Disseny d'una GUI pel software ICHNAEA

Títol:	Disseny d'una GUI pel software ICHNAEA
Autor:	Carles Baró, Joan
Data:	20 de juny de 2013
Director:	Luis Antonio Belanche Muñoz
Departament del director:	LSI

Titulació:	Enginyeria Informàtica
Centre:	Facultat d'Informàtica de Barcelona (FIB)
Universitat:	Universitat Politècnica de Catalunya (UPC) BarcelonaTech

Índex

1. Introducció 7
1.1 Antecedents 8
1.2 Objectius 9
1.3 Estructura de la memòria 11

2. Anàlisi de requeriments. 13
2.1 Requeriments funcionals 13
2.2 Requeriments no funcionals 14

3. Especificació
Diagrama de casos d'ús 17
3.1 Entorn web 17
3.2 Casos d'ús de l'aplicació mòbil 24

4. Disseny i implementació 31
4.1 Servidor
4.1.1 Introducció 32
4.1.2 Disseny i implementació del servidor 34
4.1.3 Servidor d'aplicacions 40
4.1.4 Web services 42
4.1.5 Base de dades 45
4.1.6 Integració amb ICHNAEA 51
4.1.7 Client Web 56

4.2 Client mòbil
4.2.1 Introducció: Android 65
4.2.2 Disseny de l'aplicació 67

5. Planificació i costos 74

6. Conclusions i treball futur 77

7. Bibliografia i referències 79

Apèndix I

1.1 Android client user manual 82

1.2 User web interface 84

1.3 Administrator manual 86

Apèndix II

2. Instal·lació i configuració

2.1 Servidor 89

2.2 Base de dades 90

2.3 Intèrpret R 91

2.4 Aplicació ICHNAEA 91

2.5 Client Android 92

1. Introducció

El problema anomenat MST (Microbial Source Tracking) té a veure amb la determinació de l'origen de la contaminació fecal de l'aigua mitjançant indicadors químics i microbiòtics. Aquest és un problema d'abast mundial, ja que afecta severament la qualitat de vida al ser els individus exposats a contaminació bacteriana portadora de malalties, sobretot en els països en vies de desenvolupament.
Un cop determinat un focus de contaminació resulta interessant veure a què és degut, si és d'origen humà o animal i de quin tipus, de cara a establir responsabilitats i actuar en la seva prevenció.

Mitjançant tècniques d'aprenentatge automàtic (machine learning) s'ha aconseguit resoldre satisfactòriament per a casos amb una concentració de pol·lució alta i mitjana. La tendència en els estudis MST és a intentar definir indicadors específics pels diferents orígens fecals i a la vegada establir metodologies que a partir d'aquests indicadors obtinguem resultats acurats.

Hi ha diferents enfocaments, i tècniques emprades per a resoldre el problema MST, però en totes però cal obtenir mostres i analitzar-les buscant components patògens, generalment de tipus bacterià, ja que aquests es troben abundantment en la matèria fecal. Això fa que s'aconsegueixi uns conjunts de dades que permetran "ensenyar" els sistemes MST de cara a obtenir resultats.

1.1 Antecedents

Aquest projecte treballarà amb el software ICHNAEA [1]. Aquest software tracta el problema MST no solament tractant el lloc d'origen de la contaminació sinó també els factors de la dilució (diferències en la concentració) i el temps que fa que aquest contaminant es troba en l'aigua. A més, un avantatge d'ICHNAEA és que permet a l'usuari ensenyar el sistema i aquest indica quins són els millors indicadors que l'usuari ha de tenir en compte per fer l'anàlisi.

Informàticament parlant, ICHNAEA és un conjunt de rutines escrites en llenguatge de modelatge estadístic R, la qual cosa té diverses implicacions:

1. qui les hagi d'usar ha de conèixer la sintaxi d'aquestes rutines i el funcionament de l'entorn d'R.

2. al ser aquestes rutines desenvolupades com a projecte d'investigació on principalment interessaven els resultats estadístics, no es va tenir en compte en fer-les genèriques, la qual cosa fa que hi trobem algunes variables o paràmetres "hard-coded".

3. S'han d'executar en local, en la mateixa màquina on està l'R.

La qual cosa fa que es desenvolupi aquest projecte, el qual no tractarà amb ICHNAEA des del punt de vista d'anàlisi sinó que desenvoluparem una utilitat informàtica per a aquest.

1.2 Objectius

En aquest projecte es té per objectiu el desenvolupar un entorn d'usuari, una GUI (Graphical User Interface), que faciliti la interacció amb les rutines de l'ICHNAEA. Concretament ens interessa específicament la rutina que ens permet fer el què anomenem un "test de predicció", de tal manera que un usuari li pugui passar una lectura de dades en forma de matriu i mitjançant un entrenament previ (que queda fora de l'abast d'aquest projecte), el sistema li retornarà una sèrie de valors amb els possibles contaminants de la mostra introduïda.
Aquest "entorn d'usuari" com ho he anomenat, serà un client que correrà en dispositius portàtils amb sistema operatiu Android i que en definitiva oferirà la possibilitat que un usuari, des de qualsevol lloc on tingui connexió a Internet pugui introduir dades de lectures, obtenir-ne resultats i també consultar lectures i simulacions que hagi realitzat anteriorment.

Així doncs, ell projecte consta de dues parts:

- un servidor: que és l'encarregat de realitzar les simulacions i guardar-ne els resultats per la seva posterior consulta, a més de tasques de gestió d'usuaris i sessions.
- el client: que ens serveix per introduir dades que seran processades pel servidor i permet consultar-ne els resultats i és el que correrà sobre Android.

Hi ha una sèrie de motius pels quals el projecte es separa en dos "subprojectes", definint una part de servidor i una altra de client:

- No es desitja que els possibles usuaris del sistema tinguin accés al codi font de l'aplicació ICHNAEA.

- Es desitja realitzar un control sobre qui hi accedeix, ja que pot arribar a esdevenir una aplicació de distribució controlada. Cal tenir en compte que hi ha empreses que realitzen serveis semblants a uns costos elevats.
- Hi ha una sèrie de limitacions tècniques que impedeixen i/o dificulten la possibilitat que el simulador/analitzador ICHNAEA funcioni en un dispositiu portàtil Android.

1.3 Estructura de la memòria

A partir d'aquí aquesta memòria s'ha estructurat seguint la següent organització:

Al segon capítol fem un anàlisi de requeriments funcionals i no funcionals del projecte.

Al tercer capítol es fa l'especificació del projecte, enumerant i descrivint els diversos casos d'ús d'aquest. Aquests s'han subdividit en dues parts, una per cada "subaplicació" desenvolupada.

El quart capítol tracta de com s'ha implementat el projecte, les tecnologies emprades i el disseny que s'ha tingut en compte. S'ha dividit aquest en dues parts que cobreixen el desenvolupament del servidor per una banda i la del client mòbil (Android) per l'altra.

Tenim un cinquè capítol amb la planificació del projecte i l'anàlisi de costos d'aquest.

Un sisè capítol de conclusions i possibles millores del projecte.

A més hi ha un setè apartat amb la bibliografia emprada o destacable.

Finalment tenim dos annexes, on en el primer hi haurà el manual d'usuari de l'aplicació, que cobrirà el client mòbil i l'entorn de gestió per web i un altre annex on hi haurà unes indicacions sobre com instal·lar tot el projecte.

2. Anàlisi de requeriments

2.1 Requeriments funcionals

Són aquests els requeriments que la nostra aplicació haurà de complir i que no depenen de la tecnologia a usar:

- Alta i baixa d'usuaris i edició d'aquests
- Crear projectes
- Crear tasques que s'executaran concurrentment en el servidor, concretament del tipus test de predicció de l'ICHNAEA
- Obtenir llistats de les tasques de l'usuari, extreure'n els resultats i permetre guardar-los localment.
- Carregar des de la memòria del dispositiu matrius de dades i també tenir la possibilitat de guardar-les.
- Editar des de l'aplicació les dades (matrius) que es passin al servidor, inclou modificar valors i eliminar files i/o columnes.
- Configurar l'entorn

2.2 Requeriments no funcionals

A continuació tenim els requeriments no funcionals que ha de tenir el projecte, o sigui els que no depenen del què fa sinó del com ho fa. Aquests els dividim en diferents categories:

Usabilitat

Les aplicacions desenvolupades han de ser fàcilment usables, no induir a error en el seu ús i que els problemes que es donin o errors que mostri siguin entenedors i clars. A més s'ha de poder veure correctament en qualsevol dispositiu, sigui quina sigui la mida de la seva pantalla, versió de sistema operatiu o navegador (en el cas de l'entorn web).

Rendiment

El rendiment ha de ser acceptable, essent els principals colls d'ampolla l'accés a la xarxa i en alguns casos concrets el disposar de terminals poc potents. En tot cas no ha de ser acceptable un temps d'espera de més de $2 - 3$ segons.

Seguretat i integritat

S'ha de prevenir l'accés no autoritzat al sistema, la qual cosa ja es té en compte amb la contrasenya per accedir-hi, però també cal tenir en compte possibles atacs al sistema intentant enviar paràmetres incorrectes o especialment formats per a tenir accés a àrees o serveis restringits.

Mantenibilitat

S'ha procurat tenir un codi net, ben estructurat i fàcil de mantenir, amb comentaris on facin falta, amb noms de funcions i classes clars i no ambigus, que indiquin clarament què fan.

Reusabilitat

El codi d'aquest projecte ha d'estar pensat per tal que sigui reusable i a la vegada ampliable, de fet en el disseny intern del servidor s'ha tingut en compte la possible ampliació de funcionalitats.

Interoperatibilitat

Per la pròpia arquitectura emprada en el projecte, aquest ja és de per sí compatible amb diverses configuracions de software/hardware, heterogènies. Com veurem en els *Requisits software*, ha de tenir possibilitat de funcionar sense problemes amb clients mòbils d'una àmplia gama de versions i per la part servidora funcionar sobre sistemes operatius de diferents fabricants.

Concurrència

Aquest projecte ha de donar servei a diversos usuaris de forma simultània, per la qual cosa cal que les peticions dels clients es puguin executar de forma paral·lela, sense interferir-se ni bloquejar el servidor.

Requisits hardware

Tenim dues vessants de cara al hardware on ha de funcionar, per una banda el servidor, que degut a la possibilitat d'accés concurrent de diversos usuaris que llencin execucions des dels seus clients, necessitarem una màquina de potència mitjana. Un sistema Intel Core2Duo és suficient per a càrregues amb pocs usuaris.
Per altra banda tenim el terminal mòbil on correrà l'aplicació client. En principi qualsevol aparell amb el sistema operatiu Android 2.3 ja serveix.
Tan el client com el servidor han de tenir connexió a Internet.

Requisits software

El client Android, com s'ha comentat, ha de tenir la versió 2.3 o superior del sistema operatiu.

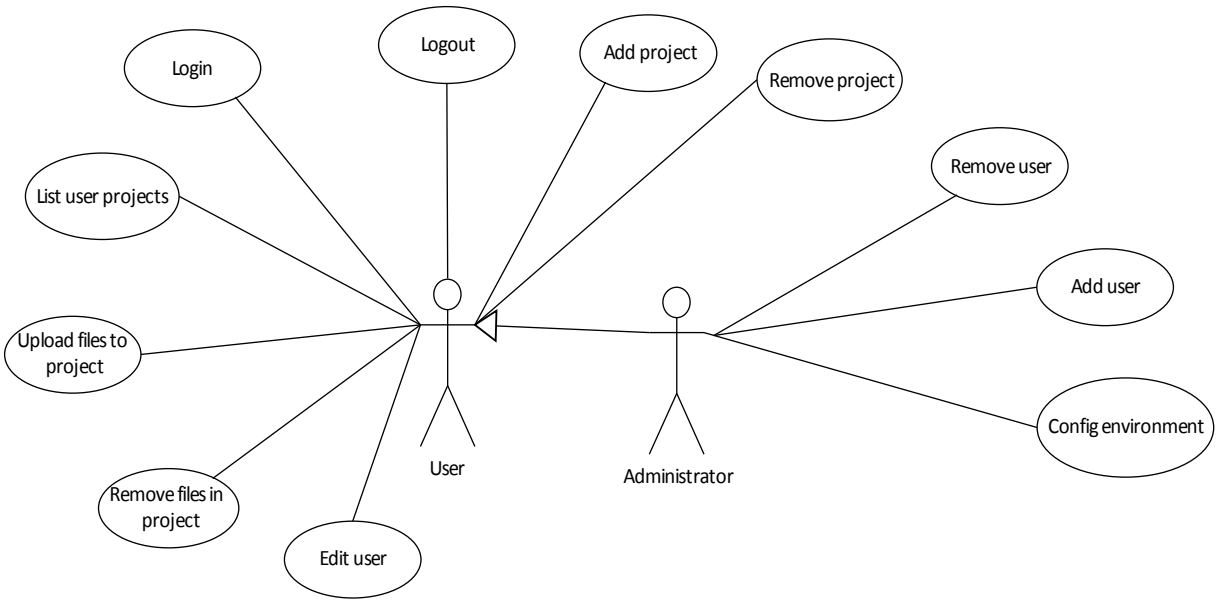
El servidor pot ser una màquina Windows Server (les versions workstation no són aptes per una limitació en el nombre màxim de connexions simultànies) o Unix (Linux, MacOS X, AIX, etc.). Han d'instal·lar-se el gestor de bases de dades MySQL, almenys la versió 5, el servidor d'aplicacions GlassFish 3.1. Una JVM també és necessària, però en principi ja s'instal·la junt al GlassFish.

3. Especificació

Diagrama de casos d'ús

Aquí mostrem el diagrama de casos d'ús del projecte. Podem veure que hi ha dos diagrames, ja que separo els casos d'ús del client web del client per a dispositius mòbils al ser dues aplicacions bastant diferents i tot i que hi ha casos d'ús semblants o equivalents, aquests es gestionen

3.1 Entorn web



Comencem amb la descripció dels casos d'ús de l'entorn web.

Cas d'ús	Identificar-se al sistema (Login)	
Actors	Usuari, Administrador	
Descripció	Identifica a un individu com a usuari legítim del sistema	
Flux normal	**Curs d'esdeveniments**	**Resposta del sistema**
	1. Un usuari indica el seu usuari i contrasenya al sistema.	2. Es valida l'usuari i se li permet l'accés al sistema, creant una sessió per aquest. 3. Si l'usuari és un administrador se l'envia a la pantalla de selecció d'usuaris, si és un usuari normal a la pantalla d'edició del seu propi usuari.
Alternativa	2. error si l'usuari no existeix, la contrasenya és incorrecta o l'usuari està deshabilitat	

Cas d'ús	Sortir del sistema (Logout)	
Actors	Usuari, Administrador	
Descripció	Un individu indica que vol sortir del sistema i per tant s'elimina la sessió associada a aquest.	
Flux normal	**Curs d'esdeveniments**	**Resposta del sistema**
	1. Un usuari indica que vol sortir del sistema.	2. Es destrueix la seva sessió i es motra la pantalla de Login
Alternativa		

Cas d'ús	Afegir un usuari (Add user)	
Actors	Administrador	
Descripció	Un administrador vol donar d'alta un nou usuari (administrador o no) per a què pugui treballar al sistema.	
Flux normal	**Curs d'esdeveniments**	**Resposta del sistema**
	1. Un administrador indica que vol afegir un usuari.	
		2. Se li presenta una pantalla amb un formulari per introduir les dades del nou usuari.
	3. S'introdueixen el nom d'usuari que tindrà, el nom real i les restriccions necessàries.	
		4. Es comprova que les dades introduïdes són correctes.
		5. Es comprova que no hi hagi un usuari amb el mateix nom.
		6. Es crea l'usuari i el directori corresponent en el sistema de fitxers.
		7. Torna al menú de selecció d'usuaris.
Alternativa	4,5. Si error saltem al punt 3	

Cas d'ús	Eliminar un usuari (Remove user)	
Actors	Administrador	
Descripció	Un administrador esborra un usuari del sistema.	
Flux normal	**Curs d'esdeveniments**	**Resposta del sistema**
	1. Un administrador indica que vol eliminar un usuari.	2. Es comprova que aquest usuari existeix i si té tasques executant-se.
	3. Es demana confirmació de l'eliminació.	4. S'esborra l'usuari, tots els seus projectes i tasques i totes les dades associades. 5. Anem a la pantalla de selecció d'usuaris.
Alternativa	2. Error: l'usuari no existeix (improbable) o té alguna tasca en execució.	

