



http://

Índex

[1. Objectiu de la pràctica](#)

[2. Requeriments](#)

[3. Introducció a la pràctica](#)

[Comunicació en les arquitectures client-servidor](#)

[Ports més utilitzats pels diferents protocols web](#)

[Nosaltres ens centrarem amb el servei HTTP i veurem amb més detall com ho fan el client i el servidor per passar-se informació utilitzant el protocol HTTP.](#)

[Peticions de comunicació client servidor via GET i POST \(HTTP\)](#)

[Simulant una petició GET \(utilitzant telnet\)](#)

[4. Enunciat de la pràctica](#)

[Part 1](#)

[Part 2](#)

[5. Recursos addicionals](#)

[Instal·lació NODE.JS \(Linux\)](#)

[Simulació petició](#)

[Simulació petició amb un servidor utilitzant el NODEJS](#)

[Algunes webs d'interès](#)

[Codi font dels programes utilitzats](#)

Objectiu de la pràctica

- Comprensió de l'arquitectura client-servidor.
- Conèixer els protocol base de comunicació utilitzat en aplicacions web (HTTP).
- Conèixer el NodeJS (i com podem crear un servei web molt lleuger en aquest entorn).
- Normalitzar l'ús del github.

2. Requeriments

- Màquina virtual amb una distribució Linux (*XUbuntu* recomanada).
- Un navegador web (*google-chrome* o *firefox* preferiblement)
- Baixar-vos el codi font que us he preparat en el github :
https://github.com/armand-fje/j23_protocolHTTP.git
- Tenir instal·lat l'entorn NodeJS (ens permetrà llençar el servidor *testServidor.js* escrit en javascript del codi facilitat).

3. Introducció a la pràctica

El motiu d'aquest exercici és que aprengueu a seguir la transmissió d'informació (via peticions) que es fan entre un client i servidor utilitzant el protocol HTTP. Per fer-ho, us he preparat un projecte github (https://github.com/armand-fje/j23_protocolHTTP.git). En aquest projecte veureu que en el README.rd hi ha els detalls i les explicacions. Veureu que hi ha un fitxer *testClient.html* que és un simple formulari web que ens servirà per enviar la informació que contingui a un servidor web. Per utilitzar-lo només necessitem el navegador web.

Per altra banda, tenim un altre fitxer, anomenat *testServer.js* que pròpiament serà el servidor encarregat de rebre les peticions dels clients i donar-los resposta. Caldrà per tant que tingueu instal·lat el servei del *Node.js* per utilitzar aquest servidor.

Abans d'iniciar la pràctica anem a veure amb més detall quina comunicació hi ha en una arquitectura client-servidor.

Comunicació en les arquitectures client-servidor

En el següent esquema es mostra el conjunt de peticions que efectuen els diversos clients (usualment serà via navegador web) amb el servidor. Les peticions del client cap el servidor les anomenarem *request*, i les respostes del servidor cap el client *responses*.

Ports més utilitzats pels diferents protocols web

Els serveis més usuals de la web funcionen utilitzant diferents protocols.

Núm. Port	Servei (protocol)
80	HTTP
442	HTTPS
21	FTP
110	POP3
25	SMTP
23	Telnet

FELIPE IGLESIAS

UF1 -
Protocols base
funcionament d'un
servidor web.

DAW2
MP08

Llista més completa de ports Altres ports seria :

http://www.webopedia.com/quick_ref/portnumbers.asp

Nosaltres ens centrarem amb el servei HTTP i veurem amb més detall com ho fan el client i el servidor per passar-se informació utilitzant el protocol HTTP.

Peticions de comunicació client servidor via GET i POST (HTTP)

Ara veurem les peticions que una pàgina web client rep del servidor.

Obriu el navegador google chrome, premeu les tecles *Ctrl+Alt+J* i aneu a la pestanya *Network*.

Petició GET



En la Fig 1 es mostra el contingut de la petició GET quan anem al google i cerquem per una paraula clau.

