3CM5

Instituto Politécnico Nacional

ESCOM

Redes de Computadoras

Práctica 5

Servidor HTTP

Alumno:

Naranjo Miranda Javier Said

Flores García Sergio

Grupo: 3CM5

**Introducción**

Un servidor web o servidor HTTP es un programa informático que procesa una aplicación del lado del servidor, realizando conexiones bidireccionales o unidireccionales y síncronas o asíncronas con el cliente y generando o cediendo una respuesta en cualquier lenguaje o Aplicación del lado del cliente. El código recibido por el cliente es renderizado por un navegador web. Para la transmisión de todos estos datos suele utilizarse algún protocolo. Generalmente se usa el protocolo HTTP para estas comunicaciones, perteneciente a la capa de aplicación del modelo OSI. El término también se emplea para referirse al ordenador.

Los programas Web rompieron con la idea clásica de documentos enlazados de hipertexto. Surge la necesidad de enviar datos desde las páginas Web al servidor (por ejemplo el usuario y la contraseña). Cuando esto sucedió tanto el HTTP como el HTML fueron modificados para soportar el envío de parámetros y formularios desde el cliente al servidor. Para esta comunicación se inventaron dos métodos llamados Get y Post.

[GET](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/GET)

El método GET solicita una representación de un recurso específico. Las peticiones que usan el método GET sólo deben recuperar datos.

[HEAD](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/HEAD)

El método HEAD pide una respuesta idéntica a la de una petición GET, pero sin el cuerpo de la respuesta.

[POST](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/POST)

El método POST se utiliza para enviar una entidad a un recurso en específico, causando a menudo un cambio en el estado o efectos secundarios en el servidor.

[PUT](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/PUT)

El modo PUT reemplaza todas las representaciones actuales del recurso de destino con la carga útil de la petición.

[DELETE](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/DELETE)

El método DELETE borra un recurso en específico.

**Desarrollo**

**Código**

HTTP.java

package http;

import java.net.\*;

import java.io.\*;

import java.util.\*;

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

import java.util.concurrent.ThreadPoolExecutor;

public class HTTP

{

public static final int PUERTO=8001;

ServerSocket ss;

class Manejador extends Thread{

protected Socket socket;

protected PrintWriter pw;

protected BufferedOutputStream bos;

protected BufferedReader br;

protected String FileName;

protected String type;

String line="";

public Manejador(Socket \_socket) throws Exception

{

this.socket=\_socket;

}

public void run()

{

try{

int i=0;

bos=new BufferedOutputStream(socket.getOutputStream());

pw=new PrintWriter(new OutputStreamWriter(bos));

File dir = new File("C:/Users/saidnDocuments/NetBeansProjects/HTTP");

String[] ficheros = dir.list();

int n = 0;

DataInputStream dis = new DataInputStream(socket.getInputStream());

byte [] b = new byte[2000];

n=dis.read(b);

line = line + new String(b,0,n);

//System.out.println(line);

if(line==null)

{

pw.print("<html><head><title>Servidor WEB");

pw.print("</title><body bgcolor=\"#AACCFF\"<br>Linea Vacia</br>");

pw.print("</body></html>");

socket.close();

return;

}

System.out.println("\nCliente Conectado desde: "+socket.getInetAddress());

System.out.println("Por el puerto: "+socket.getPort());

System.out.println("Datos: "+line+"\r\n\r\n");

if(line.indexOf("?")==-1)

{

getArch(line);

if(FileName.compareTo("")==0)

{

SendA("index.htm");

//System.out.println("Datos: "+FileName+"\r\n\r\n");

}

else if(FileName.compareTo("recursos")==0){

pw.println("HTTP/1.0 200 Okay");

pw.flush();

pw.println();

pw.flush();

pw.print("<html><head><title>SERVIDOR WEB");

pw.flush();

pw.print("</title></head><body bgcolor=\"#AACCFF\"><h1><br>Recursos Dentro Del Servidor</br></h1>");

pw.flush();

pw.print("<ul>");

pw.flush();

if (ficheros == null)

pw.print("<p>No hay ficheros en el directorio especificado</p>");

else {

System.out.println("Ficheros encontrados");

