TREBALL FINAL DE GRAU

# Nucli d'un servidor web


UOC
Universitat Oberta de Catalunya
www.uoc.edu

**Autor: David Delgado Dueñas**

**Tutor: Enric Peig Olivé**



David Delgado Dueñas

Santa Coloma de Gramenet (BCN) – Espanya

ddelgadodu@uoc.edu

# Taula de contingut

# ÍNDEX D'IL·LUSTRACIONS Y TAULES

## IL·LUSTRACIONS



## TAULES

PREFACI

# ABSTRACT

## A WEB SERVER'S CORE

The Web Service is nowadays the most important service on the Internet. Since Internet (as ARPANET) was created, at late 60s, has a long history until in 1994 the investigators of CERN released to the world the WWW protocol. Since that moment, the path of the Internet has been closely linked to Web Service and Web Servers.

The communication between computers has TCP as protocol since the first steps of the rudimentary Internet, and later it made a little change to become in TCP/IP protocol in order of being more reliable. Web Servers use this protocol for the transport of data, but they have another protocol to control how Web Servers and clients communicate: HTTP.

A Web Server is a program that runs on a computer and controls the flow of HTTP traffic between server and client. This traffic is an interchange of messages and resources with a predefined format. HTTP has a very strict format with a defined number of methods and states.

The objective of this TFG is the development of the core of multi-OS tiny Web Server executable without installation, with an easy configuration and implementing the HTTP protocol. To achieve these objectives the server has been programmed in the Java language that permits the use in many Operating Systems without a change in code. In order to make it easy to use, it has a graphical interface to configure a couple of variables, a button to start the service and an emulation of console to see what the server is doing. This graphical interface has been programmed under Java using a JSwing Frame.

# NUCLI D'UN SERVIDOR WEB

El servei Web és avui dia el servei més important a Internet. Des de que Internet (com ARPANET) va ser creat, a finals dels 60, hi ha una llarga història fins que el 1994 els investigadors del CERN van mostrar al món el protocol WWW. Des d'aquest moment, les passes d'Internet han estat estretament vinculades al servei Web i als servidors web.

La comunicació entre ordinadors té TCP com a protocol des dels primers passos d'una Internet rudimentària, més tard es va fer un petit canvi per convertir-se en el protocol TCP / IP per tal de ser més fiable. Els servidors web utilitzen aquest protocol per al transport de dades, però tenen un altre protocol per controlar com els servidors web i els clients es comuniquen: HTTP.

Un servidor web és un programa que s'executa en un ordinador i controla el flux de trànsit HTTP entre el servidor i el client. Aquest trànsit és un intercanvi de missatges i recursos amb un format predefinit. HTTP té un format molt estricte amb un nombre definit de mètodes i estats.

L'objectiu d'aquest TFG és el desenvolupament del nucli d'un petit servidor Web multiplataforma, executable sense necessitat d'instal·lació, amb una fàcil configuració i implementant el protocol HTTP. Per aconseguir aquests objectius, el servidor s'ha programat en llenguatge Java que permet l'ús en molts sistemes operatius sense canvis en el codi. Per tal de fer que sigui fàcil d'utilitzar, té una interfície gràfica per configurar un parell de variables, un botó per iniciar el servei i una emulació de la consola per veure el que el servidor està fent. Aquesta interfície gràfica ha estat programat en Java utilitzant un marc JSwing.

PART 0

# PLANIFICACIÓ DEL TREBALL

## Presentació

Des de bon començament, un treball final de grau cal que sigui un treball sobre una matèria engrescadora per l'alumne. Després de molt repassar les possibles opcions per dur a terme el meu treball final de grau (TFG), vaig escollir fer-lo sobre Xarxes de Computadors.

Entre les opcions presentades pels docents vaig trobar "Nucli d'un servidor web", on es detallava que cal realitzar el cor principal d'un servidor web. Vaig escollir fer aquesta opció donat que em permetrà obtenir majors coneixements en els àmbits següents:

- Programació, en aquesta cas Java.
- Servidors Webs
- Llenguatge i protocol HTML

Amb aquesta visió i aquests objectius en quant a coneixement vaig a començar el meu periple final en aquest Grau d'Enginyeria Informàtica que estic realitzant a la UOC després de 5 anys.

# Objectius

Pel que fa als objectius des de l'àrea de Xarxes de Computadors, es plantegen una sèrie de requeriments que poden ser entesos com el lloc al que volem arribar amb aquest treball, o dit d'una altra manera allò que volem obtenir. Aquest objectius a obtenir són:

- Visió de la situació actual: Arribar a saber quin és la situació actual pel que fa a l'ús de servidors web. Caldrà veure com s'estan fent servir, quina es la distribució de les solucions al mercat. Per altra banda aquest estudi previ també ens permetrà obtenir una visió real de on podem situar el nostre producte si el seu desenvolupament continués.

- Avaluació de la utilitat: Buscar el nínxol que podria cobrir el meu programa, en principi havia pensat en que sigues un tiny-server que permetés donar accés al servidor sense instal·lació.

- Avaluació de la usabilitat (educació, empreses públiques/privades): Pot tenir un paper educatiu donada la seva senzillesa d'ús? En un primer moment no ho veig com un servidor corporatiu, però si que ho veig com un servidor fàcil d'usar o per tots els públics. El seu ús, per tant, en un principi és no professional?

  Caldrà acabar de resoldre aquestes qüestions durant el procés del TFG.

- Disseny de l'aplicació: El disseny de l'aplicació en un principi hauria de dur algun tipus de configurador, però donada la seva naturalesa de servei en segon pla no crec que sigui necessari una interfície durant el servei. En aquest apartat potser es podria fer servir JPanel que permet un panell de configuració exportable a diferents sistemes operatius. Doncs amb aquest plantejament inicial caldrà veure si es possible i adient el disseny proposat.

- Implementació: La implementació del projecte es durà a cap amb Java. Serà la mateixa solució aportada en Java la resposta aquest objectiu.

- Bibliografia i treballs relacionats: En aquest apartat entenc que cal obtenir informació i utilitzar-la per poder portar a cap el projecte. Aquestes referències quedaran plasmades al projecte però alhora caldrà mostrar-les de forma ordenada en la bibliografia del treball.

Així doncs, ens plantegem la creació de una solució a un problema de l'àrea de xarxes, en aquest cas concret l'obtenció d'un servidor web senzill i portable. En la seva implementació haurà de ser multithread, també haurà de ser multiprocés donat que vull separar la part visual del servidor. Estarà implementat en Java per poder funcionar en entorns Windows, Linux,...

## Productes Finals

Un cop sabem quins son els objectius d'aquest TFG, cal que aquests es materialitzin. Evidentment, en un treball universitari cal tenir en compte que el principal producte és l'intangible de l'aprenentatge, però en un món materialista com el nostre caldrà que reflectim aquest intangible amb els següents productes:

1. Memòria del TFG: Document on es recollirà la redacció de tot el treball realitzat. En aquest cas anirà des de la preparació prèvia, o cerca d'informació, fins a la documentació al voltant del segon producte creat: el programari.

2. Nucli del servidor web: El producte final i objecte d'aquest TFG. Tot gira al voltant de com ens documentem, el creem i documentem aquest programari.

3. Presentació del TFG: Un document audiovisual que sintetitzarà la essència d'aquest treball per poder arribar a fer entenedor el procés d'un semestre en pocs minuts.

# Planificació

Aquest TFG es planteja com una assignatura més del Grau i per tant podem trobar PACs que fan la funció de seguiment del treball. En aquest sentit una primera divisió del treball la podem fer mitjançant l'ús de les fites temporals proposades pels docents.

*Taula 1: Temporització bàsica del treball*

| Activitat | Inici | Final |
|---|---|---|
| PAC 1 | 24/02/2016 | 13/03/2016 |
| PAC 2 | 17/04/2016 | 01/05/2016 |
| PAC 3 | 23/05/2016 | 05/06/2016 |
| Memòria | 19/06/2016 | 19/06/2016 |
| Presentació | 26/06/2016 | 26/06/2016 |

Mitjançant aquesta temporització podem obtenir una aproximació de la planificació:

PAC1: Temporització (19 dies)

Assolir un calendari per dur a terme el TFG. En aquest primer període caldrà definir com i quan es duran a terme les activitats.

El primer període definirem què volem fer. Quin serà el tema del TFG i com l'abordarem
El segon període caldrà fer una formalització expressa del allò que farem per aconseguir els objectius plantejats
En un tercer període caldrà fer una petita recerca per saber les tecnologies i materials que ens caldran.
Seguirem per el disseny de la temporització plasmant un calendari per dur a terme el TFG
Per últim farem la redacció del plantejament inicial i de la temporització

Fites:

- Temporització en Diagrama de Gantt

- Entrega del document PAC 1

Període entre PAC 1 i PAC 2 ( 34 dies)

En aquest període entre PACs, aprofitarem per fer recerca per ampliar la documentació abans de posar-nos de ple amb la programació del servidor web.

Caldrà que en un primer moment cerquem informació sobre els protocols HTTP, com funcionen els servidors web i com es relaciona profundament els protocols amb els servidors. Això donarà lloc a un enteniment profund del que fa un servidor web i de com funciona el protocol HTTP per poder exportar-lo a la programació del nucli del servidor web.

Seguidament caldrà fer una visió dels productes més habituals en el mon dels servidors webs. En aquests apartat, i donada la pretensió que sigui el més portable possible, caldrà veure si hi ha productes similar, si n'hi ha veure com funcionen i si no n'hi ha entendre perquè no.
Amb tot el recollit en els passos anteriors procedirem a la redacció d'un document on es detallin els aspectes estudiats.

En aquest punt caldrà començar a plantejar el nostre producte, per això caldrà determinar els serveis que podrà dur a terme el producte, fer un a pressa de requisits i fer un anàlisi dels requisits per que pugui complir-los.

Fites:

- Document amb la introducció teòrica del TFG.
- Document amb la determinació del producte.

PAC2: Seguiment (15 dies)

En aquest període ja posarem en marxa la programació del projecte. Farem la implementació inicial del servidor i acabarem el període amb una "Beta" del producte.

Fites:

- Entrega del document PAC 2 que compilarà el documents de la fita anterior
- Entrega de la primera implementació del servidor.

Període entre PAC 2 i PAC 3 ( 22 dies)

En aquest període si tot es correcte passarem a donar els últims retocs al programari, si per contra trobem incorreccions disposarem de temps per no haver de canviar la planificació i poder-les afrontar. Durant aquest període caldrà redactar la documentació del programari.

Fites:

- Implementació final del servidor
- Document que contingui la informació relacionada amb el programari i el seu funcionament.

PAC3: Seguiment (13 dies)

Com que un dels productes d'aquest TFG és un programa informàtic caldrà fer una sèrie de proves amb el programari i documentar-les, durant aquest període ens dedicarem a provar el servidor i corregir les falles que puguem trobar.

Fites:

- Entrega del servidor funcional
- Redacció de la documentació de les proves
- Entrega de la compilació de documentació de la fita anterior i d'aquesta com a PAC 3.

Període entre PAC 3 i Memòria ( 13 dies)

En aquest període aprofitarem per donar els últims retocs a la documentació del projecte, afegir la documentació de la instal·lació i de la configuració i per últim a donar per tancada la memòria per la seva entrega.

Fites:

- Enllestir una memòria de TFG per obtenir el millor resultat possible.

Memòria (1 dia)

En aquest apartat tot i que ens limitem a l'entrega de la memòria, sempre quedarà lloc per alguna repassada nerviosa d'última hora.

Fites:

- Entregar la memòria del TFG

Període entre Memòria i Presentació (6 dies)

En aquest període ens dedicarem a preparar la presentació amb la major cura possible per poder transmetre tota la feina que hem realitzat durant el semestre.

Fita:

- Preparar una presentació adient.

