Universitat Autònoma
de Barcelona

SISTEMA DISTRIBUÏT D'INDEXACIÓ DE FITXERS

Memòria del projecte

d'Enginyeria Tècnica en

Informàtica de Gestió

realitzat per

Daniel Casadevall Pino

i dirigit per

Rafael Cortés Fité

Escola d'Enginyeria

Sabadell, *Juny* de *2010*

El sotasignat, ***Rafel Cortés Fité***,

professor de l'Escola d'Enginyeria de la UAB,

CERTIFICA:

Que el treball al que correspon la present memòria
ha estat realitzat sota la seva direcció per en ***Daniel Casadevall Pino***

I per a que consti firma la present.
Sabadell, ***Juny*** de ***2010***

Signat: ***Rafael Cortés Fité***

Resum

Els sistemes de cerca de fitxers són cada vegada més potents, alhora que fàcils d'utilitzar per a l'usuari. Sistemes operatius com Mac OS X integren algorismes de cerca i indexació tant potents i ben dissenyats que fan que l'usuari rebi una sensació de cerca instantànea sense cap penalització en rendiment.

No obstant, aquesta potència es perd quan es parla de sistemes en xarxa. A l'hora de centralitzar informació i realitzar cerques sobre aquesta, les eines de les que es disposa no son tan avançades.

Hem creat aquest programari amb la premisa d'aconseguir un sistema de cerca com els actuals, però integrant continguts de tota una xarxa local i amb possibilitat d'exploració des de qualsevol navegador, independentment del sistema operatiu utilitzat.

Pel que fa a l'apartat acadèmic, s'ha volgut crear una aplicació que fes us de la tecnologia AJAX, molt utilitzada a la web 2.0 i amb un futur immediat prometedor. D'aquesta forma, s'han posat en pràctica els coneixements assolits durant la carrera, alhora que s'han explorat nous camps d'estudi i s'ha pogut experimentar en primera persona el procés de creació d'un projecte de principi a fi.

Índex

1. Introducció

1.1. Resum del Projecte

L'objectiu principal d'aquest treball és el d'aconseguir implementar un sistema d'indexació de recursos distribuït en diversos nodes d'una xarxa mitjançant l'ús de bases de dades que, juntament amb mètodes de sincronització el més dinàmics possibles, ens permeti emmagatzemar els continguts de vàries estacions de treball, intentant reduir la redundància de fitxers al màxim. Alhora, atorgarem a l'usuari una interfície gràfica per a explorar aquesta base de dades de forma senzilla i intuïtiva.

Avui en dia existeixen molts sistemes d'indexació de base de dades per a fitxers locals o en xarxa. La gran majoria d'aquests es troben implementats en els principals sistemes operatius (Mac Os, Ubuntu o Windows Vista per exemple) i ofereixen una cerca bastant ràpida, inclús dins del contingut de fitxers. No obstant, no abunden al mercat les solucions independents del sistema operatiu, que pugin ser implementades en un entorn corporatiu en una xarxa local i alhora ofereixin la potència de cerca esmentada. Per aquesta raó vam decidir realitzar aquest treball de final de carrera, en un intent de cobrir aquest buit del mercat amb una solució de codi font obert.

La indexació de continguts es durà a terme a partir d'una màquina servidor, des de la qual es gestionarà la connexió amb les diverses estacions de treball. Amb els continguts d'aquestes màquines, es crearà i actualitzarà la base de dades en un fitxer local al servidor.

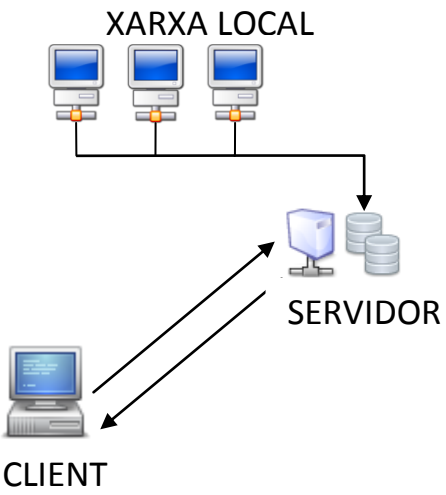


Fig 1. Entorn de treball del projecte

Per a aconseguir la interacció entre un usuari i el servidor, haurem de crear una aplicació accessible via web que serveixi com a interfície per a navegar a través d'aquesta estructura de dades situada al servidor. Volem donar èmfasis al disseny d'aquesta part, ja que resultarà la cara visible d'aquest projecte de cara a l'usuari.

El projecte estarà clarament dividit en dos fronts. El primer serà el control de la xarxa i base de dades en l'ordenador servidor i el segon l'obtenció de la informació a l'ordenador client amb comunicació directa al servidor. Així, serà important escollir tecnologies de programació adequades, que ens permetin treballar en conjunt de forma eficient.

Client	Servidor	Base de dades
Java	PHP	MySQL
HTML/Javascript	Perl/CGI	Oracle
Flash	ASP	XML
Visual Basic Script	.NET	

Taula 1. Alguns possibles llenguatges de programació a utilitzar

1.2. Estat de l'art

La solució que plantegem es troba a mig camí entre els cercadors d'Internet, que fan una indexació de una xarxa remota, amb poca

informació sobre el seu contingut (totes les pàgines web) i els diferents programes d'indexació d'escriptori, que creen una base de dades dels elements locals d'un ordenador, incloent informació bastant extensa en alguns casos.

Pels primers existeixen moltes solucions conegudes, totes gratuïtes, encara que la majoria no són amb codi font lliure. La més utilitzada avui en dia és **Google**, seguida per **Yahoo** i **MSN**.

Pel que fa als cercadors d'escriptori, trobem **Spotlight** com la millor referència a la funcionalitat que volem implementar. Es tracta del cercador de fitxers utilitzat pel sistema operatiu Mac Os X i per l'iPhone, que ens permet cerca paraules dins de documents o arxius per nom, data de modificació, dimensions , etc... Amb una potència similar però no tant ràpid tenim el sistema de cerca de directori d'**Ubuntu**. També cal esmentar el sistema d'indexació incorporat a les versions de **Windows Vista**, que tot i esser una mica lent compleix amb la funcionalitat bàsica i també permet cercar dins dels fitxers.

Com a solucions independents del sistema operatiu tenim, per exemple, **Windows Search**, basat en la versió pel sistema operatiu de Microsoft i de pagament. Tanmateix trobem el famós **Google Desktop**, que intenta combinar la cerca d'escriptori amb la cerca web.

Tant els motors de cerca com els cercadors d'escriptori empren sistemes d'indexació i refresc de la informació semblants, però ens interessen més els darrers, per semblança en el tipus d'informació emmagatzemada.

Tots aquests sistemes solen permetre a l'usuari cercar informació que es troba dins el fitxer, així com dins de les *metadades*, informació referent a l'arxiu en sí (data de creació, dimensions de l'arxiu, etc..).

La creació de la base de dades inicial sol ser lenta, però després d'aquest pas la actualització de contingut es realitza de forma periòdica i transparent a l'usuari, o quan un fitxer és modificat.

Si busquem un programa que s'adapti al perfil del nostre projecte, indexant informació d'una xarxa local en una base de dades que pugi ser

accedida de forma remota, trobem majoritàriament aplicacions creades per empreses per al seu propi ús, i no disponibles per a l'ús públic per raons òbvies. Així doncs, si volguéssim gestionar informació en una xarxa local de forma remota, hauríem d'utilitzar programari pensat per treballar de formar local i gestionar la xarxa des de l'ordenador client, quedant lligats a una base de dades local.

1.3. Objectius

A grans trets, podríem englobar els objectius d'aquest projecte en dos nivells.

A nivell acadèmic, l'objectiu implícit en aquest estudi és el de dur a terme un treball de final de carrera interessant per a l'estudiant, que permeti posar en pràctica un gran nombre de coneixements adquirits durant el curs i alhora veure en primera persona les diferents fases que constitueixen un projecte.

En un entorn d'explotació, s'espera poder desenvolupar un conjunt d'aplicacions que permetin realitzar la cerca sobre una base de dades situada en una màquina servidor, a la qual s'hi podrà accedir de forma remota a través de web. Aquesta base de dades podrà esser formada pel contingut de varies màquines connectades en xarxa local i gestionades pel servidor.

Amb aquest software, es pretén englobar la funcionalitat d'indexació de dades que ja aporten les aplicacions integrades a molts sistemes operatius avui en dia, la organització de informació en una base de dades centralitzada i la flexibilitat d'accés remot amb interfície web independent del sistema operatiu.

2. Estudi de viabilitat

2.1. Introducció

Tot projecte té un cost, ja sigui de recursos humans (en aquest cas l'estudiant i el professor majoritàriament), tècnics, materials, etc..

Es de vital importància realitzar aquest estudi de requeriments i recursos necessaris per a dur a terme el projecte, assolint un mínim d'objectius establerts. D'aquesta forma, conclourem si és possible finalitzar el nostre treball dins del termini establert i utilitzant una quantitat de recursos raonable per a un estudiant.

2.2. Objecte

2.2.1. Descripció de la situació actual

Actualment existeixen varis serveis d'indexació de contingut de disc dur que ofereixen serveis de cerca bastant potents, amb opcions de previsualització de contingut i cerca de text dins dels fitxers. No obstant, moltes d'aquestes aplicacions depenen del sistema operatiu utilitzat i les que no ho fan segueixen centralitzant la base de dades en un ordenador no accessible remotament de forma directa.

A qualsevol empresa que emmagatzemi informació sobre la qual s'està treballant des de diferent ordenadors li interessa que es puguin realitzar cerques exhaustives de forma remota, ja sigui des de casa o en xarxa, a través d'una interfície web independent al sistema operatiu utilitzat. No obstant això, i encara que les bases de dades són utilitzades arreu del món, son pocs els sistemes que integrin indexació de contingut en xarxa, independent del sistema operatiu i no centralitzada.

Si a tot això li sumem que les solucions que puguin existir d'aquest perfil són desenvolupades per empreses per al seu propi ús i/o no de codi

obert, tenim una mancança d'oferta en el sector bastant important. Tenint en compte que el poder cercar de forma àgil dins dels arxius d'una empresa des d'allà on es trobi el treballador és una activitat bastant habitual, pensem que aquest projecte pot ser útil per a moltes persones.

2.2.2. Perfil d'usuari

Degut al perfil de l'aplicació, l'usuari principal seran els treballadors de qualsevol PYME que mantingui un arxiu dels seus treballs al llarg del temps, o simplement que vulgui tenir un espai organitzat amb tots els documents en els quals es va treballant.

Un possible exemple molt adequat seria una empresa de desenvolupament de software, que vulgui saber quins fitxers han estat modificats el dia a dia i accedir a la base de dades fora de la feina per cercar els fitxers sobre els que treballar, o certes funcions dins del codi entre centenars de fitxers.

Un exemple d'àmbit totalment diferent seria una empresa de disseny tèxtil, que desa tots els patrons que ha fet al llarg dels anys. Si l'empresa te una vida considerable, la quantitat de fitxers dels que estem parlant podria arribar als milers, fent el nostre sistema de cerca a través de tota la xarxa molt interessant per l'usuari.

Com veiem, el nostre usuari no està emmarcat en cap àmbit concret, i no té per què tenir coneixements avançats d'informàtica. Així, caldrà que la interfície d'usuari sigui clara i ben organitzada, de forma que l'usuari estàndard pugui trobar el que vol fàcilment, sense tenir cap coneixement tècnic previ.

El fet que la base de dades es trobi en un servidor al qual hi pugin accedir varis usuaris alhora, ens obliga a establir diferents perfils d'usuari per cada treballador, els quals no podran modificar la base de dades alhora si ho intenten. Tanmateix, ens interessarà definir dos tipus d'usuari diferents:

Administrador: Serà l'usuari sense cap restricció, que podrà modificar la base de dades quan ho cregui necessari. Crearà còpies de seguretat i comprovarà la integritat del sistema de forma periòdica.

Gestor: Realitzarà les funcions de control d'accés a les dades de cara al treballador. Ajudarà als treballadors a utilitzar eficientment les funcionalitats del sistema. Es podria definir aquest perfil com a un pont entre les tasques de l'administrador i el treballador. No podrà modificar directament el contingut de la base de dades, però sí els permisos d'accés.

Treballador: Aquest usuari podrà cercar i iterar la base de dades igual que el coordinador, així com veure els fitxers que han estat modificats recentment, però no podrà modificar la base de dades directament.

2.2.3. Objectius

El nostre objectiu principal serà aconseguir crear una interfície web que permeti cercar sobre una base de dades situada en una xarxa externa a l'ordenador client, gestionada per una màquina servidor gràcies al software que crearem.

A partir d'aquesta premissa, podem definir un seguit d'objectius que ens ajudaran a assolir el producte final:

1. Indexar el contingut de diverses estacions de treball en una base de dades situada a la màquina servidor, emmagatzemant informació sobre cada fitxer i part del contingut.
2. Crear un aplicatiu web per a poder accedir a la base de dades del servidor, realitzant cerques ràpides amb paràmetres tals com la mida del fitxer, la data de modificació o paraules contingudes dins del fitxer.
3. Aconseguir que el servidor retorni informació dels elements demanats sense obrir-los. Així, s'emmagatzemarà a la base de dades

contingut tal com la primera pàgina d'un arxiu PDF o DOC, paraules clau d'un fitxer de text, una imatge en miniatura de les imatges en formats comuns, etc..

4. Crear un sistema d'actualització de base de dades suficientment dinàmic. En comptes de realitzar lentes actualitzacions periòdiques de l'índex, s'aniran comprovant els continguts a mida que s'iteren els directoris. Per exemple, si un directori ha estat creat en una carpeta que estem veient, aquest apareixerà com a nou element a la interfície gràfica i s'afegirà automàticament a la base de dades.

5. Aconseguir una comunicació fluida entre servidor i client, que treballaran amb tecnologies diferents. Voldrem temps d'espera curts per a petites actualitzacions quan sigui necessari, en favor de llargues esperes per grans refrescos de la base de dades.

6. Dissenyar una interfície d'usuari agradable, que no limiti les possibilitats ofertes pel nostre aplicatiu i que sigui agradable i intuïtiva.

7. Controlar l'accés i refresc de la base de dades per varis usuaris de forma simultània, de forma que es pugui accedir i modificar la informació continguda en el servidor des de varies estacions sense provocar conflictes.

8. Crear tres jerarquies d'usuari i atorgar un control de l'aplicació superior en funció de quin tipus estigui autentificat (Control limitat en cas del treballador, parcial per al gestor i total per a l'administrador).

9. Gestionar la xarxa local des del servidor de forma transparent a l'usuari, unificant tots els continguts de diferents màquines en una sola base de dades.

10. Maximitzar la compatibilitat amb diferents navegadors de la nostra aplicació web.

2.3. Descripció del sistema

2.3.1. Estructura

El sistema a realitzar estarà format per una aplicació executada pel navegador de l'usuari, que farà d'interfície entre la màquina client i el servidor i permetrà accedir a la base de dades des de qualsevol lloc amb accés a Internet. Aquesta interfície tindrà la famosa estructura de "vista d'arbre" i "vista de llista", o **treeview** i **listview**, que trobem en la majoria de gestors de fitxers com el mateix Windows (Fig. 2).

