

|  |
| --- |
| Clàudia Ramírez Martin | Sergi Mas Pujol |

|  |  |
| --- | --- |
| Disseny Aplicacions Telemàtiques | MEMÒRIES DE LES PRÀCTIQUES |

Aquesta memoria mostra la feina que s’ha dut a terme durant el quatrimestre de primavera de 2016. En primer lloc la primera pràctica mostra la creació d’una página web molt simple, amb hiperenllaços, la mostra de data i hora del dia actual i un comptador de les visites rebudes. Els llenguatges emprats són html, c i scripting.

La segona pràctica es basa en un simple sistema de missatgeria interactiva amb entorn gràfic programat amb html i el codi d’interacció de missatges fent ús de cgis usant c++.

La tercera pràctica es basa en al mateix funcionament que la pràctica 2, però usant JAVA. La primera part està programada entorn gràfic amb html, ús de bases de dades sql i la interacció de missatges amb Java fent ús de servlets. Mentre que la segona part fa ús de html de forma gràfica, sql per a bases de dades i fent ús de SPA programat amb Javascript i Angular per a la interacció.

**Índex**

**PRÀCTICA 1**: Disseny d’una página Web simple…………………….

PRIMERA PART: Introducció…………………………………………….

SEGONA PART: experimentat amb HTTP……………………………….

TERCERA PART: experimentat amb els CGIs……………………………

**PRÀCTICA 2**: Disseny d’un sistema interactiu de missatgeria amb CGIs………..

Introducció………………………………………………………………....

Gestió de bases de dades……………………………………………………

Gestió de CGIs………………………………………………………………

Gestió de Cookies……………………………………………………………

Codis Font……………………………………………………………………

**PRÀCTICA 3**: Aplicació interactiva amb gestió de sessions i bases de dades………

Base de Dades…………………………………………………………………

Diferència entre la primera i la segona part…………………………………..

PARTS DE LA PRÀCTICA 3

PRIMERA PART: Ús de Servlets …………………………………..

Introducció……………………………………………………..

Accés al servei………………………………………………….

Sintaxis necessària………………………………………………

Codis Font………………………………………………………

SEGONA PART: Ús de SPA…………………………………………….

Introducció……………………………………………………….

Accés al servei……………………………………………………

Codis Font………………………………………………………..

**PRÀCTICA 1: Disseny d’una página Web simple**

**PRIMERA PART: Introducció**

**Objectiu**: Creació d’una pàgina simple amb URL <http://soft0.upc.edu/~USER/>, experimentació amb el descriptor MIME, incorporació de CGIs i creació d’hipervincles.

Observant aquest path podem dir que té 2 segments. Cada segment correspon a un directori, excepte l’últim que fa referència a un fitxer, en el cas que estigui buit fa referència a l’índex. Com a observació dir que ~ correspon a pàgines personals. D’aquesta manera, el nostre ~USER seria ldatusrXX.

Per tal de crear la nostra pàgina és necessari treballar sobre l’arbre físic soft0.upc.edu. Per tant, creem els directoris necessaris. En aquest cas tal i com podrem observar a la següent imatge hem creat:

* Dins de USER una carpeta públic\_html
* Dins públic\_html un índex.html i una altra carpeta anomenada pràctica1
* Dins la pràctica1 creem index.html i exemple.html

D’aquesta manera, fent un restart del servidor apache al terminal:

* *etc/init.d/apache2 restart*

I posant a un navegador la URL <http://soft0.upc.edu/~ldatusr12> visualitzarem l’índex. Aquest índex l’hem modificat al nostre criteri donada la plantilla de mostra.

A continuació hem fet unes proves per tal de veure el comportament del descriptor MIME. Al canviar el sufix .HTML per .TXT podem observar que veiem el text literalment, és a dir el propi codi HTML. El format .TXT ens dóna un text pla, al canviar el sufix el servidor donarà el tipus MIME, és a dir que ens indicarà el tipus de contingut, la manera com l’ha d’interpretar.

És important saber que si canviem el nom d’un directori això ens afectarà a la URL i també l’haurem de modificar.

Per a incorporar un CGI (Common Gateway Interface), cal crear un arxiu amb el codi font i donar-li el format adequat en aquest cas amb l’extensió .CGI. Posteriorment al terminal cal atorgar-li els permisos adients, accedint prèviament a la carpeta que ho conté:

* *chmod +x datahora.cgi*

Finalment per afegir-li l’hipervincle, hem modificat l’arxiu index.html de tal manera que:

*<A HREF="./fechahora.cgi"> Data i Hora </A>*

Per modificar el codi del CGI accedim a l’arxiu i canviem el tipus MIME a text/plain.

*echo "Content-Type: text/html”  echo "Content-Type: text/plain”*

Un cop realitzat aquest canvi al refrescar la pàgina podem observar que al clicar a l’hipervincle ens mostra un text pla, de la mateixa manera que anteriorment. És a dir, veurem el codi font.

Aprofitant els recursos donats hem creat un hipervincle dins la pràctica1. D’aquesta manera el codi que hem hagut de crear ha sigut:

*<P><A> HREF=./Practica1</A> Practica 1</P>*

Al clicar sobre Pràctica 1 veiem el contingut que hi ha dins la carpeta. En el nostre cas hem afegit els arxius .cgi necessitats a la pràctica en format plain per a poder veure’s el codi.

**SEGONA PART: experimentant amb HTTP**

Aquesta segona part l’hem realitzat des de fóra del laboratori. Per això hem hagut de muntar un servidor Apache al nostre ordinador personal.

El servidor ja està escoltant al port 80 per tant, per a poder fer un GET haurem d’executar la següent línia de comanda al terminal:

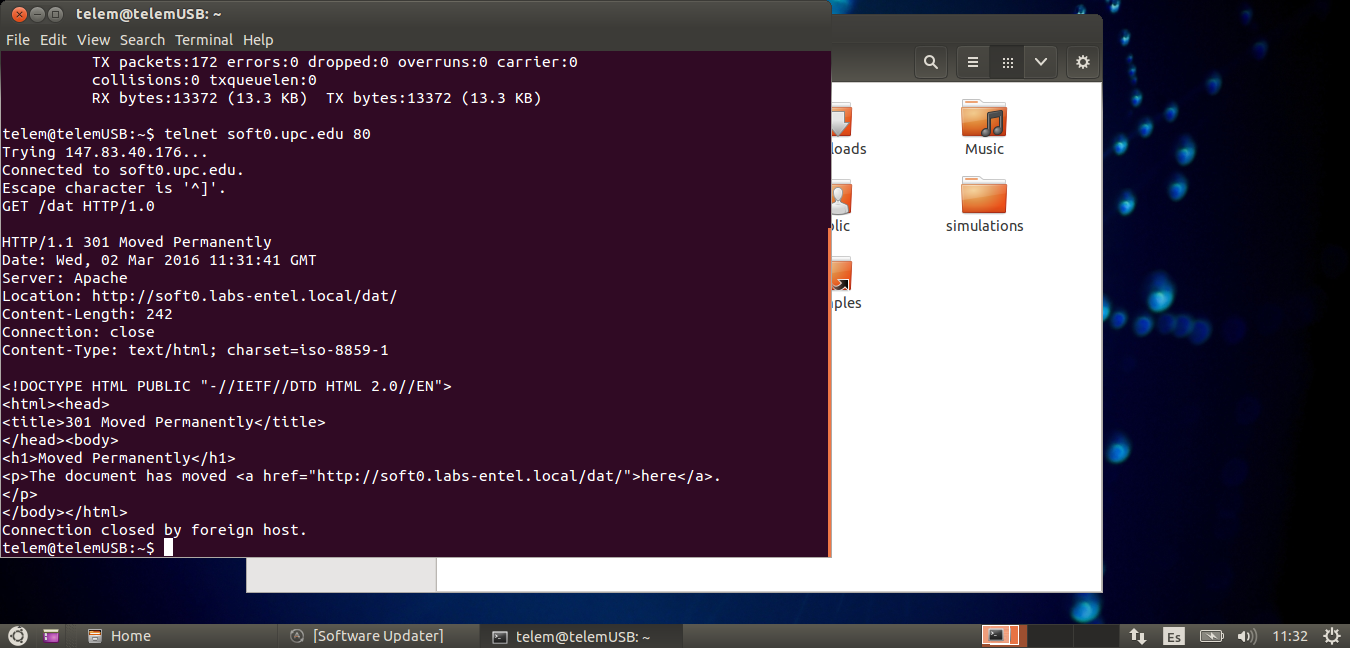
(El programa telnet és interactiu de forma que tot el que s'introdueixi per la consola s'envia al servidor, i tot el que es rebi del servidor es mostra en la consola.)

* *Telnet localhost 80*

Per a fer un GET de index.html:

* *GET / HTTP/1.0*

Podem observar que ha descarregat un arxiu en format HTML.



Quan fem un get obté el contingut d’un recurs. Aquest mètode s’utilitza quan es navega. No modifica l’estat i és idempotent. Cal destacar que la petició no porta contingut i la resposta sempre en porta.

La URL ha de ser de la forma: user@host:port/path

El mètode post és general per a modificar l’estat dels recursos. No és considerat idempotent i porta contingut tan a la petició com a la resposta.