Cas d'ús	Configurar programa (Configure environment)	
Actors	Administrador	
Descripció	Ens permet definir paràmetres com els directoris on trobarem l'R, on es crearan els dels usuaris i altres característiques del servidor.	
Flux normal	**Curs d'esdeveniments**	**Resposta del sistema**
	1. Un administrador vol canviar algun paràmetre de la configuració del servidor.	2. Es guarden els paràmetres modificats a la configuració
Alternativa		

Cas d'ús	Editar usuari (Edit user)	
Actors	Usuari, Administrador	
Descripció	Permet canviar dades de l'usuari, com el nom, contrasenya o si està habilitat o deshabilitat (això només ho pot fer un administrador).	
Flux normal	**Curs d'esdeveniments**	**Resposta del sistema**
	1. Un individu vol canviar dades associades a un usuari.	
		2. Se li presenta una pantalla amb un formulari per introduir les dades del nou usuari. Es presenta omplert amb les dades actuals que té aquest.
	3. S'introdueixen les modificacions corresponents.	
		4. Es comprova que les dades introduïdes són correctes.
		5. Es guarden les modificacions.
Alternativa	4. Error, tornem a 3.	

Cas d'ús	Afegir projecte a un usuari (Add project)	
Actors	Usuari, Administrador	
Descripció	Un usuari vol afegir un projecte.	
Flux normal	**Curs d'esdeveniments**	**Resposta del sistema**
	1. Un usuari vol donar d'alta un projecte amb el nom indicat.	
		2. Es comprova que no hi hagi un projecte d'aquell usuari amb el mateix nom.
		3. Es crea el projecte i el directori que contindrà totes les dades d'aquest.
		4. Si és el primer projecte de l'usuari, es marca com a projecte per defecte on es realitzaran les simulacions.
Alternativa	2. Error si ja existeix. Anem a 1	

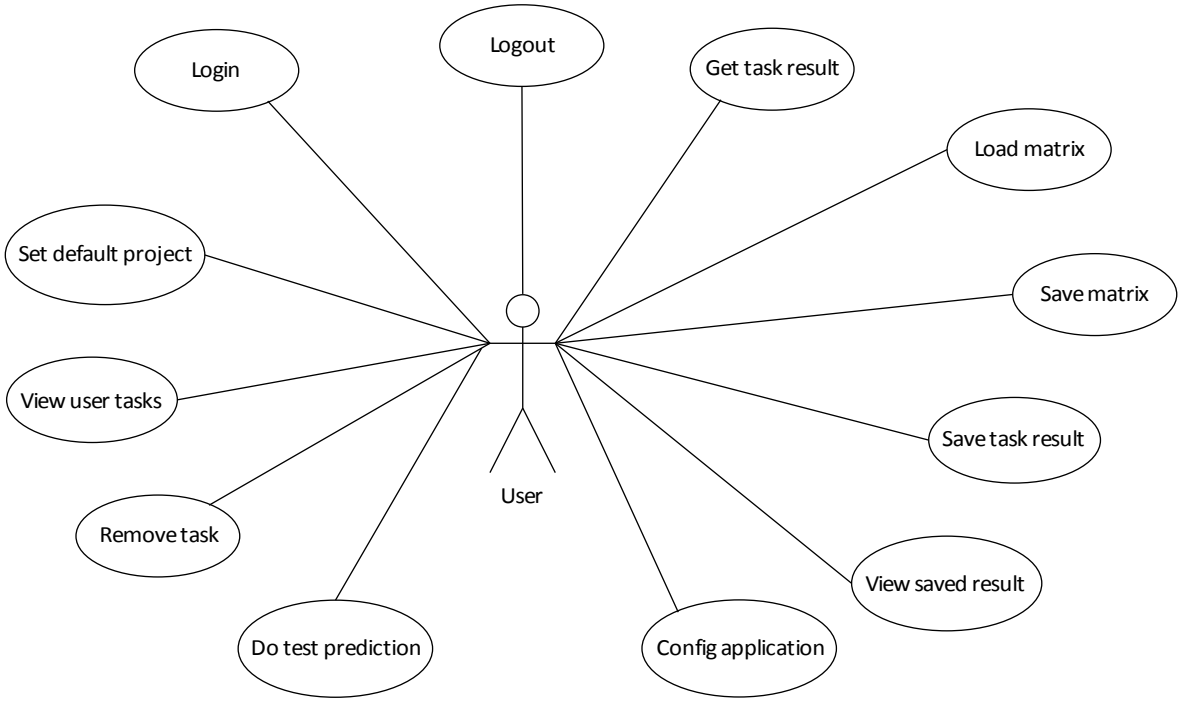
Cas d'ús	Eliminar projecte (Remove project)	
Actors	Usuari, Administrador	
Descripció	Esborrem un projecte i totes les tasques associades	
Flux normal	**Curs d'esdeveniments**	**Resposta del sistema**
	1. Un usuari vol eliminar un projecte.	2. Es comprova que no hi hagi cap tasca engegada per aquell projecte. 3. S'esborra el projecte, les tasques associades i tots els fitxers d'aquestes.
Alternativa	2. Error, no es pot eliminar el projecte.	

Cas d'ús	Pujar fitxer al servidor (Upload file)	
Actors	Usuari, Administrador	
Descripció	Pugem un fitxer, normalment de tipus .Rdata que serveix per a complementar les dades usades en el simulador ICHNAEA.	
Flux normal	**Curs d'esdeveniments**	**Resposta del sistema**
	1. Un usuari vol pujar un fitxer cap al servidor, a un projecte determinat. 2. Es selecciona el fitxer des de disc i es dóna l'ordre de pujar-lo.	3. Es guarda el fitxer al servidor, a l'arrel del projecte al qual pertany. 4. Si el fitxer ja existia, es sobreescriu i no s'afegeix cap entrada nova a la llista de fitxers d'aquell projecte (seria redundant).
Alternativa		

Cas d'ús	Eliminar fitxer d'un projecte (Remove files in project)	
Actors	Usuari, Administrador	
Descripció	Esborrem un fitxer d'un projecte, tant de la llista de fitxers com físicament de disc.	
Flux normal	**Curs d'esdeveniments**	**Resposta del sistema**
	1. Un usuari vol esborrar un fitxer. 2. Marca el fitxer i indica que s'esborri.	3. S'esborra el fitxer i l'entrada d'aquest en la base de dades del sistema.
Alternativa		

Cas d'ús	Marcar com a projecte per defecte (Set default project)	
Actors	Usuari, Administrador	
Descripció	Es marca el projecte indicat com el projecte on es llençaran les simulacions si l'usuari no ho canvia (tan a l'entorn web com en el client mòbil).	
Flux normal	**Curs d'esdeveniments**	**Resposta del sistema**
	1. L'usuari vol establir el seu projecte per defecte. 2. Es marca el que volem que ho sigui.	3. S'apunta el nou projecte com a defecte.
Alternativa		

3.2 Casos d'ús de l'aplicació mòbil



Cas d'ús	Identificar-se al sistema (Login)	
Actors	Usuari	
Descripció	Identifica a un individu com a usuari legítim del sistema	
Flux normal	**Curs d'esdeveniments**	**Resposta del sistema**
	1. Un usuari indica el seu usuari i contrasenya al sistema.	2. Es valida l'usuari cridant al WS login i se li permet l'accés al sistema, creant una sessió per aquest. 3. Es passa a la pantalla principal de l'aplicació
Alternativa	1. Es permet entrar a l'aplicació sense identificar-se, però llavors no es té accés a les funcions remotes (web services) 2. Error, es torna al pas 1.	

Cas d'ús	Sortir del sistema (Logout)	
Actors	Usuari	
Descripció	Un individu indica que vol sortir del sistema i per tant s'elimina la sessió associada a aquest. De fet es crida automàticament al tancar l'aplicació.	
Flux normal	**Curs d'esdeveniments**	**Resposta del sistema**
	1. Un usuari indica que vol sortir del sistema.	2. Es crida al WS logout. 3. Es surt de l'aplicació.
Alternativa		

Cas d'ús	Carregar matriu de treball (Load matrix)	
Actors	Usuari	
Descripció	Es carrega una matriu des d'un fitxer en format .csv per a ser editada en l'àrea de treball.	
Flux normal	**Curs d'esdeveniments**	**Resposta del sistema**
	1. Usuari indica que vol carregar un fitxer. 3. Es selecciona un fitxer.	2. El sistema mostra una finestra per navegar pel sistema de fitxers i seleccionar un fitxer. 4. Es llegeix el fitxer des de disc. 5. Es comproven els valors llegits i es crea una matriu que es mostra.
Alternativa		

Cas d'ús	Guardar matriu de treball (Save matrix)	
Actors	Usuari	
Descripció	Guarda a disc la matriu amb què estem treballant. Només es guarden les files i columnes de la matriuque no estan marcades com a seleccionades.	
Flux normal	**Curs d'esdeveniments**	**Resposta del sistema**
	1. Usuari indica que vol guardar la selecció actual de la matriu.	
		2. El sistema mostra una finestra per navegar pel sistema de fitxers, triar un directori i indicar el nom del fitxer a guardar.
	3. Es selecciona un directori i triem un nom pel fitxer.	
		4. Es guarda la submatriu actual.
Alternativa		

Cas d'ús	Configurar aplicació (Configure application)	
Actors	Usuari	
Descripció	Permet indicar i guardar paràmetres de funcionament de l'aplicació, com el servidor remot d'ICHANEA, l'usuari i contrasenya per defecte.	
Flux normal	**Curs d'esdeveniments**	**Resposta del sistema**
	1. Usuari vol canviar la configuració de l'aplicació.	
		2. Es mostra una pantalla per a canviar els paràmetres de funcionament.
	3. S'introdueix la nova configuració	
		4. Es guarda la informació introduïda a la memòria del terminal.
Alternativa		

Cas d'ús	Marcar com a projecte per defecte (Set default project)	
Actors	Usuari	
Descripció	Es marca el projecte indicat com el projecte on es llençaran les simulacions..	
Flux normal	**Curs d'esdeveniments**	**Resposta del sistema**
	1. S'indica que es vol canviar el projecte per defecte.	
		2. Es mostra una pantalla amb la llista de projectes de l'usuari.
	3. Es selecciona un nou projecte.	
		4. El sistema demana confirmació de l'acció.
	5. Es respon afirmativament.	
		6. Es crida el WS corresponent per a canviar el projecte per defecte.
Alternativa		

Cas d'ús	Visualitzar tasques de l'usuari (View user tasks)	
Actors	Usuari	
Descripció	Permet veure la llista de tasques de l'usuari, diferenciant les finalitzades de les que estan corrent i permet seleccionar-les.	
Flux normal	**Curs d'esdeveniments**	**Resposta del sistema**
	1. L'Usuari vol veure la llista de tasques.	
		2. El sistema crida al WS corresponent per obtenir la llista de tasques de l'usuari.
		3. Es mostra la pantalla amb les tasques.
Alternativa		

Cas d'ús	Realitzar test de predicció (Do test prediction)	
Actors	Usuari	
Descripció	Envia la matriu actual al servidor ICHNAEA per tal que faci un Test de Predicció a partir dels valors d'aquesta.	
Flux normal	**Curs d'esdeveniments**	**Resposta del sistema**
	1. S'indica que es vol fer un test de predicció.	2. Es seleccion la submatriu actual que no està seleccionada i s'envia cap al servidor cridant al WS 'test_prediction'. 3. Retornem al sistema l'identificador de la tasca generada.
Alternativa		

Cas d'ús	Obtenir resultat de tasca (Get task result)	
Actors	Usuari	
Descripció	Dóna el resultat de l'execució de la tasca indicada. Aquest cas d'ús pot ser cridat de forma independent i indicant l'última tasca executada (veiem el cas *Do test prediction*) o bé a partir d'una triada per nosaltres (veiem el cas *View user tasks*).	
Flux normal	**Curs d'esdeveniments**	**Resposta del sistema**
	1. Volem veure una tasca	2. El sistema crida al WS per a obtenir els resultats de la tasca, si aquesta ha finalitzat. 3. S'obre una pantalla on podem veure els resultats de l'operació.
Alternativa		

Cas d'ús	Guardar resultat de la tasca (Save task result)	
Actors	Usuari	
Descripció	De la tasca finalitzada que estem visualitzant guardem a disc els resultats de cara a consultar-los posteriorment.	
Flux normal	**Curs d'esdeveniments**	**Resposta del sistema**
	1. Indiquem que volem guardar els resultats.	
		2. El sistema ens mostra una pantalla per seleccionar en quin directori ho guardem i amb quin nom.
	3. Indiquem el nom del fitxer a guardar.	
		4. Es guarden les dades indicades un fitxer de text.
Alternativa		

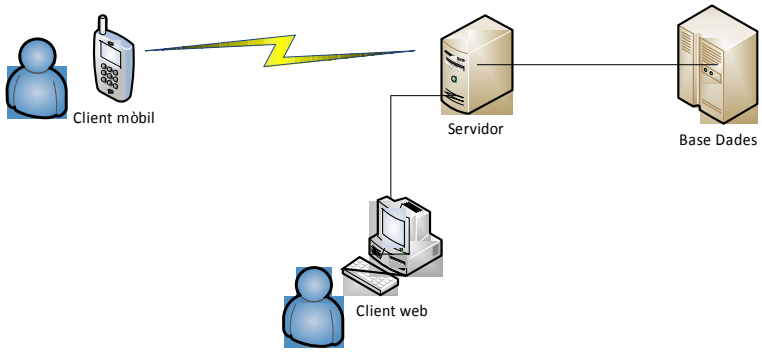
Cas d'ús	Visualitzar fitxer de resultats (View saved result)	
Actors	Usuari	
Descripció	Carrega un fitxer de resultats guardat prèviament i el mostra per pantalla.	
Flux normal	**Curs d'esdeveniments**	**Resposta del sistema**
	1. Usuari vol veure un fitxer de resultats.	
		2. Se'ns mostra la pantalla per seleccionar un fitxer amb els resultats.
	3. Es tria el fitxer a obrir.	
		4. S'obre el fitxer i se'n llegeix el contingut.
		5. Es mostren els resultats en una nova pantalla.
Alternativa		

Cas d'ús	Eliminar tasca (Remove task)	
Actors	Usuari	
Descripció	Des de la pantalla de Get task result triem d'eliminar la tasca finalitzada, perdent així també els resultats obtinguts (si no els hem guardat prèviament).	
Flux normal	**Curs d'esdeveniments**	**Resposta del sistema**
	1. Usuari vol eliminar una tasca.	
		2. Se'ns demana confirmació.
	3. Es dóna una resposta afirmativa.	
		4. Es crida al WS corresponent per eliminar la tasca.
Alternativa		

4. Disseny i implementació

En el següents apartats es descriuen quines tecnologies s'han usat en el projecte, quines decisions s'han pres i possibles alternatives a aquestes.

Degut a què el projecte té dues parts molt diferenciades i en les que s'usen unes tecnologies molt diferents, aquest capítol del disseny i implementació s'ha dividit en dos blocs: un dedicat al servidor i l'aplicació d'usuari associada i un altre bloc pel client mòbil.
De fet el projecte desenvolupat, un cop finalitzat té aquesta estructura de funcionament:



En la qual ha calgut dissenyar i implementar totes les instàncies que hi apareixen.

4.1 Servidor

4.1.1 Introducció

En el sentit més estricte del terme, el servidor consisteix en una part fonamental de l'aplicació desenvolupada, sense aquesta el client no té sentit, ja que

En els propers apartats veurem com s'ha desenvolupat. Aquests he optat per dividir-ho en les següents seccions:

- Implementació del servidor : una vista per les consideracions de disseny que s'han tingut en compte i les classes més destacades
- Servidor d'aplicacions
- Web Services
- Base de dades
- Integració amb ICHNAEA
- Entorn de gestió web

Tot i que el projecte s'ha desenvolupat usant una única màquina com a servidor, no és gens complicat dividir-lo entre diversos sistemes:

- L'aplicació desenvolupada en Java que defineix els web services i que de fet és el nucli del projecte i que corre sobre GlassFish.
- La base de dades pot estar perfectament en una altra màquina, ja que com veurem, usem MySQL i a aquest s'hi accedeix per xarxa.
- Els fitxers de dades poden també estar muntats per xarxa, per SMB en Windows o NFS en Unix.

Fins i tot hi ha combinacions més distribuïdes encara. No costaria gaire modificar el codi font per a què les simulacions s'executessin en CPUs alienes dins un cluster de màquines.

Amb tot, en un moment donat es va valorar la possibilitat que les rutines d'ICHNAEA s'executessin en el mateix dispositiu mòbil [2], ja que es donaven les següents circumstàncies:

- Android no deixa de ser una capa de presentació i una API que corre damunt d'un nucli Linux.
- De l'intèrpret d'R se n'ofereix tot el codi font per a poder ser compilat on es necessiti.