Presentació (1 dia)

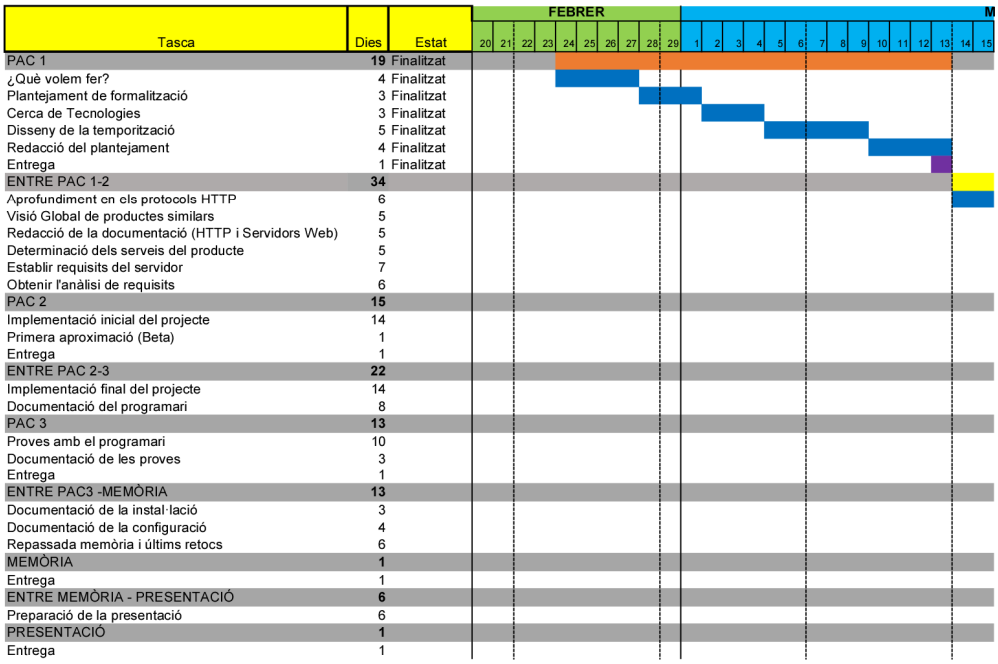
Ja no queda res més a fer que dur a terme la presentació que hem estat preparant durant la setmana anterior sobre un treball en el que hem posat tota la carn a la graella durant el semestre.

Fita:

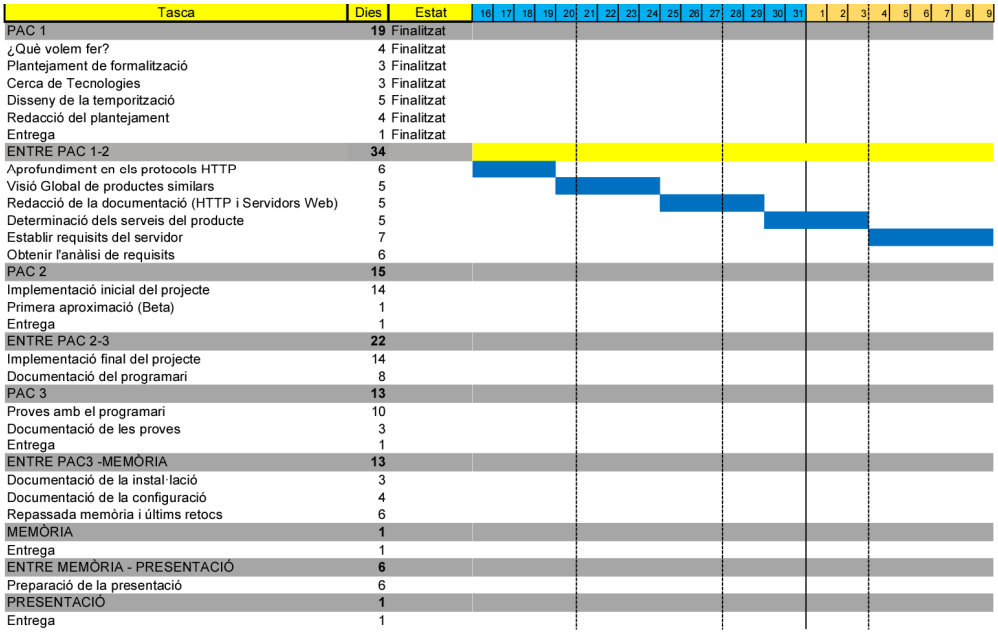
- Fer una presentació immillorable.

# Temporització. Diagrama de Gantt

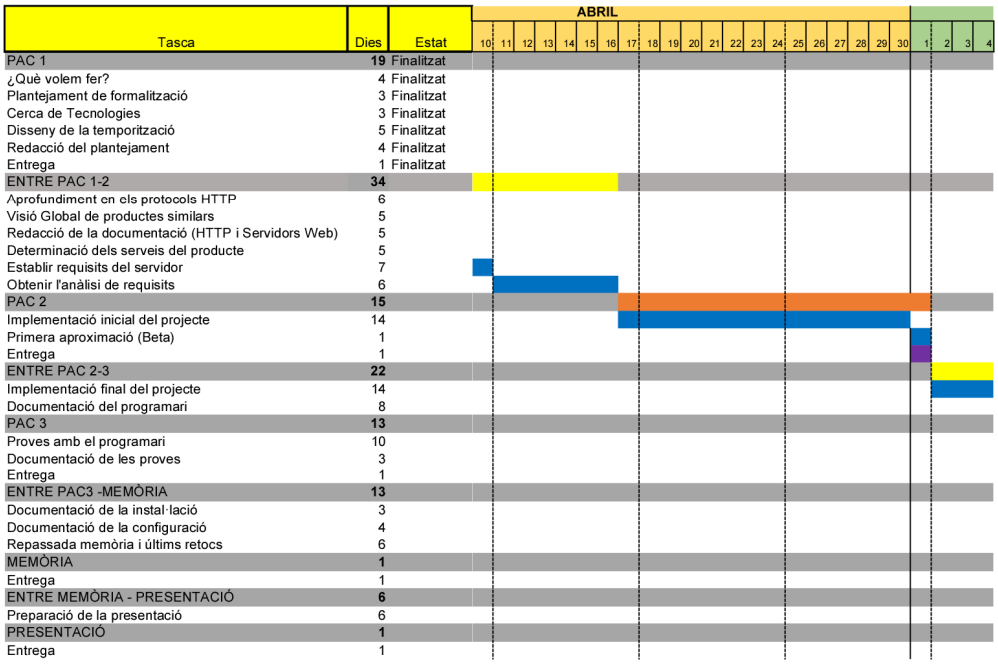
En aquest apartat adjunto un document Excel amb el diagrama complert i un enllaç (https://www.tomsplanner.es/public/ddelgadotfg ) on es pot trobar el mateix diagrama perquè sóc conscient de la dificultat per llegir les imatges en que s'ha de dividir el diagrama



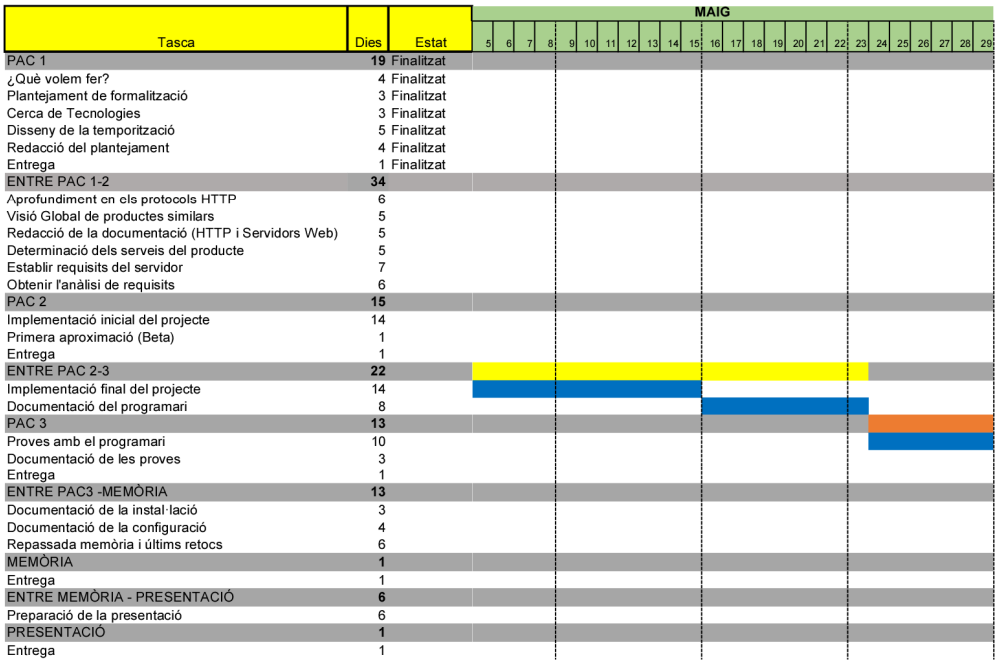

| Tasca | Dies | Estat |
|---|---|---|
| PAC 1 | 19 | Finalitzat |
| ¿Què volem fer? | 4 | Finalitzat |
| Plantejament de formalització | 3 | Finalitzat |
| Cerca de Tecnologies | 3 | Finalitzat |
| Disseny de la temporització | 5 | Finalitzat |
| Redacció del plantejament | 4 | Finalitzat |
| Entrega | 1 | Finalitzat |
| ENTRE PAC 1-2 | 34 | |
| Aprofundiment en els protocols HTTP | 6 | |
| Visió Global de productes similars | 5 | |
| Redacció de la documentació (HTTP i Servidors Web) | 5 | |
| Determinació dels serveis del producte | 5 | |
| Establir requisits del servidor | 7 | |
| Obtenir l'anàlisi de requisits | 6 | |
| PAC 2 | 15 | |
| Implementació inicial del projecte | 14 | |
| Primera aproximació (Beta) | 1 | |
| Entrega | 1 | |
| ENTRE PAC 2-3 | 22 | |
| Implementació final del projecte | 14 | |
| Documentació del programari | 8 | |
| PAC 3 | 13 | |
| Proves amb el programari | 10 | |
| Documentació de les proves | 3 | |
| Entrega | 1 | |
| ENTRE PAC3 -MEMÒRIA | 13 | |
| Documentació de la instal·lació | 3 | |
| Documentació de la configuració | 4 | |
| Repassada memòria i últims retocs | 6 | |
| MEMÒRIA | 1 | |
| Entrega | 1 | |
| ENTRE MEMÒRIA - PRESENTACIÓ | 6 | |
| Preparació de la presentació | 6 | |
| PRESENTACIÓ | 1 | |
| Entrega | 1 | |



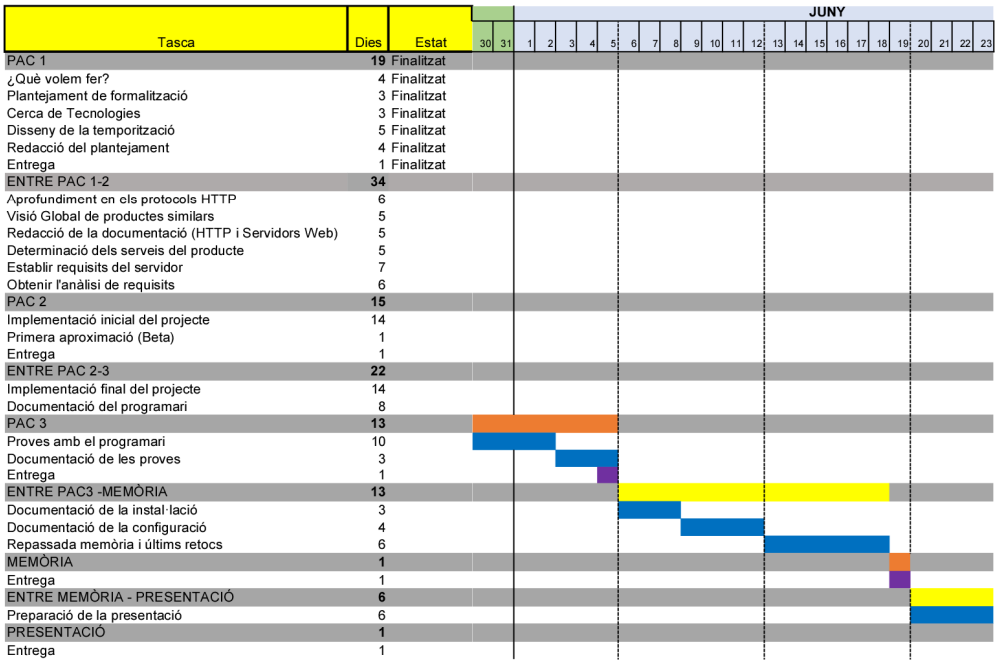

| Tasca | Dies | Estat |
|---|---|---|
| PAC 1 | 19 | Finalitzat |
| ¿Què volem fer? | 4 | Finalitzat |
| Plantejament de formalització | 3 | Finalitzat |
| Cerca de Tecnologies | 3 | Finalitzat |
| Disseny de la temporització | 5 | Finalitzat |
| Redacció del plantejament | 4 | Finalitzat |
| Entrega | 1 | Finalitzat |
| ENTRE PAC 1-2 | 34 | |
| Aprofundiment en els protocols HTTP | 6 | |
| Visió Global de productes similars | 5 | |
| Redacció de la documentació (HTTP i Servidors Web) | 5 | |
| Determinació dels serveis del producte | 5 | |
| Establir requisits del servidor | 7 | |
| Obtenir l'anàlisi de requisits | 6 | |
| PAC 2 | 15 | |
| Implementació inicial del projecte | 14 | |
| Primera aproximació (Beta) | 1 | |
| Entrega | 1 | |
| ENTRE PAC 2-3 | 22 | |
| Implementació final del projecte | 14 | |
| Documentació del programari | 8 | |
| PAC 3 | 13 | |
| Proves amb el programari | 10 | |
| Documentació de les proves | 3 | |
| Entrega | 1 | |
| ENTRE PAC3 -MEMÒRIA | 13 | |
| Documentació de la instal·lació | 3 | |
| Documentació de la configuració | 4 | |
| Repassada memòria i últims retocs | 6 | |
| MEMÒRIA | 1 | |
| Entrega | 1 | |
| ENTRE MEMÒRIA - PRESENTACIÓ | 6 | |
| Preparació de la presentació | 6 | |
| PRESENTACIÓ | 1 | |
| Entrega | 1 | |

