 13.0.0.13

Proxy: 23.0.0.23

Servidor final: 22.0.0.22

2. ¿Qué diferencia la petición HTTP que realiza el cliente de la petición que realiza el proxy?

El protocolo de HTTP cambia, porque cuando enviamos del cliente al proxy está con la versión 1.0 y del proxy al servidor final está en 1.1.

3. Identifica el nombre de la máquina donde se encuentra el servidor HTTP.

El nombre de la máquina es: Apache/2.2.9 (Debian).

4. ¿Se puede saber de la petición que realiza el proxy que dicho proxy tiene almacenada en su caché esa página?

Si la tenía almacenada, ya que al recibir la información del proxy el servidor devuelve un mensaje ‘not modified’

Entonces proxy cambiará la fecha de caducidad y se volverá a validar.

Abre la captura http8.cap y responde a las siguientes preguntas:

5. Indica qué tipo de conexiones HTTP utilizan el cliente y servidor e indica el número de

conexiones entre cliente y servidor que aparecen en la captura.

Tenemos dos conexiones una va desde el mensaje 6 al mensaje 13 con cliente HTTP 1.0 y servidor HTTP 1.1 (lo vemos en el paquete 8) y otra que va del paquete 14 al paquete 35 con HTTP cliente 1.0 y HTTP servidor 1.1.

6. Explica qué es lo que se está descargando el cliente del servidor HTTP y cuantos objetos se descarga.

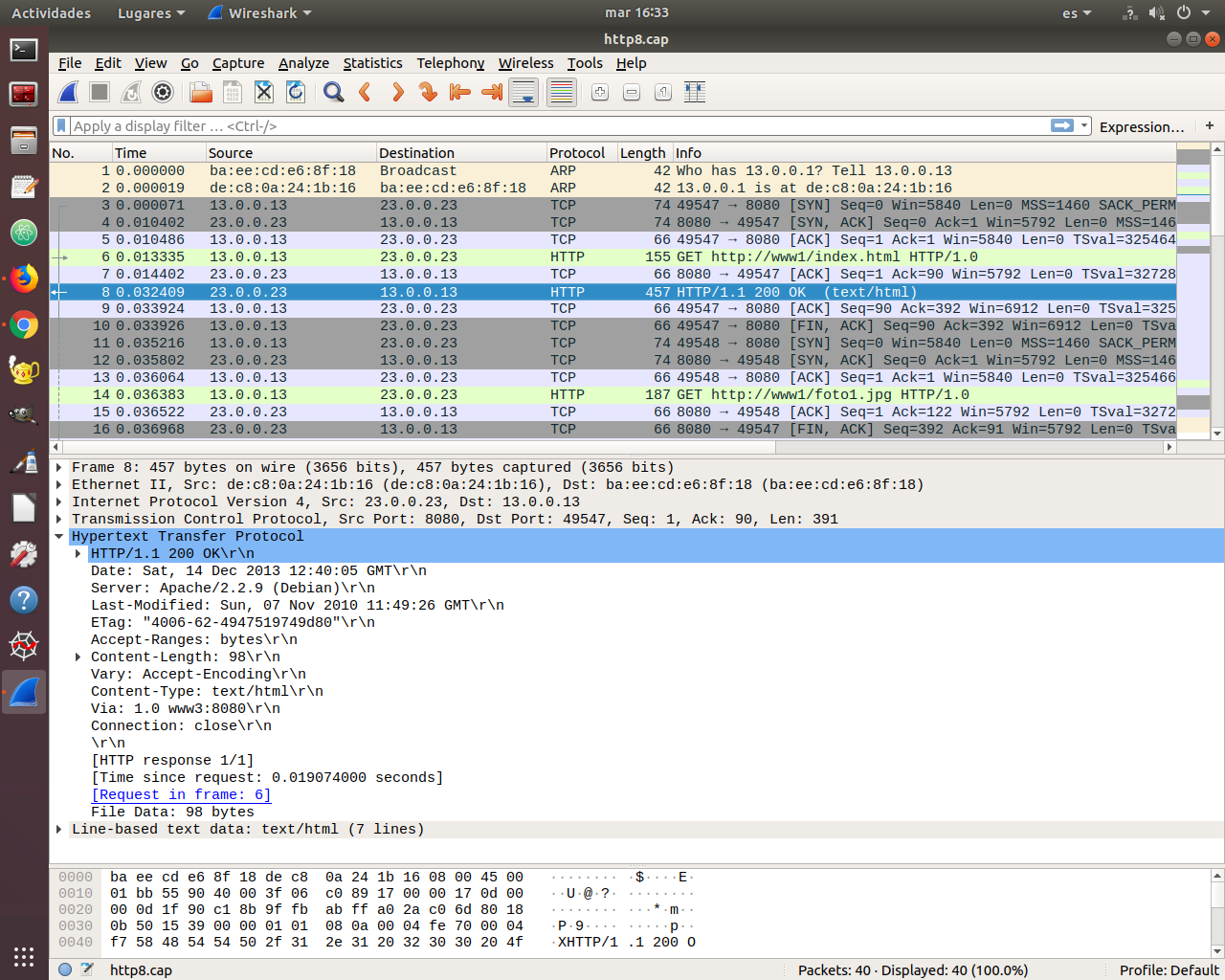
Primero pregunta por la URL <http://www1/index.html> y en la segunda conexión, se descarga el archivo JPG http://www1/foto1.jpg.