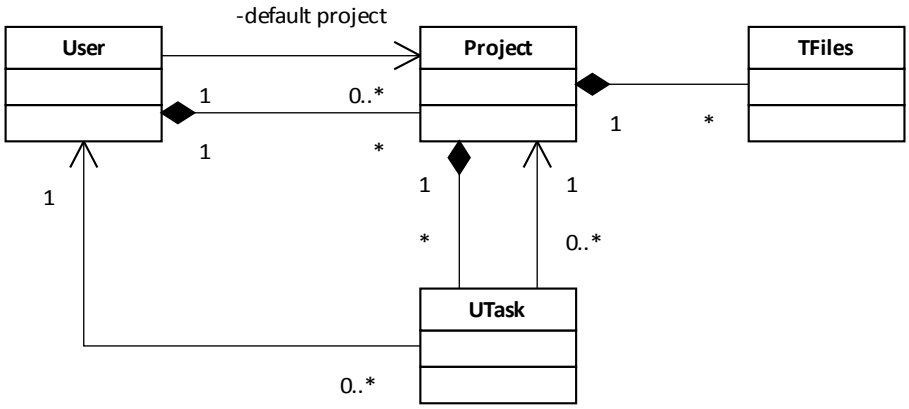
Això que en principi semblava interessant, es complica pel fet que per compilar, però sobretot per executar una versió compilada de l'R en un dispositiu Android, ara per ara cal que el dispositiu tingui l'usuari *root* d'aquest desbloquejat, cosa que no és habitual i l'intentar fer-ho en alguns casos pot donar problemes a part de la pèrdua de la garantia del fabricant. A part que l'aplicació es realitza per a fer la vida més fàcil a l'usuari, no per complicar-li enormement la instal·lació d'aquesta.
A més, hi ha el fet que en general els dispositius tenen uns recursos limitats, quant a memòria com en potència de càlcul. Si bé en el punt que hem deixar l'aplicació tampoc emprarem una potència inusual, en una possible ampliació ja es faria patent la manca d'aquesta.

4.1.2 Disseny i implementació del servidor

Classes

Fins ara hem descrit els diferents components que conformen el servidor amb petits exemples de codi. Tot seguit es mostrarà com s'ha implementat, quines classes hi ha, què fan i com s'integra tot.

Comencem pel diagrama de classes del servidor.



De forma resumida a l'esquema anterior veiem com tenim una classe User on hi guardem la informació d'un usuari, aquest usuari al seu torn té diversos projectes (classe Project), els quals tenen associats una col·lecció de fitxers que s'usaran al cridar a l'ICHNAEA (a la classe Tfiles). Per cada tasca que s'executi associada a un projecte, tenim la classe Utask.

El funcionament d'aquest esquema és simple: un usuari pot crear-se diferents projectes, els quals treballen amb diferents dades i normalment tenen a veure amb diferents llocs d'on s'han obtingut mostres. Quan es crea un projecte, hi ha la possibilitat de pujar-hi fitxers de dades precalculades mitjançant la versió actual de l'R. Aquests són fitxers d'extensió .Rdata i són els que usa aquest projecte a l'hora de cridar la simulació del Test de Predicció. Com s'ha dit al paràgraf anterior, cada

execució d'una simulació crearà una tasca nova, amb el propòsit que quan hagi acabat puguem recollir-ne els resultats.

Aquestes són doncs les principals classes del sistema i que estructuren la forma com es gestionen les dades. Ara bé, hi ha una sèrie de classes auxiliars no relacionades amb aquest esquema, però sí amb l'aplicació que enumerem tot seguit i que són les que interactuen i realitzen la funció principal del servidor:

DBMan : l'encarregada de fer d'intermediari entre les dades i l'aplicació. Té dues funcions principals, per una banda la de facilitar la gestió dels diferents objectes de l'aplicació i per altra banda s'encarrega de gestionar la persistència dins la base de dades d'aquests objectes. Ho veurem amb més detall en l'apartat referent a la Base de Dades.

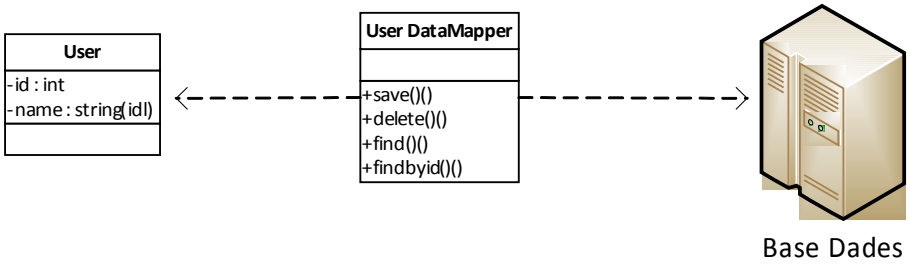
Services : en aquesta hi ha els mètodes que fan de punt d'entrada als webservices, aquesta s'encarrega de gestionar les sessions obertes i cridar els mètodes de DBMan que siguin necessaris, a més de cridar a l'ICHNAEA, d'això últim se n'encarrega la següent classe, RCaller.

RCaller : l'encarregada de crear i preparar els paràmetres que es passaran a l'ICHNAEA dins l'R i finalment executar aquest en un nou fil d'execució de forma asíncrona a les crides de Services.

Persistència

Per a la persistència dels objectes de l'anterior diagrama, s'usa un patró Data Mapper. Aquest es basa en què per cada objecte del sistema que vulguem que sigui persistent (en una base de dades) hi ha un objecte Data Mapper associat que ofereix operacions per a guardar, esborrar i consultar l'objecte original a la base de dades.

Veiem-ho:



Per sort, Java incorpora un API de forma nativa anomenada JPA (Java Persistence API), que veurem en l'apartat de la Base de Dades, i que serveix per fer la mateixa funcionalitat i de forma més fàcil i transparent. Per la qual cosa ja no ens caldrà implementar quelcom semblant.

Operacions públiques oferides com a serveis web

Normalment no enumerariem aquí la llista d'operacions dels serveis web, però crec que és important conèixer-les de cara a comprendre el funcionament intern del servidor:

- login : per identificar-se en el sistema i donar-nos un identificador de sessió
- logout : sortim del sistema i eliminem la sessió assignada.
- test_prediction : la funció que rep una matriu des del client i la passa a l'ICHNAEA per a què la processi.
- test_prediction_get_result : si la tasca que s'ha enviat amb la crida anterior ha acabat, llavors al cridar a aquest servei rebrem una llista de cadenes de caràcter amb la sortida generada per l'ICHNAEA.
- get_task_list : dóna un llistat de les tasques de l'usuari
- get_task_status : indica si la tasca indicada ha acabat o no
- remove_task : esborra la tasca i tots els fitxers associats
- set_default_project : ens permet establir el projecte per defecte. Les tasques sempre s'executaran en el projecte establert per defecte.

- get_default_project : podem canviar el projecte per defecte.
- get_project_list : llista els projectes de l'usuari.

Manteniment de sessions

Hem vist en l'apartat anterior que tenim una funció de *login* i s'ha comentat que aquesta crea una sessió. El què fa després d'això és retornar a l'aplicació client un identificador de sessió que caldrà usar en les següents crides al web service.
Abans d'entrar en detalls sobre com s'ha implementat, és important indicar breument les maneres possibles de realitzar una traçabilitat de les sessions en serveis web:

1. Manualment: l'aplicació és totalment responsable de tota la gestió de la sessió i les dades relacionades. Té la característica que el servidor ha de donar un identificador al client que aquest haurà d'usar en cada cride posterior.
2. Fent ús de les capçaleres HTTP per a guardar la cookie que ens envia el servidor i trametre-la posteriorment en cada crida als web services.

3. Mitjançant anotacions al web service (@HttpSessionScope @Stateful) de tal manera que JAX-WS (ho veurem més endavant) se n'encarregui automàticament. És la més neta i fàcil d'implantar.

Tot i que en principi semblaria que el primer mètode és el més laboriós i que de fet sembla que estiguem "reinventant la roda", el fet és que les alternatives han donat més problemes dels esperats:

- la segona si bé calia afegir lògica de control sobre la sessió HTTP, fer això al servidor no era problema, però sí al client, com veurem.

- la tercera està poc implantada encara i hi ha sistemes que no la suporten correctament. De fet, en les proves que he realitzat, sobre GlassFish li provocava inestabilitat.

De totes maneres l'elecció de la primera opció és deguda a què el client que usem no és compatible amb JAX-WS i per tant, la realització de les crides al servidor i la recepció de resposta s'ha de fer a baix nivell fent crides a funcions SOAP. És una limitació que tenim en Android i que veurem a l'apartat corresponent quan discutim sobre el client.

A nivell tècnic el manteniment de la sessió es fa tenint una llista d'objecte *Session*, un per cada sessió oberta. Quan un usuari s'identifica i aquesta és correcta, es crea una nova sessió, si en tenia una d'oberta s'elimina. A la sessió es guarda informació d'aquest usuari (objecte *User*) i a més l'adreça IP d'aquest, per evitar un possible segrest de la sessió, a més es genera un identificador numèric únic a partir de totes aquestes dades que es retorna a l'usuari en format *long* . Aquest identificador l'haurà d'usar el client en cada crida que faci a qualsevol servei web.

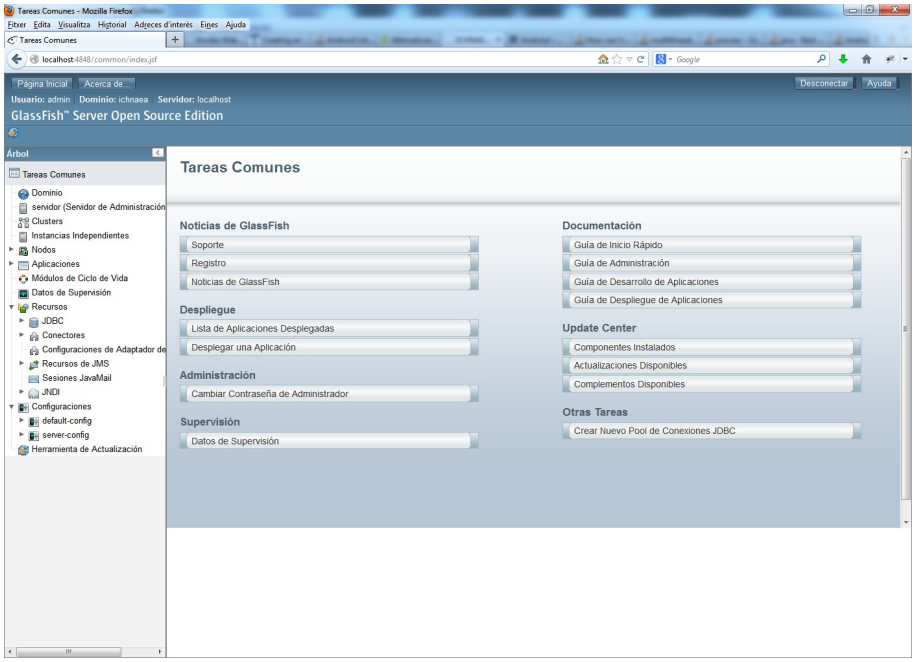
Estructura de directoris al servidor

Al servidor, caldrà tenir-hi un directori dedicat on guardar la informació de l'aplicació. Suposem que estigui en */home/ichnaea*, així dins d'aquest hi tindrem un directori per a cada usuari, per exemple: */home/ichnaea/joanc*.
Dins de cada directori d'usuari n'hi tenim un altre per cada projecte, el nom del directori en aquest cas serà "*pnnn*" on '*nnn*' serà l'identificador del projecte en forma numèrica. Els fitxers que es pugin mitjançant l'entorn web es deixaran en aquest directori.
Al directori d'un projecte n'hi trobarem uns altres que correspondran a cada tasca que s'hagi iniciat en aquell projecte. Dins d'aquests hi trobarem els fitxers *test.csv* i

output.out i *output.err*, que corresponen a l'entrada de dades per la tasca i les sortides que ha donat l'ICHNAEA per aquestes.
En tot cas els usuaris no tindran mai accés a aquesta estructura de directoris (o no seria desitjable).

4.1.3 Servidor d'aplicacions

El servidor d'aplicacions és el nucli de gestió del servidor. És qui posa en relació l'aplicació desenvolupada amb els clients, la base de dades i li dóna l'entorn d'execució. En el nostre cas s'ha optat per usar el servidor Oracle GlassFish en la seva variant OpenSource. Aquesta és una aplicació totalment desenvolupada en el llenguatge Java i per tant és multiplataforma. El sistema de configuració d'aquesta es basa en un entorn web que per defecte escolta en el port 4848 de la màquina on s'instal·li com veiem tot seguit:



Aquest servidor pot gestionar qualsevol tipus d'aplicació Java EE (Enterprise Edition) i tots els components d'aquestes, com Enterprise JavaBeans, JPA, Java Server Pages, RMI, etc.

Tot i que no és l'únic servidor capaç d'oferir aquestes funcionalitats, s'ha escollit aquest per estar molt ben integrat amb les funcionalitats que necessitem, al ser desenvolupat en col·laboració directa dels mateixos desenvolupadors del Java, és el

més actualitzat i per la seva facilitat de configuració. De totes maneres el projecte es podria desplegar en altres servidors sense massa complicació, com podrien ser Apache Axis2, IBM WebSphere o JBoss.

De fet la instal·lació d'una aplicació sobre el servidor es basa en desplegar aquesta a partir d'un fitxer de tipus *.war*, en el qual hi ha empaquetat tota una estructura de directoris amb tots els components de l'aplicació, des dels fitxers *.class* de la lògia del programa, els *.xml* de configuració de components, com JPA, fins als *.jsp* de la pàgina web.

4.1.4 Web Services

Anteriorment hem vist quines funcions ofereix l'aplicació mitjançant serveis web (web services, o WS). Aqui tractarem el què són, el seu funcionament intern i com implementar-los.
Tradicionalment la crida a funcions en una aplicació que no estiguessin en la mateixa màquina es feia mitjançant RPC (Remote Procedure Call), el qual és un estàndard que comunica codi aplicacions mitjançant la xarxa.
Recentment, amb l'aparició dels estàndards web, RPC s'ha implementat sobre d'aquests. Això ha comportat uns grans avantatges sobretot pel fet que ens permet oferir serveis a nivell global mitjançant la web. Cal tenir en compte que moltes organitzacions tenen limitat l'accés a Internet només a aplicacions que corrin en ports estàndard com el 80 d'HTTP, fent que l'accés a serveis sobre RPC no estiguessin disponibles per tothom. Una altra característica interessant és que els web services són independents del llenguatge de programació, o sigui, que poden no tenir res a veure el llenguatge amb que s'han implementat al servidor respecte el que els crida des del client.

Així trobem els protocols SOAP (Simple Object Access Protocol) i REST (Representational state transfer). Qualsevol d'aquests encapsulen les crides i missatges en el format XML, el què fa que a més siguin interoperables entre diferents arquitectures i llenguatges de programació diferents.

Com que interactuar directament amb aquests protocols pot ser pesat i farragós, podem disposar de capes d'abstracció que ho facin més transparent. En el nostre cas hem usat JAX-WS en el servidor.

JAX-WS

Aquesta és una API de Java per a implementar web services. Per al seu ús es fan servir anotacions, com podem veure:

```
@WebService
public class Services {

	....

	@WebMethod
	public long login(
			@WebParam(name="user") String user,
			@WebParam(name="pwd") String pwd)
	{
	....
```

Usant JAX-WS aconseguim no haver de tractar amb el protocol a baix nivell (en el nostre cas SOAP) que estiguem usant, facilitant així la programació dels serveis. Amb JAX-WS i les utilitats incorporades, es generen automàticament les classes necessàries per tal d'interactuar amb el protocol que corre per sota sense que el programador necessiti tenir-ne coneixements.

SOAP (Service Oriented Application Protocol)

Aquest és un protocol per a intercanviar informació de forma estructurada usant XML, pensant precisament en la implementació de serveis web. Aquest protocol normalment el trobem per damunt d'altres com SMTP o HTTP, que els pot usar per a establir les connexions.

Veiem un exemple del flux d'informació que s'intercanvien el client i el servidor: Usarem una crida senzilla, la de login, que simplement rep dos paràmetres (usuari i contrasenya) i retorna un *long* amb l'identificador de sessió o un valor negatiu si hi ha hagut algun error:

Crida del client:

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
    <S:Header/>
    <S:Body>
        <ns2:login xmlns:ns2="http://iserver.ichnaea.com/">
            <user>perico</user>
            <pwd>ofthesticks</pwd>
        </ns2:login>
    </S:Body>
</S:Envelope>
```

Resposta del servidor:

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
    <S:Body>
        <ns2:loginResponse xmlns:ns2="http://iserver.ichnaea.com/">
            <return>43500837299110</return>
        </ns2:loginResponse>
    </S:Body>
</S:Envelope>
```

Fixem-nos en els fragments marcats en negreta, on podem veure el nom de la funció, el nom dels paràmetres i el resultat retornat.