FELIPE IGLESIAS

UF1 -
Protocols base
funcionament d'un
servidor web.

DAW2
MP08

Exercici 1

Exploreu la mateixa idea cercant una altra paraula clau i intenteu identificar el moviment de la xarxa.

The screenshot shows a Google search for 'agua' on the Google.es domain. The search results show approximately 321,000,000 results in 0.62 seconds. The network tab in the developer tools is open, showing two GET requests. The first request is to 'search?q=agua&ie=utf-8&oe=utf-8' on 'www.google.com' with a status of 302 and a size of 103,17 KB. The second request is to 'search?q=agua&ie=utf-8&oe=utf-8&g...' on 'www.google.es' with a status of 200 and a size of 103,17 KB.

Método	Archivo	Dominio	Tipo	Transferido
302 GET	search?q=agua&ie=utf-8&oe=utf-8	www.google.com	html	103,17 KB
200 GET	search?q=agua&ie=utf-8&oe=utf-8&g...	www.google.es	html	103,17 KB

Al buscar la palabra AGUA en el buscador de Google, es realitzen dues peticions GET, una amb el codi **302**, que indica que la pàgina s'ha mogut i un codi **200** de petició correcta.

Simulant una petició GET (utilitzant *telnet*)

El *telnet* és un protocol que emula un terminal remot per a connectar-se a una màquina multiusuari. Igual que amb el protocol FTP, cal el programari necessari i un protocol específic per aquest servei. El port que s'utilitza és generalment el 23. Va caure en desús en favor del *SSH (Secure Shell)*, que "es pot entendre" com la versió xifrada del *telnet*.

El farem servir per fer una petita simulació.

```
telnet www.joan23.fje.edu 80
GET /l-escola/qui-som
```

Exercici 2

Intenteu fer servir aquesta mateixa idea en les vostres pàgines web (comproveu que no hagin caigut).

La idea d'aquest segon exercici és simular el funcionament de com es comunica el client i el servidor utilitzant el protocol HTTP fent peticions (*requests* & *responses*) via GETs.

Al realitzar la petició a www.pr03_ampliacion.comlu.com, mostra:

- El servidor es de tipus HTTP/1.1
- El codi **200** indica una petició correcta.
- La data del servidor.
- El tipus de servidor, Apache.
- El site està sota la tecnologia PHP.
- El contingut de 2539 caràcters de tipus text

```
felipe@debian: ~  
Archivo  Editar  Ver  Buscar  Terminal  Ayuda  
felipe@debian:~$ telnet pr03_ampliacion.comlu.com 80  
Trying 31.170.161.96...  
Connected to pr03_ampliacion.comlu.com.  
Escape character is '^]'.  
get / http/1.1  
host: pr03_ampliacion.comlu.com  
  
HTTP/1.1 200 OK  
Date: Tue, 05 Jan 2016 22:01:50 GMT  
Server: Apache  
X-Powered-By: PHP/5.2.17  
Content-Length: 2539  
Connection: close  
Content-Type: text/html
```

4. Enunciat de la pràctica

Aquesta pràctica està dividida en 2 parts.

Aquesta pràctica va orientada a que us familiaritzeu amb la manera que tenen de comunicar-se el client i el servidor via el protocol HTTP.

Feu un *clone* del codi del github (el fork que us hàgiu fet de https://github.com/elvostreusuari/j23_protocolHTTP.git). Aneu fent *commit's* de tots els canvis que us demani del codi font.

En finalitzar aquesta pràctica, creeu una carpeta `./doc` a l'arrel del vostre projecte, i afegiu-hi el `.pdf` d'aquest document (que sobre-entenc que n'haureu fet una còpia i l'estareu editant utilitzant el google docs..