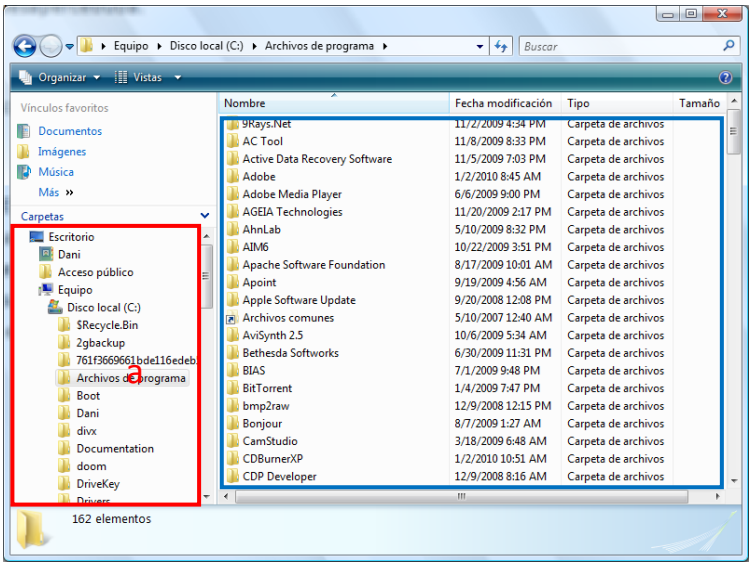


Figura 2. Listview (a) i treeview (b) mostrada en el Windows Vista.

A més a més, s'inclourà una opció de cerca amb un disseny semblant al d'altres sistemes operatius, que permetrà realitzar cerques exhaustives i mostrar el resultat en forma de llista, amb previsualitzacions del contingut dels fitxers com a icones (imatges, pàgines del document, etc..) (Fig. 3).

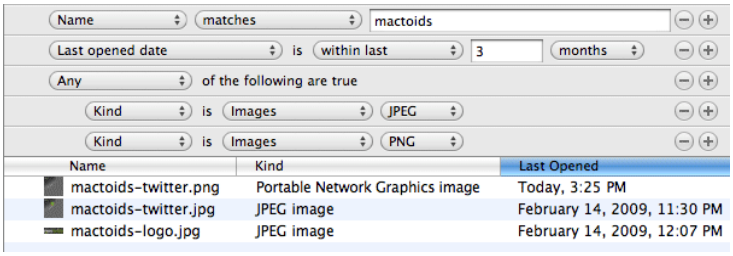


Figura 3. Resultats d'una cerca amb el sistema Spotlight de Mac Os.

Per a poder recórrer la base de dades, el client realitzarà peticions al servidor, que retornarà una copia temporal d'aquesta, sobre la qual es treballarà localment per a accelerar les cerques. Si l'usuari es troba amb elements que han estat modificats respecte la versió existent en la base de dades, aquests es mostraran com a nous, i es realitzaran les actualitzacions pertinents enviant la informació al servidor de forma transparent.

El procés de les peticions del client a la màquina servidor requerirà del desenvolupament de petits programes que realitzin aquestes tasques quan sigui necessari, i retornin la resposta corresponent. Per exemple, si un fitxer que s'ha cercat es vol obrir, el client enviarà la petició al servidor, el qual executarà un petit script que cercarà el fitxer demanat a dins la xarxa local i retornarà el fitxer a l'ordenador client.

Tot seguit veiem un esquema de l'estructura del sistema per a comprendre millor el procés explicat.

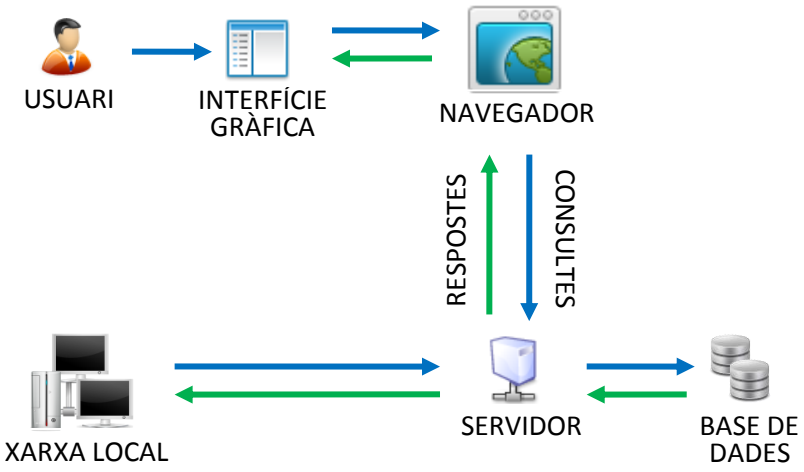


Figura 4. Estructura del sistema.

Notem que les peticions a la xarxa local per part del servidor es produiran en el moment de creació de la base de dades, quan es vulgui obrir un fitxer o quan es vagi iterant a través de la base de dades, per a comprovar la sincronització amb aquesta de forma asíncrona i transparent a l'usuari.

D'altra banda, les consultes a la base de dades es realitzaran al carregar el programa, desant-ne una copia a l'ordenador de l'usuari o al sincronitzar-la de forma manual, si un coordinador o sol·licita.

2.3.2. Tecnologies utilitzades

Com hem vist anteriorment (Taula 1), podem utilitzar una combinació de llenguatges de programació molt variada per cada part del sistema. Tot seguit es justifica l'elecció d'una tecnologia en concret per a cada cas.

Per al software que hi haurà al servidor, s'ha escollit PHP, degut a l'experiència prèvia amb el llenguatge, tant laboral com acadèmica, a la enorme quantitat de documentació trobada i a la facilitat d'integració amb les tecnologies de bases de dades XML i SQL.

Pel que fa a l'aplicació que s'executarà a l'ordenador client, hem decidit utilitzar una combinació de Javascript sobre HTML, amb taules de disseny CSS. Aquest conjunt es coneix popularment com a DHTML i proporciona una gran flexibilitat a l'hora de dissenyar pàgines web, amb contingut interactiu i efectes molt agradables a la vista (finestres flotants, columnes redimensionables, etc..).

L'elecció del llenguatge per la base de dades és complicada. MySQL seria la opció més atractiva si només tinguéssim en compte la facilitat per treballar amb aquesta tecnologia conjunt amb PHP i la quantitat de documentació existent sobre el tema. No obstant, hem decidit recórrer a XML. El motiu principal és la possibilitat de combinar una base de dades en aquest format junt amb el client en JavaScript per formar el que es coneix com a AJAX (Asynchronous JavaScript and XML), una tecnologia molt popular des que Google la va suggerir l'any 2005. Si a més a més utilitzem la llibreria Sarissa per a JavaScript, podem fer peticions asíncrones al servidor des de qualsevol navegador. Això ens permetrà, per exemple, realitzar una cerca a la base de dades i efectuar altres operacions mentrestant, sense haver d'esperar a la resposta del servidor per a actuar.

Tant el client en JavaScript com els Scripts en PHP, necessitaran un servidor web en el qual allotjar-se. Em pensat en escollirel servidor Apache, amb els mòduls que ofereixen suport per a l'execució de PHP i l'API XML i instal·lat sobre un sistema operatiu UNIX de qualsevol tipus (en el nostre cas OpenSUSE 11.1, però qualsevol plataforma LINUX ens serveix).

Ens proposem treballar amb UNIX en comptes de Windows per a instal·lar el servidor Apache degut a diferents motius. El primer és la flexibilitat que proporciona aquest sistema a nivell de servidor, podent configurar el nostre sistema sense necessitat de tenir una interfície gràfica en funcionament i amb la opció d'escollir entre un gran nombre de versions disponibles de forma gratuïta. El segon motiu és l'experiència prèvia de l'estudiant en administració de sistemes en xarxa sobre Linux, que ajudarà a l'hora de gestionar la xarxa local des del servidor, de forma transparent a l'usuari. Altres motius com la seguretat superior de la plataforma, els menors recursos tècnics necessaris o la major quantitat de documentació existent per Apache en versió Linux són també importants.

2.4. Recursos necessaris

Per a l'elaboració del projecte de principi a fi necessitarem un mínim de recursos humans i tècnics especificats a continuació.

2.4.1. Recursos Humans

Calculem que seran necessàries aproximadament unes 260 hores per part de l'estudiant per al desenvolupament d'aquest projecte. Això a banda, el tutor del projecte realitzarà entrevistes d'unes dues hores de durada mitja cada 15 dies, resultant en 24 hores compartides de discussió sobre el progrés del treball.

Finalment, el tutor haurà de dedicar una nombre d'hores indeterminades a revisar el material entregat per l'alumne (major en funció del volum de treball).

2.4.2. Recursos de Hardware

En qualsevol empresa que pugi necessitar aquest tipus d'aplicació, ja hi haurà una xarxa local configurada. Així, l'equip adicional necessari per a gestionar el sistema seria només el servidor, un ordenador amb prestacions no massa elevades, amb l'únic requisit de tenir un espai de disc dur suficient per a mantenir l'índex d'una base de dades que pugui anar creixent al llarg del temps, junt amb el propi sistema operatiu.

Determinar els recursos hardware necessaris, es una tasca complexa, ja que dependran, en gran manera, del client que exploti el sistema. Ens podem trobar petits usuaris amb xarxes LAN relativament simples o grans entorns amb cents de sistemes. Per tal de determinar l'espai en disc necessari, hem fet una proba en un ordenador local creant un índex bàsic XML dels fitxers d'una carpeta amb 581 elements. Aquest índex ha ocupat 73.725 bytes. Arrodonint, determinem un ús mitja en disc a l'índex de 127 bytes per fitxer al disc. Prenem com a exemple una empresa de disseny, que anomenarem BAPT, amb 15 anys de vida. BAPT compta amb tres ordenadors on desa tots els seus patrons, que al llarg dels anys junt amb altres fitxers relacionats han sumat la xifra de 10.000.

La xifra total en aquest cas, per a una empresa amb un volum d'informació a indexar realment elevat, puja a uns 4 MB en total

No podem estimar l'espai necessari d'una forma global, ja que depèn totalment del tipus d'activitat que realitzi l'empresa, i la quantitat de fitxers que es vulgui indexar. No obstant, podem assegurar que per a una empresa amb un volum d'informació emmagatzemada no desorbitat, amb un disc dur de 120 GB es podrà indexar quasi un bilió de fitxers, molt més del necessari en la majoria dels casos. Podem pensar que en cas d'arribar a tal quantitat d'entrades a la base de dades, el temps d'accés per cada consulta podria ser una limitació. No obstant, si algun dia

s'arribés a aquest volum de dades, el creixement tecnològic esperat en aquest període de temps fa suposar que el temps de lectura en relació al volum de dades serà acceptable.

Per tant, es viable gestionar la informació en aquest format sense comprometre les capacitats del servidor que hostatgi el servei.

A banda de la màquina servidor, farà falta un ordenador extern per accedir a l'aplicació a través d'Internet. En el nostre cas, per comoditat, durant el desenvolupament de l'aplicació podrem fer servir un sol ordenador, executant el servidor dins d'una màquina virtual amb Ubuntu 9.04 instal·lat i accedint-hi des del sistema operatiu "Host" o amfitrió (en el nostre cas Windows Vista, però no importa quin hi hagi ja que només necessitem un navegador). D'aquesta forma simularem la connexió client - servidor de forma fàcil i sense necessitar d'un ordenador extern. A l'hora de provar el sistema de forma exhaustiva, podrem configurar el servidor dins d'una xarxa local amb equips d'escriptori de configuració bàsica. Es farà servir l'ordenador personal de l'alumne com a servidor, i dos ordenadors d'escriptori reutilitzats de prestacions bastant limitades com a elements representants de la xarxa local i el client. Quan fem aquestes proves caldrà un cable de xarxa RJ-45 per connectar el servidor a la xarxa local i connexió a Internet per accedir des del client al servidor web.

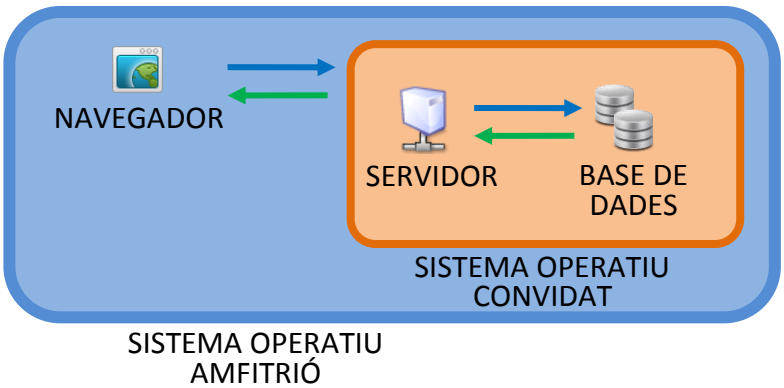


Figura 5. Estructura de l'entorn simulat a través d'una màquina virtual en un sol ordenador

2.4.3. Recursos de Software

Ja hem esmentat part del programari que s'utilitzarà a la descripció del sistema. Per al servidor, necessitarem descarregar Apache, amb els seus mòduls per a PHP i XML. El sistema operatiu utilitzat serà OpenSUSE 11.1.

Pel que fa al client, utilitzarem els navegadors Mozilla Firefox i Internet Explorer 8 com a mostra dels dos navegadors més utilitzats avui en dia per a maximitzar la compatibilitat.

La simulació de l'entorn de treball amb un sol ordenador comentada anteriorment es realitzarà gràcies al software VMWare Player, que executarà una màquina virtual amb Ubuntu 9.04 instal·lat.

Per a la programació en JavaScript, HTML CSS i PHP utilitzarem l'API gratuïta NetBeans, disponible tant en Windows com en Linux.

Per a escriure aquest document s'ha fet servir el paquet Microsoft Office 2007.

Finalment, per a depurar el nostre programa al navegador del client, utilitzarem una extensió anomenada Firebug en el cas de Firefox i les eines proporcionades per Microsoft per a desenvolupament web en el cas d'Internet Explorer.

2.5. Planificació del projecte

L'organització d'aquest projecte s'ha definit seguint el model lineal, a partir del qual dividirem la feina a realitzar en les següents fases.

Etapa	Descripció
Anàlisi	Anàlisi dels requisits mínims de comportament i rendiment per a poder dur a terme un disseny adequat.

Disseny	Definim l'estructura de dades de la nostre BBDD, les tecnologies a utilitzar, la interfície d'usuari i modelem l'algorisme del programa. Procés de traducció dels requisits mínims en una representació de software.
Implementació de la part servidor	Crearem el codi que gestionarà la base de dades, realitzant consultes i modificacions. Definirem com es llegeixen les dades de la xarxa local per a formar i sincronitzar l'índex.
Implementació de la part client	Implementarem la interfície d'usuari al navegador, establint com i quan es comunicarà amb el servidor.
Instal·lació de l'entorn	Passarem de l'entorn simulat a la instal·lació en un entorn real amb una xarxa local i accés remot.
Proves	Realitzarem intensives proves de funcionament sobre l'entorn final.
Documentació	Generarem la referència d'ús del nostre producte, així com la memòria del projecte.

Taula 3. Fases de desenvolupament del projecte

Hem dividit aquestes etapes globals en una llista de tasques a realitzar en un temps determinat. Aquests temps són aproximats, pel que contarem amb certs marges de desviació de forma que puguem acabar el projecte dins del termini determinat.