4.1.5 Base de dades

En aquest projecte usem una base de dades per tal de mantenir informació referent als usuaris, les dades amb què treballen i el què anomenem persistència dels objectes.
En el mercat hi ha múltiples bases de dades que ens podrien servir per aquests propòsits, en el nostre cas hem triat MySQL, en part per ser un dels sistemes més estesos, té una potència relativament elevada i en principi és fàcil d'instal·lar i configurar. Val a dir que es podria haver usat qualsevol altre sistema gestor de base de dades (SGBD) i de fet, canviar-ho seria totalment transparent a l'aplicació.

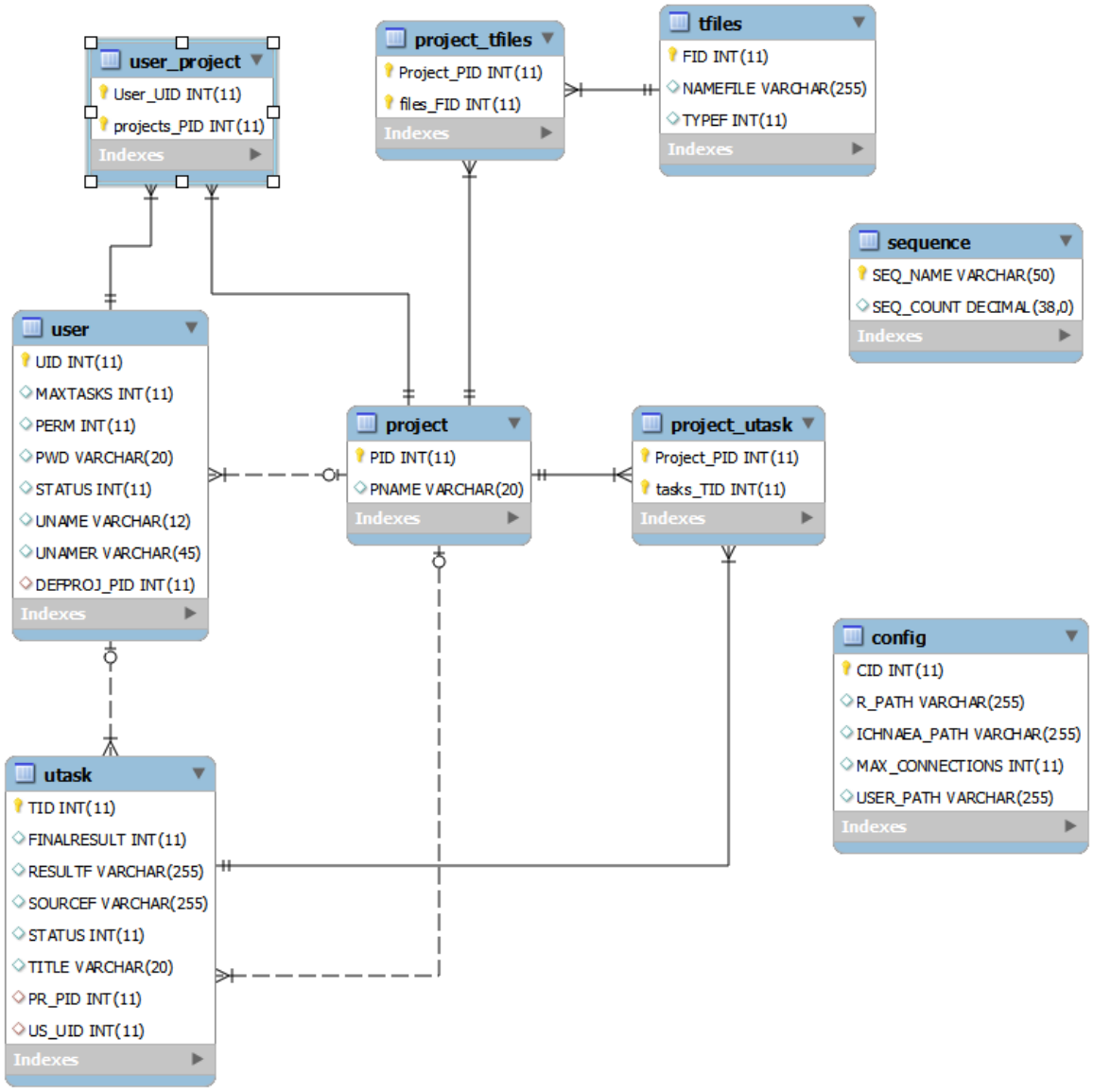
MySQL

MySQL és un sistema gestor de bases de dades relacional, multifil (thread) i multiusuari. Aquest component és el que físicament guarda les dades a disc, les organitza i manté la integritat d'aquestes. De tot aquest apartat, i com ja s'ha dit, aquest és l'únic component que podríem substituir per un altre producte de forma transparent.

JDBC

Aquesta és una API de Java (de fet vol dir Java Database Connectivity) per a accedir a bases de dades relacionals. Està pensada per a fer d'intermediari entre la BD i el codi dels programes, de forma que aquests no hagin de tractar amb les peculiaritats pròpies de cada SGBD.

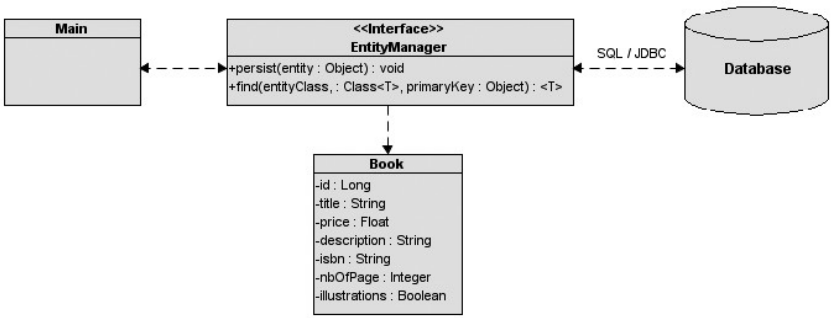
En aquest diagrama es mostra l'estructura de la base de dades. Aquest diagrama ha estat generat mitjançant l'eina *Reverse Engineer* del MySQL Workbench. A destacar que la taula *sequence* la genera automàticament el sistema per a mantenir una traçabilitat dels identificadors de claus primàries donades de forma automàtica.



Arribats a aquest punt, i de cara a simplificar el desenvolupament i simplificar l'accés a la base de dades, tot i que pel que hem vist fins ara podem

JPA

Java Persistence API és una definició d'un *framework* per a Java que té el propòsit d'encapsular les crides a la base de dades que normalment faríem per mitjà de JDBC i fer-les independents d'aquesta. Fixem-nos en aquest esquema i veiem com de fet implementa un patró DataMapper:



Hi ha múltiples implementacions d'aquest *framework*, on a més d'altres, podem destacar:

- Hibernate : és una de les més avançades, tot i que originalment és anterior a JPA, a partir de la versió 3.2 l'ha implementat.
- EclipseLink : és la que s'ha usat, es considera que és la més fidel a l'estàndard. Ja que a més deriva directament d'Oracle TopLink.
- OpenJPA

De fet, degut a la senzillesa de la base de dades i que no rebrà excessives consultes, qualsevol hauria estat perfectament vàlida per aquest projecte.

JPA s'usa aplicant anotacions als atributs de les classes Java que volem que siguin persistents, tot i que també ho podem fer indicant-ho al fitxer *persistence.xml* com veurem més endavant.

Per a fer una classe persistent (que serà guardada a la B.D.) ho fem de la següent manera:

```
@Entity
public class User {
	@Id
	@GeneratedValue(strategy = GenerationType.AUTO)
	private int uid;
	@Basic
	@Column(length = 12)
	private String uname;	// system user name
	@Basic
	@Column(length = 45)
	private String unamer;	// real name of user
	@Basic
	@Column(length = 20)
	private String pwd;
	private int perm;  //permissions : 0 : administrator, >0 : regular user
	private int status; // 1 : active, 0 : disabled
	private int maxtasks;
	private Project defproj;
	@OneToMany(cascade = CascadeType.ALL, orphanRemoval=true)
	private List<Project> projects;
	....
```

Aquest és un fragment de la classe User on hi guardem la informació referent a cada usuari del sistema. Al principi de tot, l'anotació *@Entity* indica que aquesta serà una classe que es farà persistent a la base de dades. Si no s'hi indica cap paràmetre es suposa que la taula tindrà el mateix nom que la classe. En tot el projecte les taules de la BD sempre han tingut el mateix nom que les classes.
Les altres anotacions destacades són:

@Id : indica la clau primària de la taula

@Column(length = xx) : ens permet establir característiques d'aquell atribut, en aquest cas ho usem per indicar la longitud màxima d'un String a la BD.

@OneToMany(cascade = CascadeType.ALL, orphanRemoval=true) : quan tenim una relació d'un a molts a la BD, amb la característica que quan afegim o eliminem algun element a la llista a la què fa referència, es propagui en

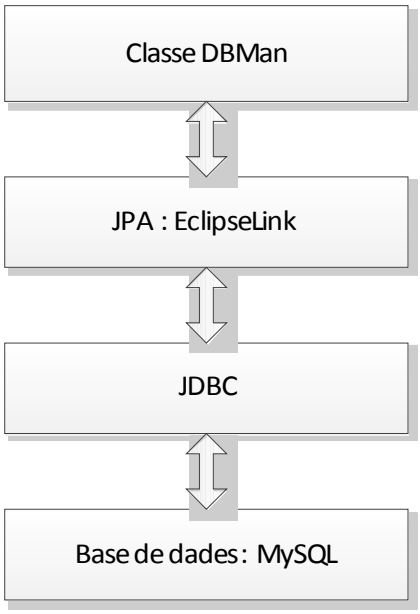
cascada. OrphanRemoval indica que si queden elements de la relació penjats després d'una eliminació, que els elimini en cascada també.

A l'hora de fer operacions que impliquin la persistència dels objectes, s'implementa mitjançant crides a les operacions d'un objecte de la classe EntityManager. Podem destacar els operacions persist(), remove(), find() o createQuery() que respectivament fan les funcions de guardar un objecte, esborrar-lo, trobar-ne un segons la clau primària donada i realitzar una consulta personalitzada. En tot cas al final tot això ho fem de forma transparent mitjançant la classe DBMan que és la que realitza les crides a les operacions indicades.

Com s'ha comentat, si volem fer el codi més net i portable, sobretot si no volem usar anotacions (que no és el nostre cas), llavors caldria indicar la informació que proporcionen aquestes dins el fitxer *persistence.xml*.
En aquest fitxer, a més hi ha la informació necessària per a connectar a la base de dades que es faci servir. Això és important, ja que si volem canviar per exemple la base de dades, ho hem d'indicar en el fitxer en qüestió. Aquest fitxer el trobarem en el servidor, on estigui desplegada l'aplicació, al subdirectori \WEB-INF\classes\META-INF :

```
<property name="javax.persistence.jdbc.url" value="jdbc:mysql://localhost:3306/ichnaea"/>
<property name="javax.persistence.jdbc.driver" value="com.mysql.jdbc.Driver" />
<property name="javax.persistence.jdbc.user" value="ichnaea" />
<property name="javax.persistence.jdbc.password" value="nobodycanseemypasswordhohoho" />
```

Com a resum, en aquest diagrama es mostra com interactua tot el que s'ha descrit:



4.1.6 Integració amb ICHNAEA

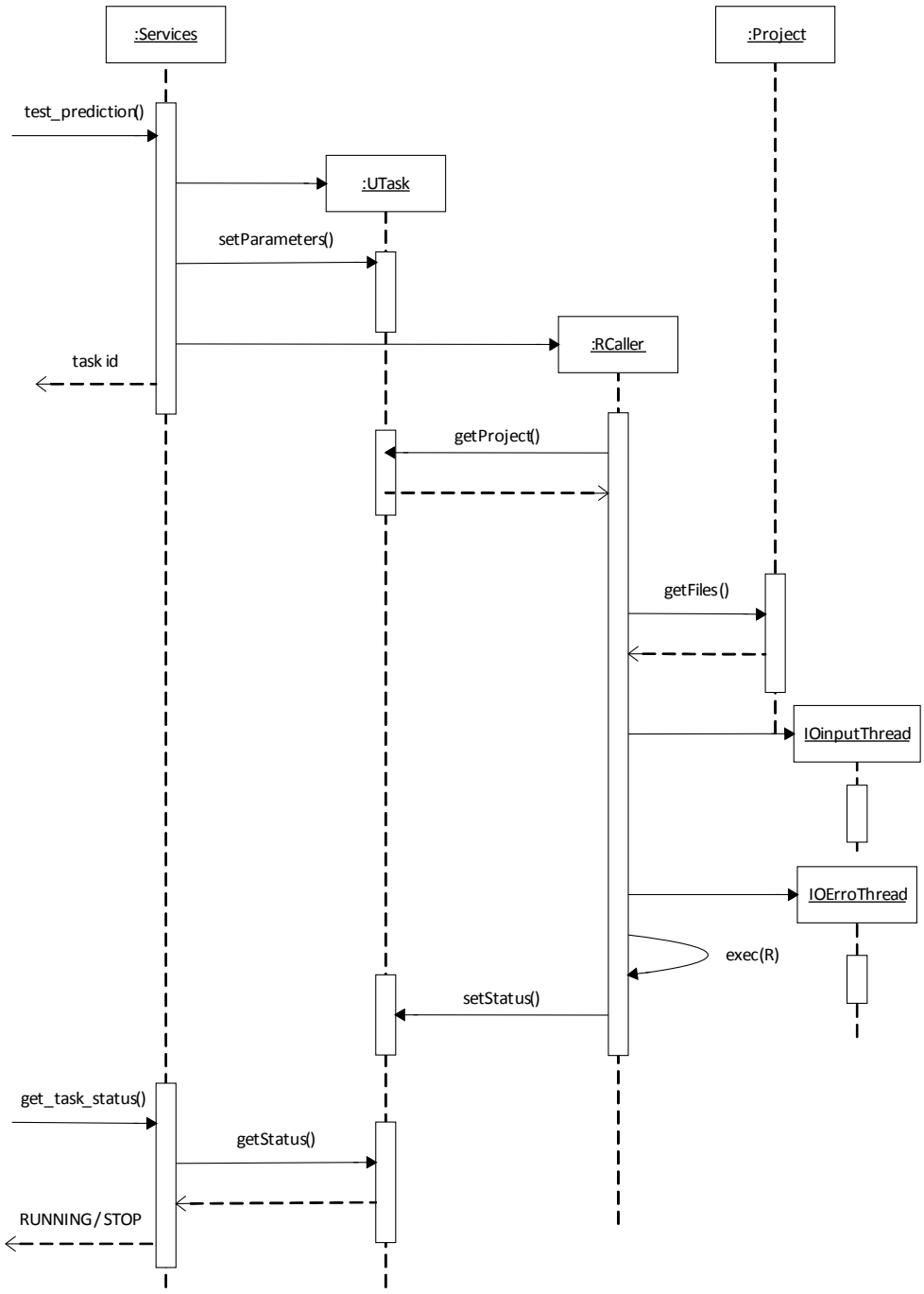
Fins ara s'ha vist quines tecnologies hem usat i com s'usen de cara a les tasques que podríem considerar "secundàries" a l'obtenció de resultats pràctics. Aquí veurem què fa l'aplicació quan ha de fer ús de l'ICHNAEA i com s'han hagut de modificar scripts d'aquest per tal que el pas de paràmetres sigui més fàcil.

Crida a ICHNAEA

La millor manera de veure com el servidor crida a l'ICHNAEA que veient en detall com funciona el servei web '*test_prediction*': aquest, després de fer les habituals comprovacions de la sessió i de correctesa de la matriu rebuda fa les següents accions:

1. Crea una nova tasca a la que li assigna els paràmetres corresponents als fitxers d'entrada i sortida que s'usaran.
2. Es crea una instància de RCaller, la qual engega un nou fil d'execució asíncronament a la crida original de la classe Services.
3. Mentrestant *test_prediction* retorna amb el valor de l'identificador de la tasca creada.
4. RCaller obté la llista de fitxers associats al projecte i crea un fitxer interpretable per R on hi ha les crides necessàries per tal que aquest carregui en memòria els fitxers.
5. Es creen dues instàncies de nous fils d'execució (IOparserThread) per tal de recollir la sortida estàndard i la d'error de l'execució de l'R (ICHNAEA) i així obtenir els resultats o els errors que es podran passar al client quan aquest ho demani.
6. S'indica a la tasca que l'execució ha acabat i que per tant, els clients poden recollir resultats.

Tot això ho veiem en el següent diagrama de seqüència, s'hi ha obviat algun pas, com el de la lectura i gravació de resultats a disc.



