SISTEMAS WEB
CURSO 2022/2023

HTTP HyperText Transfer Protocol

Petición y respuesta – Ejemplo utilizando Burp.

Oskar Casquero y María Luz Álvarez
Dpto. Ingeniería de Sistemas y Automática

ÍNDICE

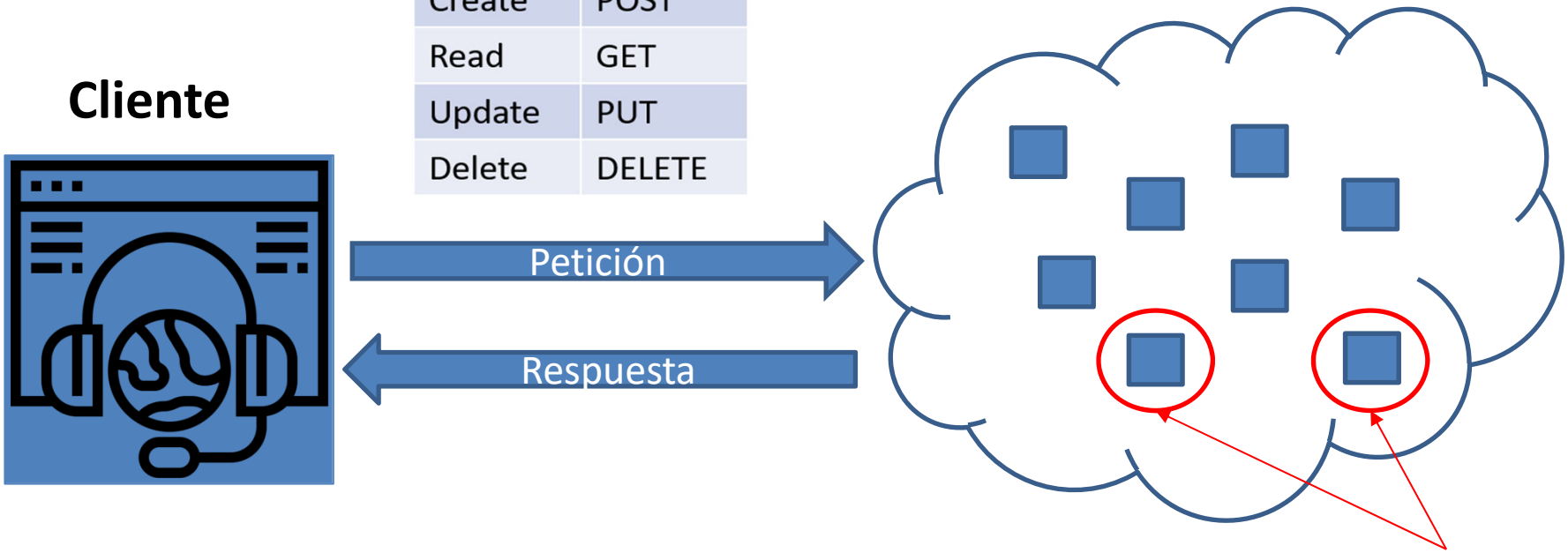
- Modelo cliente-servidor WEB
- Denominación de recursos web: URI
- Definición breve de HTTP
- Especificación HTTP: Solicitud de comentarios (RFC)
- **Funcionamiento de HTTP**
  - Entidades involucradas
  - Establecimiento de conexión TCP
  - **Solicitud de cliente**
  - Procesamiento de la solicitud en el Servidor
  - **Respuesta del Servidor**
  - **Ejemplo usando Burp**

MODELO CLIENTE-SERVIDOR WEB

Operaciones CRUD

Create	POST
Read	GET
Update	PUT
Delete	DELETE

Para realizar la petición, se debe conocer la dirección del **recurso**, por ejemplo:
http://api.thingspeak.com/channels/376999/feeds.json



NOTA 1: Un recurso puede ser un elemento concreto (una imagen, un documento, un vídeo) o un servicio:
- Indentificación: https://egela.ehu.eus/login/index.php
- Cálculo: https://ws-sendingformdata.appspot.com/
- Búsqueda: https://www.ehu.eus/bilatu/buscar/bilatu.php

NOTA 2: No tienen estado/memoria: las operaciones son independientes -> Sesiones -> ¡Cookies!