Nombre de tarea	Duración	Comienzo	Fin
Anàlisi	**12 días**	**vie 3/5/10**	**lun 3/22/10**
Estudi de viabilitat	7 días	vie 3/5/10	lun 3/15/10
Anàlisi de requisits	5 días	mar 3/16/10	lun 3/22/10
Disseny	**19 días**	**mar 3/23/10**	**vie 4/16/10**
Disseny de la Base de Dades	7 días	mar 3/23/10	mié 3/31/10
Disseny de la interfície gràfica	5 días	jue 4/1/10	mié 4/7/10
Modelat de l'algorisme bàsic	7 días	jue 4/8/10	vie 4/16/10
Implementació servidor	**14 días**	**lun 4/19/10**	**jue 5/6/10**
Codi de creació i sincronització índex	7 días	lun 4/19/10	mar 4/27/10
Codi de gestió de la Base de Dades	7 días	mié 4/28/10	jue 5/6/10
Implementació client	**14 días**	**vie 5/7/10**	**mié 5/26/10**
Codi de la interfície gràfica	7 días	vie 5/7/10	lun 5/17/10
Codi iteració i cerca de Base de Dades	7 días	mar 5/18/10	mié 5/26/10
Instal·lació de l'entorn	5 días	jue 5/27/10	mié 6/2/10
Proves	6 días	jue 6/3/10	jue 6/10/10
Documentació	7 días	vie 6/11/10	lun 6/21/10

Figura 6. Llista de tasques detallada.

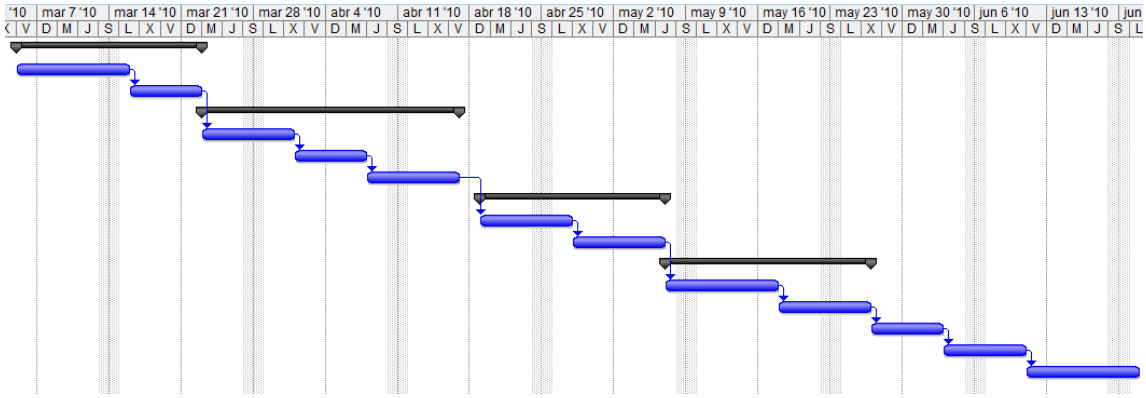


Figura 7. Diagrama de Gantt de la planificació temporal del projecte

A la figura 7 observem el diagrama de Gantt amb la distribució d'aquestes tasques al llarg del temps.

2.6. Anàlisi de costos

En el següent apartat intentarem aproximar els costos que suposaran el desenvolupament d'aquest projecte.

Com hem vist en l'apartat 2.4, aquest treball requerirà d'una sèrie de recursos tant humans, com de hardware i software adquirit que es traduïran en costos de cara a l'estudiant. Intentarem fer una previsió del preu que tots aquests elements poden arribar a tenir. Això és realment important a l'hora de determinar si el projecte és viable, ja que els recursos i costos corresponents que llistarem són els que considerem indispensables per al bon desenvolupament d'aquest treball.

2.6.1. Costos humans

Per a determinar els costos humans, s'haurà de tenir en compte el treball realitzat per l'estudiant. Hem decidit no comptabilitzar les hores que el tutor dedica tant a les entrevistes com a la correcció del treball de l'estudiant. Això es deu a que aquestes hores ja són remunerades per l'UAB.

Així doncs, es comptabilitzaran les 260 hores de elaboració del treball, junt amb les 24 hores de les entrevistes amb el tutor per part de l'alumne .

Això, assumint que el salari mig d'un estudiant d'informàtica en temps de crisis volta els 30 € l'hora, ens resulta en el següents costos humans:

Temps emprat	Preu per hora	Total
284 hores	30 € / hora	8.520 €

Cal tenir clar que això és una aproximació del cost que significa emprar aquestes hores en treballar. És el que s'anomenen **costos d'oportunitat**.

2.6.2. Costos de hardware

Com hem comentat a l'apartat de recursos de hardware. Per a la majoria del procés de desenvolupament del projecte s'utilitzarà una configuració de màquina virtual on el servidor i el client s'executen sobre la mateixa màquina. Tenint en compte que no existirà cost d'adquisició ja que s'utilitzarà el portàtil actual de l'estudiant, el nostre cost dependrà només de l'amortització d'ús de l'ordenador, junt amb la despesa en energia que suposa tenir-lo encès.

Per a calcular l'amortització, considerarem que la mitja de temps de vida d'un ordenador portatil (el que s'utilitzarà) és de 5 anys. El cost total del portatil en qüestió va ser de 1200€ en el seu moment. Si els amortitzem en 5 anys, tenim un total de 240€ l'any, (1200€ / 5 amortització lineal) o 60 € cada 3 mesos.

Farem un ús total de 260 hores, repartides en 3 hores per dia. Això equival a 87 dies d'ús.

Amortització total	Amortització anual	Amortització mensual
1200 €	240 € / any	60 € / 3 mesos

Així, podem comptabilitzar una amortització de material de 60€ en el curs del projecte.

2.6.3. Costos de software

Com ja s'ha vist a la secció de recursos de software, el cost del nostre programari és nul. Utilitzem només software de codi obert o llicència gratuïta. Ubuntu, Netbeans, Apache en són un exemple. L'únic software propietari que s'hauria de valorar seria la suite de Microsoft Office i el sistema operatiu Windows Vista. Tot i així, aquests dos elements es venien junt amb l'ordenador. El cost del qual ja hem tingut en compte. Per aquesta raó, creiem encertat assegurar que el cost en software durant el desenvolupament del projecte serà inexistent.

2.7. Avaluació de riscos

Durant el desenvolupament d'aquest projecte correrem varis riscs que poden afectar els resultats d'aquest. Alguns poden ser tècnics, mentre que d'altres poden ser deguts a una mala planificació del temps, un mal anàlisi o disseny. Tot seguit llistem algunes possibilitats:

- Un error comú es comet durant la planificació. En la creació de l'estudi de viabilitat, es preveuen dates que no s'acaben complint, augmentant els recursos utilitzats. Aquest fet té un impacte bastant greu en la qualitat final del projecte, així que és important planificar detalladament la nostra línea de temps.

- També durant l'estudi de viabilitat, es corre el risc de cometre errors de càlcul en l'anàlisi de costos. En el nostre cas, com els costos que tenim són majoritàriament costos d'oportunitat, aquest risc no és molt important.

- Per al nostre projecte hem de realitzar moltes eleccions de tecnologies diferents, que interactuaran entre elles dins el sistema instal·lat. Durant la fase d'anàlisis, és crucial escollir bé

aquestes tecnologies (PHP o ASP.NET, javascript o java, etc..) per a que puguem assolir els nostres objectius inicials marcats.

- Si realitzem un anàlisi dels objectius massa ambiciós, correm el risc d'intentar abarcar massa aspectes amb el nostre treball, fet que resultarà en un retràs en el plaç de desenvolupament important.

- Pel que fa a l'etapa de disseny, si no es realitza correctament podrem trobar-nos amb la necessitat de replantejar-nos aspectes del projecte a mig desenvolupament. Això ocasionaria pèrdues de temps i recursos importants.

- Si es realitza un trasteig del sistema pobre, existeix la possibilitat que el producte final tingui una qualitat menor de la desitjada o mostri falles de funcionament.

- Un dels riscs més importants és el possible abandonament del projecte abans d'acabar-lo. Això provocaria pèrdues econòmiques i acadèmiques molt importants per a l'estudiant.

2.8. Conclusions

Durant aquest estudi, hem observat que les solucions existents al mercat per a la indexació de dades no reuneixen totes les característiques que oferirem amb el nostre sistema.

Hem vist que l'entorn de les PYME no ofereix solucions com la que plantegem, que a més proposa utilitza tecnologies relativament noves (AJAX).

Segons la planificació realitzada, el temps de desenvolupament del projecte no supera els tres mesos, suficient per a seguir l'horari acadèmic establert. A més, aquest temps de desenvolupament suposa uns costos deguts als recursos humans i de hardware que ens són a l'abast.

Finalment, i tenint en compte que els riscos que correrem al desenvolupar al projecte són evitables o de menor importància, podem concloure que aquest projecte és viable.

3. Anàlisis del projecte

3.1. Introducció

En aquest capítol es revisaran els requeriments mínims, funcionals i no funcionals, que tindrà el nostre projecte. Avaluarem el problema que volem resoldre de forma que es puguin determinar els diferents casos d'ús i estudiar com afrontar-los.

3.2. Requeriments funcionals

3.2.1. Visió general

Per a intentar entendre el problema que volem afrontar, plantejarem la situació hipotètica d'una empresa de disseny anomenada Kyra.

Kyra consta d'un planell de treballadors reduït, no superior a 15. Encara que pugui semblar que el volum de treball d'una PYME com aquesta no és molt gran, al llarg de 20 anys aquesta empresa ha acumulat al voltant de 1000 dissenys diferents. Aquests dissenys estan formats per una mitja de 4 fitxers diferents (el disseny, dibuixos addicionals, comentaris..).

Imaginem que un treballador vol recuperar un sketch creat fa quatre anys per a un projecte nou. Actualment, en general el treballador haurà d'intentar recordar a quina estació de treball o servidor s'havia desat aquest fitxer i cercar-ne els continguts, potser a través del nom de fitxer o carpeta. Aquesta cerca sobre un volum de fitxers tan gran pot consumir una gran quantitat de temps, sobretot si no se sap on es va desar tal projecte en el seu moment.

Analitzant la situació, observem dos problemes principals a afrontar. Primer, centralitzar tots els continguts de forma que la cerca es simplifiqui

exponencialment. Segon, permetre realitzar cerques sobre aquests continguts, no depenent només del nom del fitxer (informació *metadata* o del contingut de fitxer).

Si aconseguim solucionar aquests problemes, el mateix treballador podria realitzar aquesta cerca des de la seva estació de treball, basant-se en informació addicional, com ara la data de creació o paraules que recordi associades amb tal projecte, que puguin trobar-se dins dels arxius de comentaris.

Aquesta millora proporcionaria un augment en l'eficiència de les cerques de l'empresa molt important, millorant la productivitat de forma considerable al llarg del temps.

3.2.2. Requisits funcionals d'usuari

Tot seguit identifiquem els requisits funcionals del sistema. Aquests requisits seran diferents segons el tipus d'usuari.

Login d'usuari: A partir d'un formulari que es mostrarà a l'iniciar l'aplicació, s'identificarà si l'usuari està registrat al sistema. Tot seguit, es comprovarà el tipus d'usuari del que es tracta, per a poder assignar els permisos adequats.

Exploració: De forma similar a l'explorador d'un sistema operatiu, haurà de permetre la iteració dels diferents directoris situats a les estacions de treball. Volem que la integració dels directoris de diferents màquines de la xarxa sigui transparent a l'usuari.

Cerca de fitxers: Haurà de permetre cercar fitxers dins la base de dades, prenent el nom de fitxer o informació de *metadata.* També permetrà cercar a partir del text contingut dins dels fitxers.

Descàrrega de fitxers: L'usuari haurà de poder descarregar els fitxers desitjats a la seva màquina.

Vista prèvia de fitxers: Es mostrarà una vista prèvia d'imatges en forma d'icona en miniatura. Pels fitxers de text o PDF, es mostrarà la primera pàgina.

Permisos de fitxers: El gestor haurà de poder denegar l'accés, vista prèvia o descàrrega dels fitxers desitjats, potser amb una opció al menú contextual.

Actualització de l'índex: El sistema haurà d'actualitzar la base de dades de forma transparent a l'usuari, a mida que aquesta es vagi iterant. Els canvis realitzats es mostraran en pantalla si cal.

Actualització manual: Si l'administrador ho desitja, es podrà realitzar una actualització manual de la base de dades.

Gestió d'usuaris: S'haurà de poder crear usuaris nous, així com a eliminar els actuals, o modificar el mot de pas o tipus d'usuari.

Manteniment de la xarxa: L'administrador haurà de poder gestionar i mantenir les màquines en xarxa, per tal de poder fer la seva presència transparent als altres usuaris.

Vistos els requisits funcionals, tot seguit establirem a quins usuaris pertanyen cada un. Cal notar que hi ha un requisit funcional que existirà en tots els casos, ja que és intrínsec a l'**exploració**. Es tracta de l'**actualització de l'índex**. Com que aquesta acció es realitzarà de forma automàtica i transparent a l'usuari, i sempre quan s'iterin els diferents continguts, no la hem comentat en cap cas.

3.2.2.1. Usuari no registrat

Un usuari no registrat no podrà realitzar cap de les accions llistades. Per tant, haurà de demanar al gestor que se li creï una compta d'usuari d'un tipus determinat (treballador o administrador).

3.2.2.2. Usuari treballador

Aquest usuari haurà de poder fer **login** per a identificar-se. Un cop dins el sistema, haurà de poder **explorar** el sistema de fitxers a través de la interfície gràfica. Si ho vol, podrà **realitzar cerques** dins del sistema de fitxers. Es mostrarà una **vista prèvia** dels fitxers sobre els quals tingui permisos d'accés, que podrà **descarregar** al seu ordenador.

Tindrà accés a la **Gestió d'usuaris** només per al seu perfil, i amb possibilitat de modificar el seu mot de pas com a única opció.

3.2.2.3. Usuari gestor

L'usuari gestor haurà de poder realitzar totes les accions de l'usuari treballador (**login, exploració, cerques, vista prèvia** i **descarregar**). A més, haurà de poder **modificar els permisos dels fitxers** i **gestionar els usuaris** amb possibilitat de modificar informació de mot de pas o tipus d'usuari de tots els usuaris.

3.2.2.4. Usuari administrador

Aquest usuari mantindrà les accions base de l'usuari treballador (**login, exploració, cerques, vista prèvia** i **descarregar**). Això a banda, haurà de poder realitzar **actualitzacions manuals** de l'índex per a mantenir la base de dades. A més a més, aquest usuari **mantindrà la xarxa**, assegurant-se que els continguts de les diverses estacions són accessibles de forma transparent a través del programa.

Com l'usuari treballador, podrà modificar el seu mot de pas a través de la **gestió d'usuaris**.

3.2.3. Casos d’ús

Prenent els requisits funcionals especificats al principi de l’apartat, definirem diferents diagrames de casos d’ús per a cada usuari. Aquests diagrames podran contenir els següents actors:

- Usuari no registrat
- Usuari treballador
- Usuari gestor
- Usuari administrador

Els casos d’ús venen determinats pels requisits funcionals, però no sempre són els mateixos. Veiem quins són segons el tipus d’usuari.

3.2.3.1. Usuari no registrat

L’usuari no registrat encara no té cap entrada a la base de dades d’usuaris i, per tant, no pot accedir al sistema.

Figura 8. Diagrama d’ús per a la creació d’una compta d’usuari.

3.2.3.2. Usuari treballador

L'usuari treballador, es aquell que ja ha estat donat d'alta com a usuari registrat per un gestor i per tant està autoritzar a fer ús del portal.

Tot seguit veiem un diagrama que engloba tots els casos d'ús d'aquest usuari, els quals son comuns a la resta d'usuaris.