Modificacions a ICHNAEA

Ha calgut adaptar un dels fitxers .r amb el codi d'ICHNAEA per tal de fer que treballi amb dades dinàmiques. Fins ara calia modificar manualment el fitxer per tal que carregués la matrius que calia usar. Ara en canvi es passen com a paràmetres que el programa .r reconeix i s'encarrega d'interpretar-ho. Veiem-ho:

Tenim el fitxer *main_ predictor.r*, en el qual veiem en alguns fragments com aquest carrega directament llibreries i fitxers de dades de forma directa:

```
#load( "c:/i/data_objects/processed_dfs_sci.Rdata" )

load( "/i/data_objects/best_sections_models_sci_gen_dilage_win.Rdata" )

library(car)

NUM_RECOMANACIONS <- 20
SECT_APPLIED <- 1
SECT_ADHOC <- 2
SECT_NA <- 3
ALL_APPLIED <- 1

....
load( "/i/data_objects/best_sections_models_sci_gen_dilage_sum.Rdata" )
best_section_models_sum <- best_section_models

....
test_llobregat <- read.csv( file = "/i/dades/test_delta_llobregat.csv" , head =
TRUE , dec = "," , sep = "," )
```

Això en el seu dia i en el context del projecte on es va realitzar ja era correcte, ja que en aquell projecte l'objectiu era l'obtenció de resultats d'investigació. És aquí on hem d'adaptar aquest codi, fer-lo genèric. Per fer-ho, primer de tot ens cal fer que l'script .R de l'ICHNAEA sigui capaç de rebre paràmetres de la línia de comandaments, això ho fem afegint el següent codi enlloc del que carrega automàticament les llibreries:

```
# get argsuments
args <- commandArgs(trailingOnly = TRUE)

# load .R script calling load's for needed .Rdata files

source(args[1])
```

En l'anterior codi fem que les llibreries es carreguin des del fitxer indicat en el primer paràmetre de la crida a R. Aquest paràmetre ens indicarà un fitxer .R que haurà creat el mateix servidor a partir dels fitxers associats a un projecte i internament contindrà quelcom semblant a:

```
load("/home/ichnaea/pepito/p20/best_sections_models_sci_gen_dilage_win.Rdata")
```

On hi haurà una línia per cada fitxer de l'usuari. Aquests fitxers els pujarà l'usuari mitjançant l'entorn web que veurem en la següent secció.

Més endavant, en aquest mateix fitxer d'ICHNAEA, s'ha modificat la línia que carregava un fitxer de dades, de tal manera que ens agafi el segon que li passem per la línia de comandaments, veiem-ho:

```
inputfile<-args[2]

test_llobregat <- read.csv( file = inputfile , head = TRUE , dec = "," , sep =
"," )
```

Per altra banda l'R genera molta sortida "brossa", per la qual cosa cal fer un cert filtratge, per això s'han posat unes "marques" que després el client interpretarà per tal de saber on comencen i on acaben les dades importants, això ho fem simplement posant uns simples:

```
# inici de la secció d'error
write("¡¡¡2¡¡¡", stderr())

# fi de la secció d'error
write("!!!2!!!", stderr())
```

```
# inici de la secció de sortida d'ICHNAEA
write("¡¡¡1¡¡¡", stdout())

# fi de la sortida d'ICHNAEA
write("!!!1!!!", stdout())
```

Que faran de marca al "parser" del client.

Les sortides de l'ICHNAEA es guardaran en respectius fitxers output.out i output.err del directori de la tasca associada. El web service que els retornarà ho farà llegint aquests fitxers seqüencialment i retornant-los al client.

4.1.7 Client Web

De cara a la gestió dels usuaris del sistema i que aquests puguin realitzar funcions com canviar la seva contrasenya o crear i gestionar projectes, s'ha desenvolupat un client web per tal de realitzar aquestes funcions.
Aquest no és més que un conjunt de pàgines implementades usant JSP (Java Server Pages), les quals interactuen amb el sistema mitjançant aquesta tecnologia. En els següents apartats ho veiem en detall.

HTML, CSS i JavaScript

Actualment la visualització d'una pàgina web es basa en aquests tres components enumerats. En HTML s'escriu el contingut de la pàgina, la informació. El disseny, tot i que es pot introduir en l'HTML, fa ja més d'una dècada es va anar separant usant CSS (Cascade Style Sheets), o fulls d'estil, que ens permeten definir un disseny, separat de l'HTML i que ens permet canviar aquest sense afectar els fitxers que contenen la informació de la pàgina.

Per acabar, el JavaScript, que en aquest projecte s'ha usat molt marginalment, ens serveix en aquest cas per a fer comprovacions sobre la correctesa de paràmetres en formularis abans d'enviar aquests. En aquest aspecte és important fer esment que mai s'ha de confiar cegament amb el JavaScript per fer aquesta funció de comprovació, ja que un usuari el pot tenir desconnectat, a més que no protegeix contra injeccions de peticions directament sobre les crides HTTP.

JSP

Aquesta és una tecnologia per a crear pàgines amb contingut dinàmic. El seu funcionament es basa en incrustar codi Java dins la pàgina, que el servidor d'aplicacions (Glassfish, Tomcat, etc.) que tinguem instal·lat processarà i executarà el codi, però aquest no el veurà mai el navegador web que rebi la pàgina.

Les pàgines amb contingut JSP tenen l'extensió .jsp en lloc de l'habitual .htm/.html. Internament el codi Java s'introdueix entre les etiquetes *<%* i *%>*, tot i que n'hi ha algunes variants. Veiem-ho amb un exemple, del fitxer *usersel.jsp*, del que en mostrem uns fragments:

```
<%@ page import="com.ichnaea.iserver.*" %>
<%@ page import="java.util.List" %>

<%
	List<User> users;
	DBMan um = new DBMan();

	User lus = (User)session.getAttribute("loginuser");
	session.removeAttribute("currentuser");
	session.removeAttribute("currentproject");

	users = um.GetUserList();

	if(lus==null)	// no session -> login again
	{
%>
		<jsp:forward page="index.jsp">
			<jsp:param name="foo" value="bar"/>
		</jsp:forward>
<%	}

.....

<form method="post" action="edituser.jsp" name="editu">
     <table style="text-align: left; width: 100%;" class="tseluser" border="1"
cellpadding="2" cellspacing="2">
    <tbody>
		....
    <%
		cl = ....
		....
     %>
    <tr>
        <td height="44" class="<%= cl%>" style="width: 155px;"><%= u.getUname()
```

```
%></td>
        <td class="<%= cl%>" style="width: 582px;"><%= u.getUnamer() %></td>
        <td class="<%= cl%>" style="width: 80px;">
                <button  name="subject"  type="submit"  value="<%=  u.getUid()
%>">Edit ...</button>
        </td>
    </tr>
    <% } %>
```

On podem veure que hi ha diferents tipus d'etiquetes:

<%@ %> en el nostre cas per indicar la importació de llibreries externes

<% %> entre les etiquetes hi ha codi Java pur.

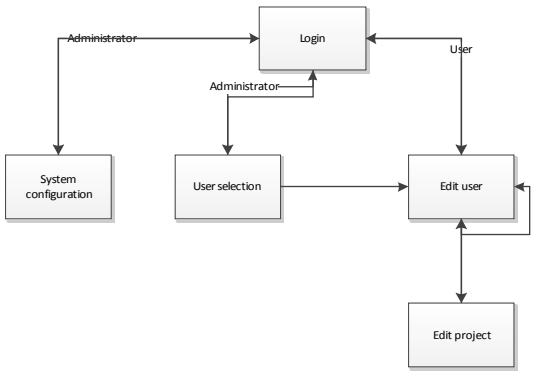
<%= %> mostra el valor de l'expressió continguda. Ideal per mostrar el valor d'una variable.

Així doncs per al fragment anterior el navegador rebrà el codi següent al carregar la pàgina:

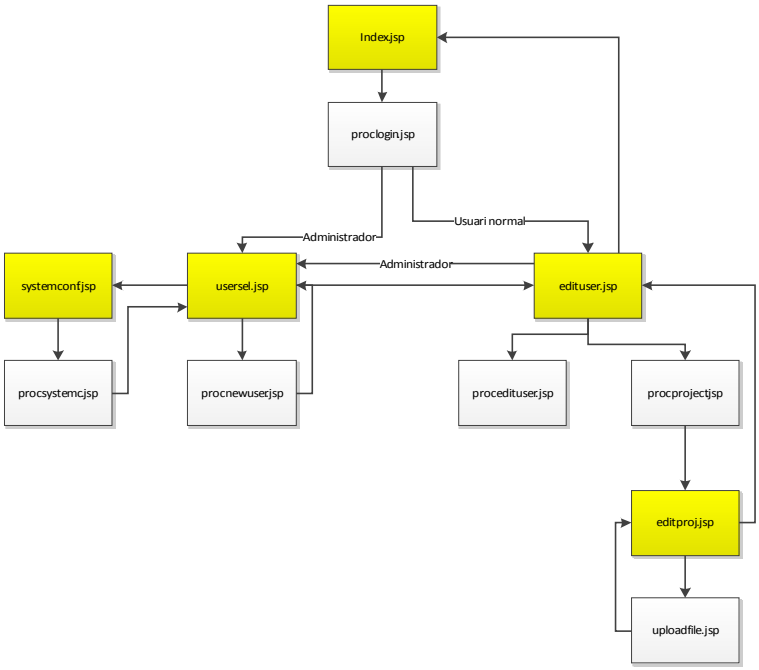
```
<table style="text-align: left; width: 100%;" class="tseluser" border="1" cellpadding="2"
cellspacing="2">
    <tbody>
    <tr>
        <td class="us_tab" style="width: 155px;">Username</td>
        <td class="us_tab" style="width: 582px;">Name</td>
        <td class="us_tab"></td>
    </tr>

    <tr>
        <td height="44" class="cladmin" style="width: 155px;">admin</td>
        <td class="cladmin" style="width: 582px;">System administrator</td>
        <td class="cladmin" style="width: 80px;">
              <button name="subject" type="submit" value="1">Edit ...</button>
        </td>
    </tr>
```

Estructura de pantalles i navegació

Aquí veiem un diagrama de les pantalles principals que l'usuari veurà:



i aquí el diagrama anterior mostrant els diferents fitxers que componen la web, en color groc s'indiquen els que l'usuari veurà, o millor dit, els que la seva funció és mostrar quelcom en pantalla.



Inicialment aquesta havia de ser una web molt senzilla que suportés unes poques funcions, per la qual cosa no es va considerar aplicar-hi un patró de disseny en el sentit estricte. En el nostre cas hauria estat un MVC (Model-View-Controller) que

essent estrictes s'hauria d'haver implementat com un conjunt de pàgines JSP únicament per a la visualització, una classe Java derivada de Servlet per a cada fitxer .jsp per a poder processar la informació entre pantalles i finalment una classe que ens serveixi de model per la interacció lògica – dades. Finalment s'ha optat per implementar la lògica dels servlets dins de fitxers .JSP, de manera que al final el sistema és més simple.

Per exemple, tenim una pantalla inicial (index.jsp) que fa el *login* i envia l'usuari i contrasenya a la pàgina proclogin.jsp on hi ha el codi *controlador* que fa les crides corresponents a la classe DBMan que fa de *model* dins del patró MVC.

En totes les pàgines d'aquesta web és fa un control exhaustiu de seguretat per tal que cap usuari es salti els permisos que té assignats. Per exemple, la pantalla de selecció d'usuaris només li apareix a qui s'identifiqui com a Administrador, però si algú hi intenta accedir manualment posant la URL al navegador havent-se identificat com a usuari normal, automàticament serà redirigit a la pàgina de *login* (*index.jsp*).

Tot això ho controlem mitjançant variables de sessió, que el mateix servidor d'aplicacions guarda en el sistema. En el cas de JSP, tenim l'objecte *session* al qual se li poden afegir objectes de tal manera que es facin persistents per aquella sessió.

Veiem-ho amb un exemple, en concret d'un fragment del fitxer *proclogin.jsp* :

```
DBMan um = new DBMan();
int retv = um.CheckUserPwd(username, password);
if(retv >= 0) // Right user
{
	// save user's id in current session
	session.setAttribute("userid", username);

	// We should check if user is an administrator o a regular user
	User us = um.GetUser(retv);
	session.setAttribute("loginuser", us);

	if(us.getPerm() == 0) // we are in front of a superuser
	{
```

```
                %>
                <jsp:forward page="usersel.jsp">
                        <jsp:param name="foo" value="bar"/>
                </jsp:forward>
                <%
        }
        else
        {
                %>
                <jsp:forward page="edituser.jsp">
                        <jsp:param name="FailReason" value="WrongPassword"/>
                </jsp:forward>
                <%
        }
}
```

En aquest exemple podem veure marcat en color blau la crida a l'operació de l'objecte encarregat de la interacció amb la base de dades, en el nostre cas DBMan. En groc veiem com a la classe global *session* hi afegim un objecte de la classe User (us) que correspon a l'usuari que ha entrat al sistema i l'hem etiquetar com a "loginuser". Aquests objectes que guardem a la sessió es consulten en les diferents pàgines que anem obrint per tal de determinar si hi ha intents d'accessos a pàgines a les que l'usuari no té autorització.

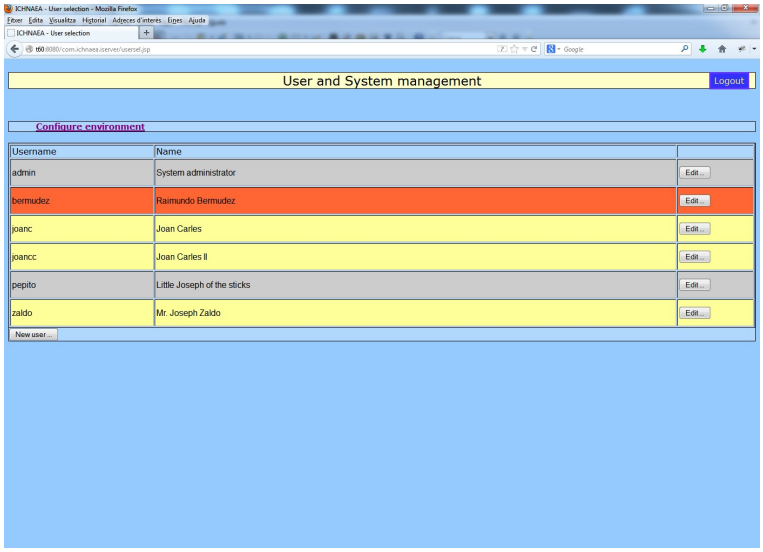
Funcionament

A nivell d'usuari, el sistema és simple. Tenim una pantalla inicial per a identificar-nos, depenent de si l'usuari té una qualificació d'administrador o no se l'enviarà a una pantalla per seleccionar usuaris i configurar el sistema o a una altra on només es permet modificar el propi usuari.

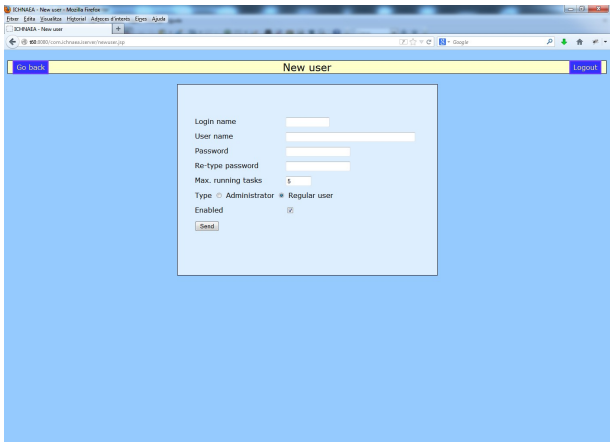
Els administradors poden crear, esborrar i editar usuaris, incloent limitar-ne l'entrada o el nombre d'execucions de tasques simultàniament (normalment ho hem establert en 5).

Així doncs, el primer que veu un administrador a l'entrar és la pantalla de selecció d'usuaris. Aquests es troben llistats en una taula i on hi ha un codi de colors que n'identifica les característiques: color gris si és un administrador, groc un usuari normal i vermell si és un usuari normal deshabilitat (un administrador no pot ser

mai deshabilitat, com a molt el podem degradar a usuari normal i després deshabilitar-lo si cal).