}

}

else

{

if(line.toUpperCase().startsWith("DELETE"))

{

File eliminated\_file = new File(FileName);

Checktype(FileName);

String sb = "";

if (eliminated\_file.delete()){

pw.print("\nHTTP/1.0 202 - OK\n");

pw.flush();

pw.print("Date: " + new Date()+" \n");

pw.flush();

pw.print("Content-Type: "+type+ " \n\n");

pw.flush();

pw.print("<html><body><h4>Archivo Eliminado</h4></body></html>");

pw.flush();

}

else{

pw.print("\nHTTP/1.0 404 - NOT FOUND\n");

pw.flush();

pw.print("Date: " + new Date()+" \n");

pw.flush();

pw.print("Content-Type: "+type+ " \n\n");

pw.flush();

pw.print("<html><body><h4>Archivo no Eliminado</h4></body></html>");

bos.flush();

pw.flush();

}

}

else {

SendA(FileName);

//System.out.println("Datos: "+FileName+"\r\n\r\n");

}

}

//System.out.println(FileName);

}

else if(line.toUpperCase().startsWith("GET"))

{

StringTokenizer tokens=new StringTokenizer(line,"?");

String req\_a=tokens.nextToken();

String req=tokens.nextToken();

System.out.println("Token1: "+req\_a+"\r\n\r\n");

System.out.println("Token2: "+req+"\r\n\r\n");

pw.println("HTTP/1.0 200 Okay");

pw.flush();

pw.println();

pw.flush();

pw.print("<html><head><title>SERVIDOR WEB");

pw.flush();

pw.print("</title></head><body bgcolor=\"#AACCFF\"><center><h1><br>Parametros Obtenidos..</br></h1>");

pw.flush();

pw.print("<h3><b>"+req+"</b></h3>");

pw.flush();

pw.print("</center></body></html>");

pw.flush();

}

else if(line.toUpperCase().startsWith("POST")){

StringTokenizer tokens=new StringTokenizer(line,"?");

String req\_a=tokens.nextToken();

String req=tokens.nextToken();

System.out.println("Token1: "+req\_a+"\r\n\r\n");

System.out.println("Token2: "+req+"\r\n\r\n");

pw.println("HTTP/1.0 200 Okay");

pw.flush();

pw.println();

pw.flush();

pw.print("<html><head><title>SERVIDOR WEB");

pw.flush();

pw.print("</title></head><body bgcolor=\"#AACCFF\"><center><h1><br>Parametros Obtenidos..</br></h1>");

pw.flush();

pw.print("<h3><b>"+req+"</b></h3>");

pw.flush();

pw.print("</center></body></html>");

pw.flush();

}

else

{

pw.println("HTTP/1.0 501 Not Implemented");

pw.println();

}

pw.flush();

bos.flush();

}

catch(Exception e)

{

e.printStackTrace();

}

try{

socket.close();

}

catch(Exception e)

{

e.printStackTrace();

}

}

public void getArch(String line)

{

int i;

int f;

if(line.toUpperCase().startsWith("GET"))

{

i=line.indexOf("/");

f=line.indexOf(" ",i);

FileName=line.substring(i+1,f);

}

if(line.toUpperCase().startsWith("POST"))

{

i=line.indexOf("/");

f=line.indexOf(" ",i);

FileName=line.substring(i+1,f);

}

if(line.toUpperCase().startsWith("DELETE"))

{

i=line.indexOf("/");

f=line.indexOf(" ",i);

FileName=line.substring(i+1,f);

}

if(line.toUpperCase().startsWith("HEAD"))

{

i=line.indexOf("/");

f=line.indexOf(" ",i);

FileName=line.substring(i+1,f);

}

}

public void Checktype(String fileName){

if (fileName.contains(".jpg")) {

type = "image/jpeg";

}

else if (fileName.contains(".pdf")) {

type = "application/pdf";

}

else if(fileName.contains(".doc")){

type = "application/msword";