Instal·leu el NodeJS a la vostra màquina virtual (linux). Aneu a l'arrel del projecte i engegueu, tal com hem vist a classe, el servidor lleuger que us he preparat (*node testServer.js*). Obriu el *testClient.html* amb el google-chrome i premeu Ctr+Shift+J (mode debug). Si mireu el codi font de *testClient.html* veureu que per defecte enviarà el contingut del formulari a *localhost* pel port 8080.

```
sudo apt-get install nodejs  
sudo apt-get install apache2
```

```
sudo git clone https://github.com/figlesiasma/j23\_protocolHTTP.git
```

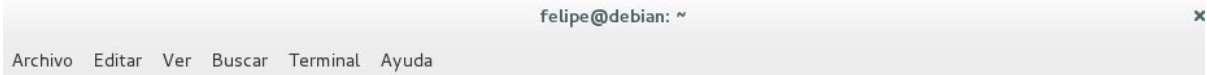

Part 1

- 1) Fes la pràctica del *telnet* (veure l'apartat 3), connecta't a l'escola (<http://www.joan23.fje.edu/>) pel port 80, i fes un GET d'alguna link (petició d'una opció). Fes una captura de pantalla i explica amb les teves paraules el què observes.

Al realitzar a petició sobre www.joan23.fje.edu/ca/serveis, es mostren una varietat de capçeleres.

El servidor utilitza el servei HTTP/1.1 i accepta la petició mitjançant el codi **200**

També mostra la data, el tipus de servidor, el llenguatge utilitzat, la codificació del llenguatge, la assignació de la cookie, etc.



```
felipe@debian: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
felipe@debian:~$ telnet www.joan23.fje.edu 80  
Trying 54.75.247.37...  
Connected to www.joan23.fje.edu.  
Escape character is '^]'.  
get /ca/serveis http/1.1  
host: www.joan23.fje.edu  
  
HTTP/1.1 200 OK  
Date: Tue, 05 Jan 2016 22:34:06 GMT  
Server: Apache/2.4.7 (Ubuntu)  
X-Drupal-Cache: MISS  
Expires: Sun, 19 Nov 1978 05:00:00 GMT  
Last-Modified: Tue, 05 Jan 2016 22:34:06 GMT  
Cache-Control: no-cache, must-revalidate, post-check=0, pre-check=0  
ETag: "1452033246"  
Content-Language: ca  
Link: <http://www.joan23.fje.edu/ca/node/378>; rel="shortlink",<http://www.joan23.fje.edu/ca/node/378>;  
rel="canonical"  
Set-Cookie: SESS9816c0cbbf8707a16f9354b753e8eb35=9x7DN_14vtzde-fqsTdFdl1Vd-i4Sz5VInWHLVqMkeY; expires=F  
ri, 29-Jan-2016 02:07:38 GMT; Max-Age=2000000; path=/; domain=.joan23.fje.edu; HttpOnly  
Vary: Accept-Encoding  
Transfer-Encoding: chunked  
Content-Type: text/html; charset=utf-8
```

- 2) En el protocol HTTP hi ha més accions que es poden fer a més del GET i el POST. Per exemple, explica que fa amb les teves paraules la opció TRACE, OPTIONS i CONNECT. Utilitza la tècnica del telnet per provar aquestes opcions. (Pots cercar aquesta informació per internet o mirar aquesta pàgina per exemple :

<http://www.vozidea.com/protocolo-http-desarrolladores-dos>)

- TRACE: s'utilitza per comprovar que les peticions son rebudes correctament i amb finalitat de depurar el codi. Està desactivat a www.joan23.fje.edu indicat amb el codi HTTP 405

```
felipe@debian:~$ telnet www.joan23.fje.edu 80
Trying 54.75.247.37...
Connected to www.joan23.fje.edu.
Escape character is '^]'.
TRACE / HTTP/1.0
Host:www.joan23.fje.edu
```

```
HTTP/1.1 405 Method Not Allowed
Date: Wed, 06 Jan 2016 00:14:25 GMT
Server: Apache/2.4.7 (Ubuntu)
Allow:
```

- OPTIONS: mostra els mètodes suportats per el servidor. En aquest cas, la petició és correcta amb el codi HTTP 200 però no mostra l'etiqueta Allow que mostra els mètodes suportats pel servidor, que voldrà dir que estan desactivats.

```
felipe@debian:~$ telnet www.joan23.fje.edu 80
Trying 54.75.247.37...
Connected to www.joan23.fje.edu.
Escape character is '^]'.
OPTIONS / HTTP/1.0
Host:www.joan23.fje.edu
```

```
HTTP/1.1 200 OK
```

- CONNECT: és el mètode per connectar-se a un servidor extern usant un servidor proxy.