DENOMINACIÓN DE RECURSOS WEB: URI

- Aclaraciones con respecto a los términos URI y URL.
  - **URI (Universal Resource Identifier)**: es una cadena de caracteres **US-ASCII** que permite identificar un recurso en Internet. Según la RFC 3986, Sección 3, su sintaxis es la siguiente:

    URI = scheme ":" "//" authority ["/" path] ["?" query] ["#" fragment]

  - **URL (Universal Resource Locator)**: es un URI que, además de identificar un recurso, permite localizarlo en Internet, porque la propia cadena describe la forma de acceder dicho recurso. Ejemplo:

    https://egela.ehu.eus/course/view.php?id=69485 (página de la asignatura SW en eGela)

    Lectura de la URL: En el servidor *egela.ehu.eus* se encuentra un recurso, este recurso es accesible mediante protocolo *HTTPS*, cuya ruta completa es */course/view.php*.

    A través de este recurso, el servidor web *eGela* puede generar la página web de una asignatura, adaptando el contenido a la asignatura. Para realizar esto, es necesario pasar un parámetro, de nombre *id*, cuyo valor permite indicar la asignatura concreta a partir de cuyo contenido se debe generar la página web.

- Se recomienda utilizar el término URI. Es decir, un recurso se referencia mediante su URI. Dependiendo del esquema("scheme") utilizado, el URI también podrá actuar como dirección del recurso.

DEFINICIÓN BREVE DE HTTP

- **HTTP** es un protocolo de **nivel de aplicación** originalmente diseñado para la transferencia de recursos de tipo hipertexto entre un cliente y un servidor.
  - ***Protocolo de nivel de aplicación****:* HTTP da soporte directo a aplicaciones (ej., a un navegador, a un servidor web) para el envío y recepción de datos.
  - ***Transferencia:*** HTTP utiliza un modelo de transacciones que sigue un esquema de ***petición-respuesta***: la aplicación cliente realiza una solicitud y la aplicación servidora responde a dicha solicitud.
  - ***Recurso:*** HTTP referencia un recurso mediante el URI (*Universal Resource Identifier*).
  - ***Hipertexto:*** texto que contiene elementos a partir de los cuales se puede acceder a otra información.
    - Ejemplos: una página web (Wikipedia)
    - En el caso de una página web, el lenguaje utilizado para definir el hipertexto es HTML.

ESPECIFICACIÓN HTTP: SOLICITUD DE COMENTARIOS (RFC)

- Una RFC (Request For Comments) constituye un memorando que unos expertos hacen llegar al IETF (Internet Engineering Task Force) para que sea valorado por la comunidad con el objetivo discutir y consensuar estándares para Internet.
- HTTP v1.1 (HTTP/1.1) se definió originalmente en la RFC 2616.

  Actualmente, las especificación de HTTP/1.1 se recoge en las siguientes RFCs:
  - RFC 7230: HTTP/1.1 Message Syntax and Routing
  - RFC 7231: HTTP/1.1 Semantics and Content
  - RFC 7232: HTTP/1.1 Conditional Requests
  - RFC 7233: HTTP/1.1 Range Requests
  - RFC 7234: HTTP/1.1 Caching
  - RFC 7235: HTTP/1.1 Authentication

  En este tema estudiaremos diversos aspectos recogidos en estas RFCs y otras relacionadas.
- En mayo de 2015 se publicó la RFC 7540: HTTP/2.
  - Esta nueva versión surge de la necesidad de agilizar la carga de las páginas web actuales, las cuales
    - están compuestas por un número elevado de imágenes, javascript y CSS.
    - y realizan peticiones asíncronas mediante AJAX.
  - HTTP/2 está basado en el protocolo SPDY de Google (en uso desde 2010).
  - **La semántica y la sintaxis de los mensajes es la misma que la de HTTP/1.1. Únicamente cambia la forma en que se expresan y envían los mensajes HTTP por la red ("on the wire").**

- Página web del grupo de trabajo sobre HTTP del IETF: http://httpwg.github.io/

- Existen otras organizaciones encargadas de estandarizar otra serie de tecnologías relacionadas con Sistemas Web; por ejemplo, el W3C (World Wide Web Consortium) coordina los estándares de HTML, CSS y DOM.