7. Observa en las cabeceras HTTP el tipo de contenido de cada uno de los objetos.

La primera en un texto en html y segunda una imagen en jpg.

8. ¿Podrías saber si los paquetes capturados se corresponde a la comunicaci ́on entre un cliente y un proxy HTTP, entre un cliente y servidor final HTTP o entre un proxy y el servidor final HTTP? ¿Por qué?

Se trata de un cliente con un proxy, ya que en paquete 6 se hace un GET, en el paquete 8 HTTP, vemos que tiene la cabecera “vía” que esa cabecera solo la tienen los proxy.



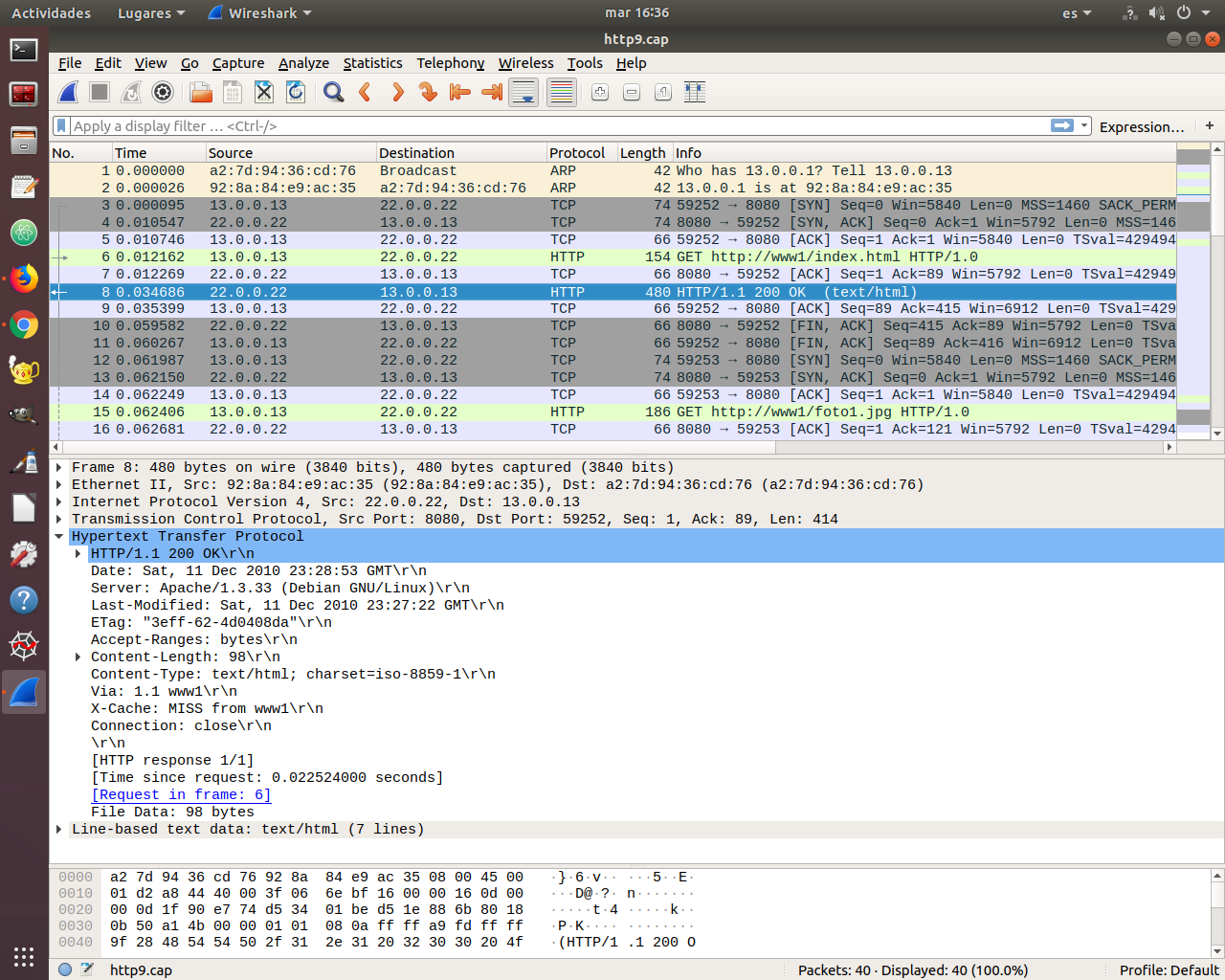
9. Sabiendo que la comunicación se ha realizado a través de un proxy HTTP, mira las cabeceras HTTP que envía dicho proxy para ver si en ellas existe alguna que muestre cuál es su nombre.