Després de la realització de les pertinents proves podem concloure que la diferència entre el mètode GET i POST es basa en que el mètode post sí conté informació en la petició i que execucions reiterades comporten efectes diferents.

**TERCERA PART: experimentant amb els CGIs**

Es demana la generació dinàmica d’un document de resposta sense tractament d’entrada. És per això que s’ha decidit portar el recompte del nombre de visites a la nostra pàgina.

Aquest comptador s’ha codificat en forma d’script de la shell (.sh). Aquest script genera un document anomenat comptador.txt on emmagatzema el valor que es va actualitzant.

*V=$(cat comptador.txt)*

*Expr $V + 1 > comptador.txt*

*cat comptador.txt*

En primer lloc dir que hem creat un document anomenat comptador.txt on l’hem inicialiltzat a 0. Serà aquest valor sobre el qual treballarà el CGI. El funcionament del programa es basa en bolcar el valor de comptador sobre la variable V . A continuació la següent línia de codi incrementa la variable i la torna a bolcar sobre el fitxer. Per últim es fa una crida al .TXT per a mostrar el valor.

Durant el procés se’ns ha presentat un problema. Al fer el primer bolcatge de l’arxiu comptador sobre la variable V s’ha produït un error de sintaxi. Aquest ha estat detectat gràcies a l’editor de text Gedit que ens permet veure si una variable ha estat assignada o no. Al detectar que no ho havia sigut, hem acabat concloent que entre assignacions no pot haver espais en blanc.

*V = $(cat comptador.txt)  V=$(cat comptador.txt)*

Per últim abans de comprovar el seu correcte funcionament cal donar permisos a l’script. Per a dur a terme això hem executat al terminal:

* *chmod +x ./comp1.cgi*

A continuació es mostra el codi HTML i la representació gràfica:

**Pràctica 2: CGI interactiu amb gestió de sessions i bases de dades**

1. **Introducció**

La pràctica 2 es basa en la implementació d’una aplicació de missatgeria. En podem distingir tres parts essencials. En primer lloc un inici de sessió, en segon lloc el propi sistema de missatgeria i per últim el tancament de sessió. Per a poder dur a terme aquest desenvolupament és necessària la gestió de bases de dades i de cookies.

És necessari per a poder desenvolupar el projecte el domini de 3 conceptes: gestió de CGIs, gestió de bases de dades i administració de Cookies.

1. **Gestió de CGIs**

Els cgis fan la funció de transport de la informació. En el projecte farem servir aquests elements per a poder detectar quan cliquem un botó i per a poder extreure la información que s’ha escrit en els diferents formularis. A la següent explicació l’element request és el nostre cgi.

1. Fem query per extreure la informació

*Query q = postParams(request);*

En el cas que només volgués extreure un paràmetre d’informació:

1. Fem ús de l’element Maybe per saber d’on procedeix l’acció. L’element name ha de coincidir amb el nom del botó que fa l’acció, és a dir s’ha de tenir en compte al codi HTML quan es fa <INPUT NAME=”name”>

*Maybe<string> mb\_a = LookupParam (“name”, q);*

1. Comprovem que hi ha informació dins l’objecte Maybe fent servir un If, i a continuació transforma el maybe a un string mitjançant la funció fromJust. D’aquesta manera podrem manegar la informació de manera senzilla.

*If(hasParam(mb\_a)){*

*String a = fromJust(mb\_a);*

*}*

En el cas que volguéssim extreure un seguit d’elements:

1. Com ara hi ha varis elements que volem extreure, no fem un Maybe, sinó que directament muntem una llista amb varis elements. Això serà útil per a quan volguem eliminar varis missatges.

*List<string> ls\_a = LookupParams(“name”, q);*

Per gestionar el click dels botons

If(!isNothing(mb\_a)){

}

1. **Gestió de Bases de Dades**

En aquest projecte és important treballar amb bases de dades per a poder gestionar l’accés a l’aplicació i l’intercanvi de missatges. La base de dades inclou 3 taules. En primer lloc una taula que conté 2 camps anomenats usuaris i contrasenyes utilitzats per al procés de loggin. En segon lloc una taula que conté \_\_\_\_\_ que gestiona la introducció/eliminació de missatges i per últim la taula msg-seq que porta el recompte d’intercanvi de missatges.

Per a poder treballar amb les bases de dades, és necessari seguir els següents passos:

1. Establim la connexió

DBConnection dbc(db\_dir);

1. Seleccionem la taula sobre la qual volem treballar

dbc.select(“nomTaula”);

1. La primera fila d’una taula d’una base dades només incorpora el nom de cada camp, per tant, per a poder accedir a la primera fila real de dades és imprscidible aquest pas:

dbc.next();

Les bases de dades permeten obtenir la informació que hi tenen emmagatzemada, afegir informació i també eliminar-la.

**Obtenir informació**

* 1. Per obtenir informació de la base de dades

dbc.get(“fieldName”);

* 1. Per obtenir informació de forma contínua (recórrer la base de dades)

while(dbc.next()){

String info = dbc.get(“fieldName”);

}

**Introducció d’informació**

1. Creem vector d’Strings per a introduir les dades dels camps

list<string> dades;

1. Introduïm la informació al vector

datos.push\_back(“info\_camp1”);

datos.push\_back(“info\_camp2”);

1. Introduïm el vector a la base de dades

dbc.insert(“nomTaula”, dades);

**Eliminar informació**

1. Per eliminar informació d’una base de dades hem d’identificar què volem esborrar. El primer camp identifica el nom de la taula, el segon correspon al nom del paràmetre que farem servir per a identificar-lo i el tercer camp és el valor que ha de tenir el camp per a ésser eliminat.

dbc.delete\_rows(“nomTaula”, “camp”, “valorCamp”);

1. **Gestió de cookies**

És molt important la gestió de Cookies en un sistema de missatgeria. És necessari per a poder saber qui és l’usuari que està enviant un missatge i per a poder esborrar-les en el moment de sortir de la sessió i això evitar problemes de seguretat.

1. Guardem les cookies al iniciar la sessió i ho enviem a l’usuari

El funcionament del Pair<SessionMap, SaveSession> és una mica particular. SessionMap és un objecte del tipus map que conté Strings de parella <Clau,Valor>= <User, Pass>. SaveSession és un mètode.

Pair<SessionMap, SaveSession> session = ClientLoadSesson(request);

Com el funcionament és particular, per poder treballar amb l’objecte SessionMap hem de extreure aquest primer camp, que correspon al nom d’usuari que ha iniciat sessió. Per tant, en aquest cas, name conté el nom d’usuari que haurà iniciat sessió.

Session.first.insert(pair <string , string> (“User”, name ));

Mitjançant la següent comanda es carregarà l’usuari al SaveSession i enviarà una Cookie a l’usuari.

Response hs = session.second.first(session.first);

1. Extracció de la informació de dins la Cookie

Pair<SessionMap, SaveSession> session = ClientLoadSesson(request);

String userName = Session.first.at(“User”);

1. Eliminem les cookies

Pair<SessionMap, SaveSession> session = ClientLoadSesson(request);

Session.first.erase(“User”);

Response hs = session.second.first(session.first);

1. **Codi Font**

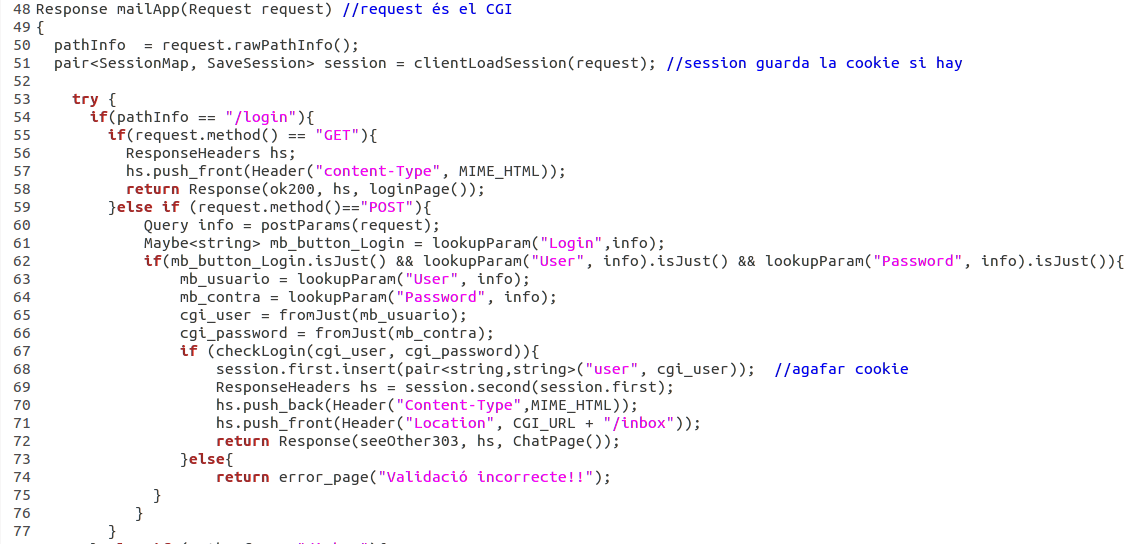
\*\*Cal possar els peu de pagina

La página d’inici de sessió té dos camps a introduir per l’usuari. S’ha d’introduir el nom i el password.