- 3) T'hauràs adonat que alguna vegada apareix l'error 404 quan no es pot accedir una pàgina a internet. A què es deu? Explica també el 200, 202, 204, 302, 400, 401 i 500.

(Pots cercar aquesta informació per internet o mirar aquesta pàgina per exemple :

<http://librosweb.es/tutorial/los-codigos-de-estado-de-http/>)

404. La petició d'un recurs que el servidor no troba per què s'ha eliminat/modificat

200. Petició correcta i completada.

202. Petició correcta però no completada.

204. Petició correcta però no hi ha contingut.

302. El recurs s'ha mogut i el servidor redirigeix automàticament.

400. Sol·licitut incorrecta.

401. Sol·licitut no autoritzada. Quan hi ha un servei login pel mig.

500. Error intern del servidor.

Part 2

- 4) Amb la pàgina del *testClient.html*, i prement Ctrl+Shift+J, i ves a la pestanya de "Network". A continuació refresca la pàgina (Ctrl+F5) i fes una captura del que estàs veient. Què vol dir el missatge "200 OK" de la columna que posa "Status"? Quin valor hi posa a la columna *Type*? Quin tipus de petició (*Method*) es tracta?

The screenshot shows a web browser window with the address bar displaying `localhost/j23_protocolHTTP/testClient.html`. The page content includes a form with a "Nom:" label, a text input field, an "Accio A" button, a "Text:" label, another text input field, and an "Envia" button. Below the form, the "Developer Tools" panel is open, showing the "Network" tab. The network log displays a single request to `testClient.html` with a status of 200, method of GET, and type of document. The timeline shows the request was completed at 21ms.

Name	Method	Status	Protocol	Type	Initiator	Size	Time	Connection	Server	Timeline
testClient.html	GET	200	http/1.1	document	Other	782 B	21ms	Keep-Alive	Apache/2.4.10 (Debian)	

1 requests | 782 B transferred | Finish: 21 ms | DOMContentLoaded: 494 ms | Load: 489 ms

Per realitzar aquest exercici i surti el codi HTTP 200, s'ha de tenir instal·lat Apache i obrir *testClient.html* dins de la carpeta HTML del servidor Apache, si no, mostrarà Finished en Status.

1. Indica que la petició ha sigut correcta i completada
2. Es de tipus Document
3. Mètode GET

- 5) A continuació (recorda que el *testServer.js* ha d'estar escoltant les peticions), amb el "mode debug" obert al google-chrome, omple el formulari i prem el botó "Envia". Fes una captura del que estàs veient.

```
nt":"Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/47.0.2526.106 Safari/537.36", "referer": "http://localhost/j23_protocolHTTP/testClient.html", "accept-encoding": "gzip, deflate, sdch", "accept-language": "es-ES, es; q=0.8, ca; q=0.6, en; q=0.4"} Capçalera (header): {"host": "192.168.1.14:8080", "connection": "keep-alive", "user-agent": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/47.0.2526.106 Safari/537.36", "accept": "*/*", "referer": "http://192.168.1.14:8080/?nom=felipe&op=accioA&notes=Hola", "accept-encoding": "gzip, deflate, sdch", "accept-language": "es-ES, es; q=0.8, ca; q=0.6, en; q=0.4"}
```

The screenshot shows a web browser window with the address bar displaying `192.168.1.14:8080/?nom=felipe&op=accioA¬es=Hola`. The page content shows "Hola, benvingut al servei ...". Below the browser window, the Chrome Developer Tools Network tab is open, showing a list of requests. The first request is a GET request to `?nom=felipe&op=accioA¬es=Hola` with a status of 200 and a type of document. The second request is a GET request to `favicon.ico` with a status of 200 and a type of text/plain. The bottom of the Developer Tools window shows a summary: 2 requests, 298 B transferred, Finish: 767 ms, DOMContentLoaded: 542 ms, Load: 533 ms.