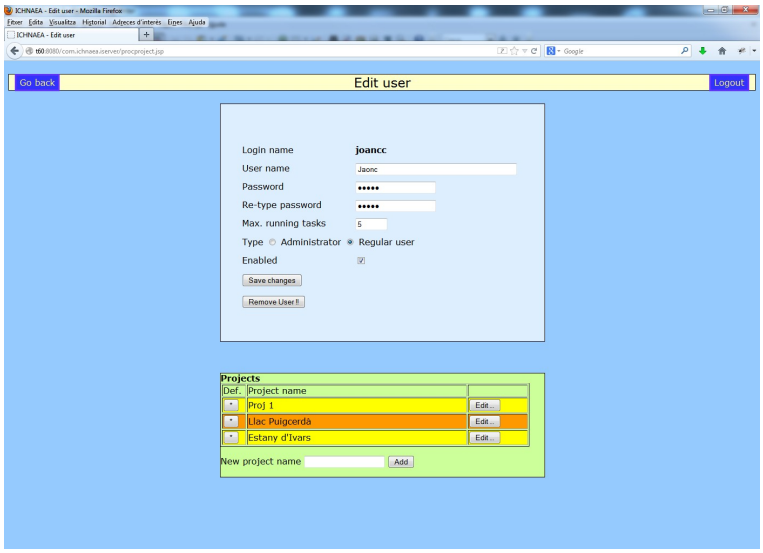


Aquí podem doncs donar d'alta un nou usuari. Aquesta pantalla ens permet introduir el nom de l'usuari, que ha de ser sense espais i és independent de majúscules o minúscules (internament el sistema el guarda en minúscules), el nom real de l'usuari, la contrasenya (mínim 5 caràcters), el nombre de tasques màximes que podrà executar simultàniament l'usuari, si està habilitat i si és un administrador o un usuari normal:

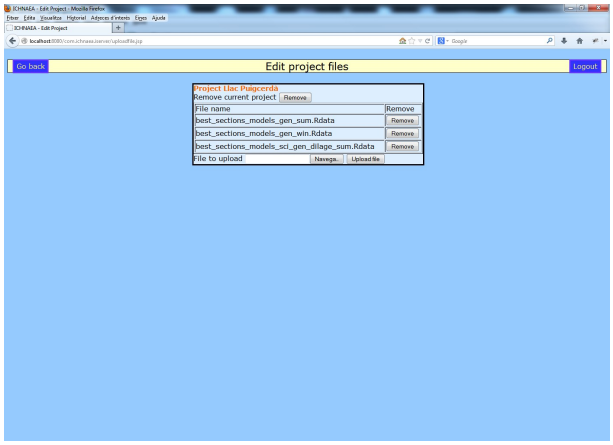


La pantalla d'edició d'usuaris segueix el mateix criteri i ens permetrà canviar-ho tot excepte el nom d'usuari. Si l'usuari amb que hem entrat al sistema és un

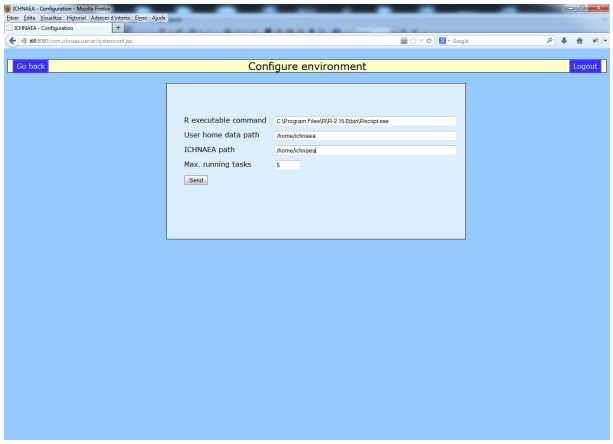
administrador ens permetrà eliminar l'usuari, canviar els permisos, i deshabilitar-lo. A més podrem afegir projectes. D'aquests projectes n'hi ha un marcat en color taronja que és amb el que es treballarà per defecte.



Tot seguit veiem la pantalla d'edició de projectes. Aquesta ens serveix per a afegir fitxers al projecte o en un moment donat eliminar-ne algun dels que hi ha. També permet esborrar un projecte sencer, amb la qual cosa es perden totes les dades que aquest guarda. Un projecte no es podrà eliminar si té alguna tasca en execució.



Els administradors també tenen finalment una pantalla on poden configurar paràmetres de funcionament del servidor, sobretot referent a la localització dels fitxers de dades i de l'ICHNAEA.



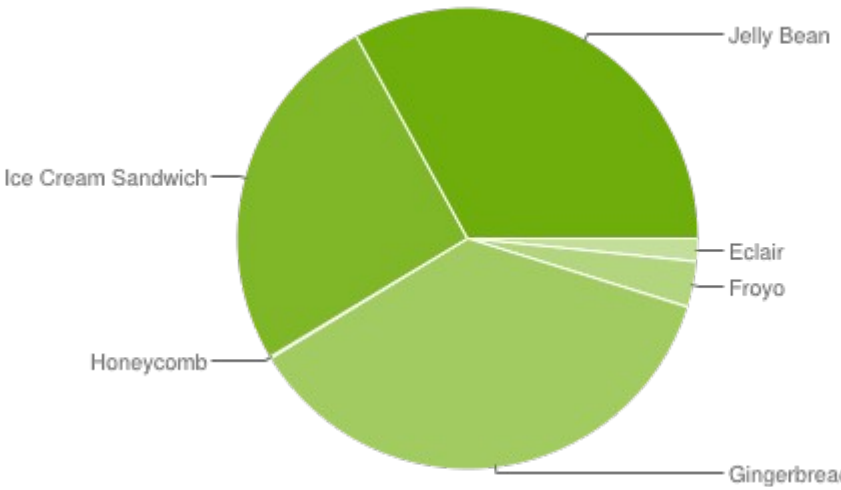
4.2 Client mòbil

El client és la part que l'usuari executarà en el seu terminal mòbil, tan pot ser un telèfon com una tauleta. Tot i que, com veurem més endavant, per la pròpia naturalesa de l'aplicació i per a una major comoditat és recomanable usar-la en un dispositiu amb una pantalla relativament gran.

4.2.1 Introducció: Android

Android és un sistema operatiu desenvolupat per a ser usat amb dispositius mòbils. Està basat en Linux que el té com a nucli en la seva versió 2.6. La seva penetració al mercat de dispositius ha arribat al punt que més del 74% d'*smartphones* venuts el primer trimestre del 2013 el porten [3].

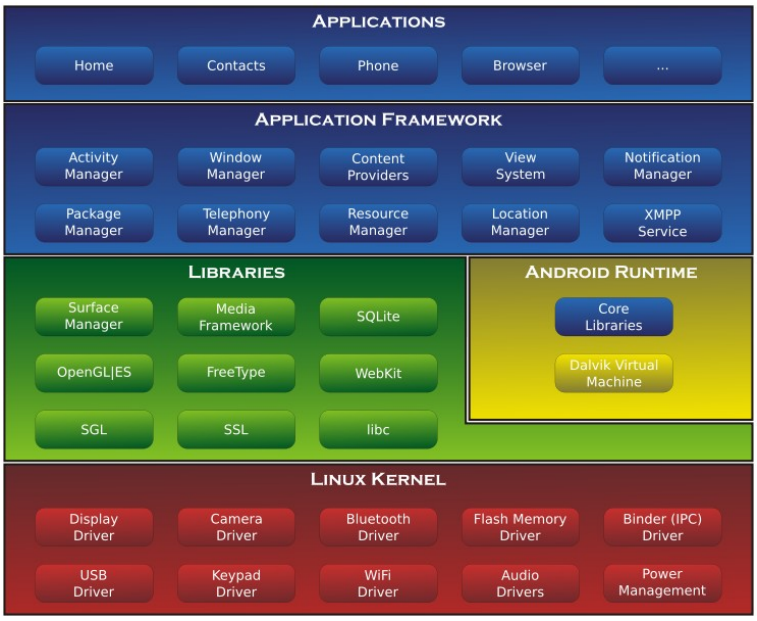
Hi ha múltiples versions d'aquest sistema, pel projecte s'ha optat per a posar un límit inferior suportat a la versió 2.3 (Gingerbread API 9), ja que la quota de mercat d'aquesta versió és molt elevada (a 3 de juny de 2013 representa encara el 36,5 %):



Actualment (juny 2013) l'última versió disponible és la 4.2.2 (Jelly Bean).

Arquitecura del sistema

Com s'ha dit abans, el nucli d'Android és un Linux 2.6, damunt d'aquest hi trobem les capes de Libraries i Android Runtime, que són respectivament una sèrie de llibreries natives compilades en el codi nadiu del processador, mentre que a Runtime hi tenim la màquina virtual Dalvik.



La màquina virtual és l'encarregada de l'execució del codi compilat específicament per a Linux a partir de Java. Originalment, degut a les limitacions dels dispositius amb Android no era possible incorporar-hi una JVM (Java Virtual Machine) completa, per la qual cosa es va dissenyar des de zero una nova màquina virtual (Dalvik) on hi corren les aplicacions, això sí compilades per a la nova màquina.
Immediatament damunt d'això tenim l'Application Framework on que conté els components necessaris de suport a les aplicacions d'usuari. Finalment a la capa superior és on hi trobem les aplicacions d'usuari i és on correrà el projecte.

4.2.2 Disseny de l'aplicació

En termes de disseny aquesta aplicació és d'una gran senzillesa. El patró de disseny adequat seria el patró en tres capes:

- capa de presentació
- capa de negoci
- capa de dades

Ara bé, tenint en compte que en el nostre cas, la capa de dades podem considerar que es troba remotament, també hi hem d'afegir que es compleix el tenir una arquitectura orientada a serveis (Service Oriented Architecture, SOA).

4.2.2.1 Capa de dades

Ens trobem que en aquesta capa a les dades hi accedim remotament. Tenim una classe WSCaller dedicada exclusivament a encapsular la lògica de crides remotes als web services.
Com vam veure al capítol del Servidor, allà s'encapsulava l'accés a la xarxa dels web services mitjançant JAX-WS de tal manera que la interacció amb els serveis web es feia d'una forma relativament transparent. En Android en canvi, tenim la peculiaritat que la màquina virtual d'aquest no suporta JAX-WS, per la qual cosa la interacció amb els web services cal fer-la a nivell SOAP, manegant les capçaleres d'aquest protocol i construint les peticions i llegint les respostes de forma totalment manual.
Disposem en Android de les llibreries ksoap2-android, que són una variant de kSOAP2. Aquesta és una llibreria d'implementació del protocol SOAP de forma lleugera i amb poc consum de recursos. De fet és la referència en dispositius Android si cal cridar a web services basats en SOAP.

Veiem en un fragment de codi com realitzem una crida a un servei des d'una rutina d'Android. Aquest fragment correspon a l'operació de *login*:

```
final String METHOD_NAME = "login";
final String SOAP_ACTION =  "http://iserver.ichnaea.com/Services";

SoapObject request = new SoapObject(co.NAMESPACE, METHOD_NAME);
SoapSerializationEnvelope envelope = new
SoapSerializationEnvelope(SoapEnvelope.VER11);

request.addProperty("user", user);
request.addProperty("pwd", pass);

envelope.setOutputSoapObject(request);

HttpTransportSE androidHttpTransport = new HttpTransportSE(URL);

androidHttpTransport.call(SOAP_ACTION, envelope);

SoapPrimitive  resultsRequestSOAP = (SoapPrimitive) envelope.getResponse();

session_id = Long.parseLong(resultsRequestSOAP.toString());
```

Es marca en groc la crida al servei, en blau com obtenim el resultat. Com es pot constatar hi ha una complexitat afegida respecte si s'usés JAX-WS, en el qual això seria transparent. A més si el que intentem enviar o rebre són tipus complexos (una llista de classes, per exemple) la cosa es complica més.

Per altra banda Android té la característica des de la versió 3 que els accessos a xarxa no es poden fer des del fil d'execució principal, per això per a totes les crides als web services cal crear nous fils d'execució. Per a fer-ho ens cal implementar objectes derivats de la classe AsyncTask.

4.2.2.2 Capa de negoci

En el nostre cas, aquesta es troba reduïda a la mínima expressió, ja que essencialment el client Android només fa de pont entre la capa de visualització i el servidor. En tot cas, la classe que més s'aproximaria a la funcionalitat d'aquests capa serien:

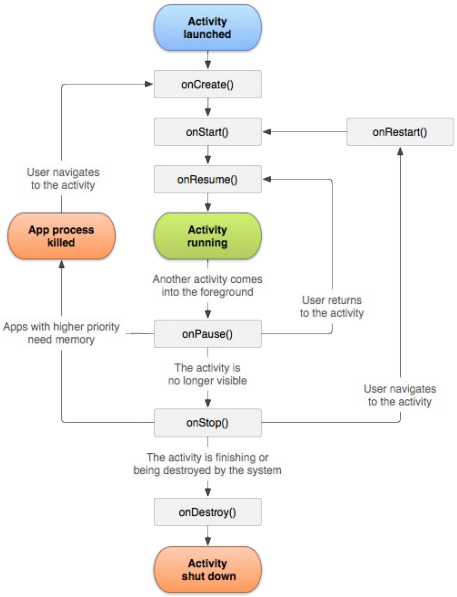
- Matrix : aquesta és l'encarregada de dibuixar la matriu en pantalla, però també en gestiona els canviar que hi fem, del tipus seleccionar files o columnes per al seu processament.
- Utils: aquesta bàsicament està feta per a proporcionar utilitats a l'hora d'interpretar els fitxers de sortida de la simulació o bé aquests quan els hem guardat en local.

Tenim a més a més dues classes externes auxiliars:

- opencsv [4]: que ens ofereix un conjunt de rutines per llegir, escriure i interpretar fitxers de text en format .csv (comma separated values), que són els que usa l'ICHNAEA.
- Android-file-dialog [5] : ens serveix per a crear uns diàlegs que ens permeten navegar pel dispositiu de cara a obrir o guardar fitxers.

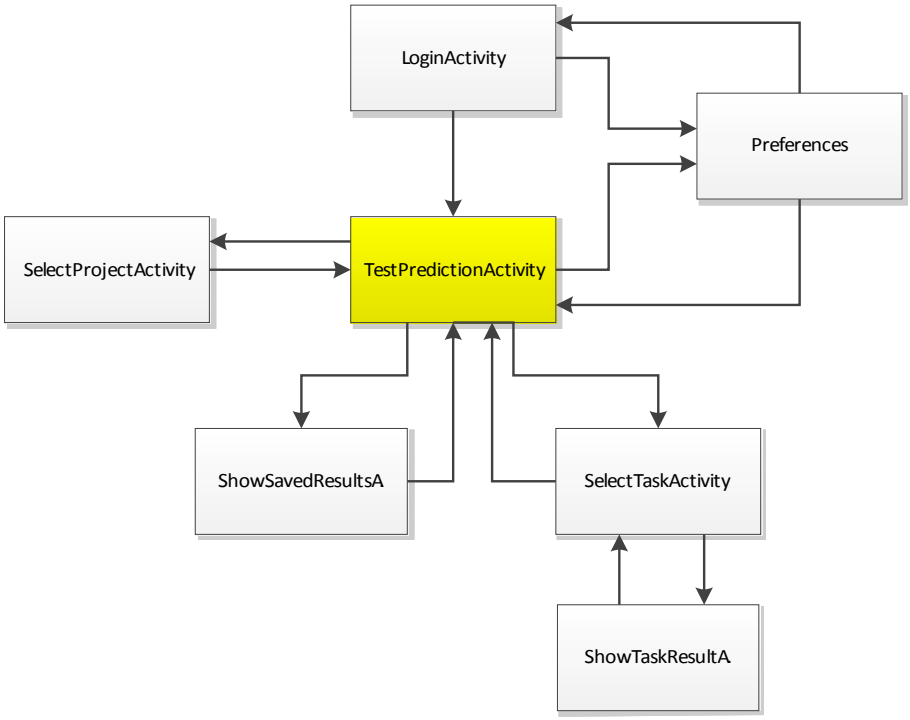
4.2.2.3 Capa de visualització

La visualització en Android es basa en el concepte d'activitat (Activity). Una activitat, de forma molt simplificada, és pot resumir com una pantalla de l'aparició. Les activitats poden cridar a altres activitats, canviant així la pantalla que veiem en el dispositiu. La peculiaritat és que s'organitza en forma de pila de manera que al sortir d'una activitat, aquesta es destrueix i es torna a la que estigui al cim de la pila. Cada pila d'activitats és pròpia de cada aplicació.



El diagrama anterior reflecteix en detall el diagrama del cicle de vida d'una activitat.

Mapa de pantalles de l'aplicació



En l'anterior esquema podem veure el diagrama de flux entre les diferents pantalles (activitats) del client. Com podem veure tot gira al mig de l'activitat "*TestPredictionActivity*".