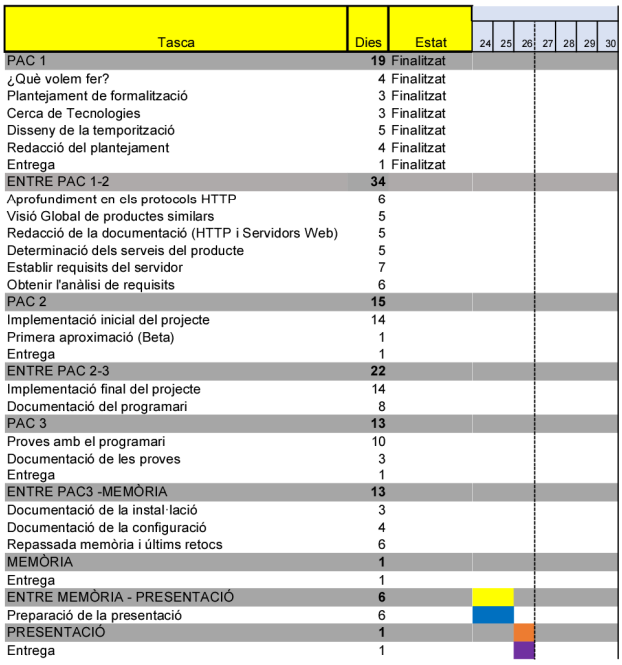



| Tasca | Dies | Estat |
|---|---|---|
| PAC 1 | **19** | Finalitzat |
| ¿Què volem fer? | 4 | Finalitzat |
| Plantejament de formalització | 3 | Finalitzat |
| Cerca de Tecnologies | 3 | Finalitzat |
| Disseny de la temporització | 5 | Finalitzat |
| Redacció del plantejament | 4 | Finalitzat |
| Entrega | 1 | Finalitzat |
| ENTRE PAC 1-2 | **34** | |
| Aprofundiment en els protocols HTTP | 6 | |
| Visió Global de productes similars | 5 | |
| Redacció de la documentació (HTTP i Servidors Web) | 5 | |
| Determinació dels serveis del producte | 5 | |
| Establir requisits del servidor | 7 | |
| Obtenir l'anàlisi de requisits | 6 | |
| PAC 2 | **15** | |
| Implementació inicial del projecte | 14 | |
| Primera aproximació (Beta) | 1 | |
| Entrega | 1 | |
| ENTRE PAC 2-3 | **22** | |
| Implementació final del projecte | 14 | |
| Documentació del programari | 8 | |
| PAC 3 | **13** | |
| Proves amb el programari | 10 | |
| Documentació de les proves | 3 | |
| Entrega | 1 | |
| ENTRE PAC3 -MEMÒRIA | **13** | |
| Documentació de la instal·lació | 3 | |
| Documentació de la configuració | 4 | |
| Repassada memòria i últims retocs | 6 | |
| MEMÒRIA | **1** | |
| Entrega | 1 | |
| ENTRE MEMÒRIA - PRESENTACIÓ | **6** | |
| Preparació de la presentació | 6 | |
| PRESENTACIÓ | **1** | |
| Entrega | 1 | |




| Tasca | Dies | Estat |
|---|---|---|
| PAC 1 | **19** | Finalitzat |
| ¿Què volem fer? | 4 | Finalitzat |
| Plantejament de formalització | 3 | Finalitzat |
| Cerca de Tecnologies | 3 | Finalitzat |
| Disseny de la temporització | 5 | Finalitzat |
| Redacció del plantejament | 4 | Finalitzat |
| Entrega | 1 | Finalitzat |
| ENTRE PAC 1-2 | **34** | |
| Aprofundiment en els protocols HTTP | 6 | |
| Visió Global de productes similars | 5 | |
| Redacció de la documentació (HTTP i Servidors Web) | 5 | |
| Determinació dels serveis del producte | 5 | |
| Establir requisits del servidor | 7 | |
| Obtenir l'anàlisi de requisits | 6 | |
| PAC 2 | **15** | |
| Implementació inicial del projecte | 14 | |
| Primera aproximació (Beta) | 1 | |
| Entrega | 1 | |
| ENTRE PAC 2-3 | **22** | |
| Implementació final del projecte | 14 | |
| Documentació del programari | 8 | |
| PAC 3 | **13** | |
| Proves amb el programari | 10 | |
| Documentació de les proves | 3 | |
| Entrega | 1 | |
| ENTRE PAC3 -MEMÒRIA | **13** | |
| Documentació de la instal·lació | 3 | |
| Documentació de la configuració | 4 | |
| Repassada memòria i últims retocs | 6 | |
| MEMÒRIA | **1** | |
| Entrega | 1 | |
| ENTRE MEMÒRIA - PRESENTACIÓ | **6** | |
| Preparació de la presentació | 6 | |
| PRESENTACIÓ | **1** | |
| Entrega | 1 | |


JUNY
Tasca
Dies
Estat
30 31 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
PAC 1 19 Finalitzat
¿Què volem fer? 4 Finalitzat
Plantejament de formalització 3 Finalitzat
Cerca de Tecnologies 3 Finalitzat
Disseny de la temporització 5 Finalitzat
Redacció del plantejament 4 Finalitzat
Entrega 1 Finalitzat
ENTRE PAC 1-2 34
Aprofundiment en els protocols HTTP 6
Visió Global de productes similars 5
Redacció de la documentació (HTTP i Servidors Web) 5
Determinació dels serveis del producte 5
Establir requisits del servidor 7
Obtenir l'anàlisi de requisits 6
PAC 2 15
Implementació inicial del projecte 14
Primera aproximació (Beta) 1
Entrega 1
ENTRE PAC 2-3 22
Implementació final del projecte 14
Documentació del programari 8
PAC 3 13
Proves amb el programari 10
Documentació de les proves 3
Entrega 1
ENTRE PAC3 -MEMÒRIA 13
Documentació de la instal·lació 3
Documentació de la configuració 4
Repassada memòria i últims retocs 6
MEMÒRIA 1
Entrega 1
ENTRE MEMÒRIA - PRESENTACIÓ 6
Preparació de la presentació 6
PRESENTACIÓ 1
Entrega 1


Tasca
Dies
Estat
24 25 26 27 28 29 30
PAC 1 19 Finalitzat
¿Què volem fer? 4 Finalitzat
Plantejament de formalització 3 Finalitzat
Cerca de Tecnologies 3 Finalitzat
Disseny de la temporització 5 Finalitzat
Redacció del plantejament 4 Finalitzat
Entrega 1 Finalitzat
ENTRE PAC 1-2 34
Aprofundiment en els protocols HTTP 6
Visió Global de productes similars 5
Redacció de la documentació (HTTP i Servidors Web) 5
Determinació dels serveis del producte 5
Establir requisits del servidor 7
Obtenir l'anàlisi de requisits 6
PAC 2 15
Implementació inicial del projecte 14
Primera aproximació (Beta) 1
Entrega 1
ENTRE PAC 2-3 22
Implementació final del projecte 14
Documentació del programari 8
PAC 3 13
Proves amb el programari 10
Documentació de les proves 3
Entrega 1
ENTRE PAC3 -MEMÒRIA 13
Documentació de la instal·lació 3
Documentació de la configuració 4
Repassada memòria i últims retocs 6
MEMÒRIA 1
Entrega 1
ENTRE MEMÒRIA - PRESENTACIÓ 6
Preparació de la presentació 6
PRESENTACIÓ 1
Entrega 1

PART 1

# UNA APROXIMACIÓ A INTERNET

## Què és internet ?

Aquesta pregunta és senzilla però es bàsica per poder començar a posar la vista sobre com funcionen els servidors web i per a que serveixen.

Internet és un conjunt de xarxes intercomunicades que fan servir protocols TCP/IP per poder formar una única xarxa lògica tot i la heterogeneïtat de les diferents xarxes físiques. Això que pot semblar una mica complicat d'entendre es pot resumir amb allò tant manit que diu Internet és la xarxa de xarxes o bé com la anomenaven al seu començament una "l'embrió d'una xarxa galàctica".

Des d'un bon començament, va ser concebuda com una manera d'unir ordinadors per aprofitar millor els recursos, de tal forma que ARPAnet no era més que la unió de diferents ordinadors per tal d'aprofitar recursos comuns i poder intercomunicar-se. En aquest punt és quan apareix un dels elements importants en aquest treball, un sistema d'hipertext encara molt primitiu.

En avançar el desplegament de ARPAnet, aquesta va anar prenent una dimensió de gran xarxa des dels 4 primers ordinadors que la composaven fins a 213 a l'any 1981 o 500 al any 1983.

Avui dia semblem molt pocs ordinadors donat que a internet es connecten milions i si mirem cap al futur potser troben connectats fins i tot els aparells més inesperats.

En aquest punt, i donat que per veure el creixement que ha tingut aquesta xarxa estem parlant de l'embrió de internet i de com evoluciona fins avui i cap al futur farem una petita cronologia d'internet per tenir clars on som i com hem arribat.

## Internet: tot va passar així

Internet avui dia és una connexió a nivell mundial però en algun punt va haver de començar-se a expandir. En una cronologia molt sintètica veurem aquesta evolució que porta a una xarxa de 4 ordinadors fins arribar a ser una xarxa global de milions d'ordinadors.

1. Els pioners: Als anys 60 la comunicació entre computadores va començar a ser una realitat. Tot parteix de l'any 61 quan es publicà el primer paper parlant de intercanvi de paquets. En els anys següents apareix el concepte de xarxa còsmica als Estats Units, tot i que com concepte ja n'hi ha referencies abans. Al any 1964 s'escriu sobre la comunicació en xarxes i és a partir d'aquest punt que la teoria es posa en pràctica.

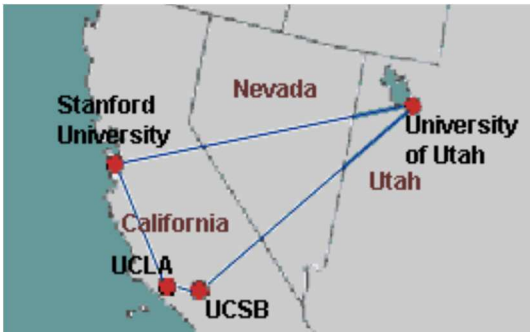



*Fig. 1: Primera xarxa del que seria internet*

2. Els experiments: A finals dels 60, la teoria ja estava prou madura per donar lloc a un element tangible, per això l'any 1965 es connecten 2 ordinadors en línia amb cable dedicat com a primer experiment. En aquests anys es duu a terme el primer esborrany del que serà ARPAnet, i després d'un seguit de planificacions i especificacions, l'any 69

s'implementa la primera ARPAnet. En aquesta xarxa hi havia 4 nodes situats a 4 grans universitats americanes, que constituirien el tronc central d'internet fins als anys 90.

3. El creixement de la xarxa: Durant els anys 70 hi ha un creixement de la xarxa dins dels EUA i cap al final de la dècada es produeix la primera connexió internacional amb la connexió via satelitat d'un node a la Gran Bretanya. En aquest període el protocol que regeix la xarxa passa de TCP a TCP/IP.

4. L'explosió i els serveis: en aquesta etapa apareixen subxarxes per tot el món i apareixen els primers serveis moderns com els mails amb registre MX o IRC, ens trobem als anys 90.

5. La internet moderna: A partir dels 90 desapareix ARPAnet, encara que alguns nodes queden integrats a internet. Apareix la primera versió comercial de connexió a la xarxa. El CERN publica la primera versió de www amb el servidor nxoc01.cern.ch. Al 92 ja hi ha 1.000.000 de computadors connectats. Al 94 www és el segon servei més usat a la xarxa per darrera de FTP. Al 95 esdevé el primer servei. Des d'aquest punt fins avui, la evolució de la xarxa ha anat lligat als serveis que s'ofereixen sobre aquest protocol i han estat els dominants en la xarxa. La progressió del número de dominis, la diversificació i mundialització dels mateixos, els serveis online, etc.

Arribats a aquest punt es trobem amb la xarxa que estudiarem, la xarxa dels servidors web.