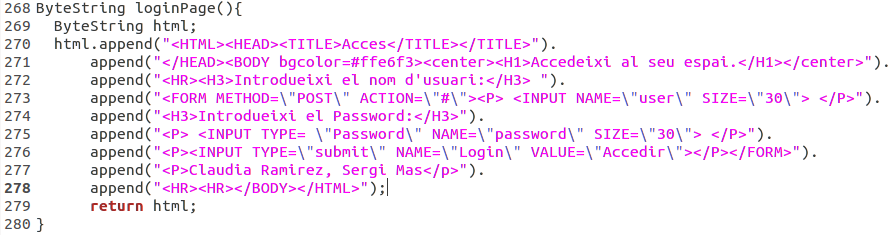
Tal i com es pot observar en el codi següent, la comprovació del usuari i la contrasenya es realitza mitjançant un post. I la utillitzacio del mètode “checkLogin”.

Per a poder realitzar totes les comprovacions necessaries cal la extracció de les dades introduides en el formolari. Aquesta acció es duu a terme amb les funcions “postParams” i “lookupParam”.

Aquests dos camps en cas d’estar registrats están a la base dades i correspondre la parella, ha de permetre l’accés al sistema, i en el nostre cas genera una redirecció cap a “chatPage”, és a dir, es genera una resposta (Response) en el qual s'inclou tot el codi HTML necessari per a la creació del sistema de missatges que estem generant.



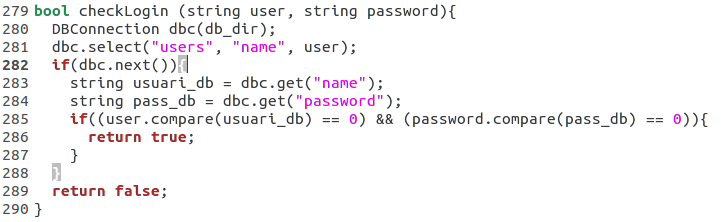
El métode “loginPage” , genera el codi HTMl necessari per a la intraducció dels paràmetres per a la autenticació.



El resultat, de forma visual, de la pagina de “login”, és el que es pot veure a continuació.



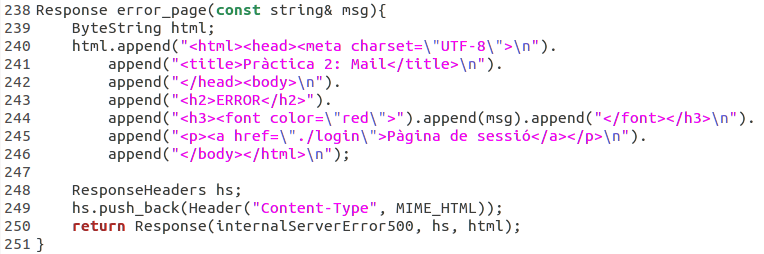
El métode “checkLogin”, com es pot observar si li passem per parametres el usuari i la contrasenya introduits, es capaç de recorrer les base de dades on hi han els usuaris registrats buscant coincidencies.



Aquesta comprovació es fa en el moment de pémer el botó d’accedir (mb\_button\_Login.isJust()). En cas contrari, ens sortirà una página informant de l’error, tal i com et pot observar en la linia 74.

“error\_page”, s'encarrega de la generació del codi HTML necessari per informar del error detectat.

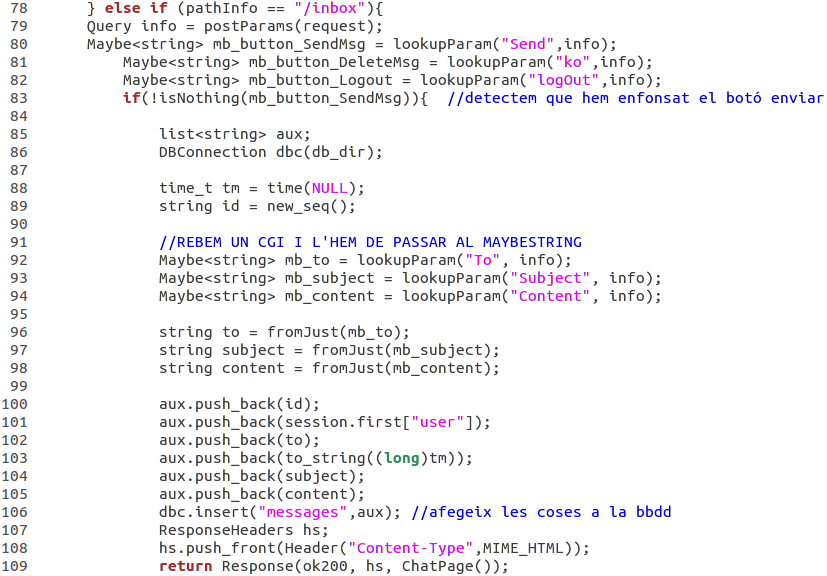
Es un métode genèric, que ens permet la visualització de diferents possibles errors o irregularitats, ja que en la seva invacació només ens cal passar per paràmetre el problema detectat, en forma de “string”.



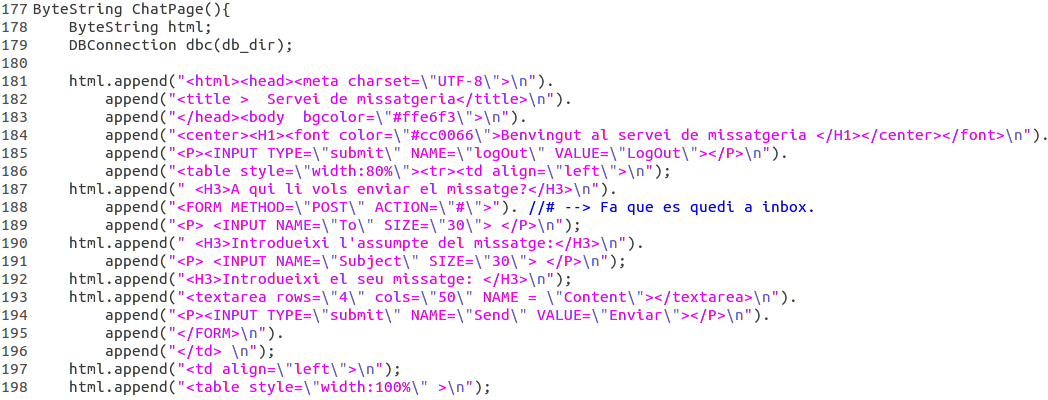


El manteniment de cada sessió es farà transferint al client (navegador) l'estat usant *cookies*. Serà usat per a quan es vulgui saber qui és l’emissor del missatge.

La resposta del CGI en cas d'inici satisfactori serà una pàgina HTML que contindrà un formulari que permeti a l'usuari enviar un missatge qualsevol o bé tancar la sessió (pàgina d'operació). La resposta a enviar missatge haurà de ser guardar el missatge en la bade de dades i novament la pàgina d'operació.  
Cada registre haurà de contenir l'identificador del missatge, l'usuari remitent, l'usuari destinatari, dia i hora, l'assumpte i el contingut del missatge.



La pàgina HTML necessari per a la introducció de nous missatges, amb la utilització de formulari, es pot subdividir en dues part. Un primera que genera el formolari en sí, per a generar nous missatges. Que correspondria al codi HMTL següent.



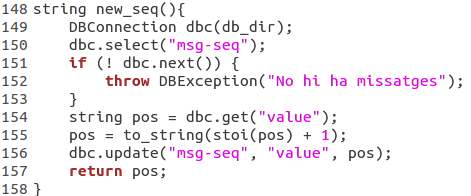
I una segona que permet la visualització dels missatges existents.



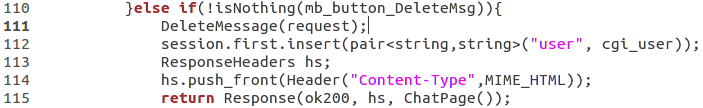
El conjunts d'aquestes dues parts, ens permet obtenir un resultat final, on la seva aparença visual és la que podem veure a continuació.



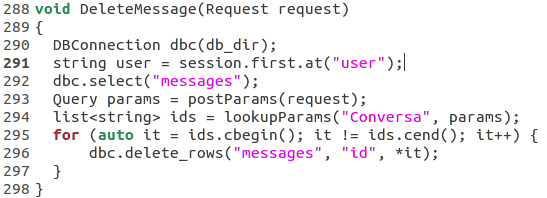
L'identificador de cada missatge, necessari per al correcte funcionament de l'aplicació ens el proporciona un mètode independent. En el nostre cas la funció “new\_seq” es capaç de generar un “string” el qual contè el identificador del nou missatge.



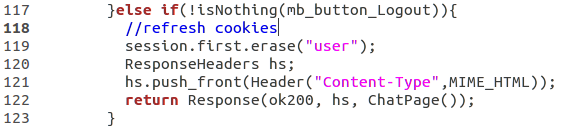
Un cop introduits els missatges dissitjats a la base de dades, es dispossa de la possibilitat de eliminar-los, mitjançant el boto corresponent. Acció de la qual s'encarrega el següent fragment de codi següent.



El métode auxiliar que s'invoca en l'eliminació, “DeleteMessage”, és el que accedeix a la base de dades dels missatges i elimina els sel·leccionats anteriorment mitjançant els “ids”.