FUNCIONAMIENTO DE HTTP

A continuación se va a describir, a través de un ejemplo, el funcionamiento del protocolo HTTP.

Concretamente, se van a responder las siguientes preguntas:

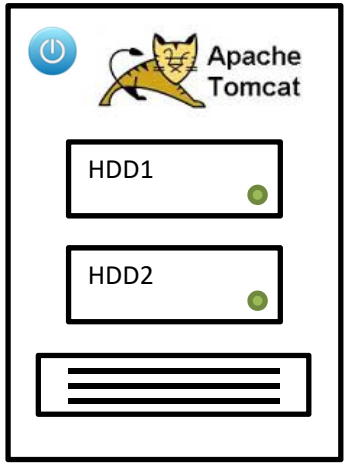
- ¿Qué ocurre cuando un usuario solicita recurso (por ejemplo, una página web) a través del navegador?
  - ¿Qué hace el navegador?
  - ¿Qué formato (sintaxis y semántica) tiene la petición?
  - ¿Cómo se procesa la petición en el servidor?
  - ¿Qué formato (sintaxis y semántica) tiene la respuesta?
  - ¿Cómo se carga una página web en el navegador?

FUNCIONAMIENTO DE HTTP
ENTIDADES INVOLUCRADAS

CLIENTE

NOTAS acerca del cliente:
- Firefox es una aplicación cliente (navegador web) que implementa el protocolo HTTP.
- Firefox puede decodificar contenido comprimido en formato gzip.
- El sistema operativo implementa la pila de protocolos TCP/IP.
- El sistema operativo tiene registrada la dirección de un servidor DNS.

SERVIDOR



NOTAS acerca del servidor:
- El alias del servidor es **sw2023.com**
- *Tomcat* es una aplicación servidora (servidor web) que implementa el protocolo HTTP.
- *Tomcat* escucha en el puerto 8080.
- *Tomcat* ofrece un recurso identificado con la URI http://sw2023.com:8080/resource.
- Dicho recurso es un documento estático disponible en texto plano y HTML, en euskera y castellano, en versión móvil y de escritorio.
- *Tomcat* no puede comprimir contenido.
- El sistema operativo implementa la pila de protocolos TCP/IP.

FUNCIONAMIENTO DE HTTP

A continuación se va a describir, a través de un ejemplo, el funcionamiento del protocolo HTTP. Concretamente, se van a responder las siguientes preguntas:

- ¿Qué ocurre cuando un usuario solicita recurso (por ejemplo, una página web) a través del navegador?
  - **¿Qué hace el navegador?**
  - ¿Qué formato (sintaxis y semántica) tiene la petición?
  - ¿Cómo se procesa la petición en el servidor?
  - ¿Qué formato (sintaxis y semántica) tiene la respuesta?

FUNCIONAMIENTO DE HTTP
ESTABLECIMIENTO DE CONEXIÓN TCP

CLIENTE

NOTAS acerca del cliente:

- Firefox es una aplicación cliente (navegador web) que implementa el protocolo HTTP.
- Firefox puede decodificar contenido comprimido en formato gzip.
- El sistema operativo implementa la pila de protocolos TCP/IP.
- El sistema operativo tiene registrada la dirección de un servidor DNS.