## Internet i la web: jo pensava que eren el mateix.

Tot i que la diferencia entre internet i web pot ser massa subtil per la majoria de la gent, tenint en compte que la universalització de l'accés a internet és un fet, és interessant posar en valor la diferencia.

Com hem pogut veure en l'apartat, tot i l'evident biaix que he fet de la historia d'internet cap als servidors web, internet existeix com a embrió des dels anys 60. Això és perquè com hem dit abans és una xarxa, la xarxa de xarxes, però "només" interconnexions entre ordinadors.

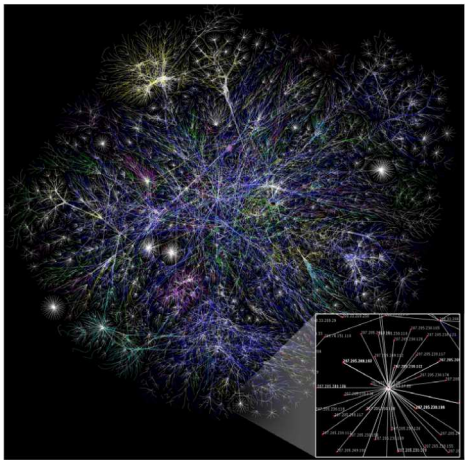


*Fig. 2: Internet avui dia, com es mostra en la imatge milers de connexions entre ordinadors*

El web és una manera de accedir a la informació. El web fa servir protocol HTTP (HiperText Transfer Protocol) per dur a terme les comunicacions. Es a adir, el web és un servei dins d'internet que no apareix fins als anys noranta tot i que ara és el dominant, cosa que no assegura que ho sigui per sempre, com dóna fet que va desbancar FTP com a servei dominant. Avui dia la web representa un tant per cent molt important del tràfic d'internet però cal recordar que no és l'únic servei, tot i que pel que nosaltres concerneix és l'únic que estudiarem d'aquí en endavant.

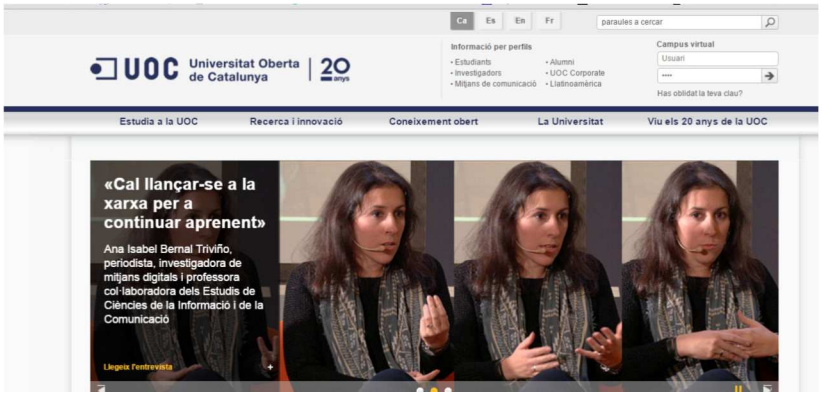



*Fig. 3: Una web, una manera de la que podem fer ús d'Internet*

# La intranet i la connexió a Internet

Un cop coneixem internet, la web i una mica sobre el seu funcionament, cal que parlem d'una de les xarxes de la xarxa de xarxes: una intranet.

Si més no, tots els servidors web formaran part d'una intranet formada pel seu router i el servidor, de fet això seria una intranet molt bàsica però ja veiem que el router és el punt d'unió entre internet i les intranets.

Així doncs una intranet, és una xarxa que no té una sortida a Internet directa on un grup d'ordinadors connectats mitjançant protocols TCP/IP comparteixen informació. A més aquesta xarxa pot tenir una connexió a internet a partir d'una porta d'enllaç que serà el router.
Dins d'aquesta xarxa haurem de donar al servidor un accés permanent als ports convenients per poder accedir al contingut des de la xarxa intranet i haurem de configurar el router per tal que aquest ports siguin accessibles des de connexions remotes per tal de tenir un servidor web funcionant a internet.

Una altra modalitat per configurar un servidor web seria donar un aspecte a la web que es pot veure des d'internet, potser més corporatiu i menys relaxat, i un altre per la intranet on podrem tenir un aspecte més relaxat per facilitar la comunicació amb les persones que formen part de la intranet. De totes maneres això ja serien configuracions de la web, que si bé formen part necessària d'aquest treball perquè haurem de donar algun contingut per demostrar el funcionament del servidor, podrien ser objecte d'un altre estudi detallat.

PART 2

# HTTP: Protocols als Servidors

## Comunicació amb un servidor web

Per dur a terme la comunicació entre un servidor web i un client, cal que tots dos facin servir un llenguatge comú que en aquest cas serà HTTP. En aquest apartat farem un ullada a com es produeix aquest llenguatge, quins components bàsics caldrà tenir en compte i com és transporten els missatges HTTP per la xarxa.

## HTTP: intercanvi de missatges

La transmissió de la informació entre servidors webs i clients és duu a terme mitjançant HTTP, les característiques del qual permeten que la informació es transmeti de forma confiable i sense perill de ser alterada en el transit.

Una transmissió bàsica entre un client i el servidor web es podria explicar de manera simplificada com l'enviament d'una petició per part del client i l'entrega del contingut indicant el seu format la longitud de la cadena i altra informació.

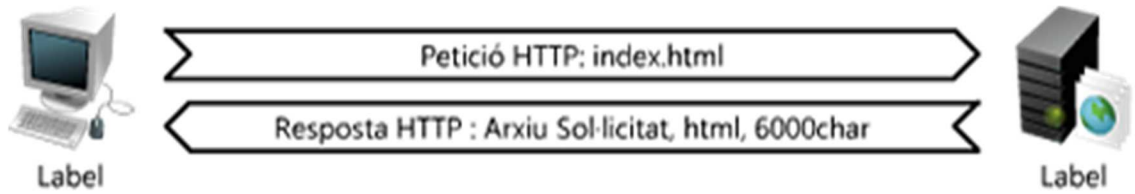



*Fig. 4: Simplificació de la comunicació Servidor-Client*

Per dur a terme la anterior comunicació, el servidor cal que gestioni els recursos web. Un recurs web és qualsevol tipus de contingut web, poden ser elements estàtics (imatges, text, html, documents, etc.) o dinàmics, és a dir, contingut generat basat en la petició feta pel client web. Per poder determinar el tipus del recurs, el servidor fa arribar una etiqueta anomenada MIME (Multipurpose Internet Mail Extensions) on s'indica el tipus de recurs que s'està enviant.

## HTTP: Objectiu localitzat

Quan es duu a terme una crida a un element del servidor, aquest el trobarà si se li demana seguint un protocol determinat. Hi ha diferents maneres d'invocar a un recurs/os dins del servidor:

1. URIs (Uniform Resource Identifier): un apuntador cap als recursos dins del servidor. Donada l'evolució de la web, avui dia casi qualsevol URI és un URL.

2. URLs (Uniform Resource Locator): un apuntador cap a un recurs de una forma precisa que indica
   - el protocol (http://)
   - La direcció web del servidor (www.uoc.edu)
   - Per últim el recurs a entregar

(/portal/system/modules/edu.uoc.portal.presentations/resources/images/uoc-logo-fase3.png)

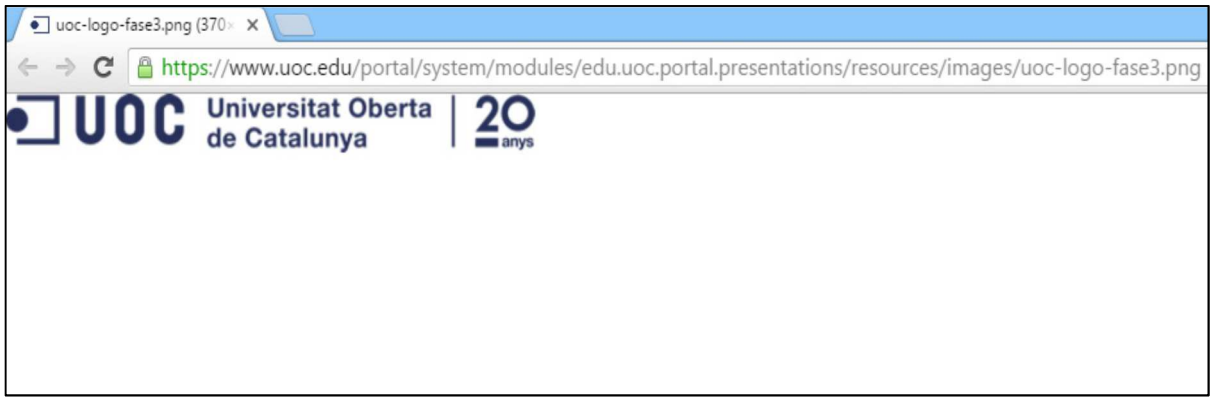



Fig. 5: Exemple d'accés a un recurs web

3. URNs (Uniform Resource Name): Aquest localitzador, encara en fase d'adopció, permet accedir als recursos independentment d'on estiguin allotjats. Es un localitzador independent de la localització que necessita una infraestructura per resoldre la localització, fet que fa que no sigui àmpliament adoptat.

## Intercanviant informació amb HTTP

La comunicació HTTP com hem vist es basa en intercanvi de missatges. Aquest missatges han de complir unes normes pròpies del protocol HTTP. Com hem dit abans el client demana el contingut i el client entrega un missatge amb el contingut i més dades. En aquest punt veurem com ho fan i quines són les dades.

Cal tenir en compte que aquestes peticions són resoltes en una sola comunicació que tan aviat s'ha acabat la comunicació es tanca, amb l'excepció de les connexions persistents.

Les diferents comandes que estan suportades en HTTP són anomenades mètodes (HTTP methods) i indiquen al servidor quina és l'acció que ha de realitzar. Entre aquest mètodes trobem:

*Taula 2: Mètodes en un servidor web*

| MÈTODE | DESCRIPCIÓ |
|---|---|
| GET | Envia un recurs des del servidor al client |
| PUT | El servidor rep informació del client per emmagatzemar-la |
| DELETE | Esborra el recurs del servidor |
| POST | El servidor rep informació del client per processar-la |
| HEAD | El servidor envia les capçaleres del document sol·licitat. |
| TRACE | Indica el recorregut pels proxies fins arribar al servidor |
| OPTIONS | Determina els mètodes que funcionen al servidor |

Una vegada sabem els tipus de mètodes que podem fer servir amb un servidor web, cal que ens centrem en com respondrà el servidor.

## Codis de Estat HTTP: La situació està així

Una vegada el servidor rep un missatge, la seva resposta anirà acompanyada de un codi d'estat. Aquest codis d'estat són la manera que té el servidor per comunicar com s'ha resolt una petició feta per un client.

El codi d'estat es componen d'una part numèrica i una part textual i els trobarem a la part inicial de cada missatge. La part numèrica facilita el processament per part de les màquines i la part textual facilita l'enteniment per part dels humans.

Els codis d'estat es classifiquen en tipus segons els seus tres dígits, indicant cadascun dels tipus un estat general segons el primer dígit i donant més informació els dos dígits últims. En la següent taula podem veure els codis, el tipus al que pertanyen i el rang d'estats definits:

*Taula 3: Codis de Estat*

| Rang | Rang Definit | Tipus d'estat |
|---|---|---|
| 100-109 | 100-101 | Informació |
| 200-299 | 200-206 | Correcte |
| 300-399 | 300-305 | Redirecció |
| 400-499 | 400-415 | Error a la part del client |
| 500-599 | 500-505 | Error a la part del servidor |

Entre aquests codis, hi ha 3 que són els més habituals:

- 200 OK : que evidentment vol dir que tot ha anat correctament
- 401 Unauthorized: que vol dir que cal usuari i contrasenya per accedir al recurs.
- 404 Not Found: que vol dir que no es pot trobar el recurs sol·licitat al URL.

## Visió global d'un missatge HTTP

Un cop hem vist quines parts ha de tenir un missatge HTTP i sabent que hi ha dos tipus, la petició i la resposta, passem a veure les parts que tenen aquest missatges:

- Línia d'inici: La primera línia del missatge que indica que s'ha de fer per a respondre o que ha succeït amb la resposta
- Capçaleres: Poden ser múltiples camps o no existir. Cada capçalera consisteix en un nom i un valor separats per dos punts. Acaba amb una línia en blanc.
- Cos: és un camp opcional que contindrà qualsevol tipus de dada que sigui part de la resposta o petició i que s'hagi de enviar.