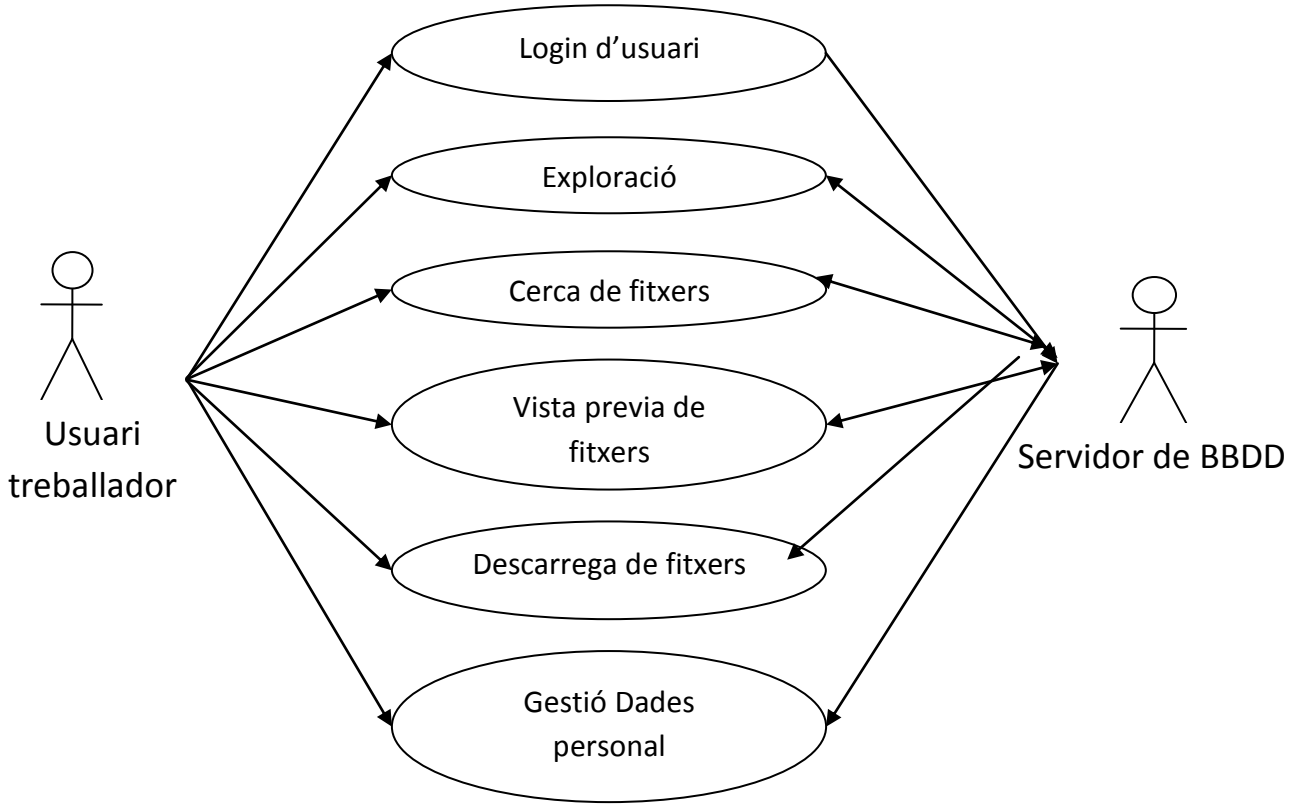


Figura 10. Casos d'ús comuns a tots els usuaris.

3.2.3.3. Usuari Gestor

Aquest usuari hereta els mateixos casos d'ús que l'usuari treballador (mostrats a la *figura 10)* i a més disposa de els seus específics. El diagrama és similar al vist anteriorment. A més, haurà de permetre la **gestió d'usuaris** i la **modificació de permisos de fitxers**.

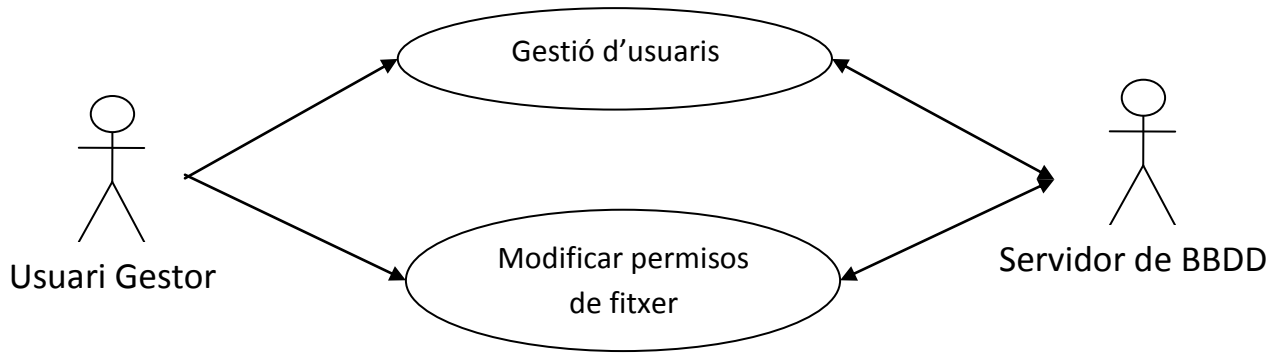


Figura 11. Casos d'ús per l'usuari gestor.

3.2.3.4. Usuari administrador

Al igual que l'usuari gestor, l'administrador també manté els casos d'ús comuns (*figura 10)*. A més ha més, ha de ser capaç de realitzar **actualitzacions de l'índex sota petició** i **mantenir la xarxa**.

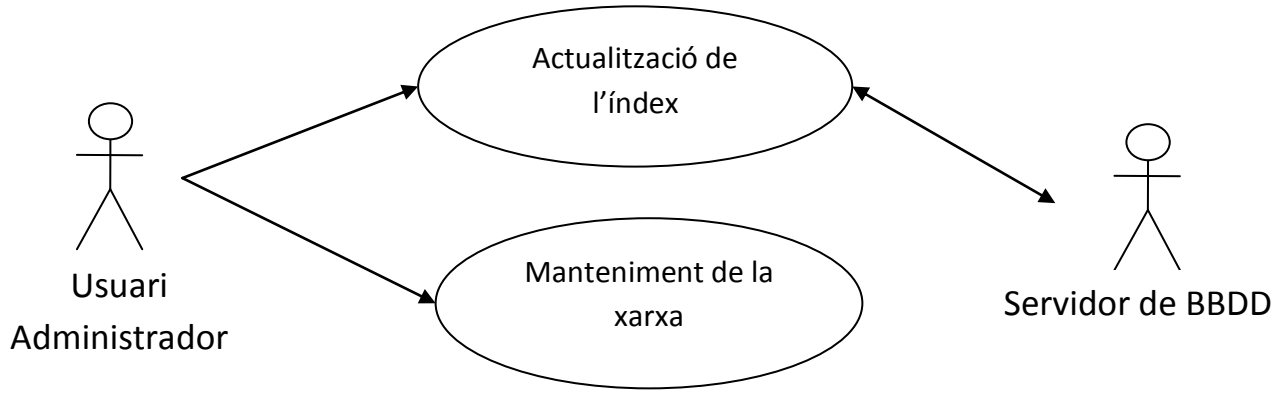


Figura 12. Casos d'ús per l'usuari administrador.

3.3. Diagrames de seqüències

Ara mitjançant diagrames de seqüència descriurem certs casos de ús que considerem rellevants. Gracies a aquests diagrames veurem clarament la interacció dels objectes a treves del temps.

3.3.1. Cas d''us d' inici de sessió

L'usuari no identificat introdueix el seu nom i el mot de pas assignat. El servidor web comprova si les dades són correctes a la base de dades, i atorga el nivell d'accés corresponent.

Actors : Usuari no registrat.

Descripció: L'usuari s'identifica al sistema per accedir a l'aplicació.

Actor	Sistema
1. **L'usuari introdueix el nom i la contrasenya.**	2. El servidor comprova a la BBDD si la contrasenya és correcta.
	3. Si les dades són correctes, el servidor web demana les dades d'usuari.
5. **L'usuari accedeix a la seva interfície gràfica.**	4. El servidor atorga el nivell d'accés corresponent a l'usuari

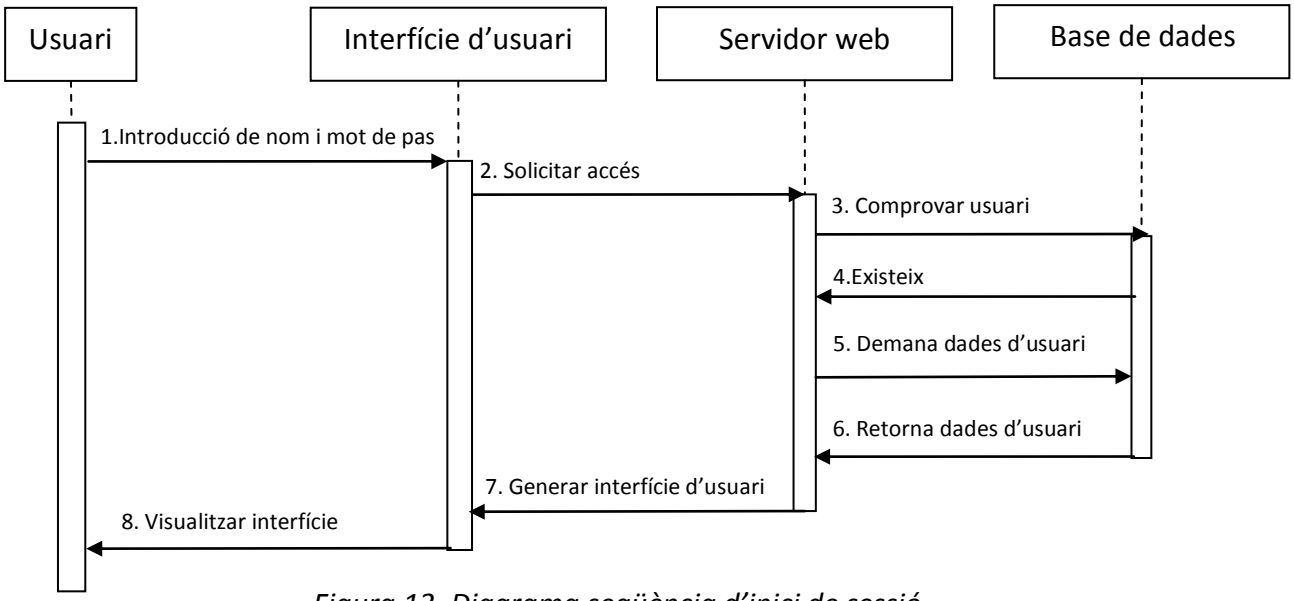


Figura 13. Diagrama seqüència d'inici de sessió.

3.3.2. Cas d'ús explorar

El servidor mostra el contingut de la base de dades en forma d'arbre. L'usuari selecciona un directori o fitxer de la llista de continguts. Es mostra el contingut del directori o la vista prèvia del fitxer segons l'element seleccionat. Si l'usuari no té permís sobre un fitxer, aquest no es mostra en pantalla. Cal notar que mentre es realitza aquesta exploració, el gestor de BBDD consulta el sistema de fitxers per a corregir l'índex en cas que hi hagi elements obsolets. Aquests canvis es mostraran en pantalla.

Actors : Usuari treballador, gestor i administrador.

Descripció: L'usuari escull un directori o fitxer i el sistema en mostra el contingut o vista prèvia.

Actor	Sistema
1. L'usuari selecciona l'element desitjat.	2. El servidor consulta els continguts a la BBDD.

	3. Si és un fitxer, es recupera una vista prèvia. Si és un directori, es cerca el contingut.
5. L'usuari rep la informació demanada.	4. S'actualitza la interfície d'usuari amb la informació obtinguda. Es mostren només els elements sobre els que l'usuari té permís. 6. En segon pla, el gestor de BBDD consulta el sistema de fitxers i notifica al servidor de possibles diferències.
8. El client ajusta la vista de continguts amb els canvis rebuts.	7. El servidor notifica al client de les diferències amb el sistema de fitxers. 9. El gestor de BBDD actualitza l'índex amb les diferències observades.

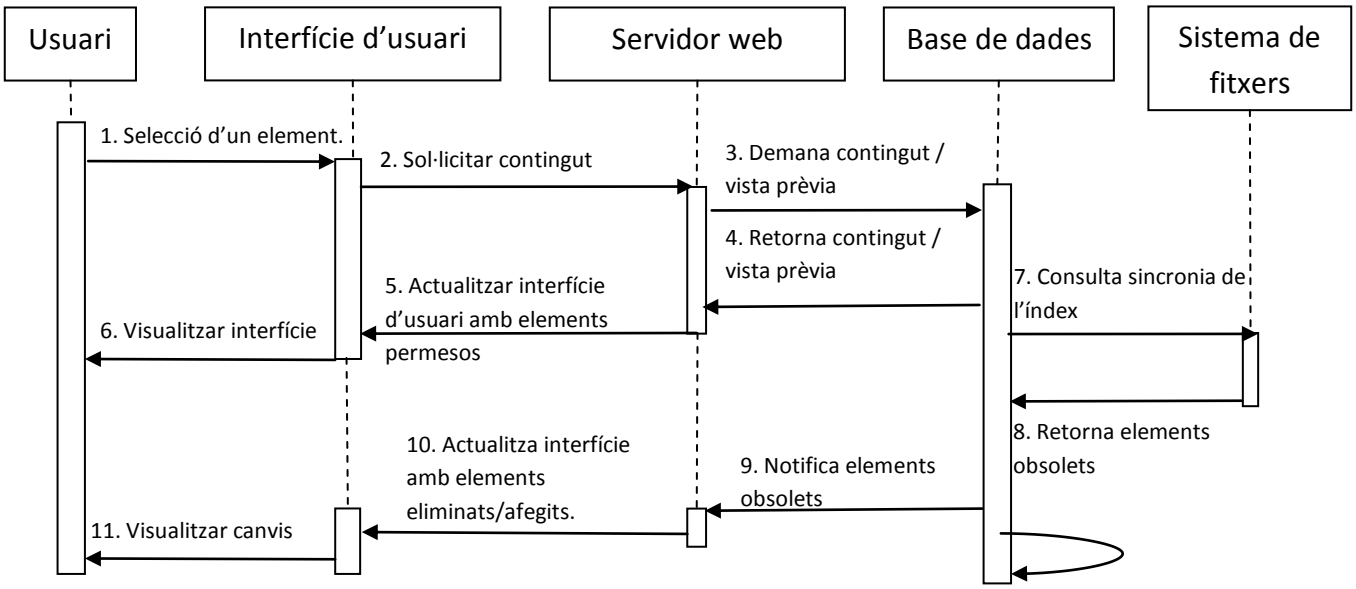


Figura 14. Diagrama seqüència per l'exploració d'elements.

3.3.3. Cas d'us cerca

El client mostra un formulari amb camps per especificar la informació a cercar (per nom de fitxer, data de creació, contingut de text...). L'usuari emplena els camps desitjats i envia el formulari. El gestor de BBDD consulta l'índex i retorna els resultats pertinents (no es mostren aquells fitxers sobre els quals l'usuari no té permís). Els resultats es mostren en forma de llista sobre la qual l'usuari pot fer clic (**cas explorar**). Es mostra el contingut del directori o la vista prèvia del fitxer segons l'element seleccionat.

Actors : Usuari treballador, gestor i administrador.

Descripció: L'usuari introdueix els paràmetres de cerca desitjats. El servidor retorna una llista amb les coincidències.

Actor	Sistema
1. L'usuari introdueix els paràmetres de cerca.	2. El servidor consulta els continguts a la BBDD.
	3. El gestor de BBDD retorna les coincidències.
5. L'usuari rep la informació demanada.	4. S'actualitza la interfície d'usuari amb la llista de coincidències sobre les quals els usuaris té permís.