Apache/2.2.9 (Debian).

Abre la captura http9.cap y responde a las siguientes preguntas:

10. ¿Podrías saber si los paquetes capturados se corresponde a la comunicación entre un cliente y un proxy HTTP, entre un cliente y servidor final HTTP o entre un proxy y el servidor final HTTP? ¿Por qué?

Se sabe que es un cliente con un proxy porque el paquete 6 se pide información con GET y después en el paquete 8 que es el mensaje que contesta vemos como es su cabecera tiene “via” que nos indica que es un proxy.



**5.1. Caché en el proxy.**

Estudia las capturas http10.cap y http11.cap, teniendo en cuenta que las direcciones 11.0.0.1 y 12.0.0.1 corresponden a la misma máquina. Responde a las siguientes preguntas:

1. Indica cuáles son las direcciones IP del cliente, proxy y servidor web. ¿Cómo lo sabes?

Cliente: 12.0.0.100

Proxy: 12.0.0.1 (subred del cliente) y 11.0.0.1 (subred del servidor)

Servidor: 14.0.0.100

2. Explica qué es lo que ocurre en estas capturas.

En la captura 10 un cliente pide una URL al proxy, en la captura 11, el proxy pide comprobar si ha habido alguna modificación en la información que tiene guardada.

3. Localiza los campos relevantes con respecto al tratamiento de caché que incluye en las líneas de cabecera el servidor. ¿Qué significan?

DATE: fecha en la que elimina esta información que le está validando al proxy.

ETag: no se ha modificado la información que hay.

Cache-control: es una información pública, es decir que se puede guardar tanto en el proxy como en el cliente si hay caché y que puede estar durante 60s.

4. Explica qué ocurre en la segunda consulta que realiza el cliente.

Pide la misma información por la que había preguntado anteriormente pero sin caché porque la tiene vacía.

5. Viendo los paquetes 14 y 16 de la captura http10.cap indica cómo se puede saber que el

contenido proviene de una caché.

Vemos como la cabecera Etag tiene la misma información que en la captura 11 porque el servidor no cambia la información, pero en la captura 10 si lo hace y el proxy envía la información desde su caché. También vemos en la cabecera Age que es distinto de 0 eso quiere decir que si la coge de su caché.

6. ¿Crees que el cliente tiene caché?