Un cop coneixem les parts veiem com queden formatades:

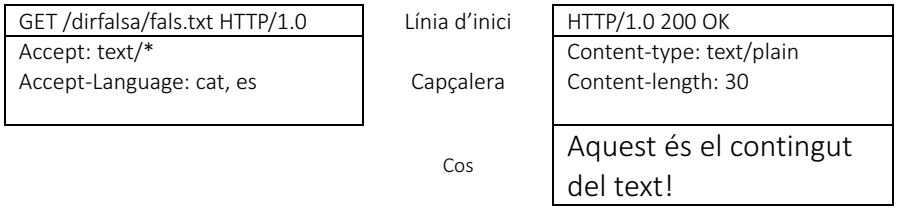



*Fig. 6: Petició i resposta HTML*

## TCP/IP: Tenim el missatge, ¿qui el mou?

HTTP és un protocol de capa de aplicació, per tant no té cap paper en la comunicació més enllà d'encapsular la informació. En aquest punt entren la dupla TCP/IP, el protocol de transport confiable més popular a internet.

Com que Internet en si mateixa es basada en TCP/IP sembla evident que un servei dins d'Internet sigui transportat mitjançant TCP/IP si volem que el missatge arribi de manera confiable.

Una vegada la connexió TCP es establerta, el missatges es poden intercanviar i podem confiar que arribaran, ho faran en ordre i sense danys.

En el anterior paràgraf diem que TCP estableix la connexió, ¿però IP per què no el nombrem? Doncs perquè TCP és un protocol de la capa de transport que s'encapsula dins de un protocol de capa de xarxa que és IP. Per tant, ens trobem que HTTP treballa sobre TCP i aquest al torn treballa sobre IP, més enllà troben l'enllaç de dades i el mitjà físic que poden variar, però des de la capa d'aplicació fins a la de xarxa un servidor web farà servir aquests protocols.

PART 3

# Els Servidors Web

## Finalment, Què és un servidor web?

Durant els apartats anteriors hem vist un munt de característiques i elements que formen part d'un servidor web, però encara no hem respost a la pregunta de què és un servidor web. Un servidor web és un programa que processa les peticions del client i proporciona les respostes del servidor.

Llavors, els servidors web com hem comentat implementen HTTP i la connexió TCP necessària per enviar el missatge, és a dir creen i mouen els missatges. Segurament estaran connectats a internet, i molt probablement tinguin una direcció fàcil de recordar pels humans més enllà de la IP, que haurà de traduir un DNS (Servidor de noms), però no és totalment necessari com hem dit en apartats anteriors, donat que tot servidor de Internet també ho és per una intranet on no caldrà un DNS si accedim directament.

Avui dia, els servidors poden ser de una diversitat molt gran, des d'un servidor que només contesti línies de codi fins a servidors molt segurs per processar transaccions econòmiques.

Entre els diferents tipus de servidors web podem trobar des dels que considerem de per instal·lar en ordinadors estàndards fins als petits que son dins del productes de consum com poden ser impressores. A més trobem servidors pre-fabricats que tenen un software i un hardware dedicat i preconfigurat.

El nostre objectiu en aquest TFG és fer un servidor senzill amb una mínima configuració però que es pugui utilitzar en qualsevol ordinador, llavors podem dir que el nostre servidor web quedarà englobat entre els petits servidors multi propòsit. No serà competència per els més utilitzats al món però sempre és una bona inspiració veure qui ho fa millor quan entres en un àmbit, per això al següent apartat veurem els més utilitzats.

## Com funciona un servidor web

Tot i que els servidors que hem anomenat són molt complexos i proveeixen al usuari de moltes funcions, bàsicament actuen com a un servidor més senzill com el que volem desenvolupar. Els servidors fan un seguit de tasques per dur a terme la seva funció:

1. Establir connexió: acceptar la connexió so el client s'ha connectat correctament o tancar-la si no.
2. Rebre la petició: llegir la petició HTTP des de la xarxa.
3. Processar la petició: Es fa la interpretació de la petició i es duu a terme la acció.
4. Accedir als recursos: Es fa l'accés al recurs indicat a la petició
5. Construir la resposta: Crear una resposta HTTP amb les capçaleres adients.
6. Enviar la resposta: enviar la resposta de tornada al client.
7. Fer l'apunt al registre: Apuntar al registre les dades sobre la transacció completada

# Els més utilitzats

A continuació farem un recorregut pels servidors més utilitzats a la web. Per començar aquest repàs, farem servir el referent en ús de servidors web, la estadística de NetCraft per a març de 2016 (http://news.netcraft.com/archives/2016/03/18/march-2016-web-server-survey.html). [Possiblement serà actualitzat en el document final a la de maig que serà la última que sortirà abans de la presentació del TFG].

En la següent gràfica podem veure la distribució de servidors que fan servir un aplicatiu en front del total:

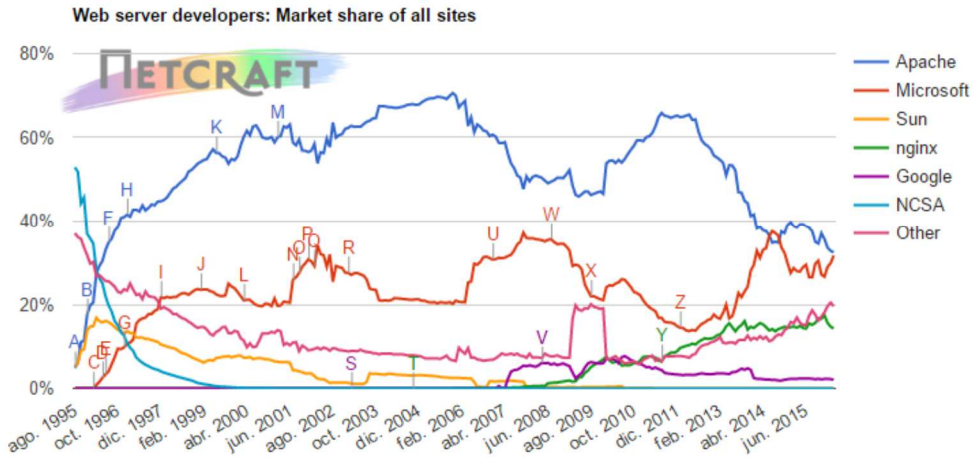



*Fig. 7: Gràfica d'ús de servidors*


*Fig. 8: Logo Apache*

El més utilitzat és Apache un projecte col·laboratiu que neix al 1995 a partir de HTTP demon de Rob McCool. A partir d'aquesta base es va desenvolupar durant uns anys a base de les aportacions que feien un grup de webmasters que anaven afinant el codi per evitar els errors amb que es trobaven. A finals de 1995 van treure Apache 1.0.

Apache és un servidor molt estable, amb molt poques falles i les que s'han trobat s'han anat solucionant per part de la seva activa comunitat de desenvolupadors.

Entre els avantatges de fer servir Apache trobem:

- Codi lliure amb una comunitat molt gran, cosa que afavoreix la retroalimentació i la millora del problemes que puguin sorgir.
- Instal·lació i configuració molt documentada degut a la gran base d'usuaris.
- Gratuïtat del software.
- Multiplataforma
- Suport per SSL i TLS, és a dir, protocols de seguretat de alt nivell.
- Bon rendiment, cosa que permet gestionar fins a un milió de visites al dia.

Però també hi ha inconvenients, encara que pocs, però destaca la mancança d'un panell de configuració.


*Fig. 9: Logo IIS*

El següent servidor web en ordre d'ús és Internet Information Systems de Microsoft. Aquest servidor basa les seves avantatges en la integració total amb la resta de productes Microsoft. Permet una integració total amb els serveis de seguretat de Windows, permet l'ús de Microsoft SQL de forma senzilla, etc.

Una de les seves virtuts es la consola de configuració gràfica que permet configurar el servidor en un entorn amigable. Una altra virtut és la seva fàcil instal·lació.

Entre els punt febles trobem que no és multiplataforma i s'ha d'utilitzar sobre Windows.


*Fig. 10: Logo NGINX*

L'últim dels software per a servidors webs que comentarem és NGINX. Aquest tot i ser 3er en el total de llocs, és el segon més utilitzats entre els llocs actius.

Aquest servidor va aparèixer al 2004 i el seu punt fort és la robustesa. Permet fins a 10.000 connexions simultànies i té una bona tolerància a falles. És de codi obert i multiplataforma.

## El nostre referent

Tal i com hem comentat en anteriors apartats i en aquest mateix, el nostre servidor ha de ser portable i petit. Amb aquestes característiques i per la seva facilitat d'ús, tot i que la meva experiència pot ser curta, he trobat un referent que sembla l'adient com a mirall: Mongoose.

*Fig. 11: Logotip Mongoose*

Mongoose és un servidor web complet en un executable creat i gestionat per la companyia CESANTA. Es un software de pagament però de llicència OpenGL, que permet l'ús de la seva versió més senzilla de franc.

Es pot configurar des del mateix navegador donant lloc a un arxiu de configuració que s'autogenera i així queden guardades les preferències de l'usuari en quant a connexió, carpeta de localització dels documents a compartir en el servidor, etc.

La seva facilitat d'ús és, comparada amb els referents, quant menys xocant. Qui hagi configurat un servidor com Apache, sap que es requereix de trastejar amb arxius de configuració, comentar i descomentar línies, etc. Amb Mangoose això és innecessari, només cal col·locar l'executable en la carpeta que volem servir i executar-lo.

A més hi ha diferents versions IOT, Standard i Binary que son dissenyades per enfocar-les a diferents aparells. De fet la principal funció d'aquest servidor és ser un servidor de sistemes embeguts, però en el nostre cas i donat que té moltes funcions de les que volem implementar ens pot fer de referent per obtenir un servidor lleuger y de fàcil configuració.

PART 4

# El Nostre Servidor Web: Requisits

## Què ha d'oferir el nostre servidor

En un principi cal pensar que estem davant d'un nucli d'un servidor web pel que més que en moltes funcionalitats ens centrarem en que sigui estable, lleuger i de fàcil configuració.

Per fer això en una primera instància el nostre servidor atendrà comandes GET ja que aquestes són la gran majoria de comandes en un web. Amb aquest mètode el nostre servidor podrà gestionar jsons mitjançant javaScript i Ajax, obtenir pàgines web estàtiques i dinàmiques amb programació a la banda de servidor, etc. De moment amb la implementació del GET tindrem un nucli funcional amb un gran espectre de possibilitats. D'altra banda, quan entrem en altres mètodes normalment calen implementacions més grans com ara bases de dades que complicarien molt la creació del servidor i augmentarien l'abast fins fer-lo potser inassolible en el període del TFG.

Pel que fa a la interfície d'usuari, basant-me en un primer moment en Mongoose vaig pensar en fer-la web, però si volem un mètode ràpid i funcional podem fer una interfície gràfica.

Haurà de ser multi-fil donat que haurà de poder atendre més d'una petició, ja des d'un mateix client o des de diferents clients, però no caldrà que sigui multiprocés perquè les peticions web es poden gestionar dins d'un mateix procés servidor. De totes maneres en ser un servidor lleuger i si té una configuració gràfica es podria canviar el port de execució i córrer més d'un procés en diferents ports de manera autònoma.

Caldrà que sigui multiplataforma, això és fàcil d'aconseguir si el programen en Java. Haurem d'obtenir el jar, que es directament executable en Linux i Windows. A més, podem fer un "Warper" per a Windows de tal manera que obtenim un executable (.exe). Si fem un executable, seria adient dissenyar una icona que podem fer fàcilment amb Adobe Ilustratror.

Haurà de ser fàcil de debugar, per tant si creem una interfície gràfica podem afegir un panell a mode de consola on ens vagi indicant quins processos està duent a terme. Així tindrem al usuari sempre informat en front de possibles problemes que sorgeixin en l'ús en etapes primerenques de desenvolupament del producte.

## Doncs, establim els requisits

Amb l'anterior apartat hem fet una visió global del que volem obtenir que es pot resumir en:

- Servidor que gestioni el mètode GET
- Interfície gràfica que faciliti la configuració i debugatge.
- Servidor multi-fil
- Procés únic
- Lleuger
- Multiplataforma
- De instal·lació senzilla o sense.
- De configuració senzilla o sense.

# Analitzem els requisits: Posem fil a l'agulla

Primerament endrecem aquest requisits per poder anar obtenint un patró de treball sobre el que basar-nos:

1. Multiplataforma: Com ja hem dit això ho resoldrem fent servir Java, per tant la nostra eina de treball serà Java. Per tant serà programat en Java.
2. Procés únic: a Java podem fer un programa que empaqueti diferents classes que faran diferents funcions així que la solució per aquest requeriment és fer ús de Java
3. MultiFil: Com que a Java existeix el mètode Runnable que podem implementar en el moment adequat per obtenir dos fils d'execució això no serà un problema per obtenir-lo.
4. Lleuger: Per més classes que implementem, que no crec que siguin més de 7 o 8, no crec que tinguem problema en obtenir un executable lleuger.
5. De instal·lació senzilla: en un principi farem que no hi hagi instal·lació per això d'aquest requisit no ens hem de preocupar.
6. De configuració senzilla: en un principi només caldrà configurar els paràmetres de ruta des d'on volem llençar el servidor i en el port que volem que treballi, això ho podem fer des de la interfície gràfica sense problemes.
7. Amb interfície gràfica que faciliti el debugatge i la configuració: A Java trobem solucions com Swing y JFrame que permeten fer interfícies gràfiques multiplataforma. En un primer moment farem un servidor gràfic però potser podem ocultar el apartat gràfic un cop estigui funcionant.
8. Ha de gestionar les peticions GET: això és ben senzill un cop obert el socket de connexió i gestionant la petició HTTP, podem veure si la capçalera conté un GET i si no és així caldrà gestionar l'error HTTP 405: Mètode no permès.