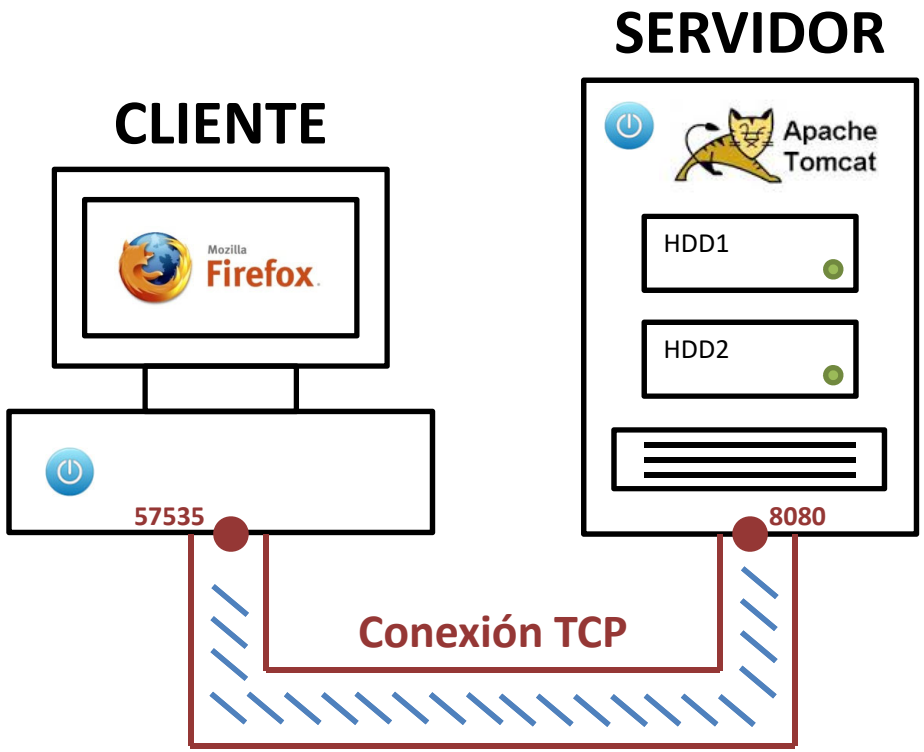
Cuando el usuario introduce la URI en la barra de direcciones, el navegador analiza las diferentes partes:

http://ws2023.com:8080/recurso
Scheme: http
Authority: Host: ws2023.com
Port: 8080
Path: /recurso

El navegador solicita al sistema operativo la resolución del nombre host del servidor. El sistema operativo resuelve esta petición a través del servidor DNS y devuelve la dirección IP al navegador.

Con este dato, el navegador solicita al sistema operativo que establezca una conexión TCP (SYN, SYN-ACK y ACK) desde un puerto local al puerto 8080 del servidor.

FUNCIONAMIENTO DE HTTP
ESTABLECIMIENTO DE CONEXIÓN TCP

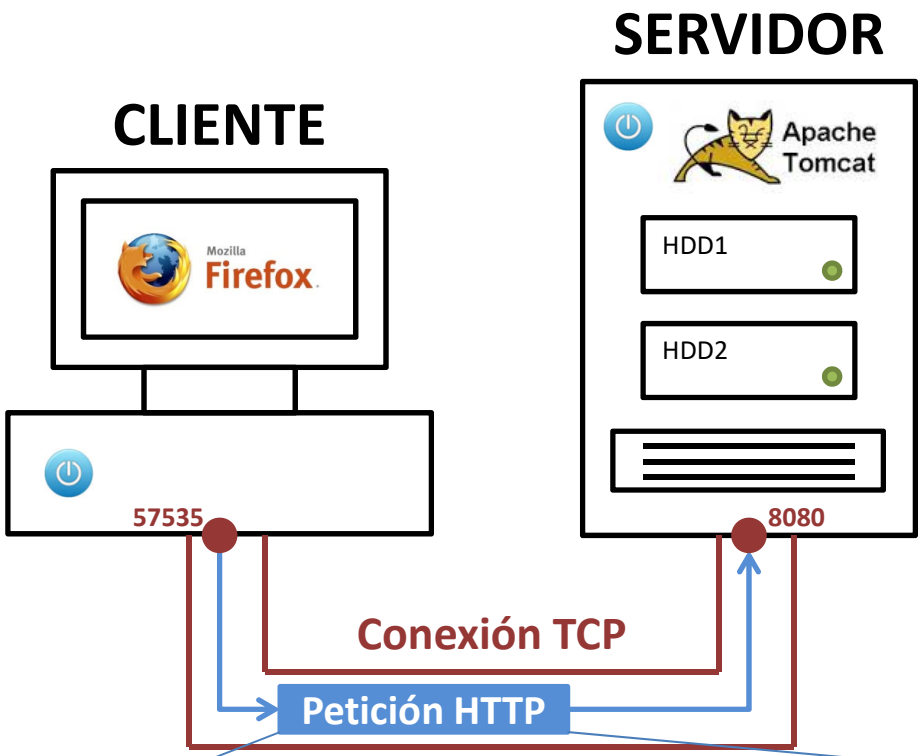


FUNCIONAMIENTO DE HTTP

A continuación se va a describir, a través de un ejemplo, el funcionamiento del protocolo HTTP. Concretamente, se van a responder las siguientes preguntas:

- ¿Qué ocurre cuando un usuario solicita recurso (por ejemplo, una página web) a través del navegador?
  - ¿Qué hace el navegador?
  - **¿Qué formato (sintaxis y semántica) tiene la petición?**
  - ¿Cómo se procesa la petición en el servidor?
  - ¿Qué formato (sintaxis y semántica) tiene la respuesta?

FUNCIONAMIENTO DE HTTP
SOLICITUD DE CLIENTE



Sintaxis de una petición HTTP

Método RequestURI HTTP/1.1
Cabeceras
CRLF*
Cuerpo del mensaje (en octetos**)

* CRLF = Carriage Return + Line Feed = \r\n = 0x0D 0x0A

** octeto = secuencia de 8 bits con valor entre 0 y 255

Petición HTTP del ejemplo

GET /recurso HTTP/1.1
Host: sw2023.com:8080
Accept: text/html
Accept-Encoding: gzip,identity;q=0.5
Accept-Language: en-US,es-ES;q=0.8
User-Agent: Mozilla Windows Escritorio

GET /recurso HTTP/1.1\r\nHost: sw2023.com:8080\r\nAccept: text/html\r\nAccept-Encoding: gzip,identity;q=0.5\r\nAccept-Language: en-US,es-ES;q=0.8\r\n User-Agent: Mozilla Windows Escritorio\r\n\r\n

FUNCIONAMIENTO DE HTTP
SOLICITUD DE CLIENTE

Sintaxis de una petición HTTP

```
Método URI HTTP/1.1
Cabeceras
CRLF
Cuerpo del mensaje (en octetos)
```

Petición HTTP del ejemplo

```
GET /recurso HTTP/1.1
Host: sw2023.com:8080
Accept: text/html
Accept-Encoding: gzip,identity;q=0.5
Accept-Language: en-US,es-ES;q=0.8
User-Agent: Mozilla Windows Escritorio
```

Método: GET

El método describe el tipo de acción CRUD (Create, Read, Update and Delete) que se desea llevar a cabo sobre el recurso. En este caso, GET → Lectura.

URI: /recurso

La identificación del recurso se puede realizar con el URI completo o con el URI relativo.

GET http://sw2023.com:8080/recurso HTTP/1.1

GET /recurso HTTP/1.1
Host: sw2023.com:8080

Cabeceras: indican las características del cliente y sus preferencias en la respuesta.

Accept: el navegador indica que acepta contenido HTML.

Accept-Encoding: el navegado indica que prefiere contenido comprimido (en formato gzip), aunque también acepta contenido no comprimido (identity).

Accept-Language: el navegador indica que su preferencia de idioma es el inglés y su segunda opción es el castellano.

User-Agent: el navegador se identifica como Mozilla sobre una plataforma Windows de escritorio.

CRLF: CR (retorno de carro) y **LF** (salto de línea).

Cuerpo del mensaje: en este caso está vacío.

FUNCIONAMIENTO DE HTTP

A continuación se va a describir, a través de un ejemplo, el funcionamiento del protocolo HTTP. Concretamente, se van a responder las siguientes preguntas:

- ¿Qué ocurre cuando un usuario solicita recurso (por ejemplo, una página web) a través del navegador?
  - ¿Qué hace el navegador?
  - ¿Qué formato (sintaxis y semántica) tiene la petición?
  - **¿Cómo se procesa la petición en el servidor?**
  - ¿Qué formato (sintaxis y semántica) tiene la respuesta?