}

else if(fileName.contains(".ico")){

type = "image/x-icon";

}

else if(fileName.contains(".txt")){

type = "text/plain";

}

else if(fileName.contains(".png")){

type = "image/png";

}

else{

type = "text/html";

}

}

public void SendA(String fileName,Socket sc)

{

//System.out.println(fileName);

int fSize = 0;

byte[] buffer = new byte[4096];

try{

DataOutputStream out =new DataOutputStream(sc.getOutputStream());

//sendHeader();

FileInputStream f = new FileInputStream(fileName);

Checktype(fileName);

int x = 0;

while((x = f.read(buffer))>0){

out.write(buffer,0,x);

}

out.flush();

f.close();

}catch(FileNotFoundException e){

}catch(IOException e){

}

}

public void SendA(String arg)

{

try{

int b\_leidos=0;

String sb = "";

BufferedInputStream bis2=new BufferedInputStream(new FileInputStream(arg));

byte[] buf=new byte[1024];

int tam\_bloque=0;

if(bis2.available()>=1024)

{

tam\_bloque=1024;

}

else

{

bis2.available();

}

Checktype(FileName);

int tam\_archivo=bis2.available();

if (line.toUpperCase().startsWith("HEAD")) {

sb = sb+"HTTP/1.0 200 ok\n";

//sb = sb +"Content-Type: "+type+ " \n";

sb = sb +"Content-Length: "+tam\_archivo+" \n";

sb = sb +"\n";

bos.write(sb.getBytes());

bos.flush();

}

else{

sb = sb+"HTTP/1.0 200 ok\n";

sb = sb +"Content-Type: "+type+ " \n";

sb = sb +"Content-Length: "+tam\_archivo+" \n";

sb = sb +"\n";

bos.write(sb.getBytes());

bos.flush();

while((b\_leidos=bis2.read(buf,0,buf.length))!=-1){

bos.write(buf,0,b\_leidos);

}

bos.flush();

bis2.close();

}

}

catch(Exception e)

{

System.out.println(e.getMessage());

}

}

}

public HTTP() throws Exception

{

ExecutorService Pool = Executors.newFixedThreadPool(5);

System.out.println("Iniciando Servidor.......");

this.ss=new ServerSocket(PUERTO);

System.out.println("Servidor iniciado:---OK");

System.out.println("Esperando por Cliente....");

for(;;)

{

Socket accept=ss.accept();

Pool.submit(new Manejador(accept));

System.out.println("Pools en hilo: "+((ThreadPoolExecutor)Pool).getPoolSize());

}

}

public static void main(String[] args) throws Exception{

HTTP sWEB=new HTTP();