En conclusió un cop analitzats els requisits, podem concloure que ha de ser programat en Java, amb la inclusió de un o varis mètodes Runnable, amb l'ús de Swing per la interfície gràfica i sense configuració mitjançant fitxers si no al moment de córrer.

PART 5

# El Nostre Servidor Web: Implementació

## Què ha de fer el nostre servidor i como ho fa

La funció bàsica que ha de fer el nostre servidor és entregar pàgines web. Per veure com ho haurà de fer farem servir un esquema de funcionament d'un servidor i anirem posant la implementació que hem fet per dur a terme el treball.

1. Obrir un socket i quedar a la espera de la connexió del client mitjançant la classe Server:

```
...
servSocket = new ServerSocket(port);

// Procesem les sol·licitus HTTP.
Msg.linia("Restem a la espera de que el client connecti");

while (true) {
        Socket socketConnexio = servSocket.accept();

...
```

2. Una vegada tenim el socket i arriba la sol·licitud caldrà gestionar-la. Primer acceptem la connexió mitjançant la classe Server:

```
...
SolicitudHttp solicitud = new SolicitudHttp( socketConnexio, path,
entrega);

// Creem un nou fil per processar-la.
//En aquest punt el nostre servidor es transforma en multi thread
realment
Thread filServer = new Thread(solicitud);

// Iniciem el fil.
filServer.start();
...
```

3. Seguidament la llegim la capçalera fent ús de la classe SollicitudHTTP:

```
...
// Referencia al stream de sortida del socket i creem un Buffer per la
trama d'entrada del socket
DataOutputStream respostaServidor = new
                DataOutputStream(socket.getOutputStream());;
BufferedReader peticioClient = new BufferedReader(new
                InputStreamReader(socket.getInputStream()));

// Recollim la petició del client, Mostra el número de sol·licitud per
tenir un control de les trames
// i seguidament la línia de sol·licitud
String lineaSol = peticioClient.readLine();
Msg.linia("----- Sol·licitud " + entrega +" ----");
Msg.linia(lineaSol);

// Recollim la línia de capçalera
//LLegim les línies de capçalera i les imprimim per pantalla
String capcalera = null;
while ((capcalera = peticioClient.readLine()).length() != 0) {
      Msg.linia(capcalera);
}
...
```

4. Extraiem la sol·licitud de fitxer:

```
...

//Passem al següent "token" on hem de tenir el nom del arxiu
sol·licitat
        String nomArxiu1 = tokenLinia.nextToken();

...
```

5. Després de processar-la, la passem per llegir-la i enviar-la al client:

```
...

if (existeix) {

// Per enviar els bytes que conformen el arxiu creem un buffer de
1KB
             byte[] buffer = new byte[1024];
             int bytes = 0;
// Copiem l'arxiu sol·licitat cap el stream de sortida del socket
de resposta.
             while((bytes = arxiuEnviar.read(buffer)) != -1 ) {
                  respostaServidor.write(buffer, 0, bytes);
             }

             //Tanquem el Stream d'arxiu
             arxiuEnviar.close();
...
```

6. Per últim fem el tancament de la petició a la classe PeticióHTTP i restem a la espera de al següent des de la classe Server:

```
...

// Tanquem els streams de resposta i petició i el socket.

        respostaServidor.close();
        peticioClient.close();
        socket.close();

...
```

## La interfície gràfica

Per obtenir una interfície gràfica des d'on configurar fàcilment els paràmetres del servidor, s'ha fet servir un panell JPane de JFrame on s'ha col·locat un botó i dues caixes de text amb les seves respectives etiquetes. Sota d'aquests podem trobar una emulació de la consola Java on van a parar tots els missatges que el programa va emetent.

El fet d'escollir un panell JFrame amb la resta d'elements (JButton, JTextArea,...) es degut a la seva exportabilitat a diferents sistemes operatius sense haver de canviar el codi, així tal i com s'ha pogut comprovar durant les proves de usabilitat el servidor fa la seva funció tant en Windows 8.1 com a Xubuntu Linux.

Però sense cap més preludi en la següent imatge podem veure com funciona el servidor, pel que fa a l'apartat gràfic en Windows 8.1:

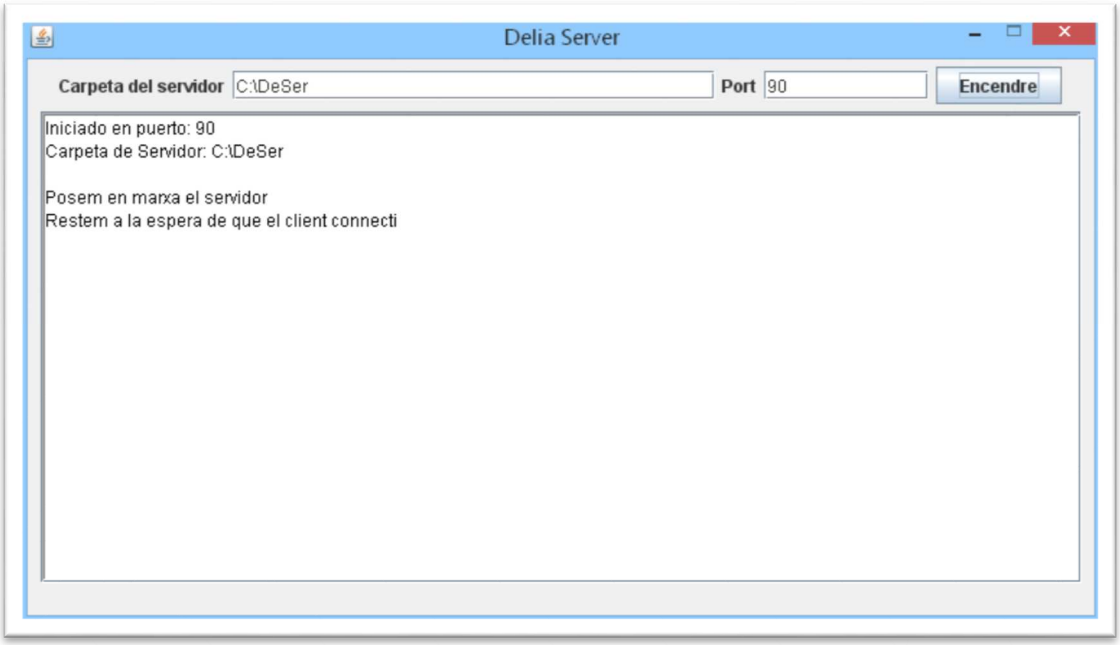



*Fig. 12: Interfície Gràfica del servidor*

Pel que fa la implementació, el panell es crida des de el main:

```
...
Panel p = new Panel();
...
```

Per seguidament afegir els elements que compondran la interfície i fer-la visible:

```
...
//Afegim els elements al panell gràfic
      p.addCajaText("Serv","Carpeta del servidor", "C:\\DeSer", 30);
      p.addCajaText("Port", "Port", "90", 10);
      p.addBoton("Encendre", "Aceptar i possar en marxa");
      p.addConsole();

//Creem el panell gràfic i el fem visible
      p.pintar();
      p.setVisible(true);
...
```

Aquestes trucades afegeixen els elements mitjançant mètodes descrits a la classe Panel. En el següent quadre veiem el mètode per afegir un botó, les etiquetes i la consola:

```
...
//Funció per afegir un botó
public void addBoton(String name, String toolTip){
	//Agafem el següent butó lliure de l'array
	buttons[numButtons] = new JButton(name);
	//L'afegim al principal
	principal.add(buttons[numButtons]);
	//Indiquem la ayuda
	buttons[numButtons].setToolTipText(toolTip);
	//Incrementem el apuntador de l'array de butons
	numButtons+=1;
}

//Funció per afegir una caixa de text amb la seva etiqueta
public void addCajaText(String tField, String desc, String valorDef,
int llarg){
	//Agafem el següent butó lliure de l'array
	campsText[numCamps]= new JTextField(valorDef, llarg);
	//Li assigmen el nom
	campsText[numCamps].setName(tField);
	//Li afegim una etiqueta
	JLabel lab = new JLabel(desc);
	//Afegim tot al principal
	principal.add(lab);
	principal.add(campsText[numCamps]);
	//Augmentem el número de l'apuntador de l'array de caixes de
text
	numCamps+=1;
}

//Funció per afegir un ScrollPanel que faci de Consola
public void addConsole(){
	//Determinem el color de fons i de les lletres de l'area de text
	j.setBackground(new Color(255, 255, 255));
	j.setForeground(new Color(0, 0, 0));
	//Determimen el tipus de voral
	j.setBorder(BorderFactory.createLoweredBevelBorder());
	//Afegim un text d'ajuda
	j.setToolTipText("Consola del Servidor");
	//Evitem que es pugui escriure al area de text
	j.setEditable(false);
	//Determinem columnes i files
	j.setColumns(65);
	j.setRows(20);
	//Afegim el area de text al panell Scroll
	jScrollPane1.getViewport().add(j);
	//Afegim el panell Scroll al principal
	principal.add(jScrollPane1);
}
...
```

Aquest elements creen necessitats auxiliars com per exemple controlar els esdeveniments de botons. En la següent caixa trobem el codi per controlar el click al botó encendre, en aquest cas el botó més important donat que és el que crea l'objecte servidor.

```
...

//FUNCIÓ DEL BOTÓ ENCENDRE
//Mitjançant l'addició d'accions als buttons crearem el servidor i
el posarem en marxa
p.buttons[0].addActionListener(new ActionListener(){
      @Override
      public void actionPerformed(ActionEvent e){
            if(Server.running == false){

                  argServer[0]=p.campsText[1].getText();
                  argServer[1]=p.campsText[0].getText();
                  Server serv;
                  try {
                        serv = new Server(argServer);
                        serv.arranca();
                  }
                   catch (Exception er) {
                        Msg.linia("Error creant el servidor\n" +
                        er.toString());
                              }
                        }
                  }});

...
```

Un altra aspecte a destacar ha estat la creació d'una classe OutputStream anomenada Msg que ha servit per ser accedida de forma estàtica i fer les funcions de System.out.println() però per comptes de fer-ho cap a la consola de Java ho fa cap a l'emulador de consola que hem creat amb JTextArea i JScrollPane.

En el següent quadre podem veure la implementació d'aquesta classe que està creada per poder facilitar les tasques de debugació i gràcies a la qual els missatges que va llençant el servidor son recollits en un entorn gràfic, donat que avui dia és el més habitual que l'usuari faci servir aquesta forma d'interacció per comptes de la consola o cmd.

```
...

//Classe que crea un nou OutputStream per poder redirigir la
sortida a la consola que hem
//creat a la classe Panel
public class Msg extends OutputStream {

private static JTextArea Text;
private PrintStream out;

//Constructor
public Msg(JTextArea Text)
      { Msg.Text = Text;}

//Funció que transcriu els caràcters en la sortida
@Override
public void write(int letra){
      out.write(letra);
}

//Funció que escriu una línia
public static void linia(String s) {
      Text.append(s+"\n");
}

}

...
```

PART 6

# El Nostre Servidor Web: Proves

## Proves 1: Windows 8. 1

El desenvolupament de aquest programari s'ha dut a terme en Windows 8.1, per començar ha fer les proves he fet servir els arxius de la web creada per informar del naixement de la meva filla Delia, aquesta web està corrent en un servidor web remot i per les proves fetes hem obtingut els mateixos resultats que consultant la pàgina des de el nostre servidor que des de el servidor online.

Per començar cal provar que el programa arranca. Per Windows hem creat un petit executable mitjançant el programari *"Launch4j"* que permet construir un embolcall .exe pels arxius .jar. Això ens ha permès el ús de una *splash screen* així com la assignació d'una icona que ha estat dissenyada especialment per aquest projecte.

En la següent imatge podem veure la pantalla emergent i la icona:


*Fig. 13: Splash Screen i Icona*

Seguidament veiem que el servidor ja es mostra llest per començar a funcionar. En aquesta pantalla podem decidir quin és el directori del servidor i quin el port de servei.