La resposta al tancament de sessió serà l'eliminació de la *cookie* corresponent (o similar) i una pàgina de comiat (amb un enllaç a la pàgina d'inici, per exemple). A continuació podem veure el codi que s'invoca quan el presiona el boto de “logOut”.



PRÀCTICA 3:

1. **Base de dades**

La nostra base de dades està formada per dues taules.

En primer lloc la taula “users”, que guarda tres camps (id, name, password).

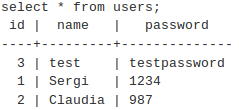
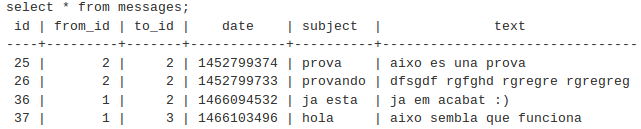


Tabla 1

En segon lloc tenim la taula messages, que guarda l’identificador del missatge (id), l'identificador de l'emissor (from\_id), del destinatari (to\_id), la data d'emissió (date), l'assumpte del missatge (subject) i finalment el contingut del missatge (text).

Si es desitja modificar aquestes taules a través del terminal tenim les següents opcions:

Tabla 2

-Esborrar contingut de la taula: delete \* from nom\_taula;

-Eliminar una fila: delete from nom\_taula where nom\_columna=valor;

**-**Afegir una fila: insert INTO nom\_taula(camp1 , camp2, camp3, …) VALUES ('valor1', 'valor2', 'valor3', …);

1. **Diferència entre la primera i la segona part**

La pràctica 3 es basa en una aplicacií interactiva amb gestió de sessions i bases de dades.

La primera part de la pràctica es basa en l’ús de servlets. Es fa ús d’un framework anomenat DatFw, útil quan una aplicación Web s’implementa al servidor.

La segona part fa ús de SPA (Servei de Pàgina Única) on es minimitza l’ús de recursos i per tant és més òptim. La principal diferencia doncs radica en que un cop carregada la página principal, el servidor únicament s’encarrega d’oferir un servei web d’accés a les contes. Tota la presentación i interacció amb l’usuari és tractada localment al client, és a dir, al navegador.

PRIMERA PART: Aplicació interactiva amb gestió de sessions i bases de dades amb ús de Servlets

1. **Introducció**

En aquesta primera part tractarem l'aquitectura MVC (**Model-View-Controller**), la gestió de sessions d'usuari i un sistema de persistència (bases de dades).

Concretament aquesta pràctica consistirà en la realització d'una aplicació interactiva, com la especificada en la pràctica 2, però amb la utilització de servlets, en lloc de CGI's.

Un servlet és una classe de java, utilitzada per ampliar les capacitats d'un servidor, els quals poden respondre qualsevol tipus de sol·licituts i poden ser executats en servidors en comptes dels navegadors web.

1. **Accés al servei**

<http://soft0.upc.edu:8080/ldatusr12-practica3/> \*escollir hiperlink Inbox

1. **Sintaxis necessària**

Interpolació de valors

Dues formes d'interpolació:

#{expr}

on expr és una expressió JAVA de tipus HTML ó bé convertible a HTML. Per ara hi ha definida la conversió a HTML dels tipus String i Integer.

En l'expressió es poden usar variables constants (finals) visibles en l'àmbit de la invocació de la plantilla.

@{expr}

on expr és una expressió JAVA de tipus dat.fw.web.app.Route<AppClass> i AppClass és el tipus de la classe principal indicada en la invocació de la plantilla.

També es poden usar variables constants visibles en l'àmbit de la invocació de la plantilla.

Condicionals

La forma general és:

$if{ expr1 }

...

$elseif{ expr2 }

...

$else

...

$end

on les expressions expr1... han de ser de tipus boolean, poden haver-hi 0 ó més seccions elseif i la secció else és opcional.

Maybe

La forma general és:

$maybe{ v :: T <- expr }

...

$nothing

...

$end

on l'expressió expr ha de ser de tipus dat.fw.util.Maybe<T> i la secció nothing és opcional.

Es defineix la variable v de tipus T que es podrà usar dins la secció maybe .

Forall

La forma general és:

$forall{ v :: T <- expr }

...

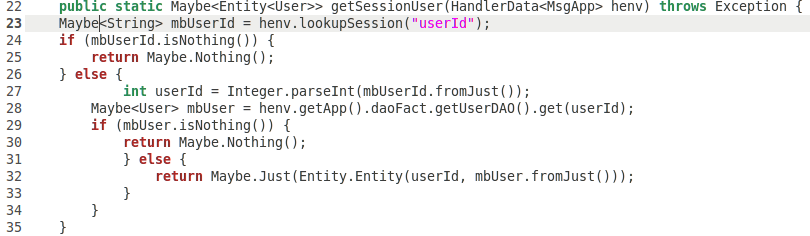
$end

on l'expressió expr ha de ser de tipus java.util.Iterable<T>.

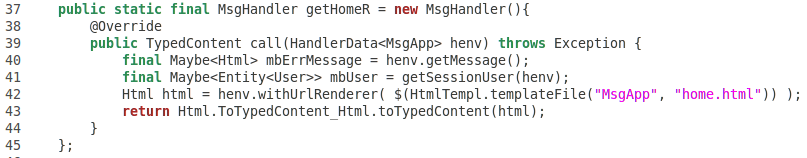
Es defineix la variable v de tipus T que es podrà usar per a cadascuna de les iteracions.

1. **Codi font**

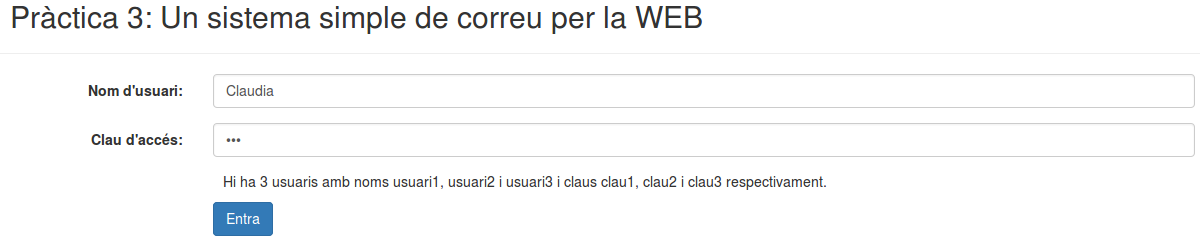
El mètode **getSessionUser,** retorna els paràmetres identificadors vinculats a l’usuari registrat.

Codi 1

El mètode **getHomeR** , mostra la pàgina principal del sistema de missatgeria implementat. Aquesta pàgina està implementada en un fitxer anomenat “home.html” (Text X), a la il·lustració X podem veure la seva representació gràfica.

Codi 2

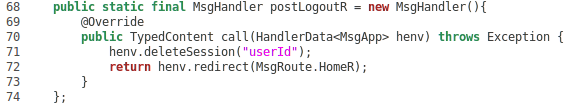
HTML 1

Il·lustració 1

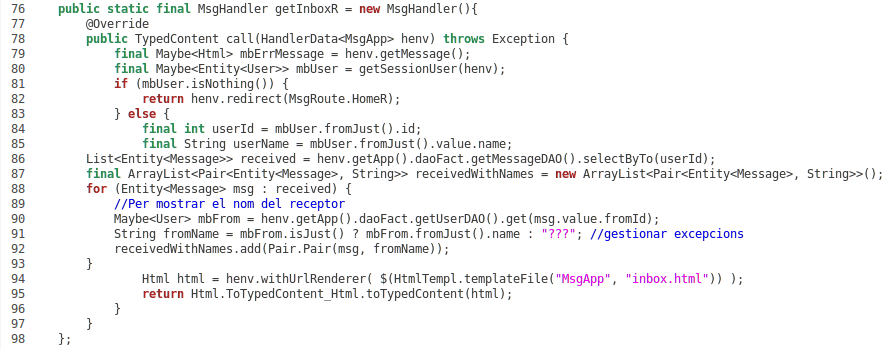
El mètode **postLoginR**, comprova que el teu nom d'identifiació coincideix amb un registre de la base de dades.

Codi 3

El mètode **postLogoutR**, permet tancar la sessió, eliminant el “userId” i redirixeix a la pàgina “homa.htlm”.

Codi 4

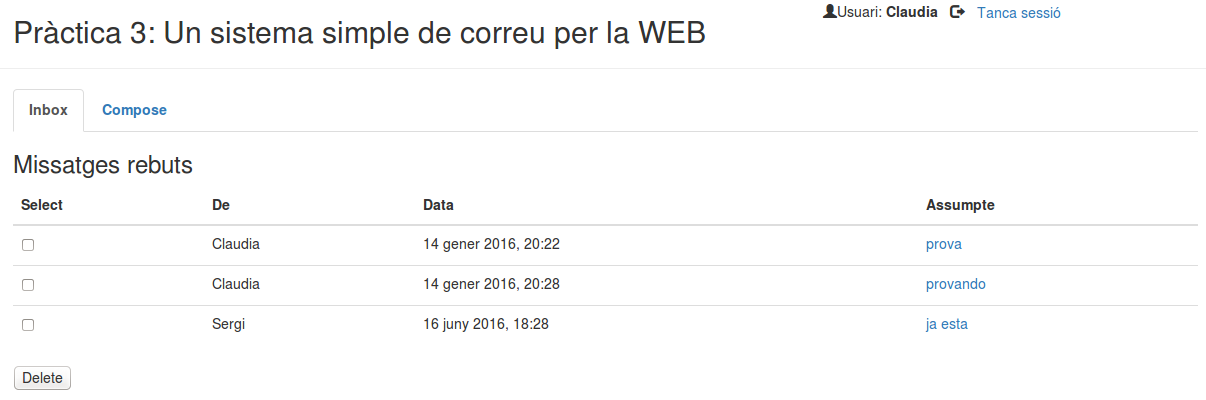
El mètode **getInboxR**, obté tots el paràmetres necessaris per a la visualització de la pàgina principal de l'aplicació, en la qual es poden visualitzar els missatges existents.