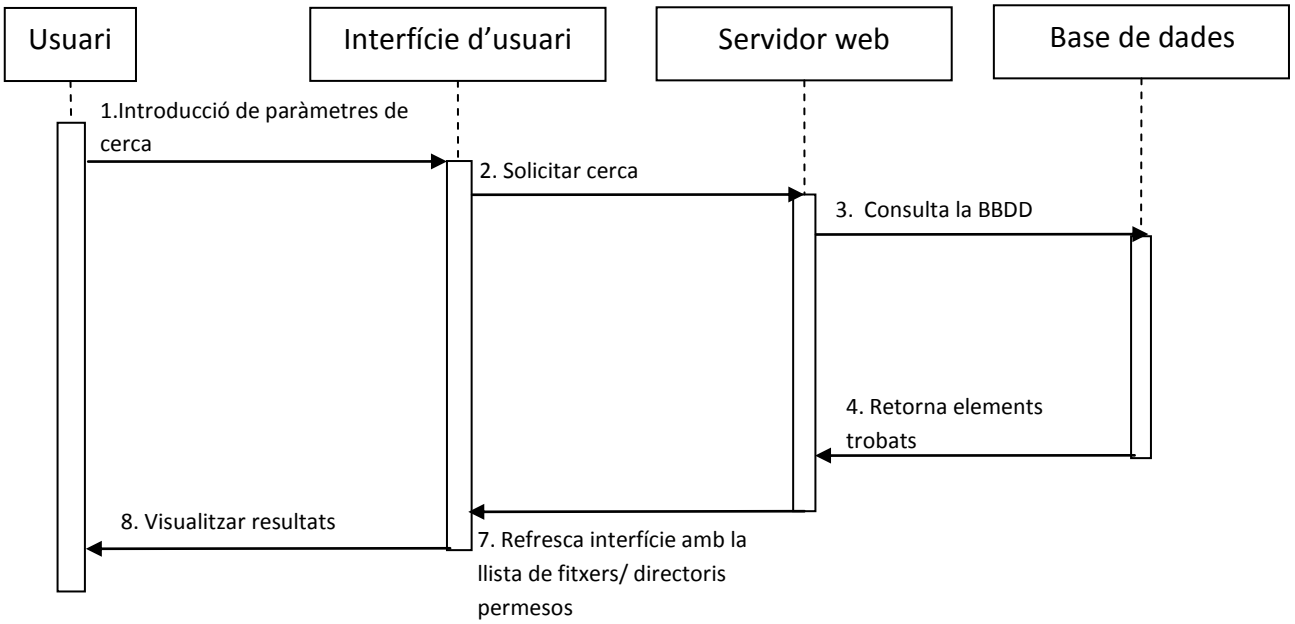


Figura 15. Diagrama seqüència per la cerca d'elements.

3.3.4. Cas d'us descàrrega:

Actors : Usuari treballador, gestor i administrador.

Descripció: L'usuari selecciona un fitxer a descarregar. El Servidor web demana la ubicació real al disc dur al gestor BDD i l'obté del sistema de fitxers per a la seva descàrrega.

Actor	Sistema
1. L'usuari selecciona el fitxer a descarregar.	2. El servidor consulta la ubicació dins del disc dur a la BBDD.
	3. El gestor de BBDD retorna el camí al fitxer.
5. L'usuari descarrega el fitxer.	4. El servidor obté el fitxer del sistema de fitxers i el retorna a l'usuari.

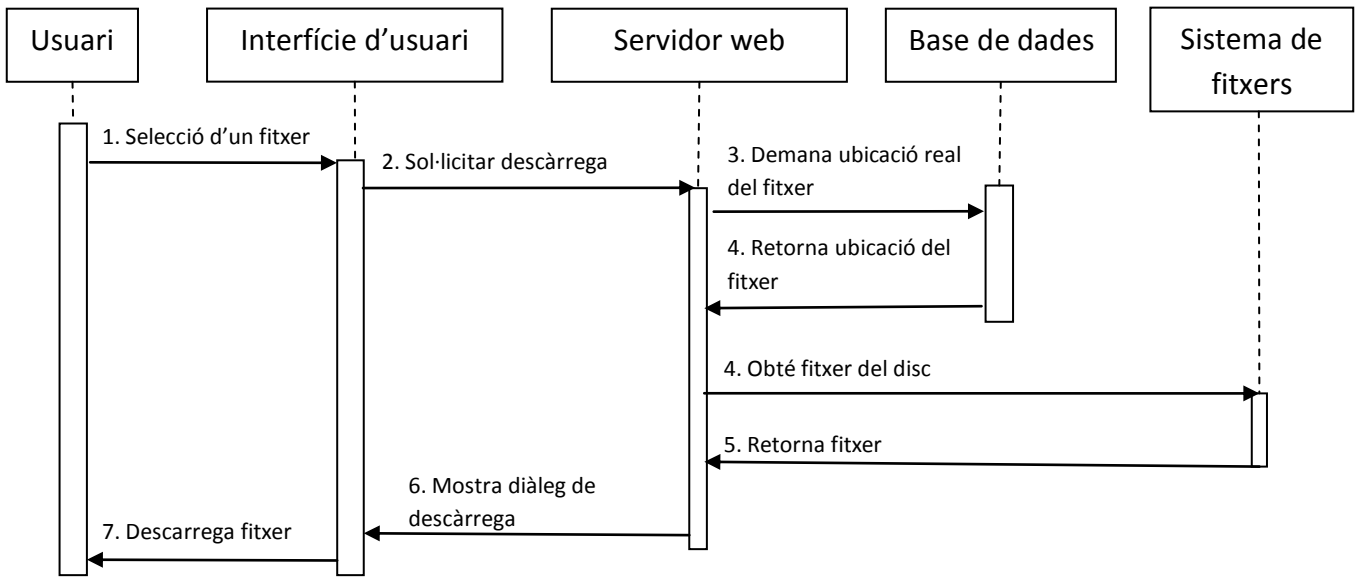


Figura 16. Diagrama seqüència per la descàrrega de fitxers.

3.3.5. Cas d'ús gestió d'usuaris:

Actors : Usuari gestor.

Descripció: L'usuari selecciona l'acció a realitzar i omple el formulari corresponent (afegir usuari, esborrar usuari, modificar usuari). El Servidor Web recol·lecta la informació proporcionada i envia la petició al Gestor de Base de Dades, que modifica l'índex d'usuaris de la forma pertinent, o retorna un error si la modificació no es pot dur a terme.

Actor	Sistema
1. **L'usuari selecciona l'acció desitjada.**	2. El servidor recol·lecta la informació i envia la petició.
	3. El gestor de BBDD intenta modificar l'índex de la forma corresponent (afegir usuari, esborrar usuari, modificar atributs de l'usuari).
	4. El gestor de BBDD retorna el missatge d'èxit o error al servidor segons si s'ha pogut completar la petició.
6. **L'usuari rep el missatge d'èxit o error.**	5. El servidor web genera el missatge corresponent.

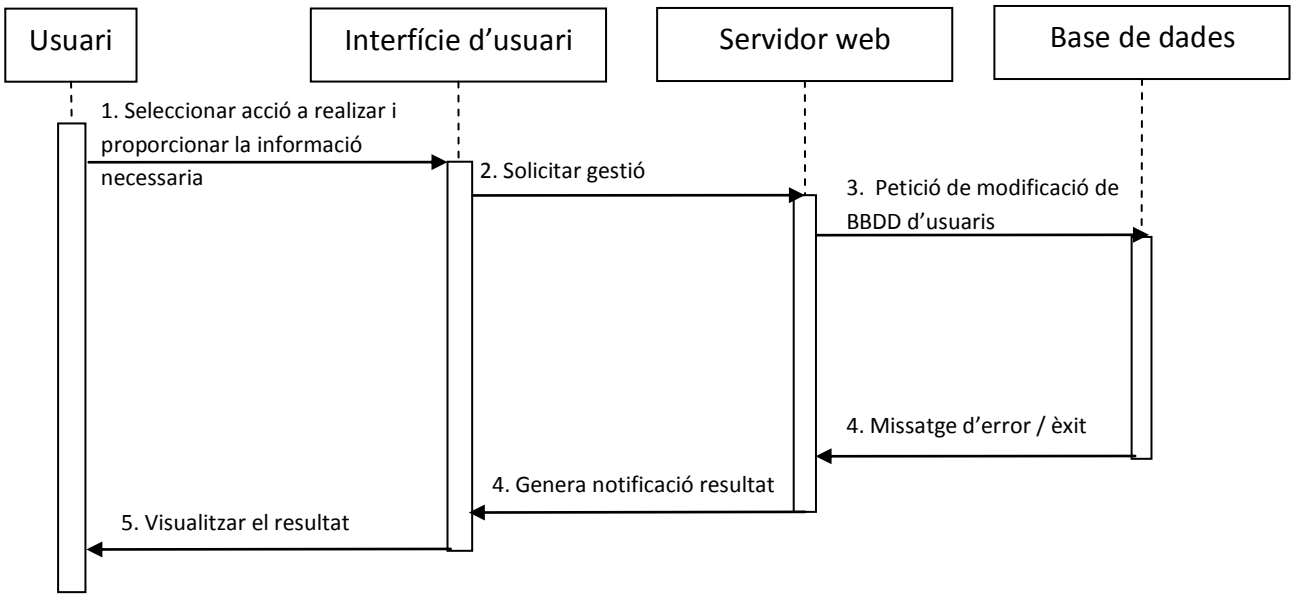


Figura 17. Diagrama seqüència per la gestió d'usuaris

3.3.6. Cas d'ús permisos de fitxers:

Actors : Usuari gestor.

Descripció: L'usuari selecciona quins treballadors tindran accés a cert fitxer o directori.

Actor	Sistema
1. L'usuari selecciona quins usuaris podran accedir a l'element.	2. El servidor envia la petició al gestor de BBDD.
	3. El gestor de BBDD intenta afegir/treure usuaris de la llista de permisos de l'element.

	4. El gestor de BBDD retorna el missatge d'èxit o error al servidor segons si s'ha pogut completar la petició.
6. L'usuari rep el missatge d'èxit o error.	5. El servidor web genera el missatge corresponent.

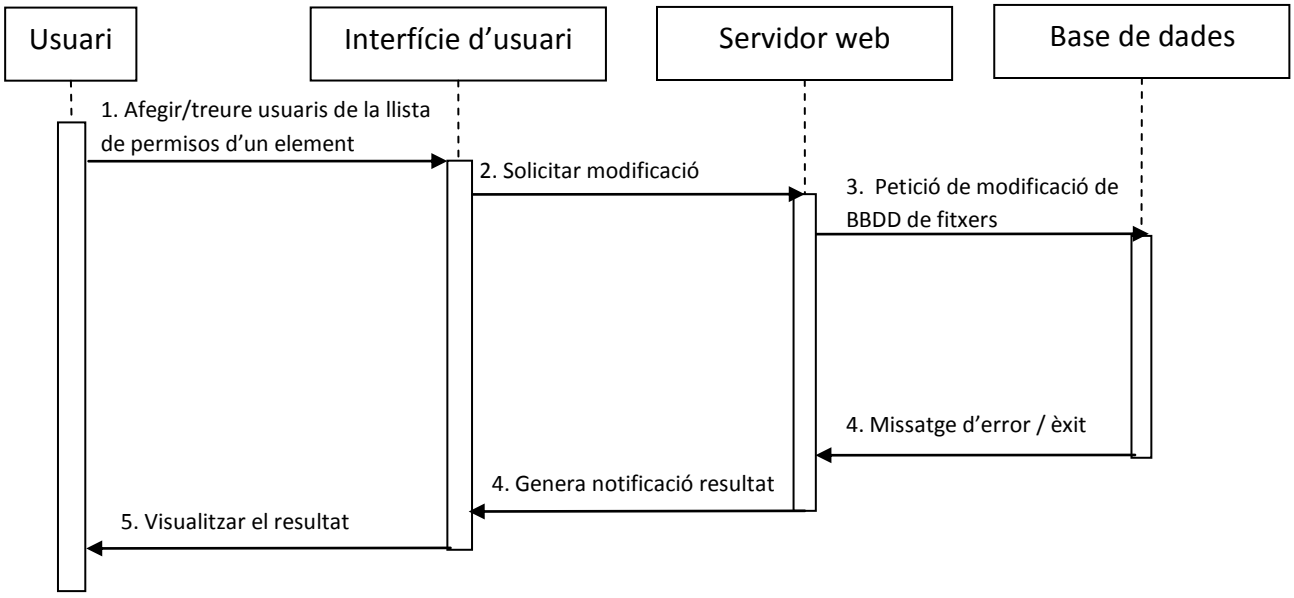


Figura 18. Diagrama seqüència per la modificació de permisos

3.3.7. Cas d'ús actualització de l'índex:

Actors : Usuari administrador.

Descripció: L'usuari realitza una petició d'actualització de l'índex. El Servidor Web ho comunica al Gestor de BBDD, que contrasta els continguts de l'índex amb el sistema de fitxers a la xarxa i actualitza les entrades necessàries.

Actor	Sistema
1. L'usuari demana una actualització manual de la BBDD.	2. El servidor web envia la petició al gestor de BBDD.
	3. El gestor de BBDD compara les entrades de l'índex amb els elements al sistema de fitxers.
	4. El gestor de BBDD actualitza els elements obsolets i retorna un missatge d'èxit. Si no es pot actualitzar, retorna un missatge d'error.
6. L'usuari rep la notificació d'èxit o error.	5. El servidor genera el missatge de resultat corresponent.

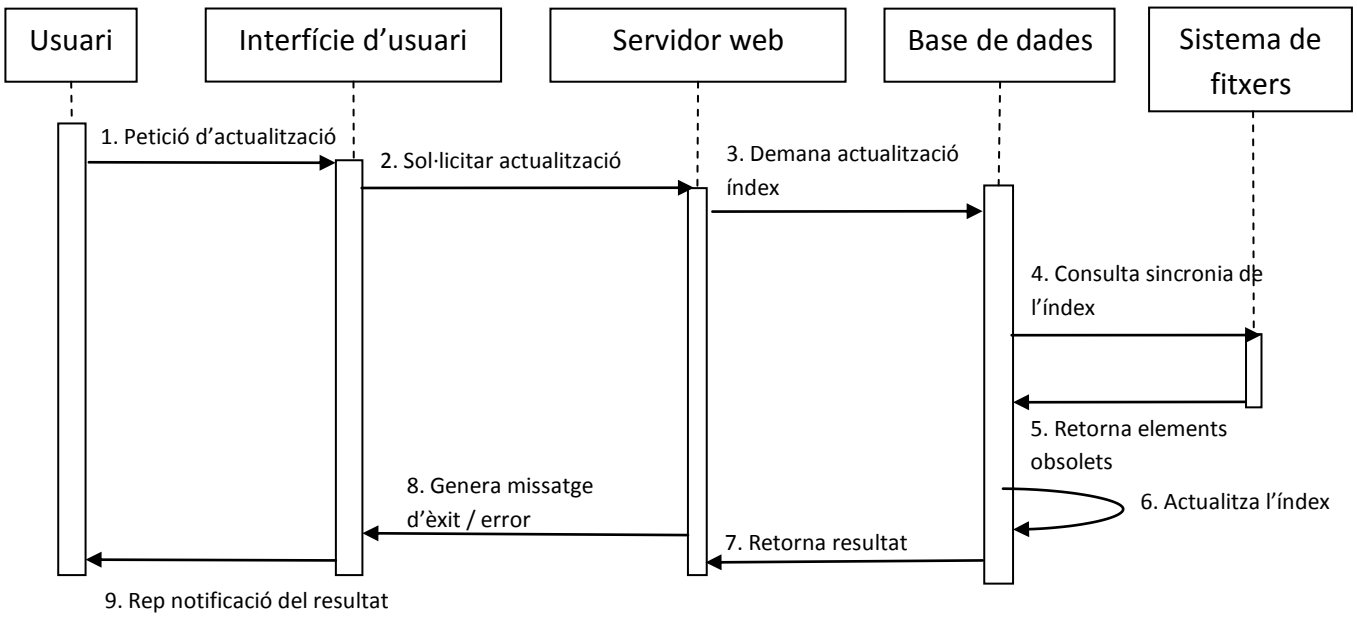


Figura 19. Diagrama seqüència per l'actualització manual d'índex

3.3.8. Cas d'ús manteniment de la xarxa:

En cas que l'administrador cregui pertinent modificar l'estructura de la xarxa local, aquest modificarà directament la configuració de directoris de forma transparent als altres actors del sistema. Per aquesta raó no s'ha afegit cap diagrama de seqüència en aquest marc.