FUNCIONAMIENTO DE HTTP
PROCESAMIENTO DE LA SOLICITUD EN EL SERVIDOR

Cuando el servidor web recibe la petición, analiza el **método** y la **URI** para saber si:

1. El recurso existe.
2. Se le puede aplicar la acción solicitada.

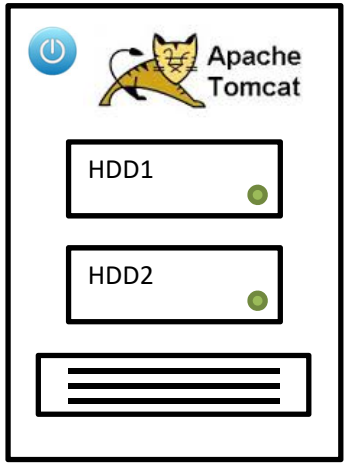
Negociación de contenido: si el recurso existe y se le puede aplicar la acción solicitada, el servidor web **analizará las cabeceras de la petición** para devolver la versión del recurso que mejor se adapte a las necesidades del cliente:

- *Accept*: text/html
- *Accept-Encoding*: gzip,identity;q=0.5
- *Accept-Language*: en-US,es-ES;q=0.8
- *User-Agent*: Mozilla Windows Escritorio

En este caso, el servidor web devuelve una respuesta cuyo contenido tiene las siguientes características:

- codificado en HTML
- sin compresión
- en castellano
- en versión clásica

SERVIDOR



NOTAS acerca del servidor:

- El alias del servidor es **sw2023.com**
- *Tomcat* es una aplicación servidora (servidor web) que implementa el protocolo HTTP.
- *Tomcat* escucha en el puerto 8080.
- *Tomcat* ofrece un recurso identificado con la URI http://sw2023.com:8080/resource.
- Dicho recurso es un documento estático disponible en texto plano y HTML, en euskera y castellano, en versión móvil y de escritorio.
- *Tomcat* no puede comprimir contenido.
- El sistema operativo implementa la pila de protocolos TCP/IP.

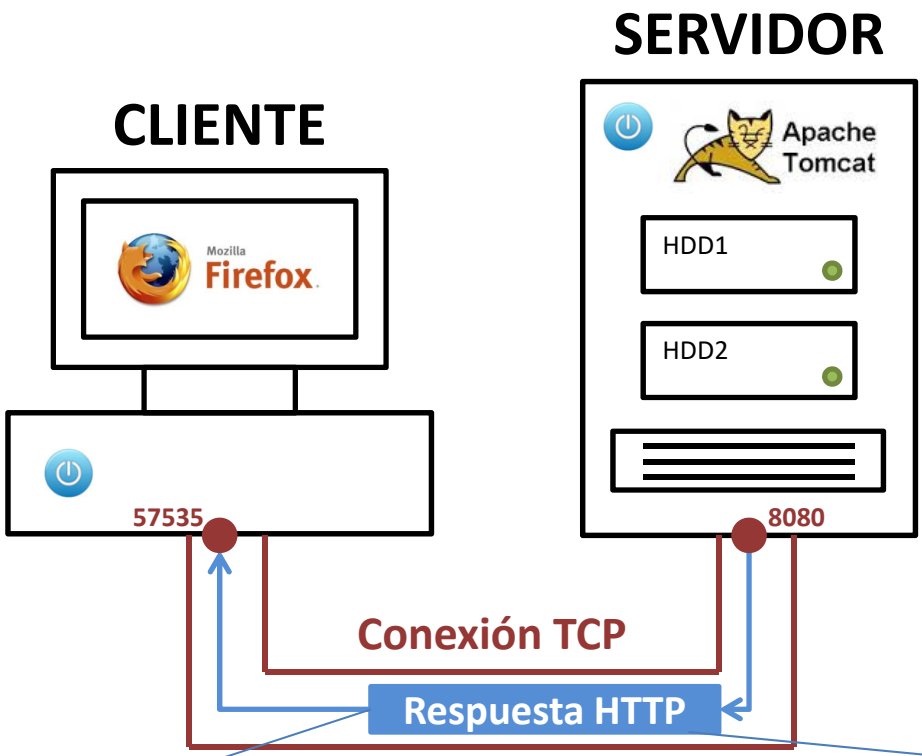
FUNCIONAMIENTO DE HTTP

A continuación se va a describir, a través de un ejemplo, el funcionamiento del protocolo HTTP. Concretamente, se van a responder las siguientes preguntas:

- ¿Qué ocurre cuando un usuario solicita recurso (por ejemplo, una página web) a través del navegador?
  - ¿Qué hace el navegador?
  - ¿Qué formato (sintaxis y semántica) tiene la petición?
  - ¿Cómo se procesa la petición en el servidor?
  - **¿Qué formato (sintaxis y semántica) tiene la respuesta?**

FUNCIONAMIENTO DE HTTP

RESPUESTA DEL SERVIDOR



Sintaxis de una respuesta HTTP

HTTP/1.1 Status Descripción
Cabeceras
CRLF
Cuerpo del mensaje (en octetos)

Respuesta HTTP del ejemplo

HTTP/1.1 200 OK
Date: Thu, 20 Nov 2015 20:25:52 GMT
Last-Modified: Tue, 17 Sep 2015 13:00:02 GMT
ETag: "1a968-3ec-4e693e61bb8b6"
Content-Length: 76
Content-Type: text/html; charset=ISO-8859-1

<html><head><title>index.html</title></head>
<body>Hello World!</body></html>

HTTP/1.1 200 OK\r\nDate: Thu, 20 Mar 2014 20:25:52 GMT\r\nLast-Modified: Tue, 17 Sep 2013 13:00:02 GMT\r\nETag: "1a968-3ec-4e693e61bb8b6"\r\n
Content-Length: 76\r\nContent-Type: text/html; charset=ISO-8859-1\r\n\r\n<html><head><title>index.html</title></head><body>Hello World!</body></html>

FUNCIONAMIENTO DE HTTP

RESPUESTA DEL SERVIDOR

Sintaxis de una respuesta HTTP

```
HTTP/1.1 Status Descripción
Cabeceras
CRLF
Cuerpo del mensaje (en octetos)
```

Respuesta HTTP del ejemplo

```
HTTP/1.1 200 OK
Date: Thu, 20 Mar 2014 20:25:52 GMT
Last-Modified: Tue, 17 Sep 2013 13:00:02 GMT
ETag: "1a968-3ec-4e693e61bb8b6"
Content-Length: 76
Content-Type: text/html; charset=ISO-8859-1

<html><head><title>index.html</title></head>
<body>Hello World!</body></html>
```

Status: 200

Código que describe el resultado de la solicitud.

Concretamente, el código 200 indica que la petición está bien formada y que ha sido procesada correctamente.

Para programas.

Descripción: OK

Descripción textual asociada al Status.

Para usuarios.

Cabeceras: caracterizan determinados aspectos de la respuesta.

Content-Length: el servidor indica la longitud (número de octetos) del contenido de la respuesta.

Content-Type: el servidor indica que el contenido es de tipo HTML y que sus octetos están codificados en latin-1 (ISO-8859-1).

FUNCIONAMIENTO DE HTTP

RESPUESTA DEL SERVIDOR

Sintaxis de una respuesta HTTP

```
HTTP/1.1 Status Descripción
Cabeceras
CRLF
Cuerpo del mensaje (en octetos)
```

Respuesta HTTP del ejemplo

```
HTTP/1.1 200 OK
Date: Thu, 20 Mar 2014 20:25:52 GMT
Last-Modified: Tue, 17 Sep 2013 13:00:02 GMT
ETag: "1a968-3ec-4e693e61bb8b6"
Content-Length: 76
Content-Type: text/html; charset=ISO-8859-1

<html><head><title>index.html</title></head>
<body>Hello World!</body></html>
```

Cabeceras: (continuación)

Date: fecha en la que el servidor creo la respuesta (formato definido en RFC 822, apartado 5 , 1s de resolución).

Last-Modified: fecha de la última modificación del recurso.