Codi 5

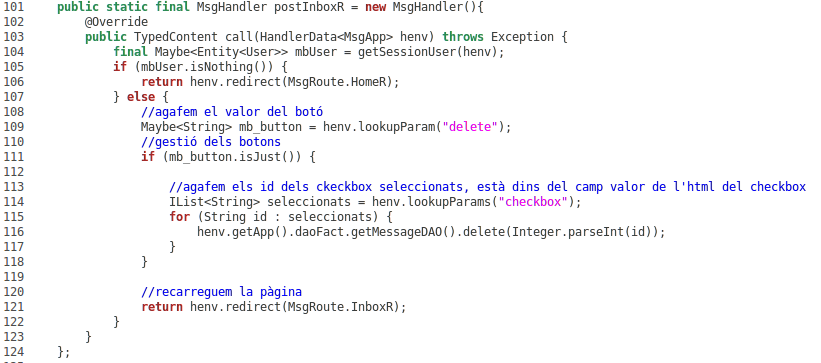
El codi html associat a la pàgina de “inbox.html”, és el que podem veure a continació:

El processat d'aquest codi html permet una visualització amb les característiques següents:

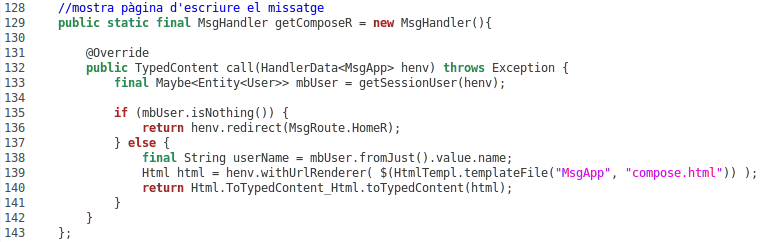
HTML 2

Il·lustració 2

El mètode **postInboxR**, ens permet mitjançant l'utilització de “checkbox” la possibilitat d'eliminar diferents missatges, des de la pàgina d'inbox. El funcionament d’aquest mètode es basa en la detecció del botó delete. Un cop es detecta que s’ha clicat, agafa els identificadors dels checkboxes què han estat seleccionats i els elimina. Per últim, recarrega la página d’inbox.

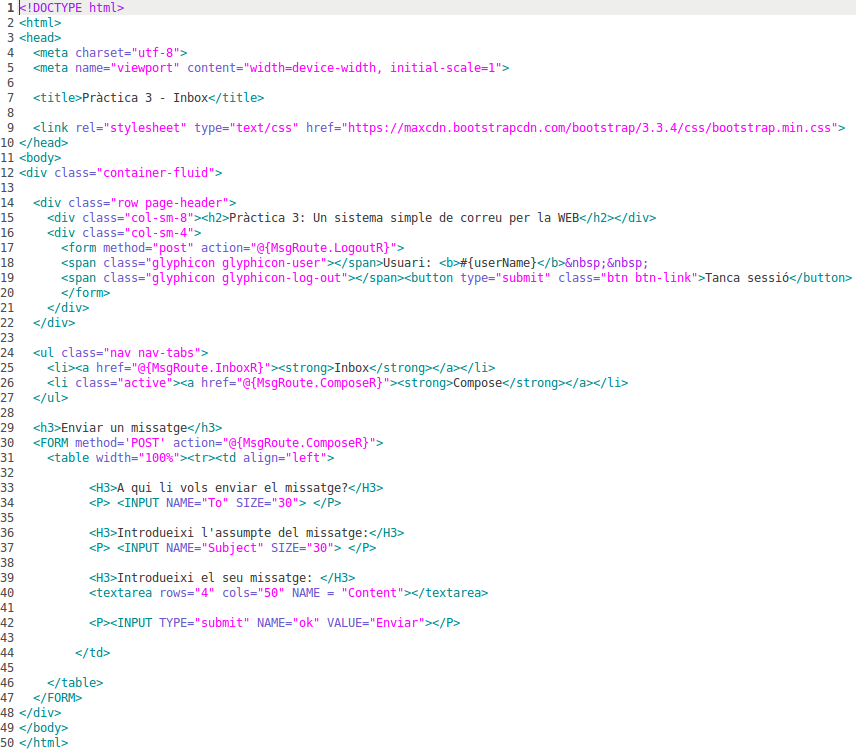


El mètode **getComposeR**, permet dins de l'aplicació la possibilitat de visualitzar el formulari d'introducció de missatges a la resta d'usuaris registrats a la base de dades.



La plantilla del formulari està programada amb “html”, el qual podem veure a continuació:

Volem destacar que la navegació es duu a terme mitjançant pestanyes. Per tant cal moure’s a la pestanya “compose”, és a dir, cal possar em mode “active” (com podem veure a la linea 26).

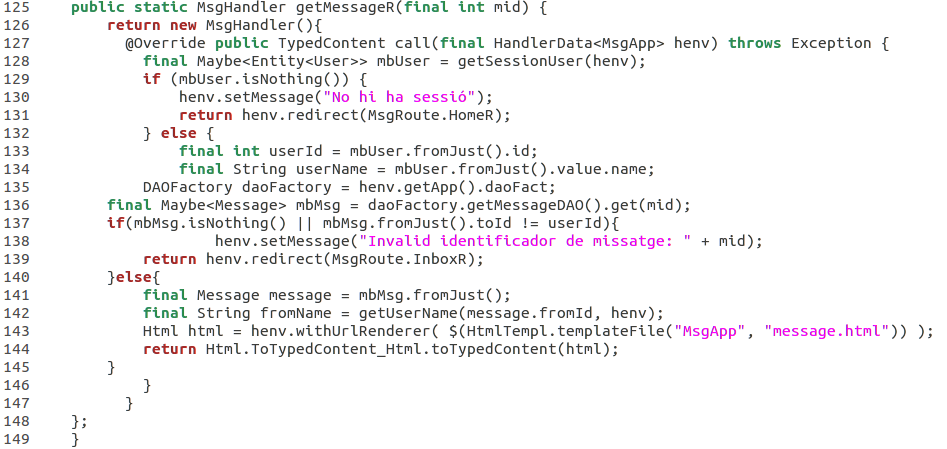
HTML 3

Un cop processat el codi “html”, el resultat gràfic que obtenim és el següent:

Il·lustració 3

El mètode **postComposeR**, s’encarrega de guardar els camps omplerts del forumlari. Quan detecta que s’ha premut el boto Enviar, agafa els valors introduits a cada camp i els emmagatzema a la base de dades. Els camps introduits sobre el codi html són: receptor, assumpte i contingut del missatge. Però per instroduir-los a la base de dades, és necessari fer-ho de forma ordenada, tal com està descrit a la bbdd, i omplint tots els camps. Per tant, com a la taula message són necessaris els camps emisor, receptor, data, assumpte i missatge, i ens falten els camps emisor i data els hem d’introduïr de la forma: el camp emisor s’agafa de la sessió i el camp data utilitzant un mètode propi de Java.

El mètode **getMessageR,** ens permet veure al veure el contingut d'un missatge de forma detallada.



Si el processat és satisfactori s'invoca “message.html”, el qual es capaç d'obtenir tota la informació necessari per a una adequanda representació.



El processat del codi anterior, permet una visualització com la que es pot veure a continuació.



**SEGONA PART: Aplicació interactiva amb gestió de sessions i bases de dades amb ús de SPA**

1. **Introducció**

El procés d’autenticació i la base de dades és compartit amb la primera part de la pràctica. Un cop logguejat l’usuari podrá escollir l’opció SPA, Aplicació de Pàgina Única. La funcionalitat és la mateixa. Permet enviar missatges, veure els missatges rebuts i eliminar els missatges rebuts. Permet eliminar un missatge concret un cop l’estem llegint o esborrar-los de la página inbox de forma múltiple fent ús de checkboxes.

Per aquesta segona part fem ús del framework client Angular JS. Està programat en Javascript i és opensource desenvolupat per google. Serveix per a creae WebApps usant SPA. Està basat en el MVC i és molt intuiritiu i de fàcil comprensió. Nosaltres el fem servir per a poder usar les platilles de forma òptima i poder manipular dades en JSON.