Name	Method	Status	Protocol	Type	Initiator	Size
<code>?nom=felipe&op=accioA&notes=Hola</code>	GET	200	http/1.1	document	Other	
<code>favicon.ico</code>	GET	200	http/1.1	text/plain	Other	

2 requests | 298 B transferred | Finish: 767 ms | DOMContentLoaded: 542 ms | Load: 533 ms

- 6) Fixa't que a la columna *Type* hi ha el valor "text/plain". Només hi ha un recurs que es carregui (la resposta que el servidor t'ha donat). Clica-hi. Veuràs que s'ha obert un altre *finestra* amb diverses pestanyes (*Headers*, *Preview*, *Response* i *Timing*). Ves a la pestanya *Headers*. Fes un copy+paste de la línia que posa "*Request URL*". Pots distingir-hi el contingut del formulari que has acabat d'enviar? Fes-ne una captura de pantalla.

Request URL:

<http://192.168.1.14:8080/?nom=felipe&op=accioA¬es=Hola>

S'ha enviat el contingut insertat als diferents camps del formulari per el metode GET.

The screenshot shows the 'Headers' tab of a web browser's developer tools. The 'General' section displays the Request URL as 'http://192.168.1.14:8080/?nom=felipe&op=accioA¬es=Hola', the Request Method as 'GET', the Status Code as '200 OK', and the Remote Address as '192.168.1.14:8080'. The 'Response Headers' section shows 'Connection: keep-alive', 'Date: Wed, 06 Jan 2016 16:49:09 GMT', and 'Transfer-Encoding: chunked'. The 'Request Headers' section lists various headers including 'Accept', 'Accept-Encoding', 'Accept-Language', 'Cache-Control', 'Connection', 'Host', 'Referer', 'Upgrade-Insecure-Requests', and 'User-Agent'. The 'Query String Parameters' section shows 'nom: felipe', 'op: accioA', and 'notes: Hola'.

×	Headers	Preview	Response	Timing
▼ General				
Request URL: http://192.168.1.14:8080/?nom=felipe&op=accioA¬es=Hola				
Request Method: GET				
Status Code: 200 OK				
Remote Address: 192.168.1.14:8080				
▼ Response Headers view source				
Connection: keep-alive				
Date: Wed, 06 Jan 2016 16:49:09 GMT				
Transfer-Encoding: chunked				
▼ Request Headers view source				
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8				
Accept-Encoding: gzip, deflate, sdch				
Accept-Language: es-ES,es;q=0.8,ca;q=0.6,en;q=0.4				
Cache-Control: max-age=0				
Connection: keep-alive				
Host: 192.168.1.14:8080				
Referer: http://localhost/j23_protocolHTTP/testClient.html				
Upgrade-Insecure-Requests: 1				
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/47.0.2526.106 Safari/537.36				
▼ Query String Parameters view source view URL encoded				
nom: felipe				
op: accioA				
notes: Hola				

- 7) Modifica el codi del *testClient.html* per que envii el missatge via method POST. Fes un copy+paste de la línia que posa “*Request URL*”.
- Pots distingir-hi el contingut del formulari que has acabat d'enviar ? (mou l'scrollbar fins a baix de tot.

1. Amb el mètode POST, la informació no s'envia via url
2. La informació introduïda al formulari es pot visualitzar en l'apartat FORM DATA.