Les operacions típiques seran afegir o eliminar una partició o estació de treball del sistema.

3.4. Requeriments no funcionals

En la enginyeria del software, un requeriment no funcional és aquell que especifica criteris que poden esser utilitzats per a avaluar un sistema, en comptes dels seus comportaments específics.

Se solen ponderar varis requisits habituals, com ara el cost, la seguretat, l'estabilitat o el rendiment (altres requisits podrien ser la documentació, plataforma, compatibilitat, suport... però no són tan populars). Per al nostre projecte, valorarem tots els esmentats menys el cost, ja que la flexibilitat d'implementació del sistema esmentada a l'estudi de viabilitat fa que el preu pugui ser mínim si realment fa falta.

3.4.1. Requeriments de seguretat

Degut a l'àmbit local i corporatiu del sistema, les mesures de seguretat del sistema es basaran en la codificació de les contrasenyes i la base de dades d'usuaris. Aquesta codificació es realitzarà amb una substitució de caràcters junt amb la codificació en base 64.

3.4.2. Requeriments de recursos utilitzats

Al tenir una base de dades que anirà creixent amb el temps, farà falta que els recursos en forma d'espai en disc utilitzats per l'índex siguin

raonables en funció del desenvolupament de la tecnologia. Com s'ha vist en l'estudi de viabilitat, aquest requeriment es compleix sense problemes.

3.4.3. Requeriments de desenvolupament

Al tractar-se d'un treball de final de carrera, avaluat en un termini de temps concret, existeix un requeriment de temps de desenvolupament de 6 mesos aproximadament. Alhora, és bastant important que aquest desenvolupament sigui progressiu per a la seva correcta avaluació i correcció per part del tutor.

3.4.4. Requeriments de rendiment

Per tal que l'aplicació que crearem impliqui una millora respecte les eines actuals de cerca incorporades a la majoria de sistemes operatius, els temps de treball hauran de ser menors als que proporcionaria una cerca directa del disc dur des de qualsevol estació. Si aquest requisit no es pogués complir, el software seguiria sent atractiu degut a la centralització de la informació, accessible des de qualsevol lloc a través d'un navegador web.

4. Disseny i Implementació del Sistema

4.1. Introducció

En aquesta etapa del nostre projecte estudiarem com complir els requeriments establerts en el capítol anterior, amb suficient detall com per a poder realitzar una implementació física del sistema. Especificarem així l'estructura final del programa, essent de vital importància prendre cura en cada element.

Durant aquest disseny, començarem per establir la configuració de maquinari necessària per a poder utilitzar el nostre software. Tot seguit, definirem les diferents capes que formaran l'aplicació (Entorn d'usuari, motor d'aplicació, capa de dades). Després, es farà un petit estudi sobre l'arquitectura de l'aplicació, on es mencionarà la interacció entre les diferents capes i l'estructuració del codi.

Un punt important del nostre disseny serà l'estudi de les diferents interfícies d'usuari, on es definirà aquest element per a cada perfil d'usuari, acompanyant les explicacions amb esquemes que representin el disseny desitjat.

Finalment, es definirà l'estructura de fitxers que contindrà el nostre codi.

4.2. Configuració de la plataforma

Abans de dissenyar la resta del sistema, cal tenir clar quines tecnologies es faran servir. A banda dels possibles llenguatges de programació a utilitzar, escollirem les llibreries necessàries, sistemes operatius i possibles interfícies de comunicació en xarxa.

Pel que fa al maquinari, vindrà definit en major part per a la configuració de programari i interfícies especificada.

4.2.1. Serveis i sistemes operatius

- Per a la configuració del servidor web a la màquina servidor, es farà servir **Apache**. S'ha pres aquesta decisió degut a la popularitat d'aquest software gratuït, que comporta la disposició de molta documentació en cas necessari.
- Per a l'entorn de proves, es muntarà una màquina virtual amb **WMware Player**. Aquesta màquina prendrà el lloc de l'ordenador servidor en aquest cas.
- Per a les estacions de treball client, es podrà fer servir qualsevol sistema operatiu. Per a l'entorn de proves utilitzarem **Microsoft Windows 7**, ja que permet utilitzar més navegadors per comprovar la compatibilitat del sistema.
- Per a la màquina servidor, tant en l'entorn de proves com en una implementació final, es farà servir qualsevol distribució de **Linux**. Per a l'entorn de proves es farà servir **OpenSuse 11.1**.
- Pel compartiment d'arxius en xarxa amb màquines executant Linux o Windows independentment, s'utilitzaran el servei **Samba**. A la màquina servidor es farà servir el **mount** per a poder accedir als directoris en xarxa de forma transparent al usuari final com si es tractessin de directoris locals.

4.2.2. Llenguatges de programació i llibreries

- Per a crear la interfície d'usuari s'utilitzarà **JavaScript** junt amb la llibreria **Sarissa**, que ens permetrà realitzar consultes al servidor en temps real i tractar la informació rebuda en format DOM*.
- Per al codi que comportarà el servidor web i el gestor de BBDD s'utilitzarà **PHP**. Es farà servir la implementació del *parser** XML

inclosa amb PHP5, per a tractar la informació en format d'arbre DOM.

- La BBDD, com veurem més endavant, s'emmagatzemarà en arxius XML i codificació **UTF-8**.
- Les imatges en miniatura es generaran amb la llibreria per PHP **Imagick.**

4.3. Capes de l'aplicació

Hem pensat el desenvolupament d'aquesta aplicació mitjançant diferents capes. Així, dividirem l'estructura del sistema en tres parts: la **capa d'usuari**, la capa **motor** i la **capa de dades**.

Aquesta abstracció ens permet desenvolupar el sistema de forma modular, aïllant els problemes que puguin sorgir o les possibles modificacions al codi i fent més fàcil d'entendre l'arquitectura interna del software.

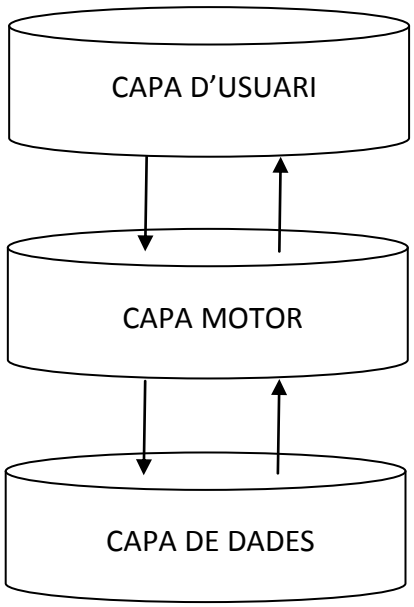


Figura 20. Abstracció del sistema en capes

A la figura anterior veiem una representació de les tres capes existents, amb les seves possibles interaccions. Es important notar que la capa usuari i la capa de dades mai es comunicaran de forma directa. En comptes d'això, es fa servir la capa motor com a intermediària.

4.3.1. Capa d'usuari

Aquesta capa s'encarregarà de generar la interfície d'usuari, mostrant els continguts pertinents a cada cas d'ús en cada moment.

Recollirà la informació proporcionada per l'usuari a partir de formularis (elements **FORM** en html) i, després de validar-la, l'enviarà al servidor. Un cop rebuda una resposta, s'encarregarà de generar els nous continguts en conseqüència (missatge d'error o actualització del contingut en pantalla).

Per a generar la interfície, utilitzarà HTML combinat amb JavaScript. Aquesta combinació, coneguda com a **DHTML**, ens permet obtenir una interfície més interactiva i dinàmica que la tradicional. Per exemple, l'usuari podrà redimensionar els elements en pantalla a voluntat, a part de visualitzar efectes gràfics de tot tipus (aparició d'elements en pantalla de forma suau, mostra de continguts nous de forma progressiva...). Això a banda, estilarem la pàgina amb fulles CSS per a obtenir un resultat encara més atractiu.

La comunicació amb la capa motor es realitzarà utilitzant la llibreria JavaScript **Sarissa**, que ens permetrà enviar i rebre missatges utilitzant peticions **XMLHttp**, mantenint la compatibilitat amb la majoria de navegadors actuals.

4.3.2. Capa motor

Aquesta capa es podria considerar l'espina dorsal del nostre sistema, ja que farà les funcions de "pont" entre la capa d'usuari i la capa de dades.

S'utilitzarà **PHP** per a gestionar les peticions rebudes des de la capa d'usuari, en format de petició **POST** o **GET**. Aquestes peticions, realitzades pel client en format **XMLHttp**, cridaran diferents parts del codi de la capa motor, que retornarà una resposta en format XML i codificació **UTF-8**.

Quan faci falta, es consultarà la capa de dades i s'interpretarà el contingut dels diferents índex gràcies a les funcions de la classe **DOM-PHP**. Tanmateix, es podran modificar els continguts de la base de dades a través d'aquestes funcions.

A més, a mida que es vagi accedint als continguts de la BBDD, la capa motor realitzarà comprovacions de sincronia entre les entrades de l'índex i els fitxers a la capa de dades, ajustant les diferencies i notificant a la capa d'usuari per a actualitzar la interfície si fa falta.

Per últim, aquesta capa decidirà quins continguts generar a la interfície d'usuari en funció del nivell d'accés corresponent. Aquests continguts es mostraran de forma dinàmica a la capa d'usuari.

4.3.3. Capa de dades

En aquesta capa es trobaran les dades accessibles per la capa motor. Dividirem aquesta capa en dos tipus de continguts: base de dades i sistema de fitxers.

La base de dades estarà formada per dos fitxers: l'índex d'usuaris i l'índex de fitxers, que contindran la relació d'usuaris i l'arbre de directoris al sistema de fitxers respectivament. L'estructura d'aquests índex, en format **XML**, quedarà explicada més endavant.

El sistema de fitxers residirà en diferents estacions en xarxa o simplement al mateix servidor. L'usuari administrador s'haurà d'encarregar d'utilitzar la eina de gestió de particions remotes **NFS** per a aconseguir simular un entorn de directoris locals a la màquina servidor.

4.4. Estructura de la Base de Dades

Degut a l'estructura en forma d'arbre inherent a un sistema de fitxers, la forma més simple de representar tal contingut passa per utilitzar també una vista en arbre.

La naturalesa del llenguatge **XML**, creat pensant en una vista d'arbre, ens permet tractar cada fitxer com a un node **DOM**, que pot tenir varis atributs (mida, tipus de fitxer, paraules clau, icona...). Això ens permet replicar l'estructura d'un sistema de fitxers de forma molt simple, alhora que afegim informació adicional.

A part de l'estructura de fitxers, que es trobarà dins de l'arxiu **index.xml**, necessitarem una relació d'usuaris amb la informació sobre tipus d'usuari, contrasenya, etc. Aquest fitxer tindrà una estructura més semblant a una taula, on cada node d'usuari estarà al mateix nivell (sota el node arrel) i no tindrà nodes fill. Els atributs de cada usuari es desaran en forma d'atributs del node xml. El fitxer d'usuaris s'anomenarà **users.xml** i haurà d'esser encriptat degut a la seva naturalesa.

Tot seguit veiem l'estructura d'arbre esmentada anteriorment del fitxer **index.xml**. A la figura es pot observar una vista de fitxers amb l'habitual arbre de directoris a l'esquerra. A baix, tenim el format resultant d'aquesta estructura real al emmagatzemar-la en format **XML**. Cada element d'aquest arbre representa un node al fitxer XML.

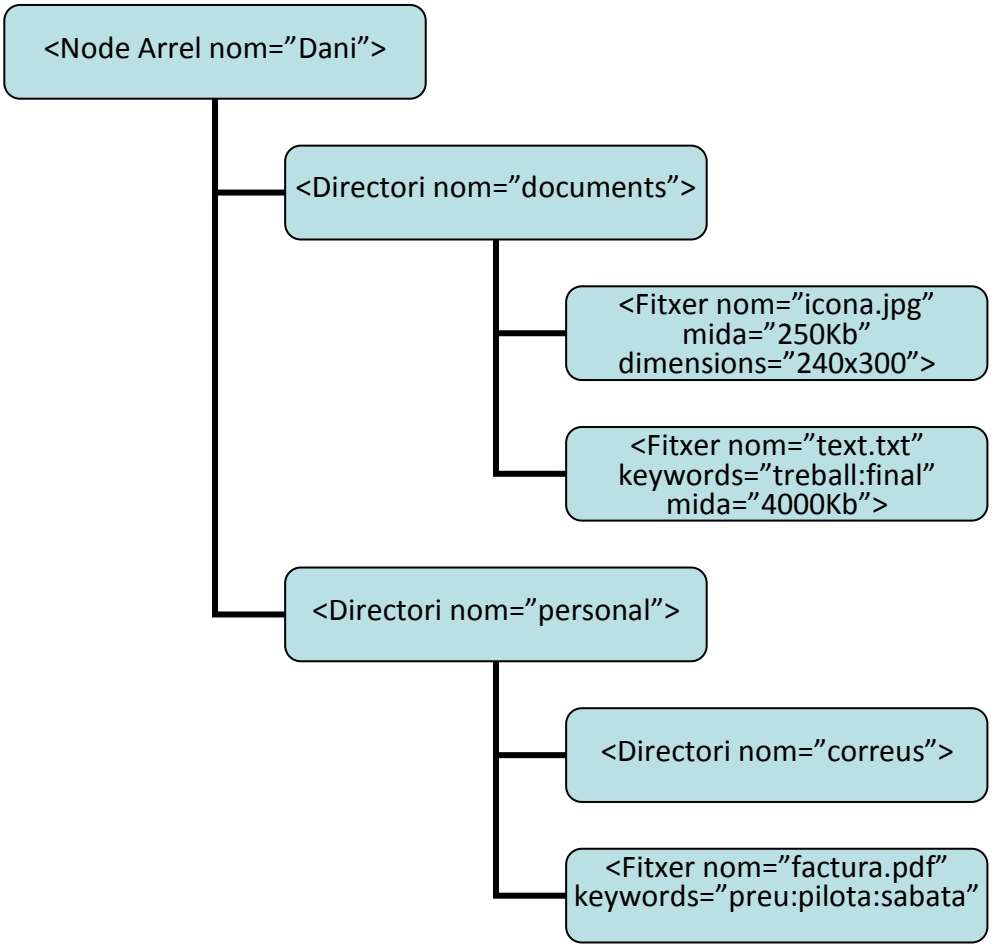


Figura 21. Estructura de nodes a l'índex de fitxers.