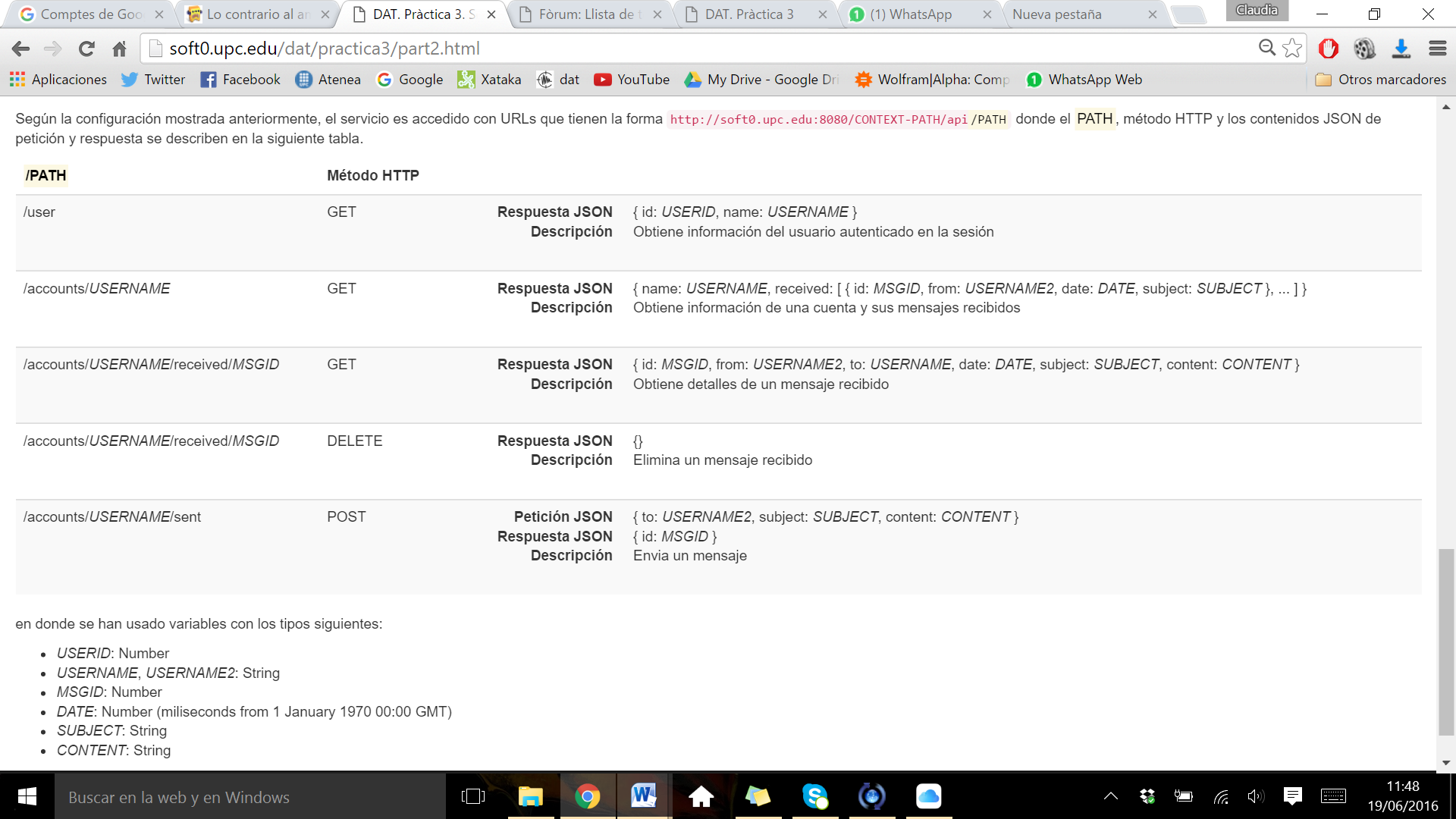
1. **Accés al servei**

<http://soft0.upc.edu:8080/ldatusr12-practica3/> \*escollir hiperlink 2ª part SPA

1. **Codi font**

L'organització d'aquesta part de la pràctica es basa en uns controladors I unes plantilles.

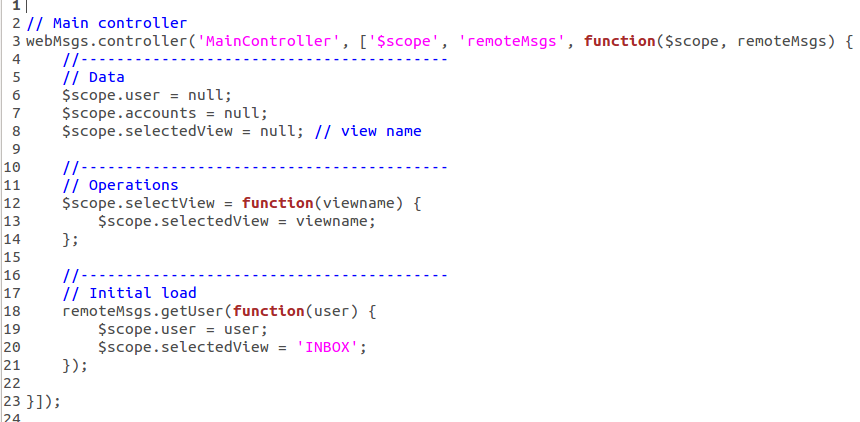
Els paths corresponents donats i les respostes JSON vénen donades per enunciat:



A continuació podem veure el controlador de la pàgina principal.

**SelectView** es fa servir per a seleccionar quina és la plantilla que volem visualitzar.

**GetUser** és emprat per a saber qui és l'usuari que ha iniciat sessió I per tant poder dir qui és l'emissor dels missatges que estem enviant.



**Els mètodes essencials de funcionament de la pràctica són:**

**getUser** s'encarrega de d'obtenir informació de l'usuari autenticat en la sesió actual.

La funció anterior ens proporciona una resposta JSON, mitjançant la qual es pot obtenir l'identificador del usuari (id: userId) I el nom associat a aquesta identificació (name: userName).

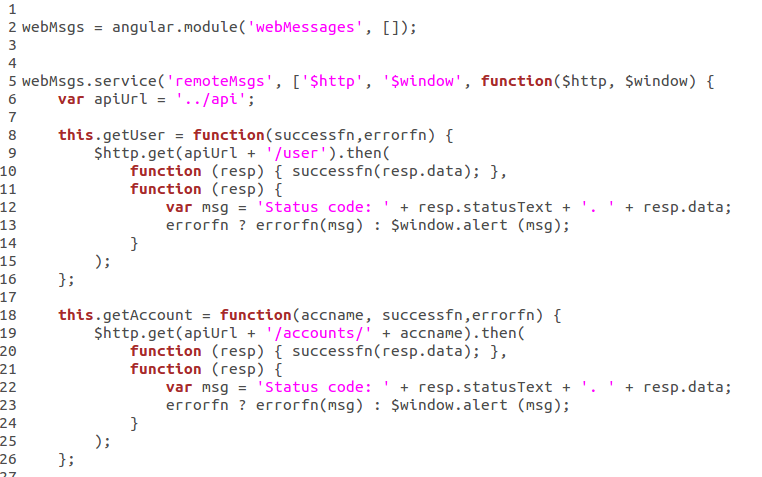
**getAccount** s'encarrega d'obtenir informació d'una conte, en concret, I els missatges rebuts, associats a aquest compte. Requereix del pas per paràmetres del nom d'usuari.

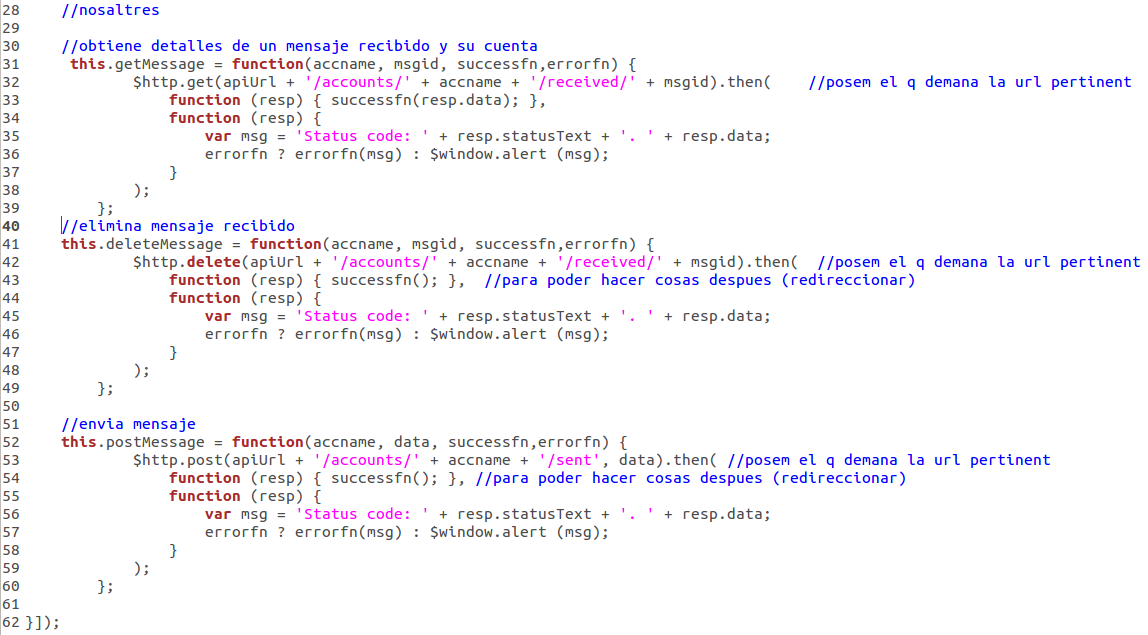
En la resposta que proporciona tenim el nom del compte, I el conjunt de misatges associats ([ id: MSGID, from: USERNAME2, date: DATE, subject: SUBJECT }, ...] )

**getMessage** s'encarrega de retornar tota la informació que prové de la id del missatge. Aquetsa informació que retorna és: id del missatge, emissor, destinatari, la data, l'assumpte I el contingut del missatge.