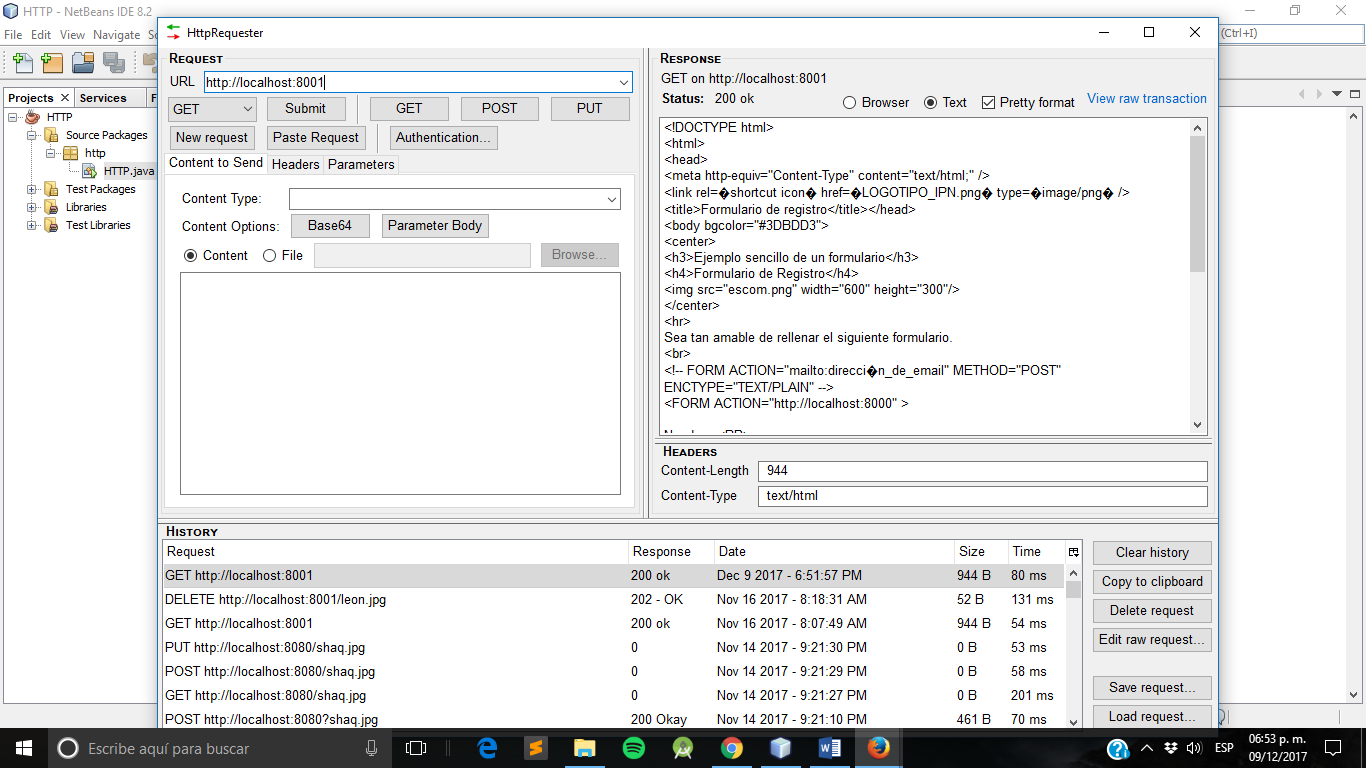
}

}

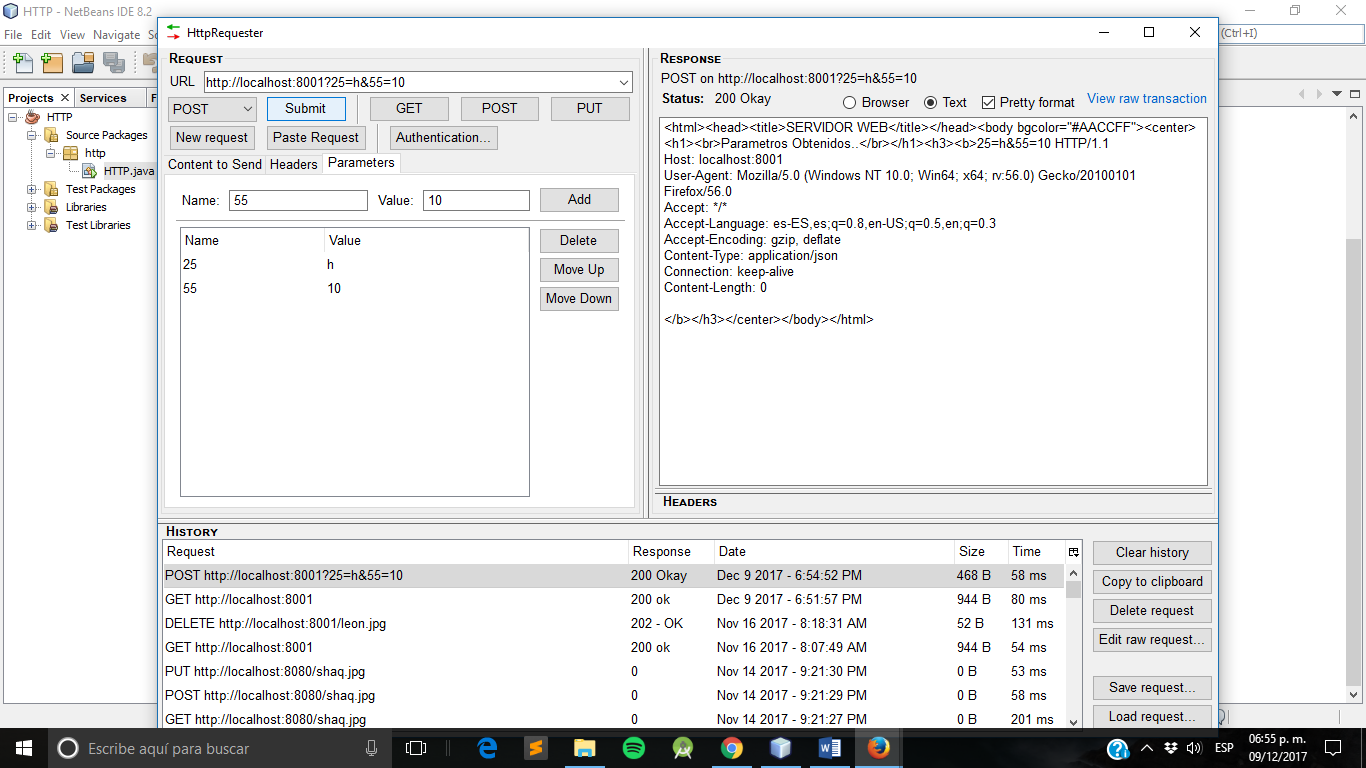
**Pruebas**

Para hacer las pruebas del servidor se utilizará una extensión de Firefox para