▼ General

Request URL: http://192.168.1.15:8080/
Request Method: POST
Status Code: 200 OK
Remote Address: 192.168.1.15:8080

▼ Response Headers [view source](#)

Connection: keep-alive
Date: Wed, 06 Jan 2016 17:05:36 GMT
Transfer-Encoding: chunked

▼ Request Headers [view source](#)

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: es-ES,es;q=0.8,ca;q=0.6,en;q=0.4
Cache-Control: max-age=0
Connection: keep-alive
Content-Length: 31
Content-Type: application/x-www-form-urlencoded
Host: 192.168.1.15:8080
Origin: http://localhost
Referer: http://localhost/j23_protocolHTTP/testClient.html
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/47.0.2526.106 Safari/537.36

▼ Form Data [view source](#) [view URL encoded](#)

nom: felipe
op: accioA
notes: Hola

- 8) Modifica el codi del *testClient.html* i afegeix una imatge (posa la imatge a una subcarpeta *.img*) i el codi pertinent perquè es vegi la imatge just damunt del formulari. Amb el “mode debug” del google-chrome obert, refresca la pantalla i fes una captura. Explica el què estàs veient al grid que posa “*Network*”.



Nom : Accio A ▼

Text :

Envia

Developer Tools - http://locali				
Elements Console Sources Network Timeline Profiles F				
View: [Icons] [] Preserve log [] Disable cache [] N				
100 ms 200 ms 300 ms 400 ms 50				
Name	Method	Status	Protocol	Type
testClient.html	GET	200	http/1.1	document
cobi.JPG	GET	304	http/1.1	jpeg

La imatge mostra un Type jpeg i Status 304, que vol dir que no ha sigut modificada desde l'última vegada que ha estat requerida per estalviar tràfic.

- 9) Afegeix també codi javascript (en una carpeta a part .js) per tal que al canviar a “opció B” (menú selector), t’aparegui un missatge (un alert que posi “has seleccionat la opció B”). Amb el “mode debug” del google-chrome obert, refresca la pantalla i fes una captura. Explica el què estàs veient al grid que posa “Network”.

S’Afegeix el link de l’arxiu extern.

```
<script type="text/javascript" src="js/funcion.js"></script>
```

Es crida a la funció mensaje amb el valor del paràmetre del select

```
<select name="op" onchange="mensaje(this.value)">
```

La funció que rep el paràmetre

```
function mensaje(opcion){
    if(opcion === "accioB") alert("has seleccionat l'opció B");
};
```

El debug mostra que s’ha requerit i mostrat correctament tots els elements, inclòs el arxiu extern funcion.js que conté la funció de mostrar un alert.

Name	Method	Status	Protocol	Type	Initiator
testClient.html	GET	200	http/1.1	document	Other
funcion.js	GET	200	http/1.1	script	testClient.html:10
cobi.JPG	GET	304	http/1.1	jpeg	testClient.html:13

3 requests | 1.5 KB transferred | Finish: 673 ms | DOMContentLoaded: 1.11 s | Load: 1.14 s


- 10) Contesta : quins elements tarden més en enviar-se? (veure *Timing*).

De més a menys: Cobi 305ms, funcions 207ms i testClient.html 11ms


testClient.html	GET	200	http/1.1	document	Other	866 B	11ms
funcion.js	GET	200	http/1.1	script	testClient.html...	446 B	297 ms
cobi.JPG	GET	304	http/1.1	jpeg	testClient.html...	181 B	305 ms

- 11) Ves a la pàgina de yahoo (www.yahoo.com) i amb el menu de debug (Ctrl+Shift+J), ordena per "Type" (clica la capçalera del grid la paraula "Type"), i tots els recursos apareixeran ordenats (html, css, imatges, ...). Refresca la pàgina (Ctrl+F5). Indica quins han tardat més temps en carregar-se i quins eren més grans ("pesaven més"). Fes-ne una captura.

El recurs més pesat

Name	Method	Status	Protocol	Type	Initiator	Size ▼	Time
 ?p=us	GET	200	h2	document	https://www.y...	89.5 KB	1.53 s

Recurs que més ha tardat en carregar.