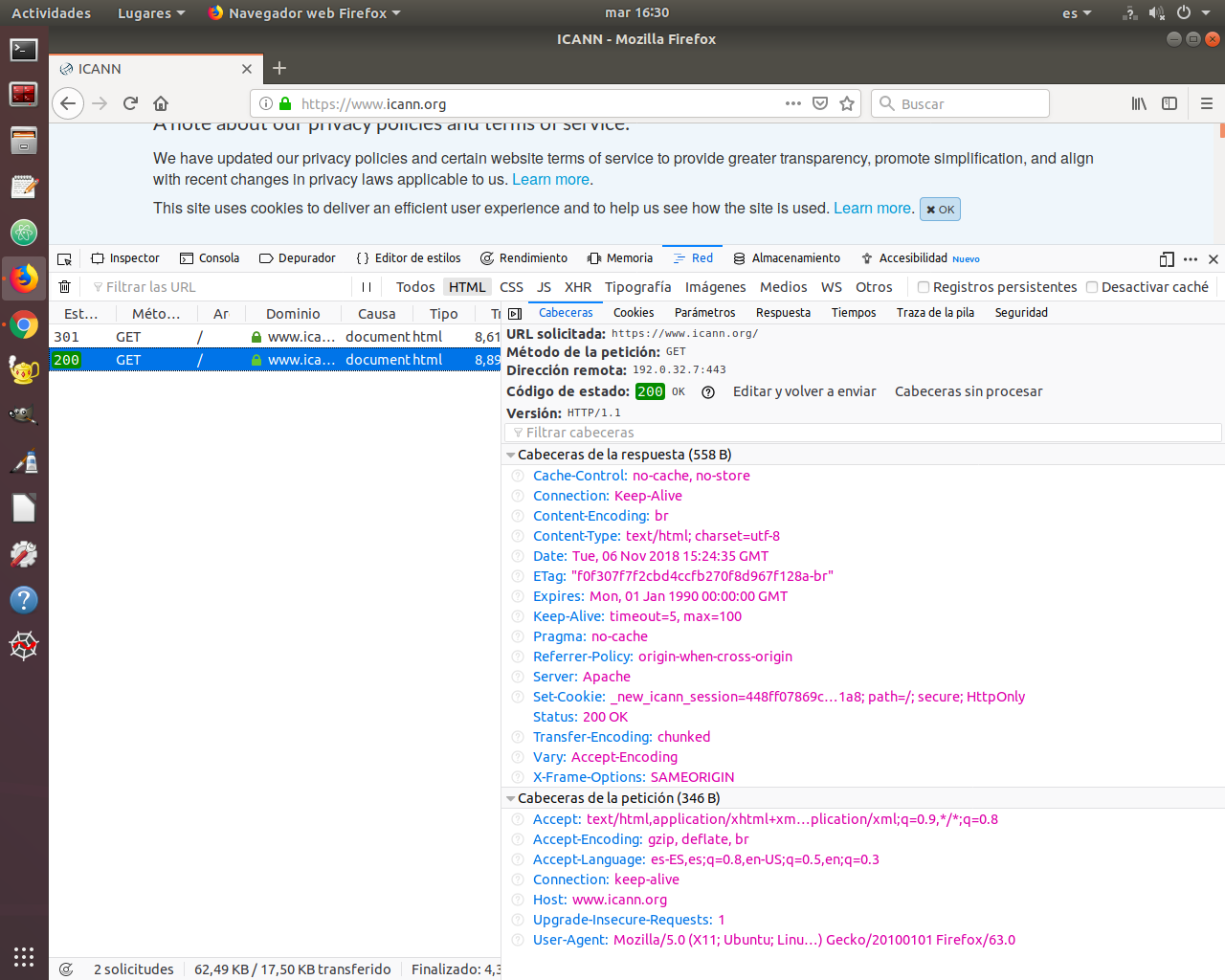
No, puesto que vemos en el paquete 16 como la cabecera Pragma pone no-cache.