así ejemplificar cada uno de los métodos del servidor http.

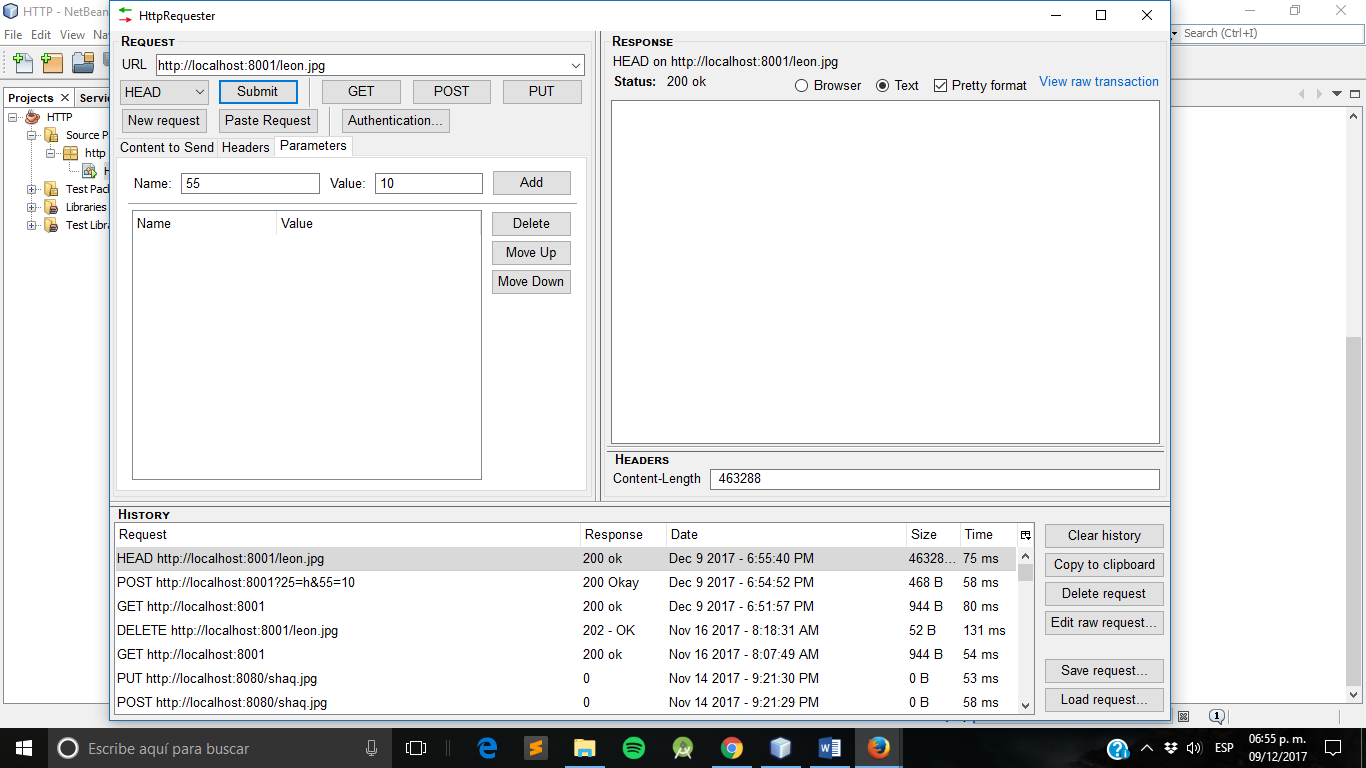
Se mostrará primero el método GET y una respuesta 200 que corresponde a un OK