ETag: identificador de entidad*; se usa para distinguir dos versiones del mismo recurso, por ejemplo:

URI: http://ws2023.com:8080/recurso

```
Last-Modified: Tue, 17 Sep 2013 13:00:02 GMT
Content-Length: 12
Content-Type: text/plain; charset=ISO-8859-1

Hello World!
```

```
Last-Modified: Tue, 17 Sep 2013 13:00:02 GMT
Content-Length: 76
Content-Type: text/html; charset=ISO-8859-1

<html><head><title>index.html</title></head>
<body>Hello World!</body></html>
```

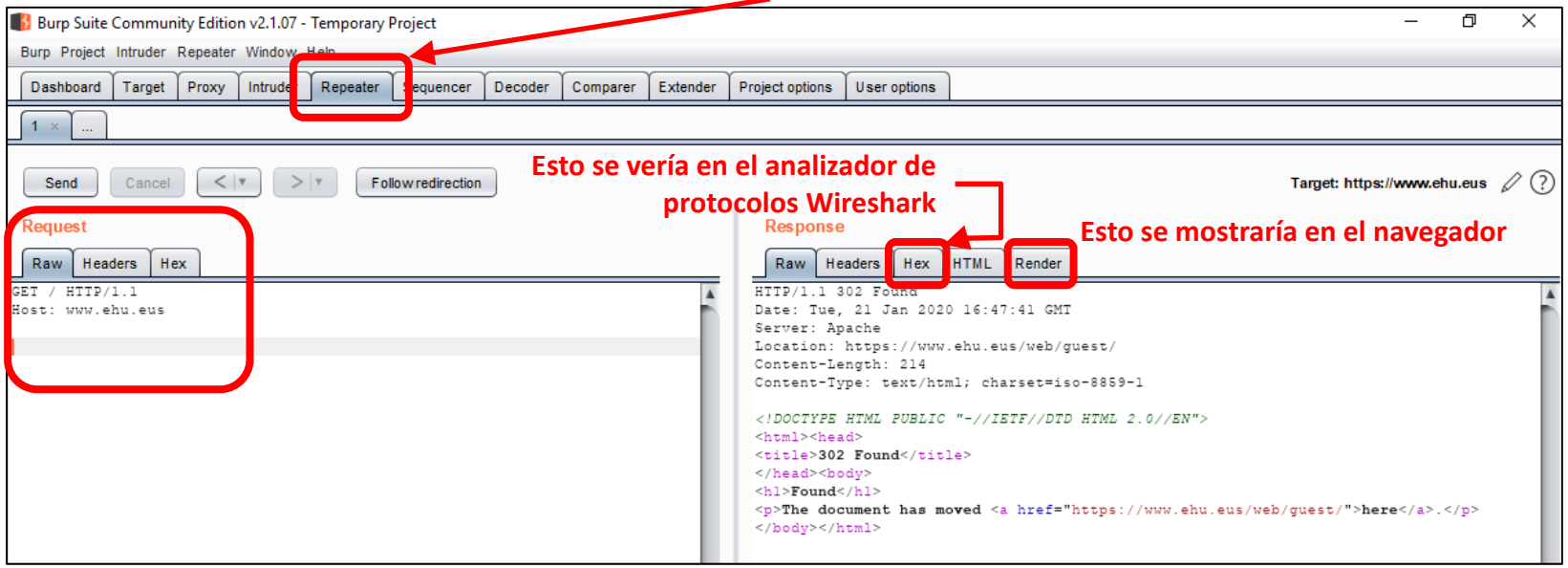
* entidad: conjunto de determinadas cabeceras y cuerpos del mensaje(RFC 2616, apartado 7).

CRLF: CR (retorno de carro) y **LF** (salto de línea)

Cuerpo del mensaje: contednio; en este caso, documento HTML (página web).

EJEMPLO USANDO BURP

– Abrir un navegador y solicitar, utilizando la barra de direcciones, el siguiente recurso: http://www.ehu.eus/
  - ¿Cuál es el URI del recurso que se muestra en el navegador?
  - Repetir la solicitud utilizando en Burp *Repeater* :



– Abrir un navegador y solicitar, utilizando la barra de direcciones, el siguiente recurso : https://www.ehu.eus/eu/home/
  - ¿Cuál es el URI del recurso mostrado?
  - Repetir la solicitud utilizando en Burp *Repeater*