**6. Caché en el navegador.**

**6.1. Sitio web ICANN.**

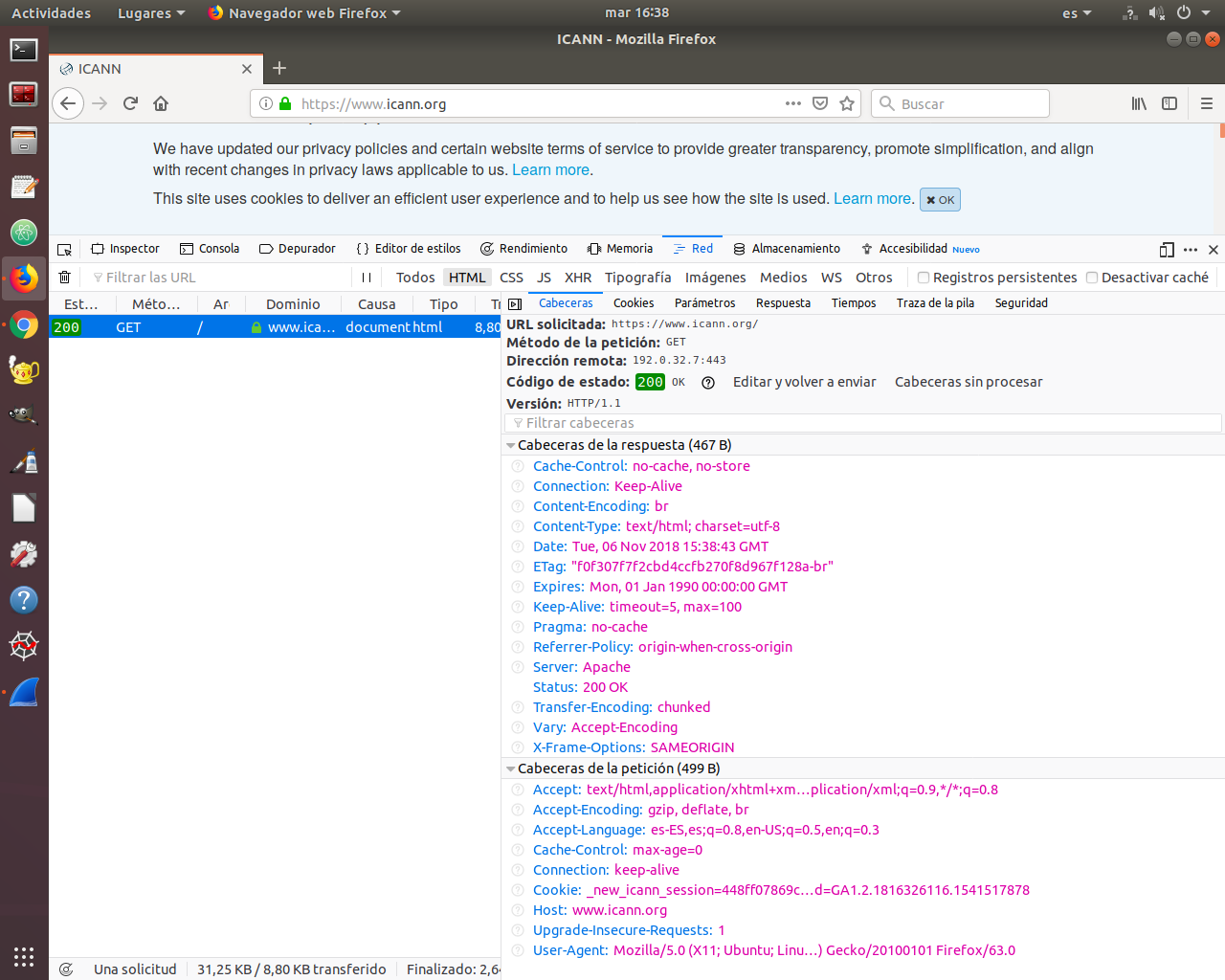
1. Despliega la pestaña de las cabeceras de la respuesta e indica qué tipo de caché se permite.

En cache-control vemos como nos dice que no tiene: no-cache, no-store.



2. A la vista de los resultados, ¿qué crees que ocurrirá si se recarga la página en el navegador?

Vemos como se vuelve a producir el mismo mensaje como si nada ya que no tenemos caché.



3. Abre otra pestaña en el navegador, selecciona de nuevo en las herramientas → desarrollador web → alternar herramientas. Teclea en esta nueva pestaña la misma URL: http://www.icann.org, observaras que se muestra nuevamente el recurso solicitado y la respuesta con el mismo (junto a la solicitud se muestra el recuadro verde intenso).