Se mostrará el método POST:

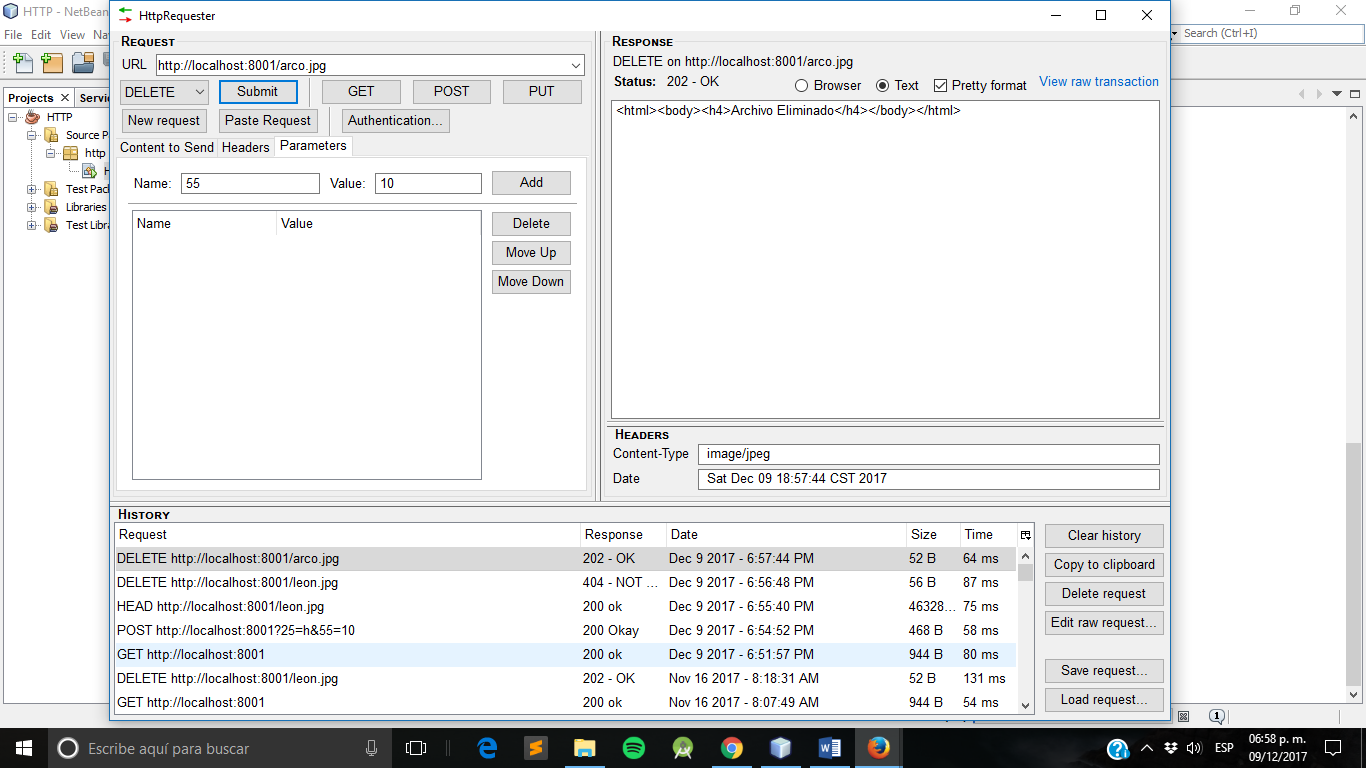


Se mostrará el método HEAD:

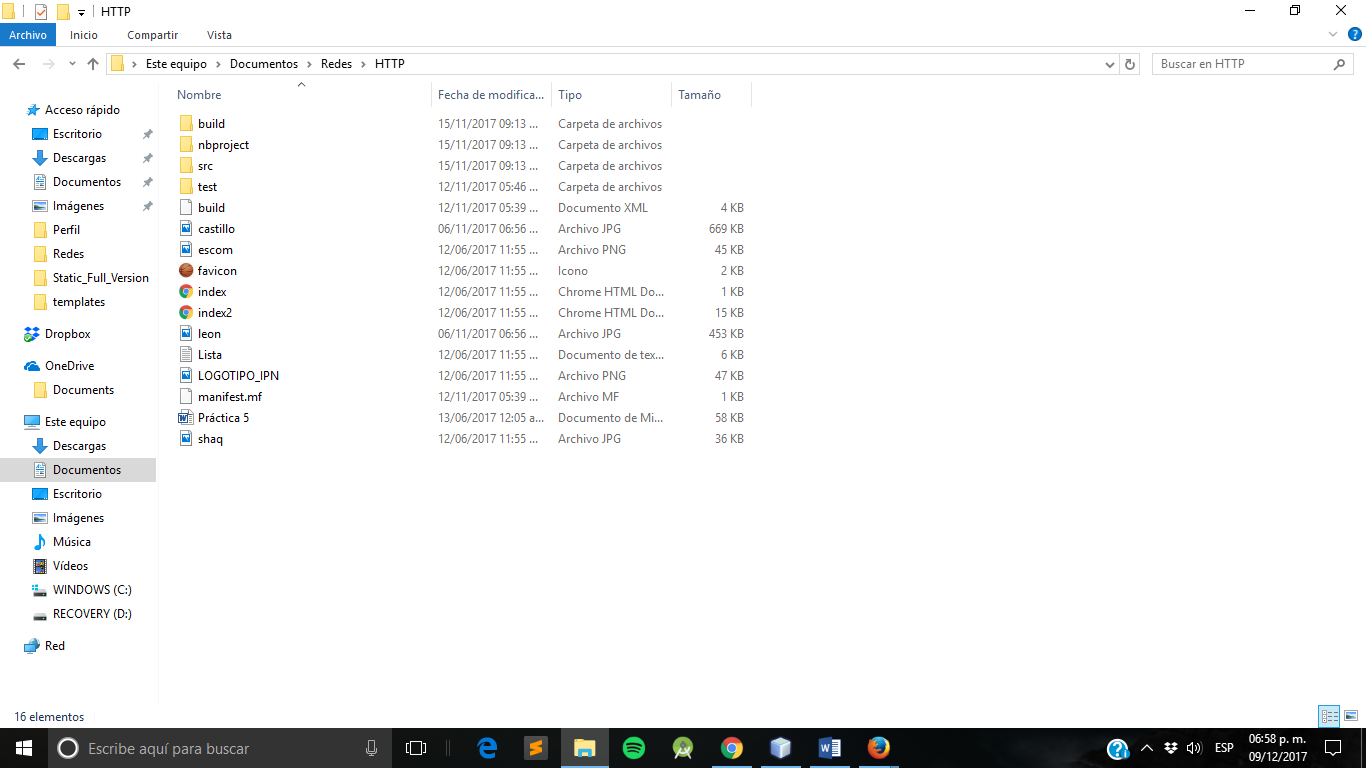


En este método se observa que solo nos da una respuesta 200 y no nos muestra contenido.