**deleteMessage** s'encarrega d'eliminar un missatge rebut.

**postMessage** s'encarrega d'agafar els camps receptor, assumpte I contingut del missatge per a guardar-los a la base de dades I poder-los mostrar a Inbox.

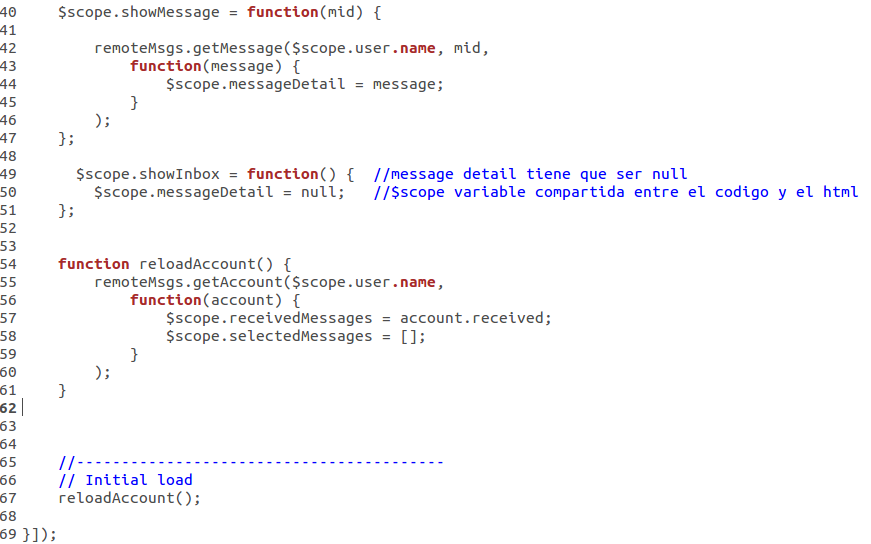




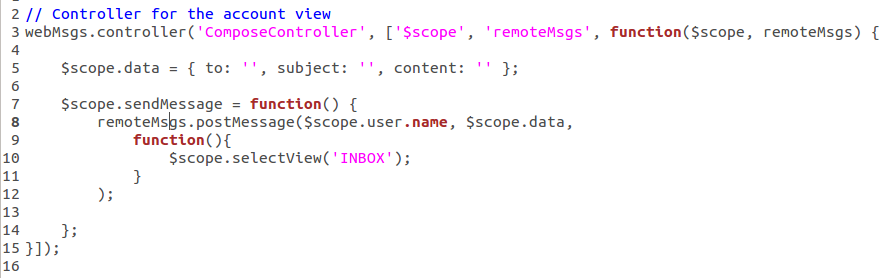
A continuació podem veure l' InboxController:

Que correspon al controlador de la secció de visualització del missatges.





El controlador de Compose conté la variable data, necessària per a poder accedir-hi des del codi html. Quan una variable té davant $scope, vol dir que és una variable accessible per a tots els controladors I els codis html. La funció sendMessage serveix per a agafar els valors del missatge que es vol enviar.

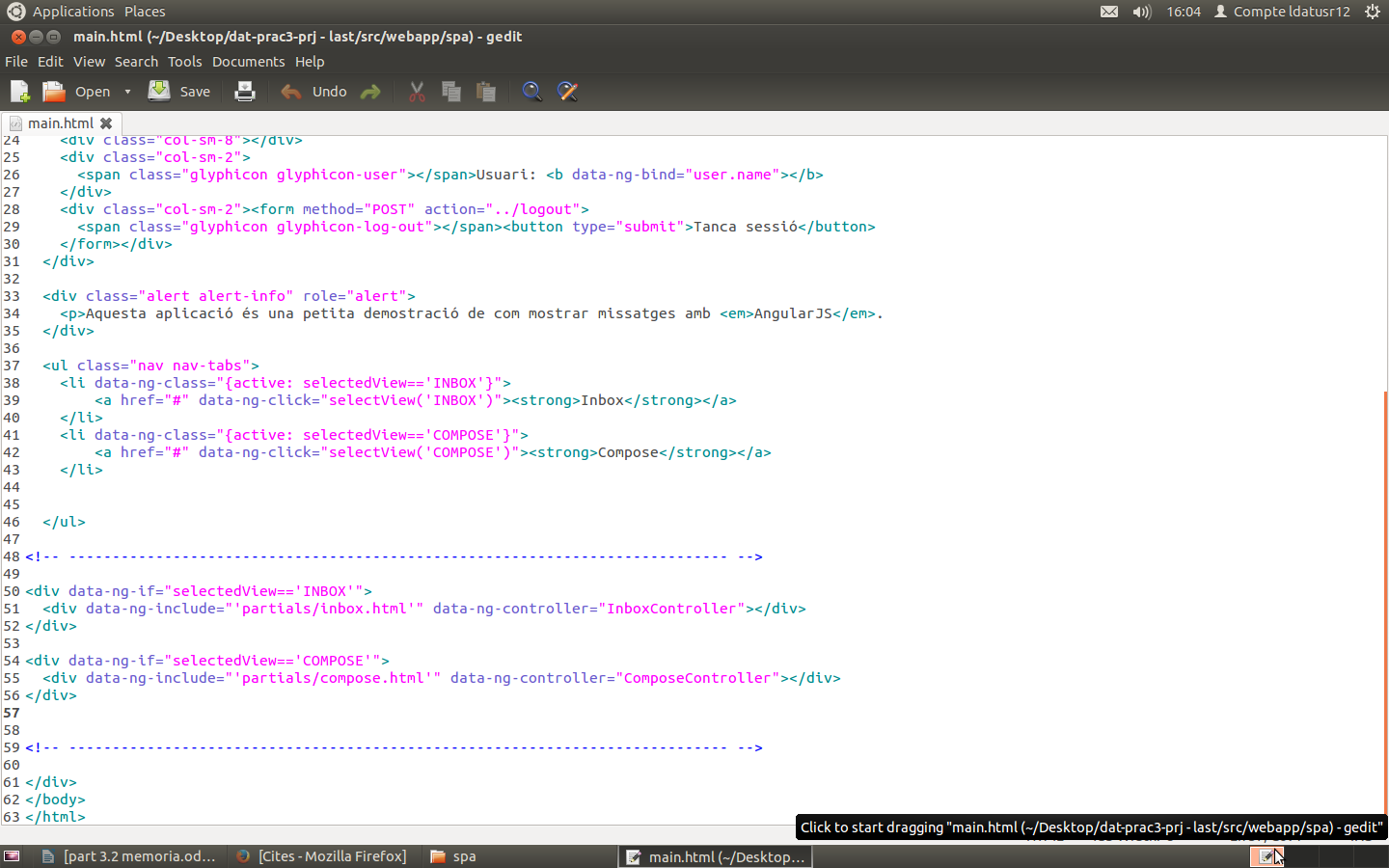
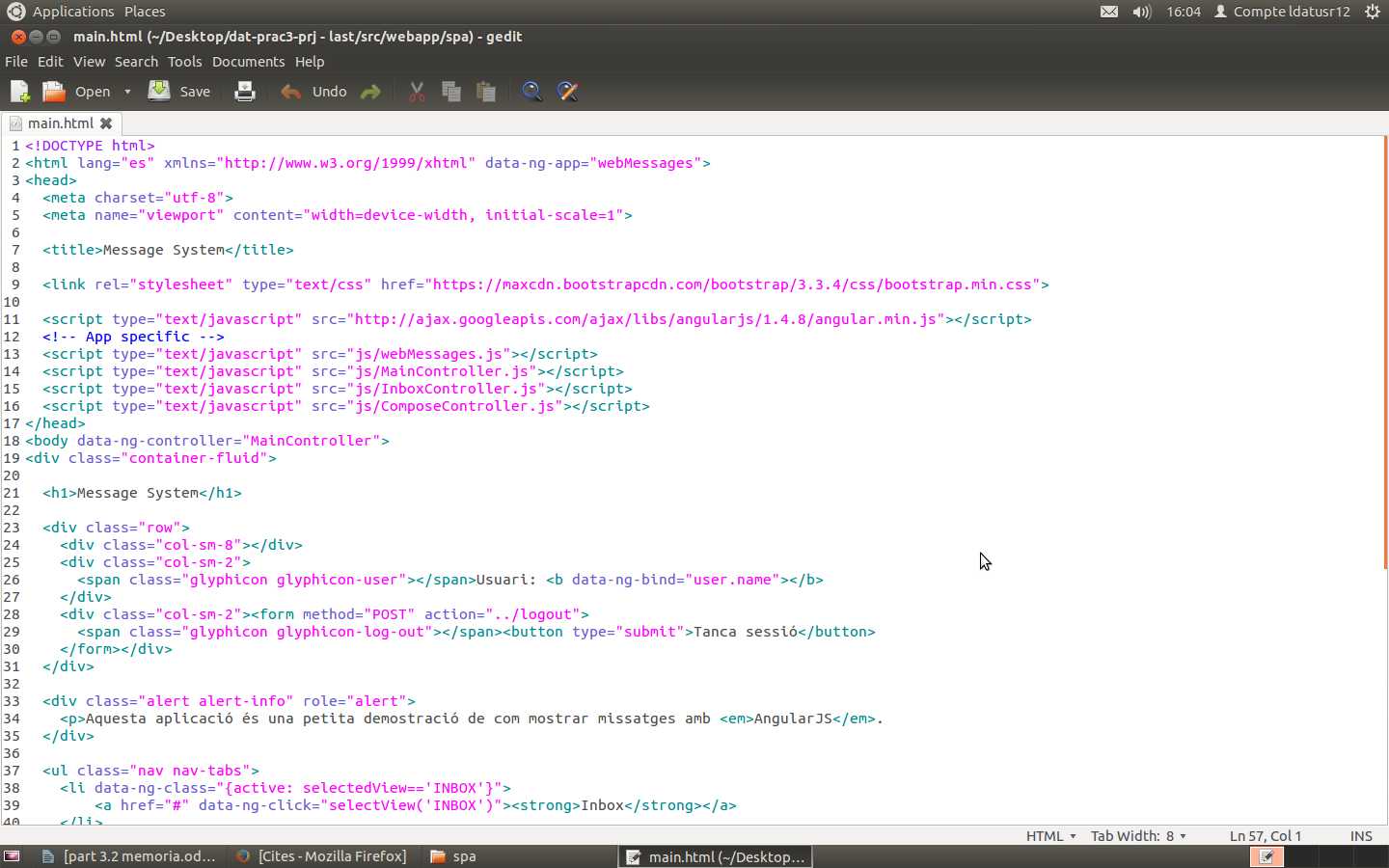