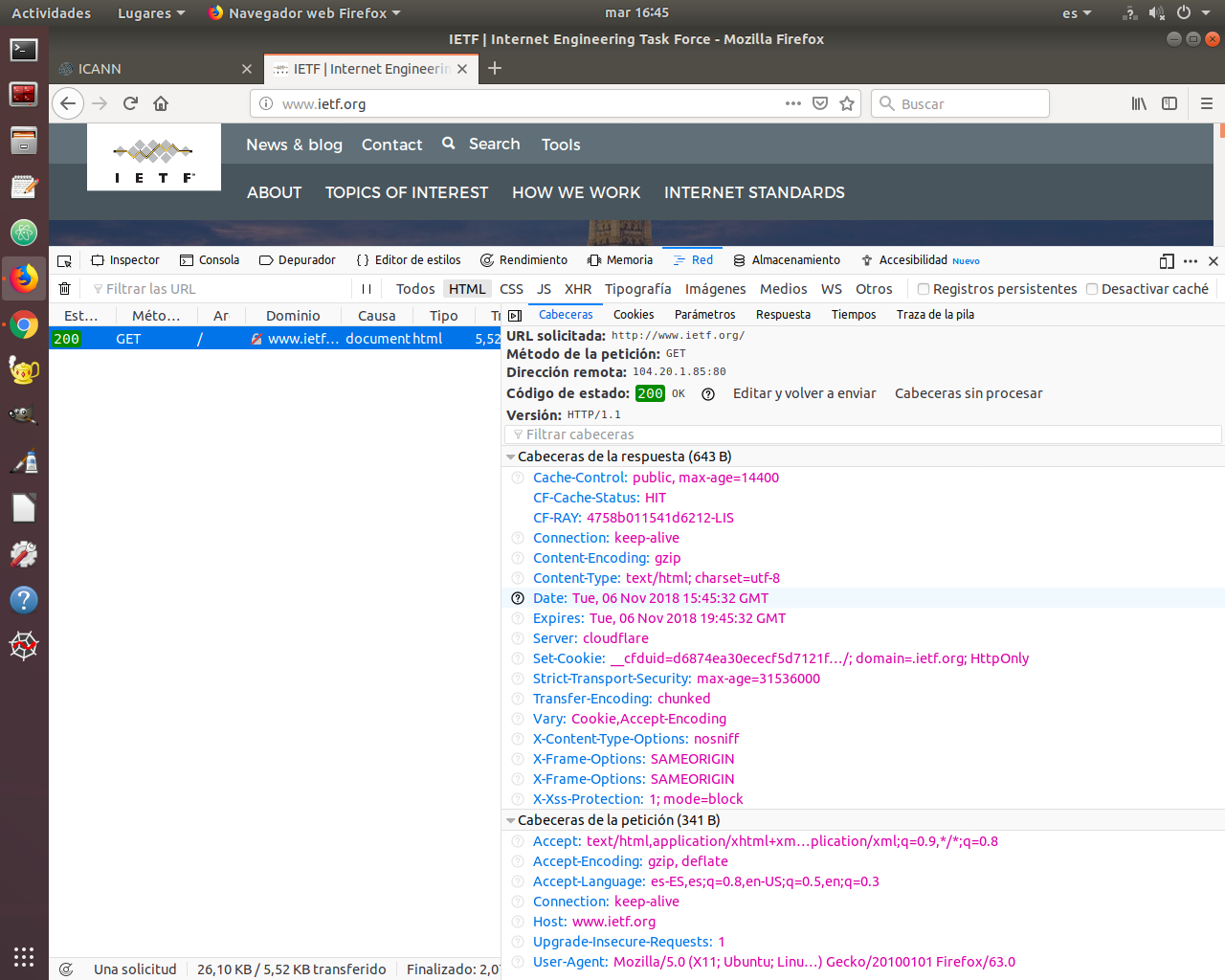
Ocurre lo mismo que anteriormente.

**6.2. Sitio web IETF.**

Carga en esa misma pestaña del navegador la dirección del IETF (Internet Engineering Task Force): <http://www.ietf.org>.

4. Selecciona la petición GET / y observa en las líneas de cabecera de respuesta qué tipo de caché se permite.

Vemos: Public, max-age=14400



5. Indica cuándo expirará el recurso en la caché. Relaciona los valores de las cabeceras: Date, Expires y Cache-Control.

Date: Tue, 06 Nov 2018 15:45:32 GMT

Expires: Tue, 06 Nov 2018 19:45:32 GMT

Caché-Control: Public, max-age = 14400

Expirará a las 3 horas, 10800 segundos que es la diferencia entre la fecha en la que pido en recurso y en la que dejará de estar disponible.

6. A la vista de los resultados ¿qué crees que ocurrirá si desde otra pestaña se solicita la misma URL?

Vemos como la fecha de expires es la misma que anteriormente puesto que lo coge desde la caché y no lo modifica.

7. Abre otra pestaña en el navegador, selecciona de nuevo en las herramientas → desarrollador web → alternar herramientas. Teclea en esta nueva pestaña la mism URL: http://www.ietf.org. Selecciona de nuevo la petición GET /. Al posicionar el ratón sobre el nombre de la petición GET, observaras que se muestra que el contenido estaba cacheado (fíjate que ahora toda la petición se muestra “atenuada”, incluyendo el recuadro verde al lado de GET /, que ahora está atenuado). Fíjate en las líneas de cabecera: Date, Expires y Last-modified y explica si han cambiado y por qué.