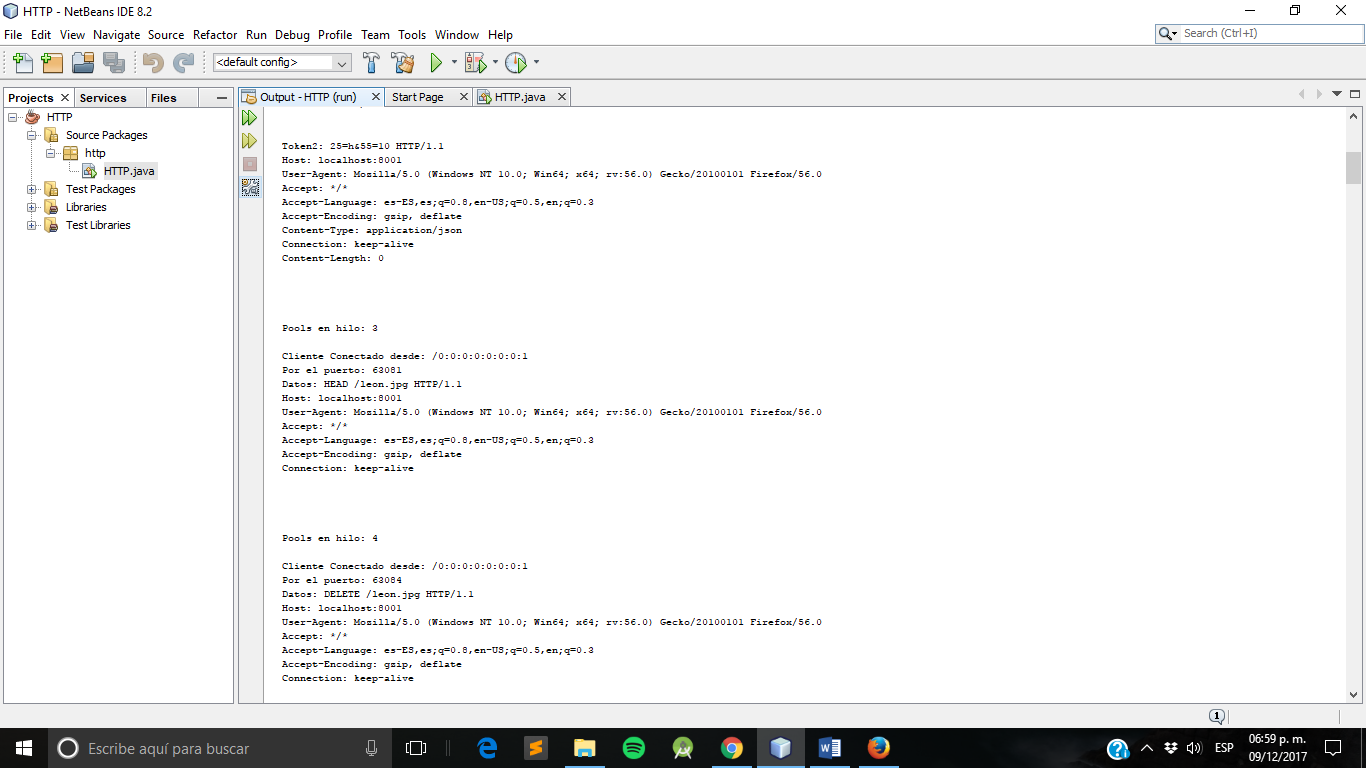
Se mostrará el método DELETE:



Se muestra que el archivo se eliminó.



Por ultimo se muestra que se utilizo un pool de conexiones para estas peticiones .



**Conclusión**

Con la ayuda de los mecanismos de sincronización y el uso de hilos podemos crear aplicaciones para atender múltiples clientes, además entendimos de una manera más explícita como es que los servidores web procesan o hacen su trabajo y como es que cada uno de los métodos se realizan según sea lo requerido por el usuario.