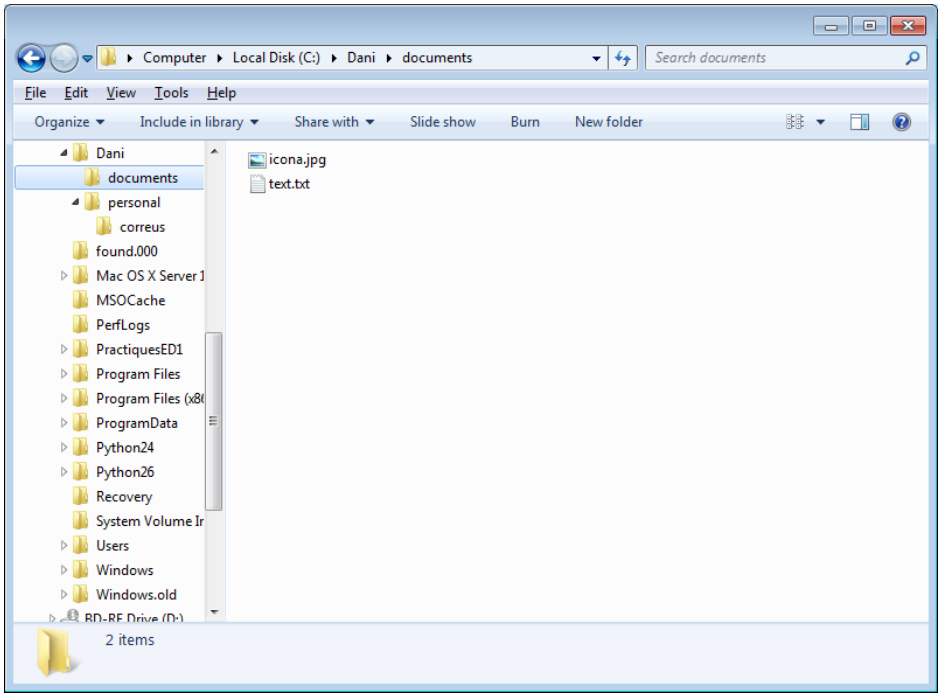


Figura 22. Vista del sistema de fitxers representat en forma de nodes a la figura 21, amb arrel al directori "Dani".

Pel que fa al fitxer **users.xml** que contindrà la llista d'usuaris amb els seus atributs, la seva estructura seguirà el següent esquema:

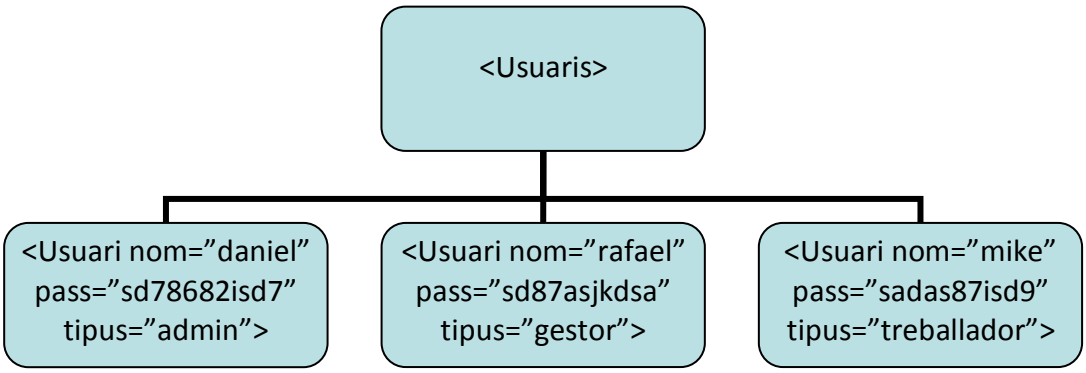


Figura 23. Estructura de nodes a l'índex d'usuaris.

Com veiem, es tracta simplement d'una taula representada en format de nodes, gràcies als atributs d'aquests. Per a augmentar la seguretat del sistema, la paraula de pas estarà codificada amb una combinació de l'algorisme **md5** i la conversió a **base 64**.

4.4.1. Disseny de l´índex de fitxers

Hem vist l'estructura que seguiran ambdós índex, però encara no hem especificat els atributs que podrà tenir cada element de la Base de Dades. Tot seguit veiem una taula amb la relació de possibles atributs i la seva descripció corresponent per als dos tipus de node (fitxer i directori).

Atribut	Descripció
Nom del node	<dir>
Url	Camí del directori al sistema de fitxers.
Id	Identitat única al node.
Group	Grup d'accés. Estableix qui podrà veure el directori.

Taula 4. Descripció del node de directori

Atribut	Descripció
Nom del node	<file>
Url	Camí del fitxer al sistema de fitxers.
Id	Identitat única al node.
Group	Grup d'accés. Estableix qui podrà veure el fitxer.
Type	Tipus de fitxer (segons l'especificació MIME).
Size	Mida del fitxer en bytes.
Dim	Tamany de la imatge.
Thumb	Icona de la imatge o vista preliminar del document PDF en base64
Keys	Paraules clau dins del fitxer
Mod	Data de modificació del fitxer

Taula 5. Descripció del node fitxer

4.4.2. Disseny de l'índex d'usuaris

Pel que fa a l'índex d'usuaris, cada node "<User>" penjarà del node arrel "<Users>" i contindrà els següents possibles atributs:

Atribut	Descripció
Nom del node	<user>
Id	Identitat única al node.
Name	Nom de l'usuari

Type	Grup al cual pertany l'usuari
Pass	Paraula de pas codificada
Email	Correu electrònic
	Tamany de la imatge.
Fullname	Nom complert especificat per l'usuari
Lastname	Cognoms de l'usuari

Taula 6. Descripció del node d'usuari

4.5. Arquitectura de l'aplicació

Com veiem a la *figura 24*, cada capa del sistema consta de varis mòduls que permetran realitzar les tasques necessàries en cada moment.

Per a implementar aquests mòduls utilitzarem un sistema de programació orientada a objectes (**OOM**). Així, representarem cada mòdul en forma de classe, que a la vegada podrà constar de diferents classes "fill". Aquestes subclasses seran clarament dividides en pantalla i ens permetran agrupar les funcions del nostre sistema de forma molt específica i clara respecte a la seva funcionalitat.

Per exemple, el mòdul "Explorador" consta de 3 subclasses. Treeview, Listview i Infoview. Cada una d'aquestes classes ocuparà un lloc en pantalla i gestionarà informació diferent (Treeview per a la llista de directoris, Listview per a la llista de fitxers i Infoview per a mostrar la informació més concreta de cada fitxer), però tots estaran relacionats. Més endavant estudiarem aquestes relacions de forma més concreta.

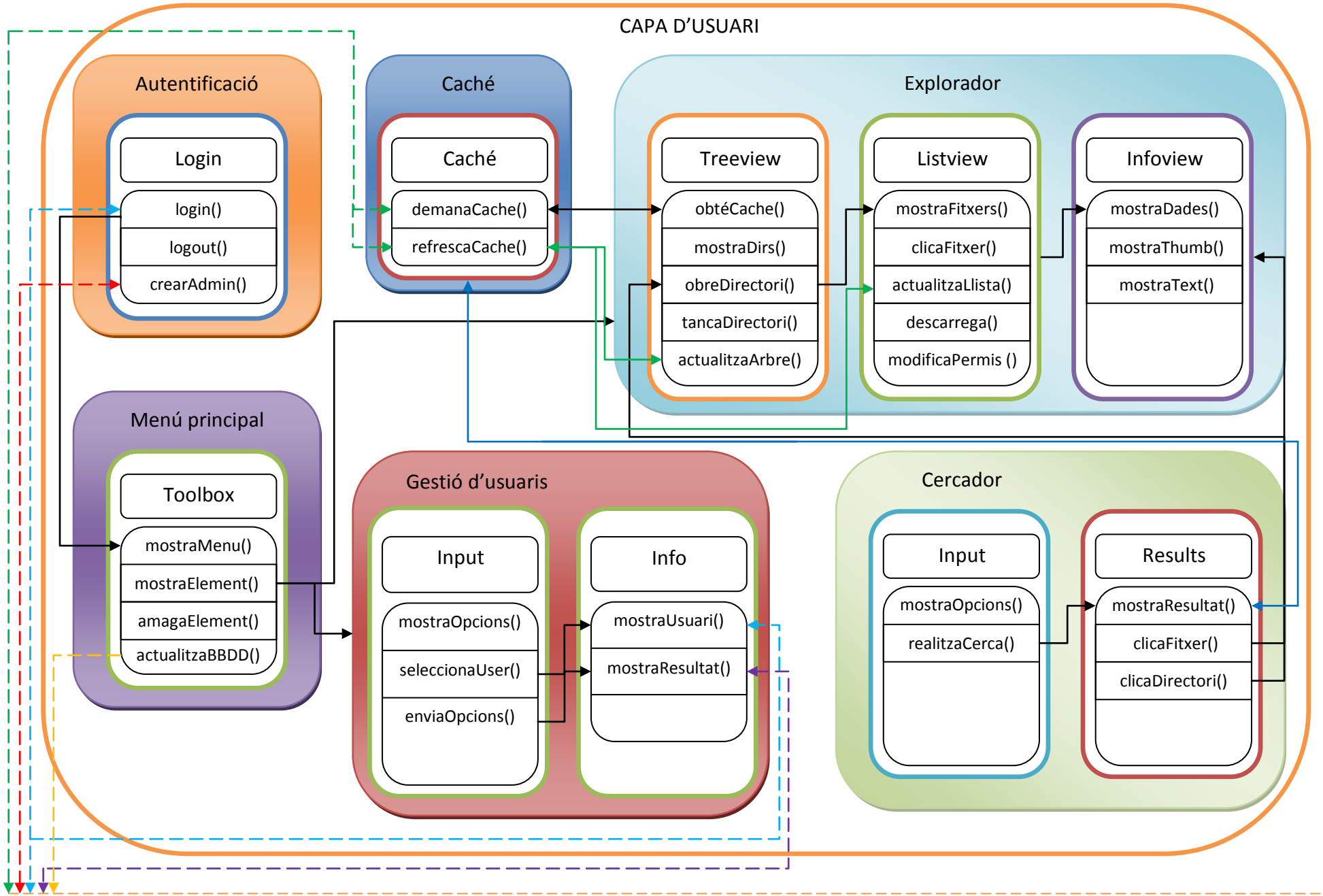
Observem que la majoria de mòduls es troben a la capa d'usuari. Això es deu a que la majoria de les funcions del sistema s'han definit per a requerir interacció humana i , per tant, tindran un lloc a la interfície gràfica.

Un mòdul a notar és la caché. Aquest recurs ens permet minimitzar les interaccions amb el servidor (i per tant els temps d'accés). El seu funcionament és simple. Al iniciar el mòdul Explorador, el client desarà una còpia de l'índex de fitxers de forma local sobre la que es realitzaran les iteracions i cerques. La capa motor serà l'encarregada d'anar refrescant aquesta caché i alhora sincronitzant la base de dades de forma transparent.

Podríem discutir la posició del mòdul de caché a la capa d'usuari o a la capa motor, ja que les seves funcions són transparents a l'usuari. El que ens ha fet decidir incloure-la a la capa d'usuari és el fet que aquesta residirà al client en tot moment. Tot i així, un sistema amb aquesta caché a la capa motor també tindria sentit.

Abans de passar a la descripció detallada de cada mòdul i les seves interaccions, veurem la figura que mostra tots aquests elements i les corresponents classes. Cada classe mostra una llista dels seus mètodes més importants.

Cal notar que aquest és un disseny preliminar, i podrà variar durant el curs del desenvolupament. És possible que s'afegeixin o treguin mètodes a l'estructura final, però la jerarquia i interaccions dels diferents mòduls de ben segur no canviaran massa.


CAPA D'USUARI
Autentificació
Login
login()
logout()
crearAdmin()
Caché
Caché
demanaCache()
refrescaCache()
Explorador
Treeview
obtéCache()
mostraDirs()
obreDirectori()
tancaDirectori()
actualitzaArbre()
Listview
mostraFitxers()
clicaFitxer()
actualitzaLlista()
descarrega()
modificaPermis ()
Infoview
mostraDades()
mostraThumb()
mostraText()
Menú principal
Toolbox
mostraMenu()
mostraElement()
amagaElement()
actualitzaBBDD()
Gestió d'usuaris
Input
mostraOpcions()
seleccionaUser()
enviaOpcions()
Info
mostraUsuari()
mostraResultat()
Cercador
Input
mostraOpcions()
realitzaCerca()
Results
mostraResultat()
clicaFitxer()
clicaDirectori()

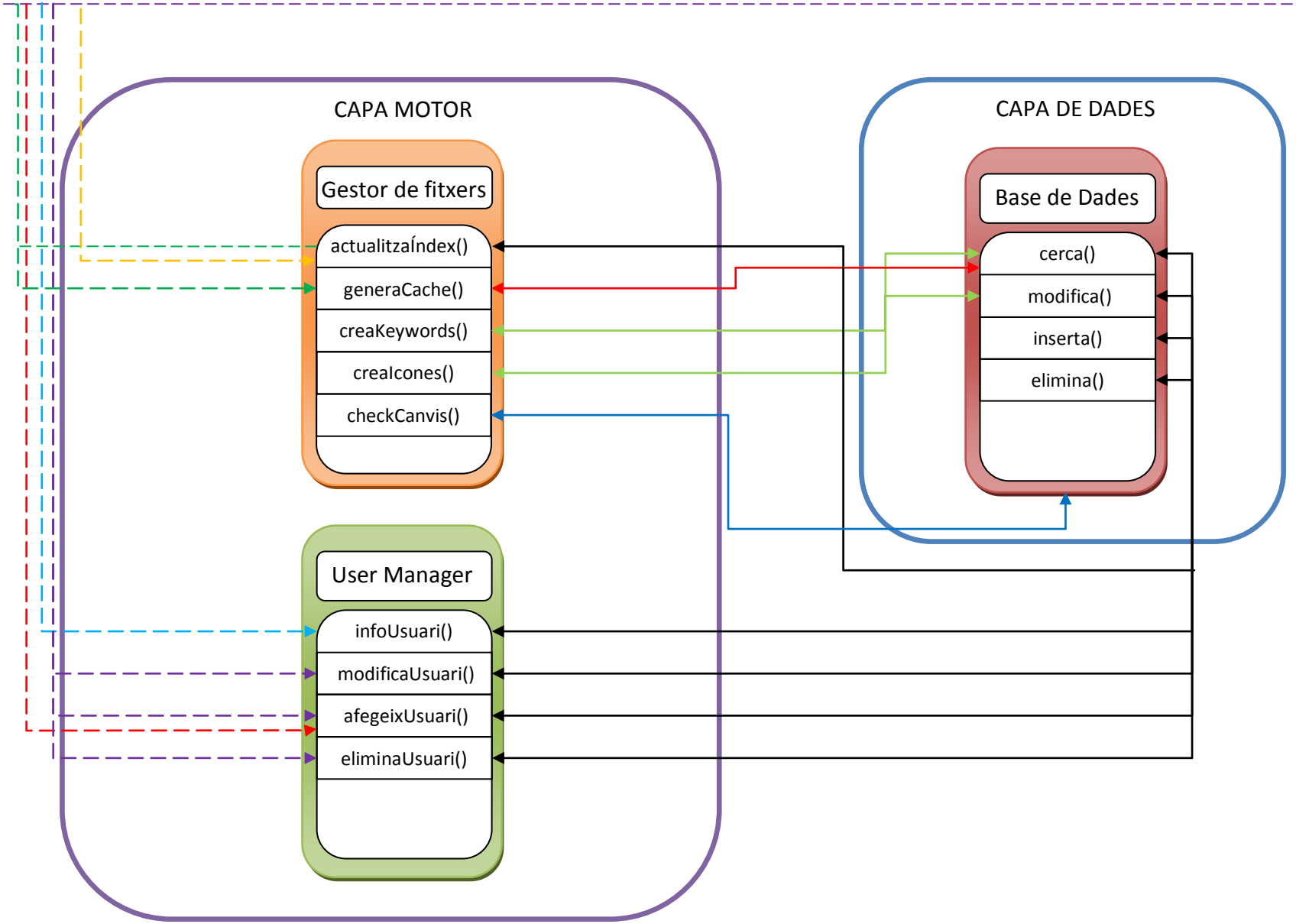


Figura 24. Disseny de capes i classes del sistema, amb la seva corresponent interacció.