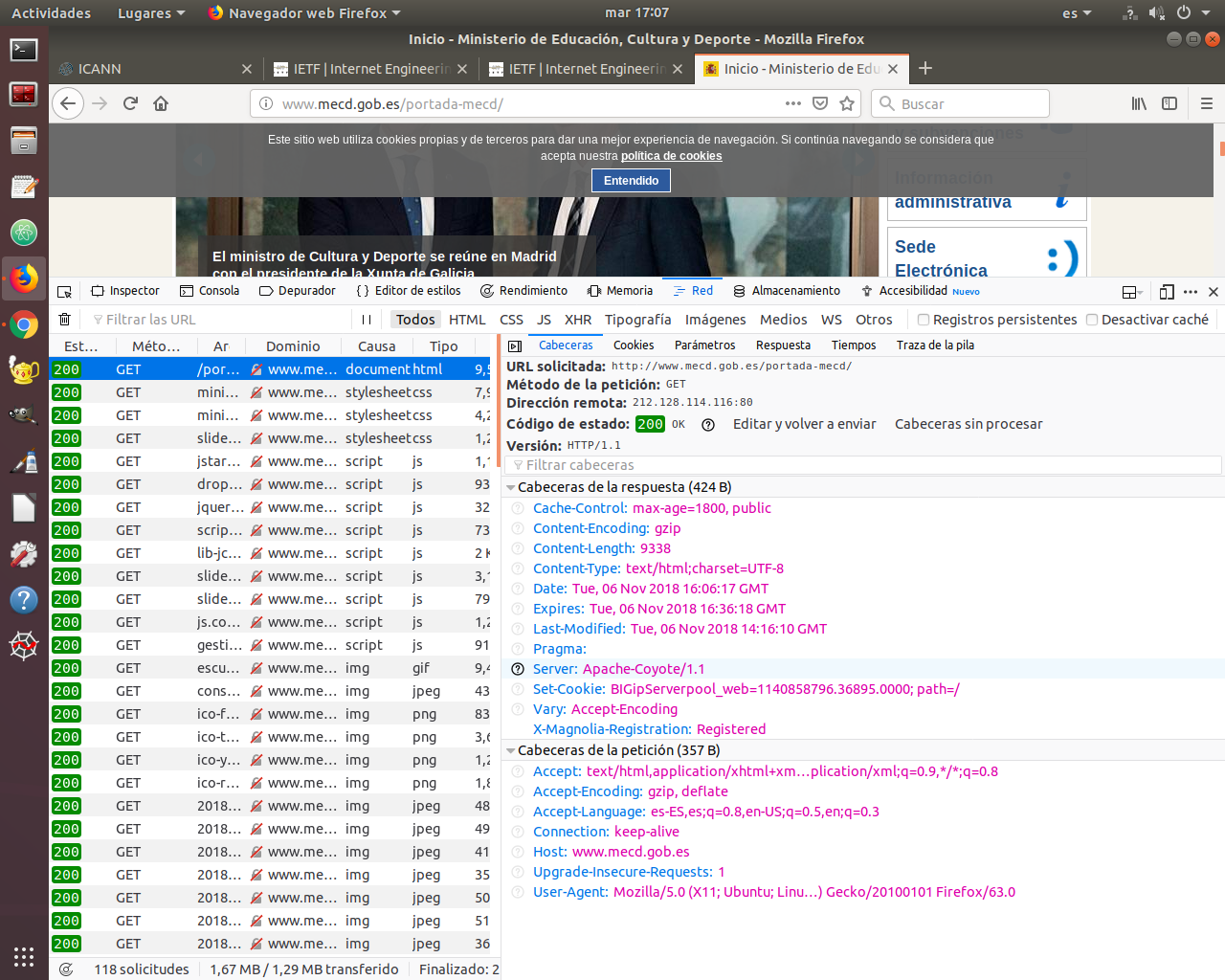
No cambia porque lo coge de su caché si lo pide cuando ya ha expirado se volverá a pedir y a recargar como si no lo hubiéramos tenido nunca puesto que ha caducado.

**6.3. Sitio web MECD.**

Carga en una de las pestañas donde tengas abiertas las herramientas del desarrollador, la dirección del Ministerio de Educación: www.mecd.gob.es/portada-mecd/, selecciona la petición GET /portada-mecd/ y responde a las siguientes preguntas:

9. Observa las líneas de cabecera de la respuesta e indica qué tipo de caché se permite.

Max age = 1800, public.



10. Indica cuándo expirará el recurso en la caché. Relaciona los valores de las cabeceras: Date, Expires y Cache-Control.