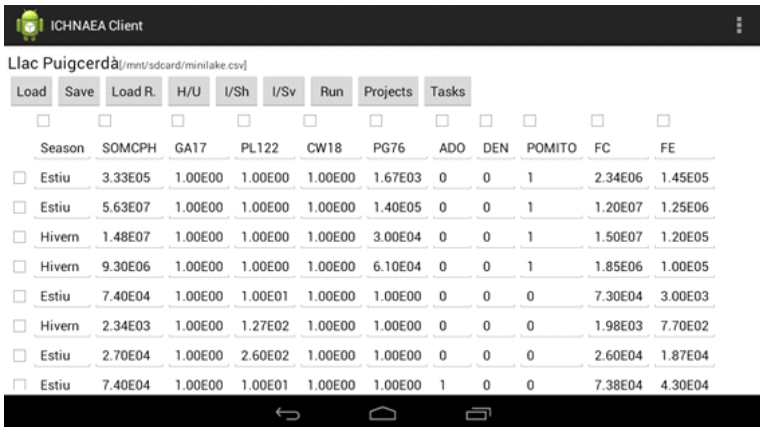
Funcionament

El primer que veu l'usuari quin inicia és la pantalla d'identificació (login). Si l'usuari està donat d'alta en el sistema i el client està correctament configurat, podrà entrar identificant-se tot i que hi ha l'opció d'entrar sense fer-ho. En aquest darrer cas no podrà accedir a les funcions que impliquin accedir al servidor. Només podrà obrir matrius, editar-les, guardar-les i visualitzar resultats guardats localment. És important indicar que només hi pot haver una única sessió oberta per usuari simultàniament. Si algú altre s'identifica amb un altre dispositiu deixa "desconnectat" al primer.

Sigui com sigui entrem a la pàgina principal, en aquesta, en la part superior hi veiem indicat quin és projecte que tenim definit per defecte i el fitxer que tenim obert. Només en podem tenir obert un a la vegada.

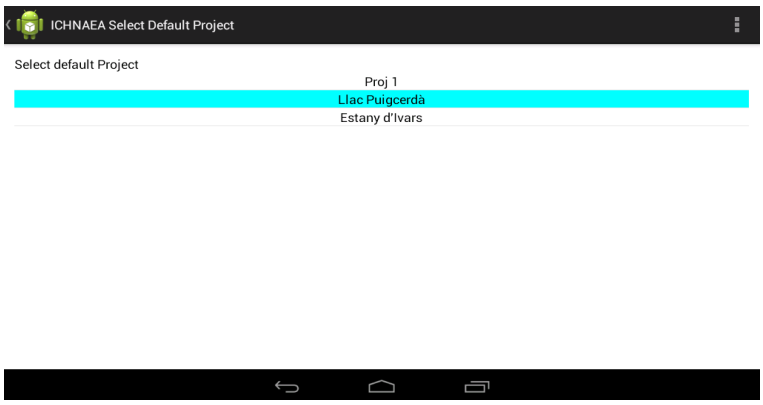
Tenim també una barra de botons amb funcions per obrir i guardar la matriu, engegar la simulació, obtenir resultats d'aquesta i canviar el projecte per defecte.

En aquest espai és on carregarem i editarem les matrius. Aquestes es visualitzen i s'editen d'una manera semblant a com ho faríem en un full de càlcul.



Season	SOMCPH	GA17	PL122	CW18	PG76	ADO	DEN	POMITO	FC	FE
Estiu	3.33E05	1.00E00	1.00E00	1.00E00	1.67E03	0	0	1	2.34E06	1.45E05
Estiu	5.63E07	1.00E00	1.00E00	1.00E00	1.40E05	0	0	1	1.20E07	1.25E06
Hivern	1.48E07	1.00E00	1.00E00	1.00E00	3.00E04	0	0	1	1.50E07	1.20E05
Hivern	9.30E06	1.00E00	1.00E00	1.00E00	6.10E04	0	0	1	1.85E06	1.00E05
Estiu	7.40E04	1.00E00	1.00E01	1.00E00	1.00E00	0	0	0	7.30E04	3.00E03
Hivern	2.34E03	1.00E00	1.27E02	1.00E00	1.00E00	0	0	0	1.98E03	7.70E02
Estiu	2.70E04	1.00E00	2.60E02	1.00E00	1.00E00	0	0	0	2.60E04	1.87E04
Estiu	7.40E04	1.00E00	1.00E01	1.00E00	1.00E00	1	0	0	7.38E04	4.30E04

Podem seleccionar quin serà el projecte per defecte, el qual serà al que s'enviaran les simulacions a partir d'aquell moment.



Tenim una activitat que ens llista les tasques que té l'usuari. Es mostren totes, les

finalitzades i les que estan corrent, es distingeixen pel color. Les finalitzades es poden obrir i llavors se'ns mostra una pantalla subdividida en dues parts on es mostra la sortida normal de l'ICHNAEA i la sortida de la consola d'error d'aquesta.

Tenim dos botons, un per guardar el resultat i també per eliminar la tasca. Això últim ens l'esborrarà totalment, perdent les dades que no haguem guardat d'aquesta.

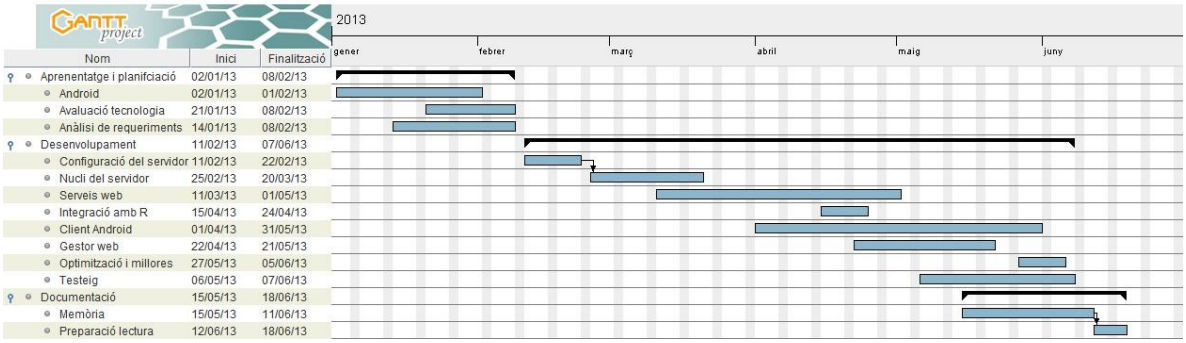
Igualment podem també llegir una tasca que haguem guardat a disc.

5. Planificació i costos

Aquest projecte es va començar a gestar la primavera del 2012, amb una sèrie de reunions mentre s'anava concretant què caldria fer i les necessitats, tot i que per motius diversos a nivell personal del projectista, la cosa va quedar aturada fins el gener de 2013 en què es va continuar amb el projecte.

Degut a tenir una feina de dilluns fins dissabte al matí, el gruix d'hores dedicat al projecte era durant el cap de setmana i part de la tarda i les nits dels dies laborables. He estimat que de mitjana hi ha hagut unes 20 – 25 hores setmanals, que si ho comptem per dies seria unes 3 – 3,5 hores diàries. De totes maneres això no és exacte, ja que a mesura que avançava el quadrimestre, la mitjana d'hores setmanals va augmentar fins a 30 hores a partir de finals d'abril, i les últimes 3 setmanes molt més encara.
En general, un factor important en els retards que hi ha hagut en alguns moments del desenvolupament (que han acabat motivant un augment significatiu de les hores emprades l'últim mes) ha estat degut al fet d'usar tecnologies totalment desconegudes a priori pel projectista, la qual cosa retarda la implementació a l'anar-se trobant petits problemes que a vegades fan replantejar part del disseny.

La planificació final, mostrada en un diagrama de Gantt ens queda com:



Nom	Inici	Finalització
Aprenentatge i planificació	02/01/13	08/02/13
Android	02/01/13	01/02/13
Avaluació tecnologia	21/01/13	08/02/13
Anàlisi de requeriments	14/01/13	08/02/13
Desenvolupament	11/02/13	07/06/13
Configuració del servidor	11/02/13	22/02/13
Nucli del servidor	25/02/13	20/03/13
Serveis web	11/03/13	01/05/13
Integració amb R	15/04/13	24/04/13
Client Android	01/04/13	31/05/13
Gestor web	22/04/13	21/05/13
Optimització i millores	27/05/13	05/06/13
Testeig	06/05/13	07/06/13
Documentació	15/05/13	18/06/13
Memòria	15/05/13	11/06/13
Preparació lectura	12/06/13	18/06/13

Com podem veure hi ha tasques que es solapen entre elles, això és degut a què hi ha tasques de desenvolupament que són independents unes d'altres, a més la fase de testeig es podia fer en paral·lel al ser en alguns casos simple. Al principi també, les fases d'anàlisi i disseny es podien compaginar amb formació de les tecnologies que finalment s'han usat.

Costos de personal

Si desglossem el projecte en diferents rols assumint les funcions pròpies d'aquests tindrem i assumint uns costos aproximats als de mercat bruts, tindríem:

Recurs	**Tasca**	**Hores**	**Preu/hora**	**Total**
Cap de projecte	Requeriments	60	60,00 €	3.600,00 €
Analista	Anàlisi i disseny	80	40,00 €	3.200,00 €
Programador	Desenvolupament, test i part de documentació	432	25,00 €	10.800,00 €
Total				**17.600,00 €**

Costos de software

Per a tot el desenvolupament s'ha usat intensivament software lliure, amb una part de software propietari per tasques secundàries, concretament hem usat:

- Java/MySQL/GlassFish com a software propi de l'aplicació, cost 0
- Eclipse com a entorn de desenvolupament, cost 0
- OpenOffice com a programari per redactar la memòria i la presentació, cost 0
- Kompozer per a la part de la pàgina web, cost 0
- Microsoft Visio Standard per a realitzar diagrames, cost 399 €

Costos hardware

Respecte el hardware, per a desenvolupar en Android no és estrictament necessari

disposar d'un terminal, ja que el mateix SDK té un simulador. El problema és que aquest simulador necessita un ordinador molt potent, per això per programar la major part del projecte s'ha usat un dispositiu real (una tauleta), quedant el simulador per unes poques execucions a resolucions diferents per veure com quedaria en terminals mòbils.

Així els costos per aquest concepte serien:

Ordinador portàtil + pantalla externa 700 €
Tauleta de marca genèrica Android 4.1 200 €

En resum, tenim:

Costos de personal	17.600,00 €
Costos materials	1.299,00 €
Total	**18.899,00 €**

6. Conclusions i treball futur

L'objectiu del projecte era crear un client Android que fes de *front-end* a les rutines d'ICHNAEA i en concret a la que permet fer una predicció de l'origen de la contaminació d'aigua a partir d'una mostra.

Com hem vist durant tot aquest informe, per a realitzar aquesta funció s'han usat i desenvolupat una sèrie de tecnologies que en principi podrien semblar exagerades, això és degut a què tota l'aplicació ha estat pensada per ser escalable i ampliable, per la qual cosa s'han deixat posades les bases quant a estructures de dades i fluxos de funcionament.

Respecte al desenvolupament d'aquest projecte caldria considerar que és un exemple interessant d'integració i aplicació de diferents tecnologies en un sistema relativament heterogeni: des del disseny d'una base de dades, a una aplicació que gestiona dades sobre aquesta, una web que interactua amb aquesta aplicació, uns serveis web que ofereixen funcions a un client implementat en un entorn totalment diferent (Android).

Coneixements aplicats i adquirits

En aquest projecte he pogut posar en pràctica tot una sèrie de coneixements adquirits durant la carrera, entre els que destaquem els adquirits en les diferents assignatures d'Enginyeria del Software, Gestió de Projectes i Bases de Dades.

M'ha servit també per a introduir-me en el món de la programació de dispositius Android, profunditzar enormement en l'entorn Java EE (Enterprise Edition) i actualitzar-me en les tecnologies relacionades amb RPC sobre web.

Possibles millores

Enumerem algunes possibles millores que es podrien fer a l'aplicació a curt termini, ordenades de menys a més dificultat tècnica:

- Traduir la interfície del client Android: això no seria massa complicat, ja que tots els missatges estan guardats com a recursos String, fàcils de traduïr.
- Possibilitat de consultar l'estat de les tasques i obtenir-ne resultats des de l'entorn web.
- Afegir l'accés a altres funcions de l'ICHNAEA: com s'ha dit, tot el sistema està preparat per a acceptar l'ampliació de funcions de forma fàcil.

7. Bibliografia i referències

7.1 Bibliografia

Goncalves, A. Beginning Java EE 6 Platform with GlassFish 3 Second Edition. Apress: 2010 ISBN 978-1-4302-2889-9

Darcey, L., Conder, S., Sams Teach Yourself Android Application Development in 24 Hours, Second Edition. Sams: 2012 ISBN 978-0-672-33569-3

Deepak Vohra, Java 7 JAX-WS Web Services. Packt Publishing: 2012 ISBN 978-1-84968-720-1

Schmitt, C. Et al., Professional CSS: Cascading Style Sheets for Web Design, Second Edition. Wiley Publishing: 2008 ISBN 978-0-470-17708-2

Pàgina web Android Developer: http://developer.android.com

7.2 Referències

[1] Sanchez-Mendoza, D., Belanche, L. A Software System for the Microbial Source Tracking Problem. Master in Artificial Intelligence (UPC-URV-UB), 2012

[2] R en Android:
http://rwiki.sciviews.org/doku.php?id=getting-started:installation:android

[3] Gartner Says Asia/Pacific Led Worldwide Mobile Phone Sales to Growth in First Quarter of 2013: http://www.gartner.com/newsroom/id/2482816

[4] Smith, G., Sullivan, S., Conway, S. OpenCSV : http://opencsv.sourceforge.net

[5] Alexander Ponomarev. Android File Dialog:
http://code.google.com/p/android-file-dialog

Apèndix I

Tot seguit hi ha el manual d'usuari de l'aplicació, consta de dos apartats principals:

1. Manual d'usuari de l'aplicació Android

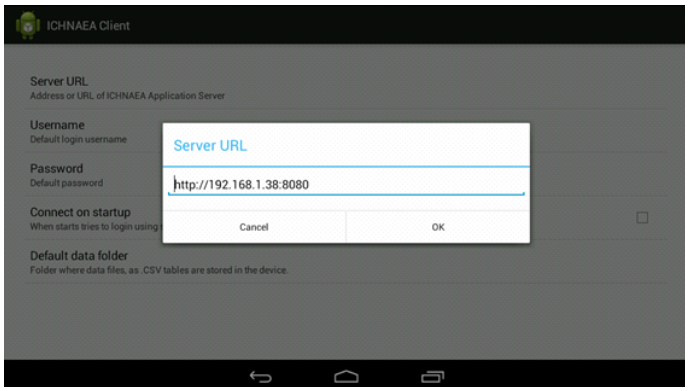
2. Manual de l'entorn de gestió d'usuaris i projectes mitjançant la web

El segon es divideix en el manual de l'usuari i el manual específic per l'administrador.

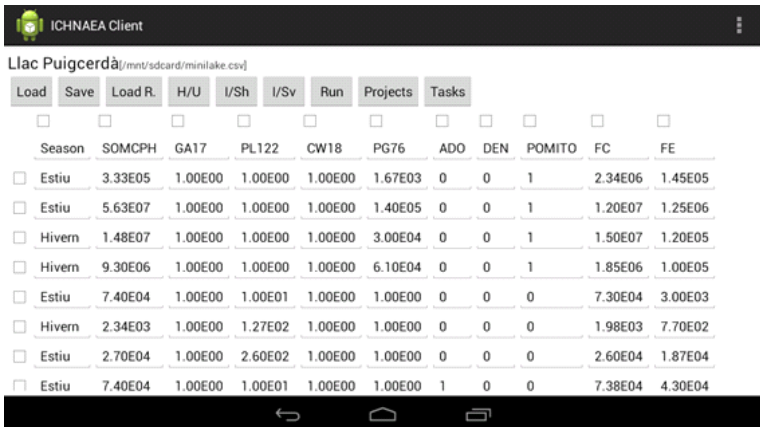
User Manual

1.1 Android client user manual.

The first page you will see is the login page, here you should enter your user name and password, although you can access to the application without credentials. In this case you can only read, modify and save matrices and also load saved previous results, but you can't use any network resource or run simulations. On the first run is important to set applications settings, specially those referred to the server address in order to be able to connect:



After doing login you are on the main screen:



Season	SOMCPH	GA17	PL122	CW18	PG76	ADO	DEN	POMITO	FC	FE
Estiu	3.33E05	1.00E00	1.00E00	1.00E00	1.67E03	0	0	1	2.34E06	1.45E05
Estiu	5.63E07	1.00E00	1.00E00	1.00E00	1.40E05	0	0	1	1.20E07	1.25E06
Hivern	1.48E07	1.00E00	1.00E00	1.00E00	3.00E04	0	0	1	1.50E07	1.20E05
Hivern	9.30E06	1.00E00	1.00E00	1.00E00	6.10E04	0	0	1	1.85E06	1.00E05
Estiu	7.40E04	1.00E00	1.00E01	1.00E00	1.00E00	0	0	0	7.30E04	3.00E03
Hivern	2.34E03	1.00E00	1.27E02	1.00E00	1.00E00	0	0	0	1.98E03	7.70E02
Estiu	2.70E04	1.00E00	2.60E02	1.00E00	1.00E00	0	0	0	2.60E04	1.87E04
Estiu	7.40E04	1.00E00	1.00E01	1.00E00	1.00E00	1	0	0	7.38E04	4.30E04

There you can see the first line where there is your current project title and the name of the current file. Below there is a group of buttons, that depending on the device screen resolution, you can see them in one or two rows. Those are they functions:

- Load: loads a .csv data file from the device.
- Save: saves current unselected columns and rows to a file in the internal storage.
- H/U: hides or shows current selected columns and rows
- I/SH - I/SV: invert selection horizontal and vertical.
- Load R: load results, loads and displays in a new screen results saved previously from an ended task.
- Run: sends current matrix to the server and runs the simulation.
- Proj.: lists your projects and allows you to change the default one.
- Tasks: lists all your tasks and allows the selection of the ones that are finished.
- Last T: loads results from the last task sent to the server.