L'entorn gràfic està organitzat en forma de templates. Es fa d'aquesta manera per evitar moltes línies de codi que fan complicat d'entendre el codi.

D'aquesta manera, a la pàgina principal, en aquest cas main.html hem d'incloure uns enllaços que hi facin crides. En aquesta aplicació tenim 2 plantilles programades en html: inbox.html I compose.html

Cada plantilla té un controlador que permet fer les accions. Aquests constroladors estan programats en javascript. Ha estat dissenyat d'aquesta manera per a facilitar I fer més còmode les modificacions del codi. Els htmls fan referències als propis controladors.

Mistajçant l'estructura data-ng-if permet que poguem moure'ns d'una plantilla a una altra depenent de l'acció que s'estigui realitzant.

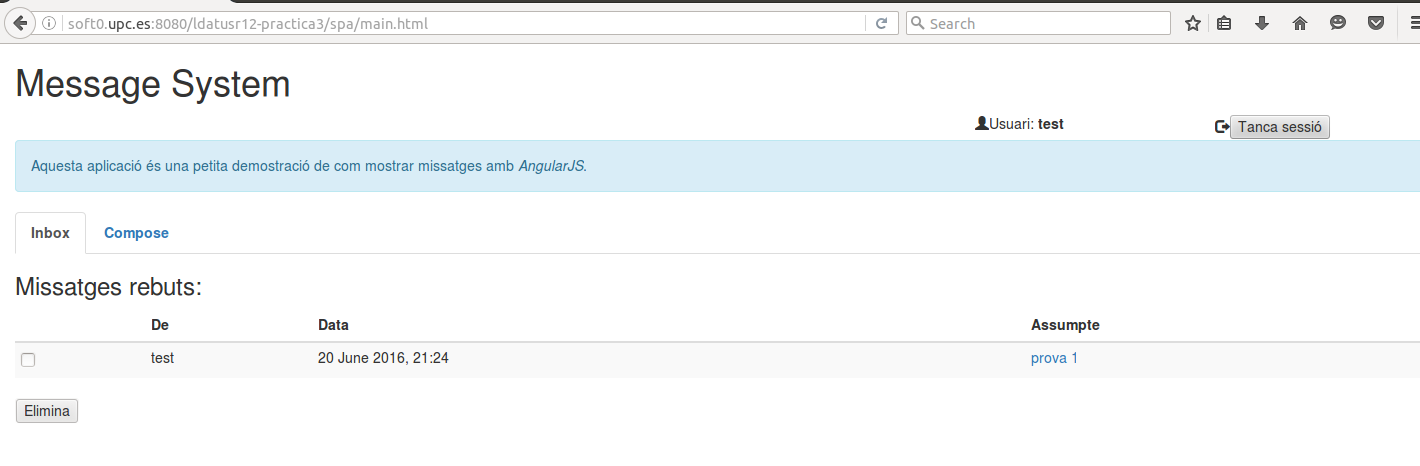


La pàgina html corresponent a Inbox pot tenir dues opcions de visualització. Per a poder diferenciar quina de les dues opcions es veurà fem servir una estructura if. L'element que comparem és messageDetail, és a dir, mirem si té detalls o no del missatge.

En primer lloc, quan messageDetail és igual a null, la pàgina d'inbox mostra la llista de missatges amb l'emissor, la data I l'assumpte del missatge. Aquesta pàgina mostra un checkbox en cada missatge I el botó de delete que permet la selecció múltiple de missatges que es volen eliminar.

L'element msg prové del paràmetre que es passa a les funcions provinent de InboxController.

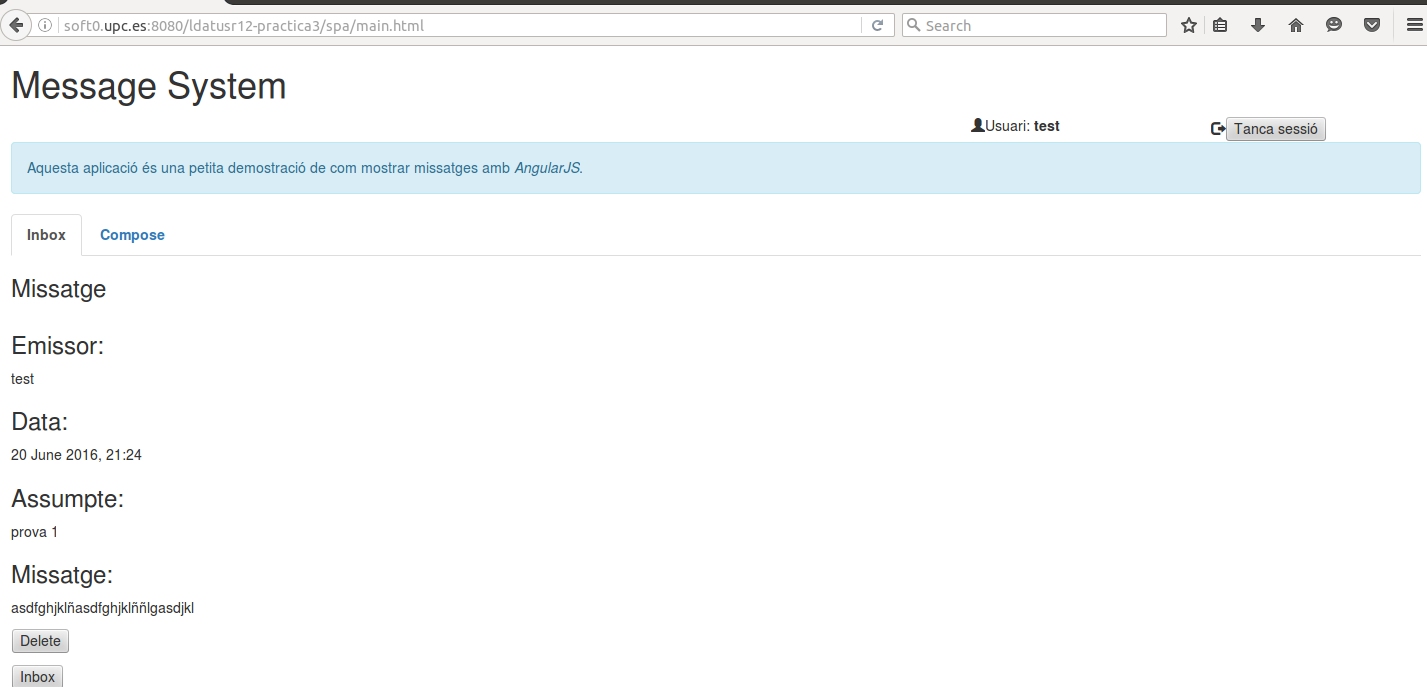




Quan messageDetail és diferent de null, és a dir, quan sí té detalls del missatge, es mostren aquests. És a dir, la pàgina que veurem serà la d'un missatge amb el seu emissor, la data d'enviament, l'assumpte I el propi contingut del missatge. Aquesta pàgina permet també l'eliminació del missatge en concret I també poder tornar a la pàgina anterior d'inbox que llista tots els missatges.

MessageDetail prové de InboxController.





A continuació podem veure el codi html que correspon al formulari d'enviament de missatge. La principal diferència amb la pràctica 2 es basa en què ara als elements d'on volem agafar els valors per introduir-los a la base de dades haurem d'afegir un camp que s'anomena:

*data-ng-model=”camps”*

L'element data que conté els detalls del missatge prové del codi ComposeController.