Name	Method	Status	Protocol	Type	Initiator	Size	Time ▼
 yahoo_en-US_f_p_142x37.png	GET	304	h2	png	g-r-min.js:1	372 B	3.95 s

NOTA: En recursos addicionals hi ha informació complementària per facilitar-vos algun d'aquests punts. Qualsevol dubte (armand.gutierrez@fje.edu).

Recursos addicionals

Instal·lació NODE.JS (Linux)

Baixar-se i instal·lar el codi font../

```
wget http://nodejs.org/dist/node-v0.4.4.tar.gz
tar -xzf node-v0.4.4.tar.gz
cd node-v0.4.4/
./configure
sudo make install
```

(Nota : `sudo apt-get install libssl-dev g++`)

Alternativament podeu utilitzar :

```
sudo apt-get install nodejs
)
```

Comprovar que la instal·lació funciona correctament.

```
$ node
> console.log('Hello World');
Hello World
```

Simulació petició

Podem per exemple simular una petició (un altre test, no es pretén que entengueu el codi) :

```
>
var http = require('http');
var options = {method: 'HEAD', host: 'stackoverflow.com', port: 80, path: '/'};
var req = http.request(options, function(res) {
  console.log(JSON.stringify(res.headers));
})
);
req.end();
```

Simulació petició amb un servidor utilitzant el NODEJS

Feu un *fork* d'aquesta pràctica, i feu el clone pertinent.

p.ex) git clone https://github.com/armand-fje/j23_protocolHTTP.git

És important que feu el *fork* perquè caldrà que efectueu una sèrie d'exercicis.

Per llençar-lo :

```
node testServer.js
```

Ara podeu comprovar en una altra pestanya del terminal que cada vegada que es fa una petició (p.ex utilitzant la comanda *curl*), escriu un petit missatge :

```
$ curl localhost:8080c
```

```
Hello Http
```

(Nota : Evidentment si no teniu el curl instal·lat *sudo apt-get install curl*)

Algunes webs d'interès

<http://nodeguide.com/beginner.html>

<http://stackoverflow.com/questions/4295782/how-do-you-extract-post-data-in-node-js>

Notes addicionals per visualitzar el contingut del request:

```
console.log(JSON.stringify(req.body));
```

```
console.log('req.body.name', req.body['name']);
```

Codi font dels programes utilitzats

Codi Servidor.

```
/*                                     */  
/* Pràctica 1 UF2 - Arquitectura client servidor */  
/*                                     */  
/*      Suport : armand.gutierrez@fje.edu */
```

```
// El protocol que faré servir
```

```
require('http');
```

```
var server = http.createServer(function(req, res) {  
  // Mostrem per pantalla la petició (request) que m'ha enviat el client ...  
  console.log("Capçalera (header): " + JSON.stringify(req.headers));  
  // Li (servidor) responc un response al client que m'envia ...  
  res.writeHead(200);  
  res.end('Hola, benvingut al servei ...\n');  
});  
// Escolto les peticions pel port 8080  
server.listen(8080);
```

Codi Client.

```
/* testClient.html */  
<html>  
<header>  
  <title> Test NodeJS </title>  
  <!--  
    /*                               */  
    /* Pràctica 1 UF2 - Arquitectura client servidor    */  
    /*                               */  
    /*      Suport : armand.gutierrez@fje.edu    */  
  -->  
</header>  
<body>  
<!-- Formulari de proves per provar NodeJs -->  
<form method="get" action="http://127.0.0.1:8080" ><p>  
  Nom : <input type="text" name="nom" />  
    <select name="op">  
      <option value="accioA" selected="selected">Accio A</option>  
      <option value="accioB">Accio B</option>  
    </select>  
</p>  
<p>  
  Text : <textarea rows="5" cols="30" name="notes"></textarea>  
</p>  
  <button type="submit"> Envia </button>  
</form>  
<!-- proves alternatives : curl localhost:8080-->  
</body>  
</html>
```