Last three options are disabled if you are logged on offline.

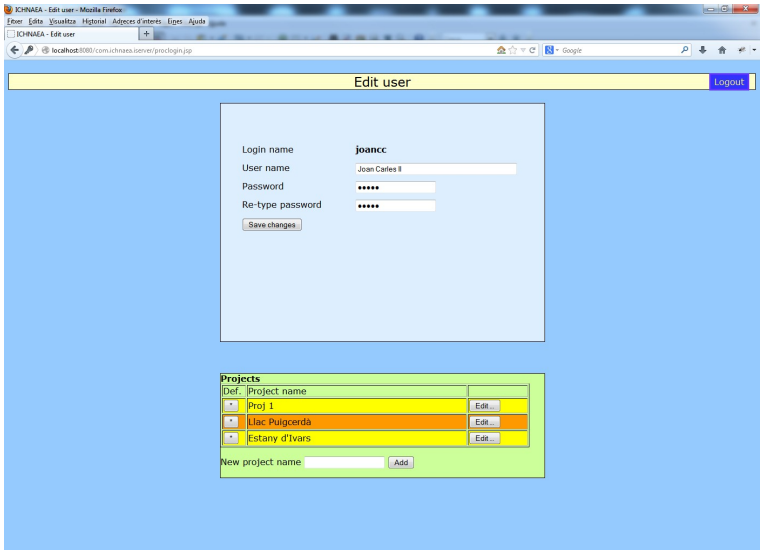
After loading a file, your display will show the matrix in a spreadsheet appearance as you have seen above.

Finally there is a Results Task screen, that is divided in two sections: the first shows the usual ICHNAEA results. The second shows errors occurred during simulation, they can be related to the data you sent or simply R problems. Below there is a button for saving results and another for removing the task and erase all information related to it.

1.2 User web interface.

ICHNAEA has a web interface mainly designed for managing user data. First of all we should connect to the web interface using your preferred web browser. Take care to know exactly the server URL, ask your system administrator for that.

The first page you will see is the login page, here you should enter your username and password. If validation is ok, you can see your user's properties page.

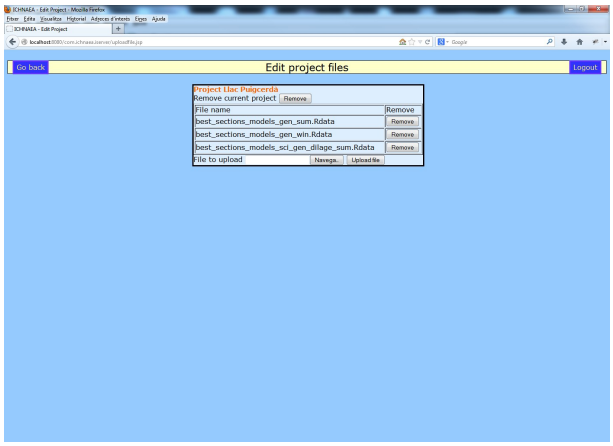


There you can change your password and your name (not login user name). You can also add projects entering a project name at “New project name” box and click “Add”. Immediately it will appear at the projects list.

You always have a default project that you can distinguish it by its orange colour. Other projects are colored in yellow, as you can see in this picture:

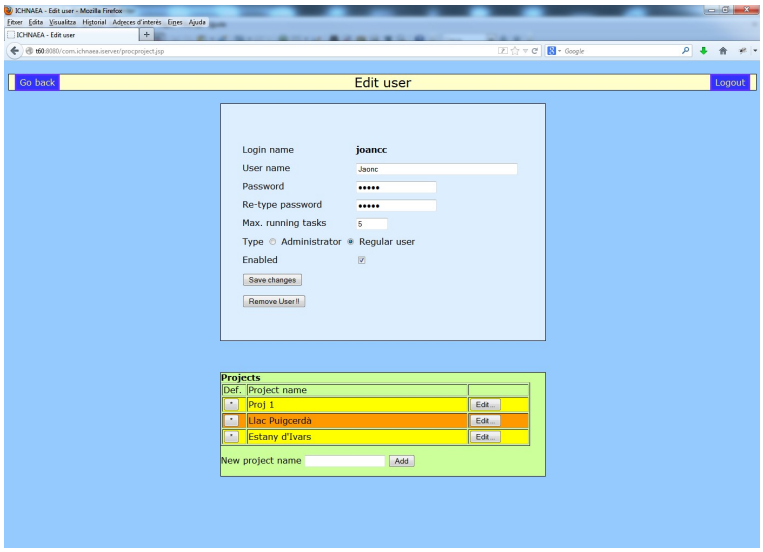
The default project can be changed clicking the button at the left side of the project's name. On its right there is a button to edit the project.

Edit a project means the action of adding. You have a button to upload a file and another to remove existing files.



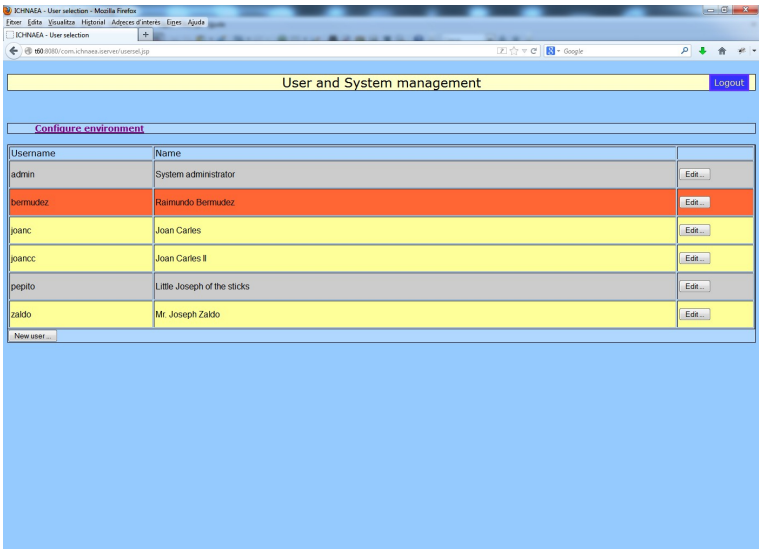
1.3 Administrator manual.

As we have seen in the previous section, regular users can only add, edit and remove their own projects. Administrator may also remove, disable or change user's role (as regular or administrator). A disabled user can't login or access any web service although their running simulations will continue. For doing so Edit User screen is slightly different:



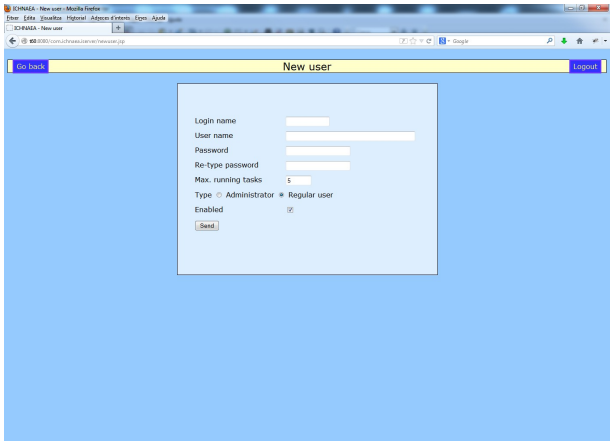
Another difference, is that administrators have tree new screens:

Select user screen



There we can select a user to modify it. Also we can add a new user to the system. In this list, users with administrative rights are shown in grey colour, regular users yellow and disabled, red. An administrator can't be disabled but you can change its permissions to regular user and disable it.

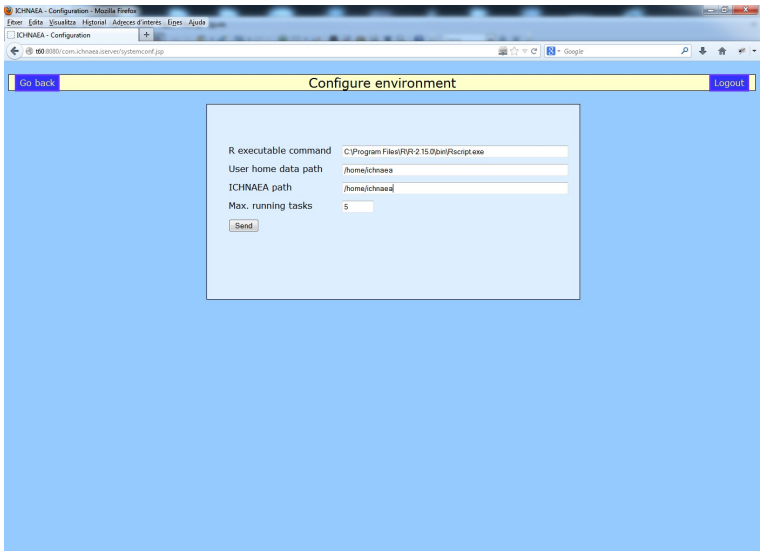
New user screen



As you can see it is similar at Edit User screen. There you can choose a new user's name. It isn't case sensitive (internally will convert it to lower case) and must not have space characters.

Configure server

The first time the application is started, after installation, there aren't any users in the system and the login screen only recognizes the 'admin' user (with password 'admin'), so it is very important to enter into the configuration page (through the link provided in User Selection page) and fill in the form with configuration parameters, like ICHNAEA folder, user folder and R executable path and program.


Go back
Configure environment
Logout
R executable command
C:\Program Files\R\R-2.15.0\bin\Rscript.exe
User home data path
/home/ichnaea
ICHNAEA path
/home/ichnaea
Max. running tasks
5
Send

Apèndix II

2. Instal·lació i configuració

Seguidament es descriuen les operacions que cal realitzar per a preparar l'entorn de cara a instal·lar el necessari per a executar el servidor i començar a processar peticions.
Cal destacar, que per la pròpia arquitectura de l'aplicació tan pot ser instal·lada en una màquina Linux/Unix com Windows. Si s'instal·la en un Windows cal parar compte que sigui una versió de servidor ja que les de *Workstation* (Windows XP, Vista, 7, 8) tenen una limitació (artificial) en el nombre màxim de connexions TCP/IP que poden tenir obertes simultàniament.

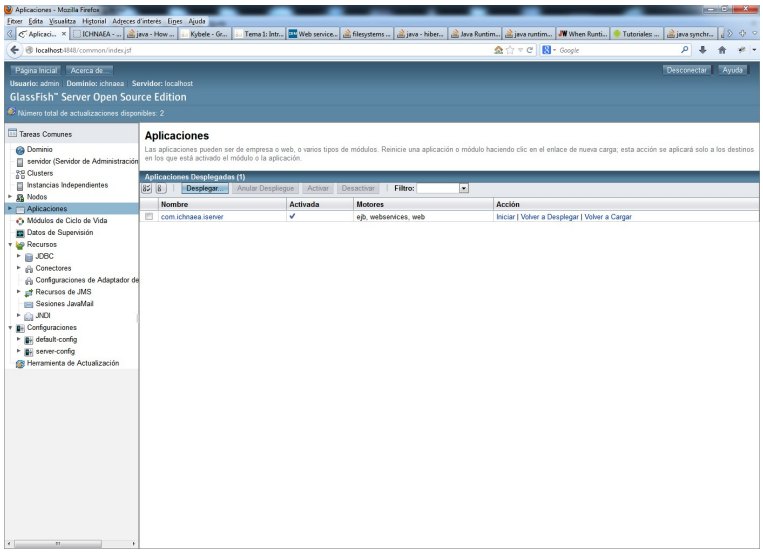
Primer de tot cal decidir una ubicació pels fitxers dels usuaris. Si treballem amb un sistema Unix, es recomana usar /home/ichnaea, sempre i quan existeixi aquest directori.

2.1 Servidor

El servidor d'aplicacions Oracle Glassfish Server Open Source Edition es pot descarregar en diferents versions depenent del sistema operatiu on s'executarà des de l'URL https://glassfish.java.net/. Allà hi trobarem versions per Windows, Unix/Linux/AIX i independent de la plataforma (el servidor està íntegrament desenvolupat en Java).

Un cop instal·lat Glassfish i havent-lo configurat convenientment, cal desplegar l'aplicació des del fitxer .war proporcionat. Això es fa des de l'entorn de gestió del Glassfish, accessible des de http://localhost:4848 i amb les credencials que li haguem indicat durant la instal·lació.
Un cop allà cal anar al menú "Aplicaciones" i prémera Desplegar, seleccionem el

fitxer .war de la nostra aplicació i ja està. Cal parar compte que si ja havia estat desplegat anteriorment cal marcar l'opció de forçar un nou desplegament.



2.2 Base de dades

MySQL el podem descarregar des de http://dev.mysql.com/downloads/mysql/. Per a monitoritzar la base de dades i disposar d'un entorn amigable per a gestionar-la, es recomana baixar i instal·lar el MySQL Workbench des de http://dev.mysql.com/downloads/tools/workbench/.

Després d'això, els passos i paràmetres usats en la instal·lació han estat els següents (per a una versió sobre Windows):

Multifunctional Database, tot i que ens podriem plantejar Transactional Database Only
Online Transaction Processing (OLTP)
Port 3306
Standard Character Set
Install As Windows Service amb nom MySQL

i activar Include Bin Directory in Windows PATH
Contrasenya de root: itxi2013 (podem posar la que ens interessi)
Finalment instal·lem el MySQL Workbench si l'hem descarregat.

Un cop s'ha instal·lat MySQL cal crear la base de dades de l'aplicació. Això ho fem executant el fitxer de comandaments SQL proporcionat.

2.3 Intèrpret R

Com que l'ICHNAEA corre sobre R cal també tenir instal·lat aquest entorn. El podem descarregar des de http://www.r-project.org/ on podem escollir baixar els binaris ja compilats per Windows, Linux o MacOS X. Actualment ja hi ha la versió 3.0, però com que l'ICHNAEA original va ser implementat en la versió 2.1 i com que quan es va iniciar el projecte també era la versió de referència, s'ha optat per continuar en aquesta.

Un cop instal·lat l'R ens cal incorporar-hi una sèrie de paquets que necessitarem, per a fer-ho cal entrar a l'entorn R i instal·lar-los mitjançant els comandaments:

```
install.packages("ipred")
install.packages("kernlab")
install.packages("randomForest")
install.packages("car")
```

2.4 Aplicació ICHNAEA

Aquesta és la col·lecció de fitxers R que conformen les rutines que s'executaran sobre l'R. Aquestes es troben efectivament guardades en fitxers de l'extensió .r i poden ser copiades al directori que desitgem, tot i que es recomana que sigui dins del directori arrel on hi ha els usuaris de l'ICHNAEA. Cal tenir en compte que sigui on sigui que la instal·lem hem d'indicar en quin directori a l'aplicació

mitjançant l'entorn de de gestió per web.

2.5 Client Android

Normalment les aplicacions Android s'acostumen a baixar i instal·lar des del Google Play (antigament conegut com Android Market). En aquest cas no li hem penjat degut a què al ser una eina tant específica i que a més requereix de connexió a un servidor ICHNAEA amb acreditació, es decideix emprar el sistema d'instal·lació des d'una memòria en local.
Per a fer-ho ens copiem o baixem l'aplicació en format .apk al dispositiu, hi cliquem al damunt i deixem que el Google Play s'engegui i s'instal·li automàticament.

El primer cop que engeguem el client Android és necessari anar a la pantalla de configuració, bàsicament per a indicar-li a quin servidor remot es connectarà. Normalment, l'adreça a introduir serà quelcom semblant a: http://serv.ichnaea.com:8080 on podem veure-hi el nom de la màquina i el port on s'ofereixen els serveis.