Date: Tue, 06 Nov 2018 16:06:17 GMT

Expires: Tue, 06 Nov 2018 16:36:18 GMT

Caché-Control: max-age=1800, public

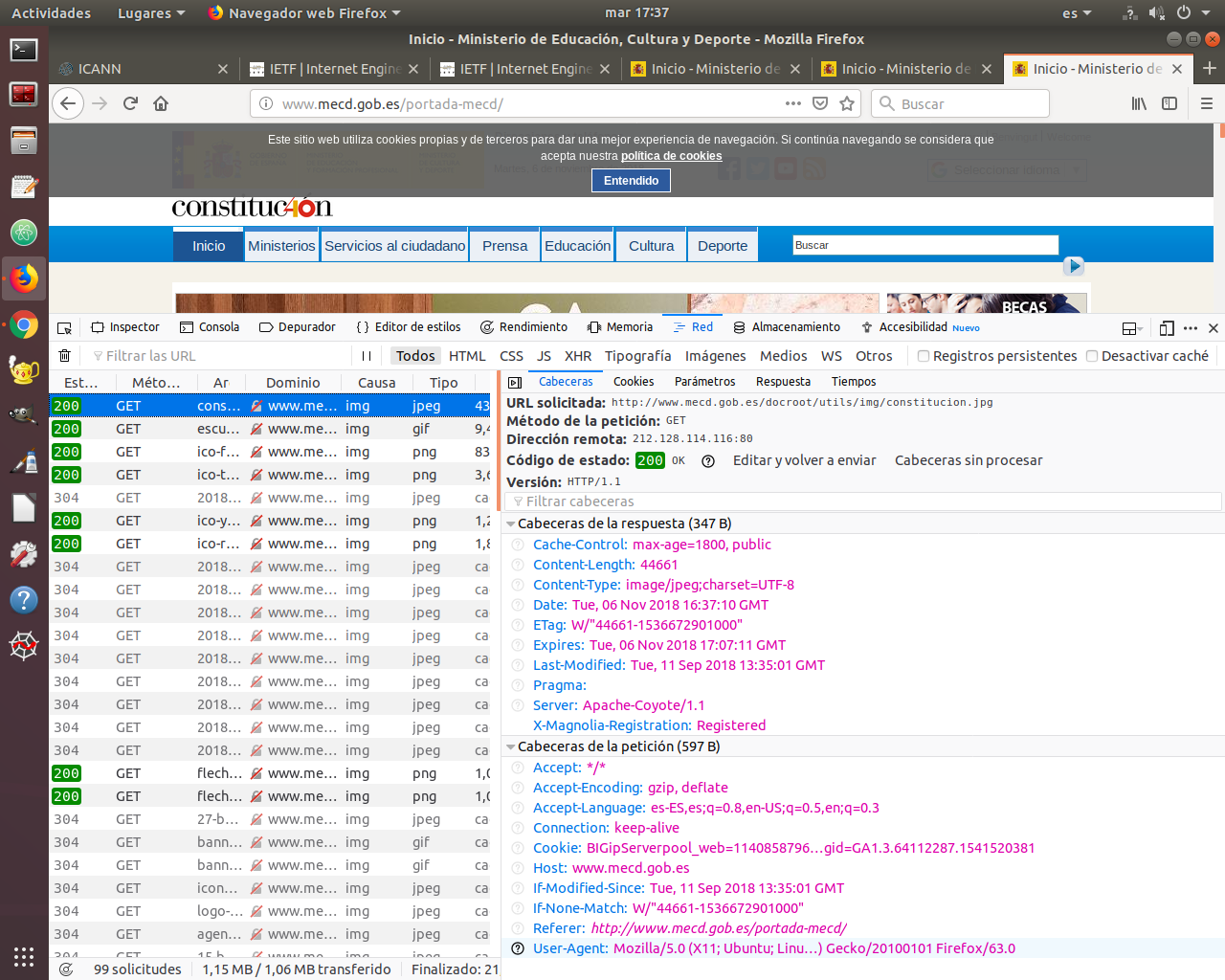
Expirará a los 30 min, 1800 segundos que es la diferencia entre la fecha en la que pido en recurso y en la que dejará de estar disponible.

11. Abre una nueva pestaña del navegador con las herramientas del desarrollador visibles. Indica qué ocurre si tecleas la misma URL.

Lo coge de la caché y por tanto no cambia.

12. Espera el tiempo necesario para que la información caduque en la caché y abre una nueva pestaña con las herramientas del desarrollador visibles. Explica qué sucede cuando vuelves a teclear esta URL.

Vemos como coge la de la caché que ya tenía pero cambia la fecha y la expiración puesto que la que teníamos antes a caducado, vuelve a expirar a los 30 minutos de la petición.



13. ¿Se está usando validación fuerte o débil?

Se trata de validación débil, ya que tiene la etiqueta ‘Last-Modified’ y no ‘E-Tag’.