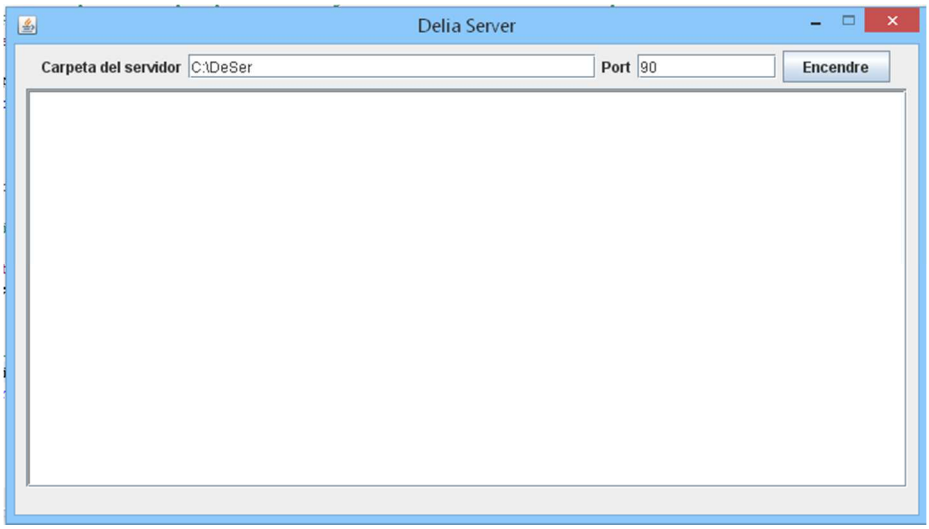



*Fig. 14: Servidor llest per ser executat*

Per últim hem de prémer el botó "Encendre" per deixar el servidor corrent. (El port predeterminat és el 90 per evitar interferències, en un funcionament convencional el port de servei http seria el 80).

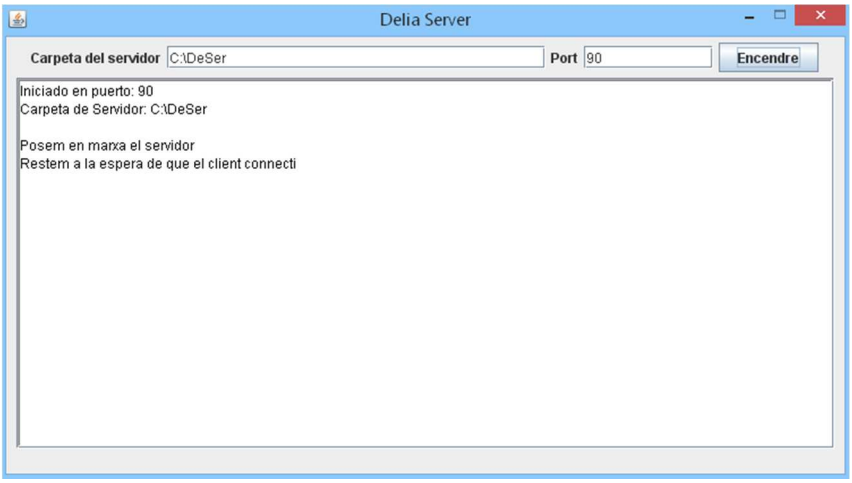



*Fig. 15: Servidor esperant connexions*

Un cop en marxa podem connectar amb localhost:90 per veure el que s'està compartint ( i comprovem que és exactament igual a la pàgina original):


*Fig. 16: El que el servidor està servint*

En aquest moment en l'emulador de consola de la interfície gràfica podem veure les comunicacions per carregar aquesta pàgina:

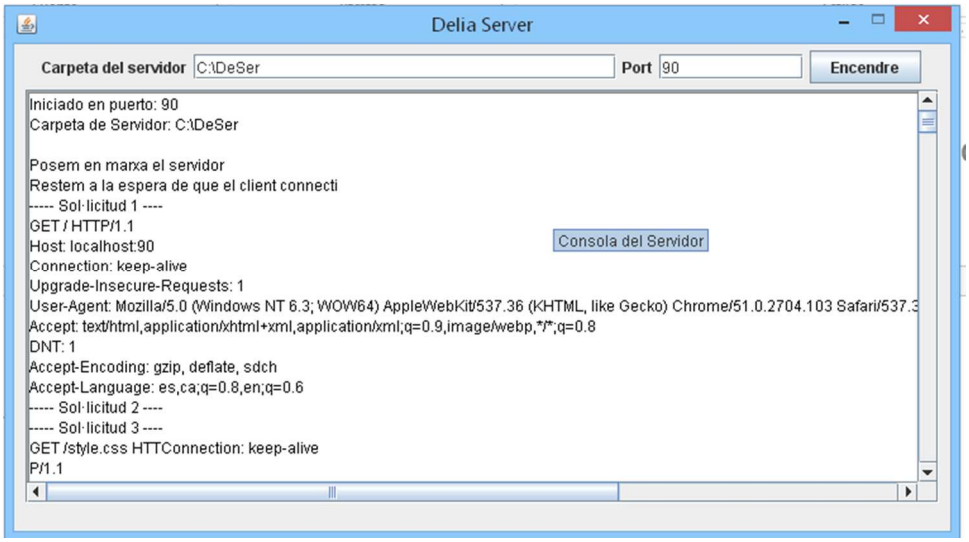



*Fig. 17: Consola amb el servidor treballant*

Quan tanquem el servidor, després de respondre sí a la finestra emergent comprovem que el servidor està aturat.

En una implementació primerenca, es va crear un botó anomenat "Ocultar", que ha quedat al codi comentat que permetia deixar el servidor corrent i tancar la interfície gràfica sense aturar el servidor. L'únic problema que trobàvem va ser que el servei queda en marxa sense poder-lo aturar d'una manera elegant, és pot fer un kill, però no s'ha cregut adient aquesta solució.

## Proves 2: Xunbutu Linux

La següent prova va ser fer córrer el servidor en un entorn Linux. Aprofitant que es disposava d'un netbook de característiques molt planeres on habitualment es fa servir Xubuntu per aprofitar les seves característiques, vam posar en marxa el servidor en aquest aparell.

Per fer-ho en aquest cas, com que els executables no són vàlids a entorns Linux per tant farem servir el jar. Perquè el nostre servidor funcioni cal que la versió de Java sigui 1.8 o superior, després de comprovar que disposem d'una versió igual a o superior a Java 8, haurem de llançar un terminal on executarem la següent comanda en el directori on es troba el servidor: "java –jar DeSer.jar". En aquesta ocasió no veurem cap splash i directament veurem la pantalla del servidor.

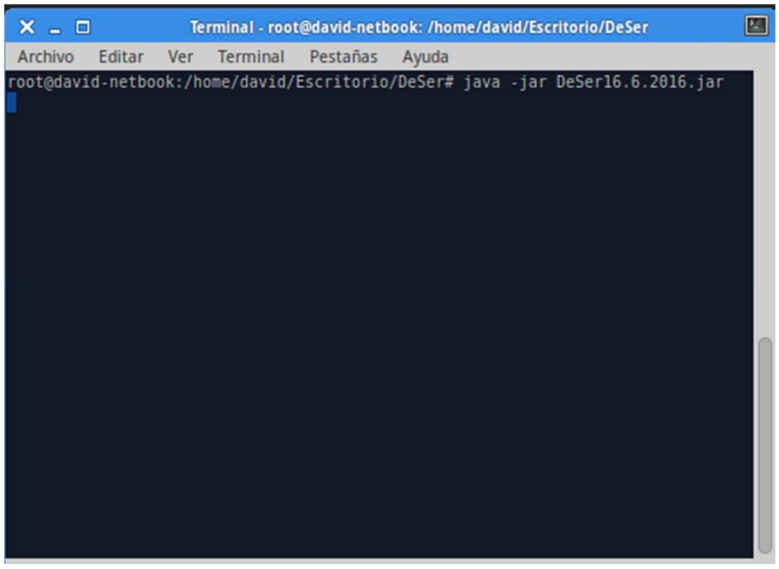



*Fig. 18: Terminal de Linux*

En aquest cas és molt important que canviem la carpeta de servei del servidor perquè sinó no podrem veure res perquè la predeterminada és per Windows. Es pot veure com el botó queda a sota en aquesta distribució perquè la pantalla és petita i flowchart agafa una nova línia per col·locar els elements.

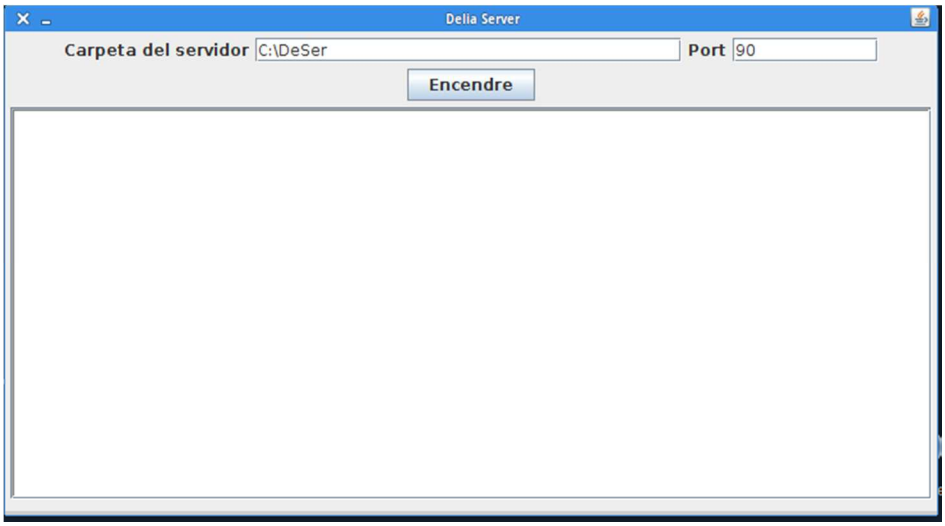



*Fig. 19: DeSer a Linux amb ruta errònia*

Si no aprofitarem per veure la pantalla d'error 404 (Aquesta pantalla està integrada al codi com un fragment HTML ).

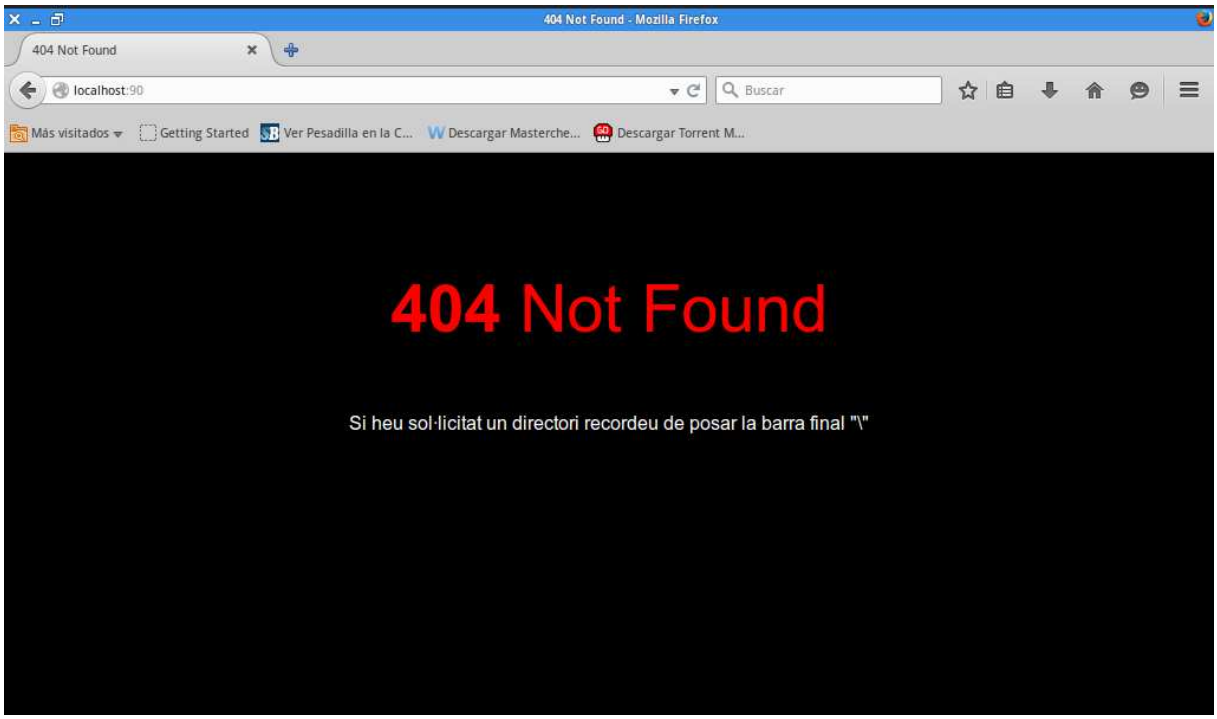



*Fig. 20: Pantalla d'error 404*

Un cop canviada veiem que el servidor corre bé:


*Fig. 21: El Servidor càrrega bé en Linux*

I recupera un JSON mitjançant una crida GET d'Ajax amb javaScript d'un exercici fet per l'assignatura de Programació Web Avançada:

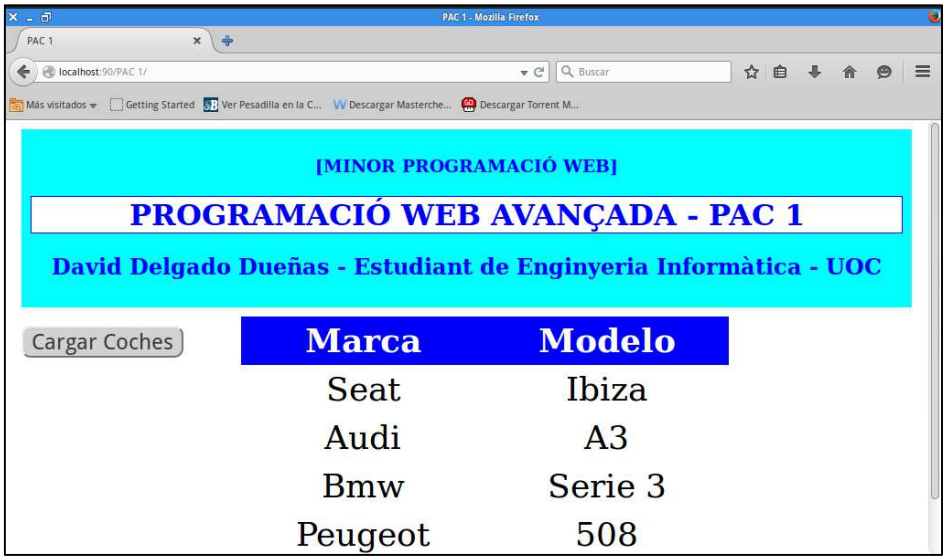



*Fig. 22: DeSer carregant un JSON*

## Proves 3: Servint per la Intranet local

Un cop hem vist que el servidor funciona i fa totes les funcions en loop local anem a provar la connectivitat en la xarxa internet local. Per fer-ho aprofitem que l'ordinador Linux està servint i des de l'ordinador amb Windows 8.1 connectem a la direcció local del ordinador Linux (192.168.1.108) en el port 90. Amb això i recuperant el JSON veiem que el servidor funciona per la intranet local.

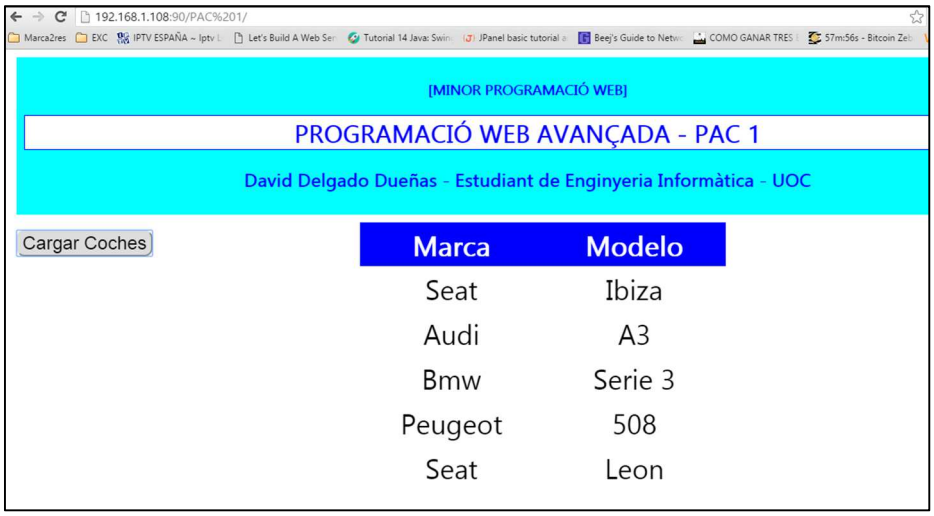



*Fig. 23: Servint des de Linux cap a Windows*

# Proves 4: Servint pel món

Per últim, anem a comprovar si podem fer el servei que el servidor ha de fer: posar els documents a Internet. Per fer-ho hem redirigit el port 90 del ordinador amb Windows 8.1 al port 90 de la direcció pública.

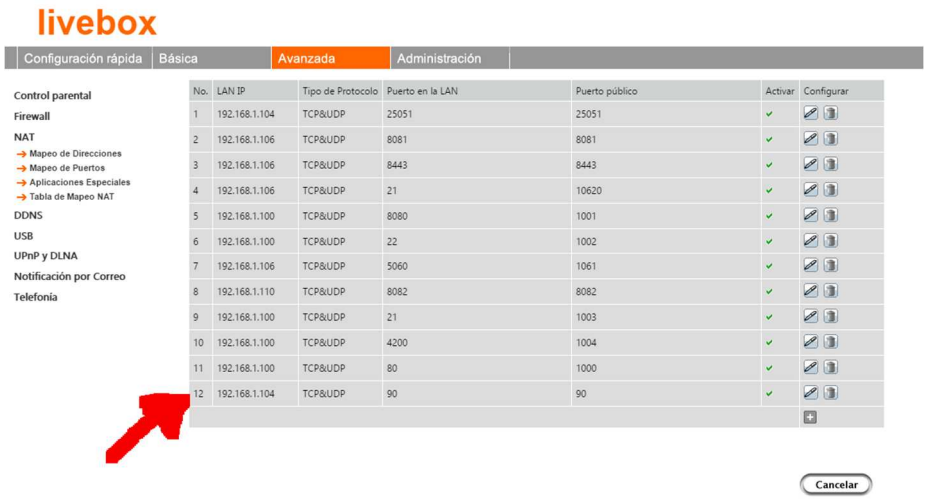



*Fig. 24: Assignació de port extern*

A continuació procedirem a fer la connexió des d'un aparell extern a la xarxa, en aquest cas un telèfon mòbil Android. Seleccionem la direcció pública de la xarxa on es troba el servidor seguit del port 90. I Veiem que el servidor serveix per al món sencer.

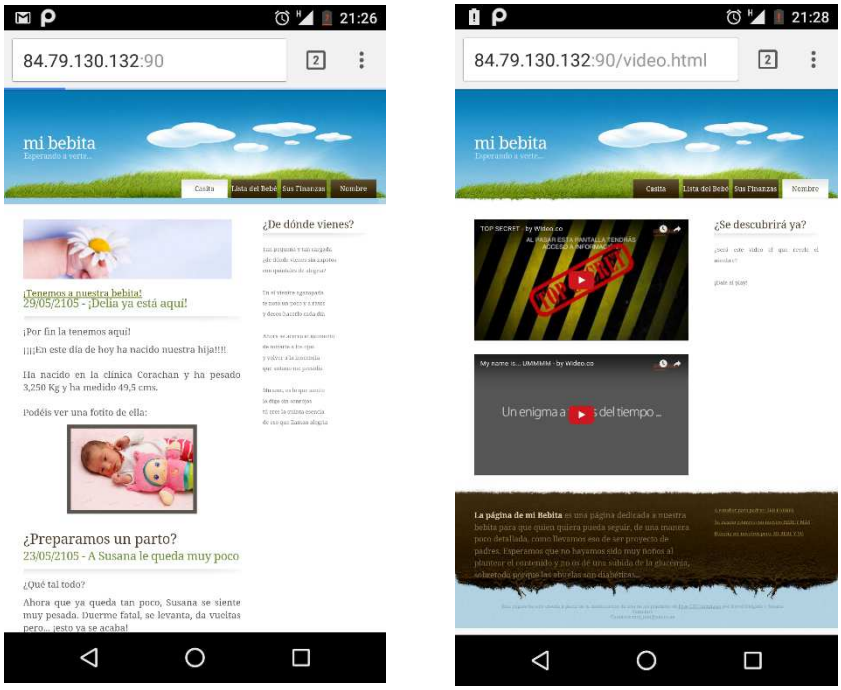



*Fig. 25: Servint en una direcció pública*

PART 7

# El Nostre Servidor Web: Instal·lació i configuració

A continuació farem una visió global de la instal·lació i la configuració del servidor. En la planificació d'aquest treball es va reservar un apartat per aquesta tasca, però ha quedat una mica desvirtuada quan tenim un servidor sense instal·lació i de fàcil configuració.

Per compensar una mica donarem els requisits que ha de tenir la màquina que corre el programa.

## Requisits

Els requisits són mínims:

- Java 1.8 (JRE ) o superior instal·lat

Poca cosa més donat que és un servidor molt lleuger. En aquest cas l'únic limitant es que està compilat en Java 8.

## Configuració

Com hem vist en les proves per poder configurar el servidor solament disposem de 2 caixes de text que automàticament son emplenades amb els valors per defecte. Aquesta és tota la configuració que requereix el servidor. Un cop emplenades les caselles, no cal fer res més si són correctes.

# BIBLIOGRAFIA


1. Roser Beneito Montagut. *Presentació de documents i elaboració de presentacions* . Material docent UOC. Barcelona: UOC.

2. Nita Sáenz Higueras i Rut Vidal Oltra. *Redacció de textos cientificotècnics*. Material docent UOC. Barcelona: UOC.

3. David Gourley and Brian Totty. *HTTP. The Definitive Guide.* . O'Reilly Media, Inc. California, USA. Primera Edició: Setembre 2002. ISBN: 1-56592-509-2

4. Sergio Luján Mora. *Programación de servidores web con CGI, SSI e IDC*. Editorial Club Universitario. Alicante, España. Primera Edició: 2001. ISBN: 84-8454-136-3

5. Marc Clifton. *Writing a Web Server from Scratch.* CODEPROJECT. Gener de 2015. http://www.codeproject.com/Articles/859108/Writing-a-Web-Server-from-Scratch (en línia) [Data última consulta: 19/06/2016]

6. Jon Berg. *Creating a simple java web server*. TURTLEMEAT. https://fragments.turtlemeat.com/javawebserver.php (en línia) [Data última consulta: 14/04/2016]

7. Redactors de Java2s. A Simple Web Server : Web Server Client. JAVA2S. http://www.java2s.com/Code/Java/Network-Protocol/ASimpleWebServer.htm (en línia) [Data última consulta: 19/06/2016]

8. Budi Kurniawan (Adaptació). How Java Web Servers Work. ONJAVA. Abril de 2003. http://www.onjava.com/pub/a/onjava/2003/04/23/java_webserver.html (en línia) [Data última consulta: 19/06/2016]

9. Assoudi. Implementing a Web Server in Java. CODEPROJECT. Juny de 2010. http://www.codeproject.com/Articles/86387/Implementing-a-Web-Server-in-Java (en línia) [Data última consulta: 19/06/2016]

10. Redactors de FREEFORMATERS. MIME Types List. FREEFORMATERS. http://www.freeformatter.com/mime-types-list.html (en línia) [Data última consulta: 19/06/2016]

11. Ophelie Lechat. The Complete List of MIME Types. SITEPOINT. Agost de2015. https://www.sitepoint.com/web-foundations/mime-types-complete-list/ (en línia) [Data última consulta: 19/06/2016]

12. Robert H'obbes' Zakon. Hobbes' Internet Timeline 23. ZAKON. Edició continuada 1993-2016. http://www.zakon.org/robert/internet/timeline/ (en línia) [Data última consulta: 19/06/2016]

13. Gabriela González. Internet y la web no son lo mismo, te explicamos por qué. BLOGTHINKBIG. Agost de 2014. http://blogthinkbig.com/internet-y-la-web/ (en línia) [Data última consulta: 19/06/2016]

14. Redactors de NetCraft. March 2016 Web Server Survey. NETCRAFT. Març de 2016. http://news.netcraft.com/archives/2016/03/18/march-2016-web-server-survey.html (en línia) [Data última consulta: 19/06/2016]

15. Redactors de la Wikipedia. Nginx. WIKIPEDIA. Març de 2016. https://es.wikipedia.org/wiki/Nginx (en línia) [Data última consulta: 19/06/2016]

16. Redactors de la web d'Apache. *What is the Apache HTTP Server Project?*. APACHE. http://httpd.apache.org/ABOUT_APACHE.html (en línia) [Data última consulta: 19/06/2016]

17. Redactors de Ibrugor. *Apache HTTP Server: ¿Qué es, cómo funciona y para qué sirve?*. IBRUGOR. Juny de 2014. http://www.ibrugor.com/blog/apache-http-server-que-es-como-funciona-y-para-que-sirve/ (en línia) [Data última consulta: 19/06/2016]

18. Redactors de la web de Cesanta. *Moongose (diverses pàgines del lloc)*. CESANTA. https://www.cesanta.com/products (en línia) [Data última consulta: 19/06/2016]