4.5.1. Disseny dels mòduls

Tot seguit farem una petita llista de tots els mòduls vistos anteriorment per cada capa, junt amb una explicació dels seus mètodes principals i les interaccions amb altres elements del sistema.

4.5.1.1. Capa d'Usuari

Login:

Aquest mòdul serà el que l'usuari veurà només arrencar el sistema. Comprovarà les dades d'usuari (nom i mot de pas) interactuant amb la capa motor i atorgarà el nivell d'accés corresponent. L'execució del mètode login() passarà el control al mòdul d'interfície.

Menú o toolbox:

Generarà un menú principal que permetrà escollir quin mòdul es desitja carregar. Contindrà mètodes per a cridar la generació de la gestió d'usuaris, l'explorador i el mòdul de cerca, segons la opció escollida. A part de la selecció del mòdul corresponent, l'usuari administrador podrà actualitzar manualment l'índex de fitxers gràcies a una entrada d'aquest menú.

Explorador:

Generarà la interfície que ens permetrà navegar a través del sistema de fitxers indexat a la BBDD. Està dividit en tres blocs: **Treeview**, per a mostrar l'estructura de directoris, **Listview**, que mostrarà els fitxers de cada directori i **Infoview**, que mostrarà la informació avançada del fitxer seleccionat. Obtindrà la informació directament de la caché i podrà ser cridat tant pel corresponent element al menu principal com pel mòdul de cerca (al demanar més informació sobre els resultats).

Cercador:

Generarà la interfície que permetrà realitzar cerques, dividida en dos blocs. La subclasse **Input** mostrarà i gestionarà els formularis amb els paràmetres de cerca, mentre que la subclasse **Results** realitzarà la cerca sobre la informació de la caché i mostrarà les coincidències trobades. A més a més, la interacció amb els elements retornats permetrà la crida del mòdul explorador per a mostrar més informació.

Gestor d'usuaris:

Generarà la interfície per a modificar la base de dades d'usuaris. Al igual que el cercador, estarà dividit en un bloc d'**Input** per a l'entrada dels paràmetres i un de **Info** que mostrarà el resultat de la operació actual, o la informació de l'usuari seleccionat per a editar. Al realitzar modificacions a la BBDD, es tractarà de l'únic mòdul visible a l'usuari amb accés directe a la capa motor a part del mòdul de **login**.

Caché:

Interactuarà amb la capa motor quan sigui necessari per a generar una còpia de la BBDD de fitxers o per a refrescar aquesta còpia local a mida que es vagi iterant l'estructura de directoris. Serà transparent a l'usuari final (no tindrà cap representació a la interfície gràfica) i evitarà que altres mòduls hagin d'accedir a la capa motor per a realitzar funcions de consulta. Això accelerarà totes aquestes transaccions de forma considerable.

4.5.1.2. Capa motor

Gestor de fitxers:

Aquest mòdul s'encarregarà de mantenir l'índex de fitxers actualitzat a partir de comparacions de sincronia periòdiques i comprovacions sota petició del mòdul de **caché** (normalment quan un usuari cliqui sobre un directori). Quan es realitzin actualitzacions,

s'aprofitarà per a generar o actualitzar la llista de paraules clau i les icones dels fitxers d'imatge. A banda de tot això, el gestor de fitxers mantindrà un enllaç amb l'índex de fitxers per a comprovar si ha estat modificat en qualsevol moment (per altres instàncies del sistema executades per altres usuaris en el mateix moment) i forçar una actualització de la caché si es dona el cas.

User Manager:

Gestionarà les diferents peticions del mòdul anàlog a la capa d'usuari, comunicant-se amb la capa de dades per a dur a terme les consultes o peticions necessàries. També interactuarà amb el mòdul de login per a autentificar l'usuari.

Cal nota que s'ha nombrat el mòdul de gestió d'usuaris a la capa motor de forma diferent per a evitar confusió amb el mòdul de mateix nom a la capa d'usuari.

4.5.1.3. Capa de dades

Mòdul base de dades:

Aquest mòdul és passiu. Només interactuarà directament amb la capa motor quan rebi una petició. En tal cas retornarà informació que llegirà dels dos índex o en modificarà el contingut.

4.6. Interfície

Tot seguit definirem el disseny de la interfície gràfica per a cada mòdul present (**Explorador, Cercador, Gestor d'usuaris, Login, Menú)**. Tanmateix, mostrarem les diferents variants per a cada tipus d'usuari.

4.6.1. Login

Figura 25. Pantalla d'inici de sessió.

Aquesta és la pantalla que veuran tots els usuaris al entrar al sistema, sempre i quan ja hi hagi un usuari administrador creat. En cas que el sistema encara no hagi estat inicialitzat (creant un administrador), es mostra el següent formulari automàticament:

Figura 26. Formulari de creació del primer usuari administrador.

4.6.2. Menú principal

Hem dissenyat aquest menú amb un objectiu minimalista, per tal de poder emprar poc espai en pantalla i així tenir-lo present en tot moment mentre veiem un mòdul o altre. Creiem que substituïr tot text per una icona representativa és una bona solució. Això significa que aquestes icones han d'esser molt explícites per tal de poder transmetre la informació desitjada.

El mòdul actualment mostrat en pantalla es representa amb un canvi de color de la icona a verd. A més, cada icona té un text descriptiu que apareix al passar el ratolí per sobre d'ella.

Com que l'usuari Gestor i l'usuari Administrador tenen accés a funcionalitats úniques de cada un, aquests compten amb una interfície una mica diferent a la de l'usuari treballador. Tot seguit veiem els tres possibles casos.

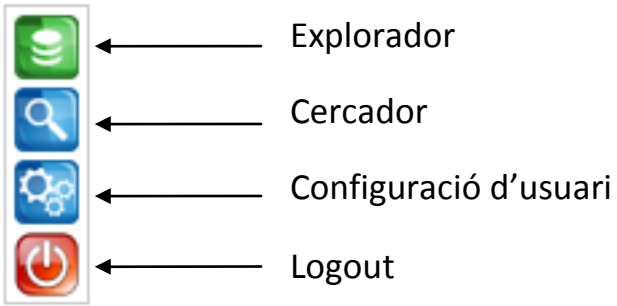


Figura 27. Menú per a un usuari treballador

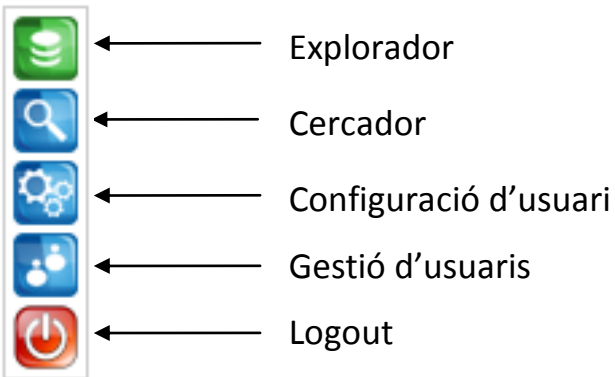


Figura 28. Menú per a un usuari gestor

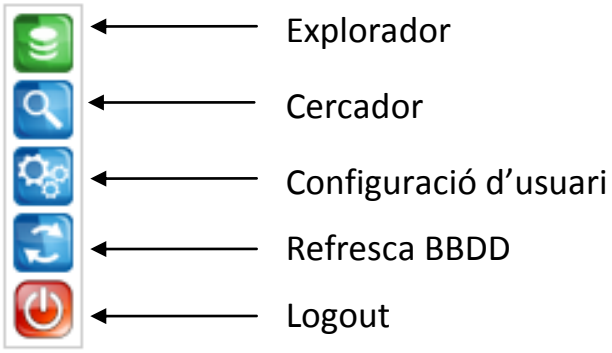


Figura 29. Menú per a un usuari administrador

4.6.3. Explorador

Aquest serà el mòdul més utilitzat i base d'altres elements com el resultat del mòdul de cerques. Per tant, hem volgut oferir el major nombre de funcions optimitzant al màxim l'espai en pantalla, però vigilant de no perdre la simplicitat i claredat que hem buscat en el disseny de la nostra interfície en tot moment.

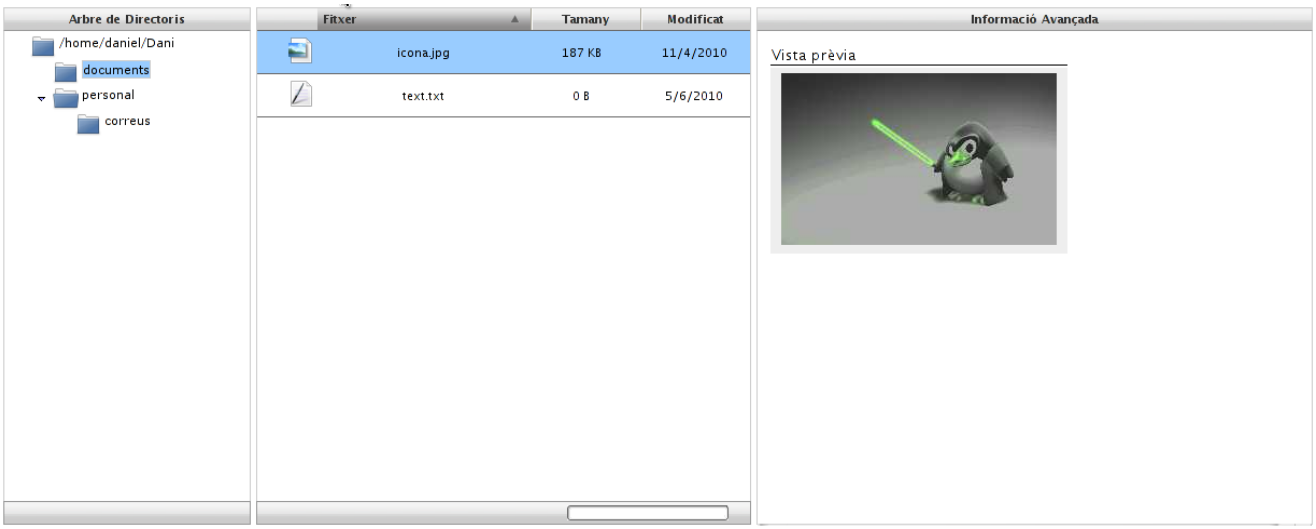


Figura 30. Interfície del mòdul explorador

A la *figura 30* veiem les tres subclasses esmentades anteriorment que formen el mòdul explorador.

A l'esquerra, tenim el **Treeview**, que ens permet iterar els directoris de la forma més ràpida fent clic sobre ells. El directori seleccionat apareixerà ressaltat en color blau. Al clicar sobre un directori, aquest "s'obrirà", afegint els seus directoris fill a la vista d'arbre actual. Si es clica sobre un directori que ja estava obert, "tancarem" l'element, amagant els seus fills de nou.

Al mig, hi ha el **Listview**, on es mostra una llista dels fitxers dins del directori actual. En un principi es mostra el nom de fitxer, tamany i data de modificació, fent possible la ordenació de la llista per qualsevol d'aquests elements. L'element seleccionat com a ordenant tindrà una capçalera a la taula una mica més fosca (per defecte el nom). A més a més, es pot ordenar de forma ascendent (fletxa cap amunt) o descendent (fletxa cap avall).

A banda dels elements comentats, cada fitxer té una icona relacionada amb el seu contingut. Si es tracta d'una imatge o document de text i es pot recuperar una versió en miniatura de l'índex de fitxers, es mostra aquesta imatge com a icona de tamany 24x24. En cas contrari, la icona mostrada és un petit dibuix representatiu del tipus de fitxer.

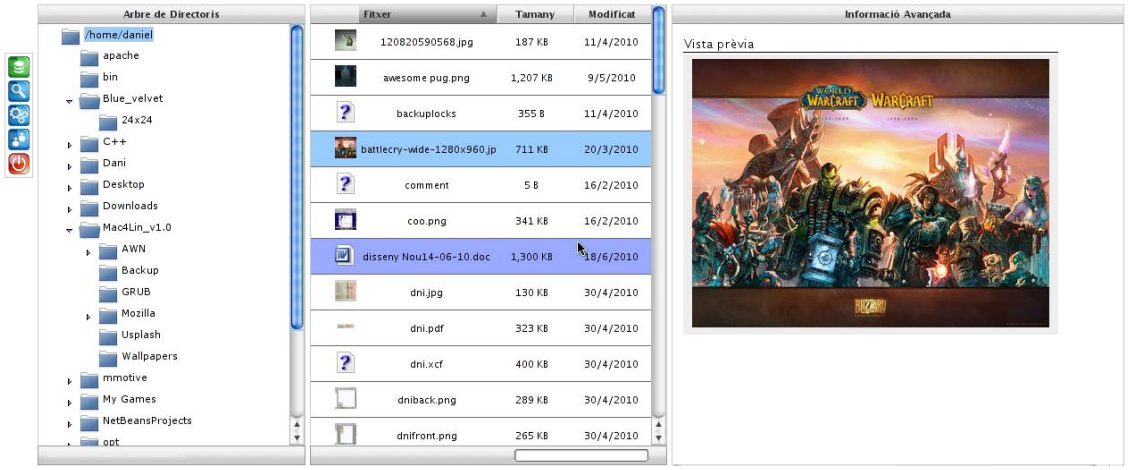


Figura 31. Explorador amb les icones pertinents i estats de color.

El fitxer seleccionat es ressalta en blau cel, mentre que el fitxer sobre el qual passi el ratolí es ressalta en lila.

A la *figura 31* es poden observar aquests colors d'estat, junt amb les icones en tamany petit que genera la capa motor.

Finalment, al fer clic dret sobre qualsevol fitxer de la llista es desplegarà un menú contextual que variarà en funció del tipus d'usuari. L'usuari veurà el següent menú que li permetrà descarregar qualsevol fitxer a la seva màquina:

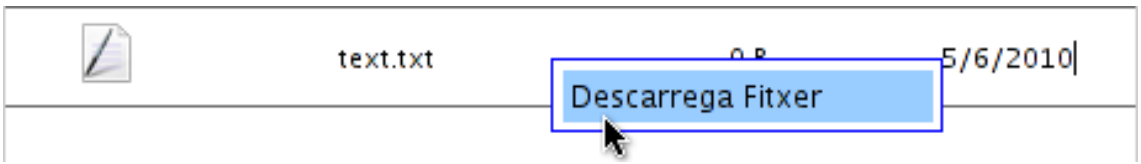


*Figura 32. Menú desplegable a la **listview** per a l'usuari treballador i administrador.*

Aquest menú contextual i els seus mètodes intrínsecs en realitat són una classe a part, cridada des del bloc **listview**. Tot i així, no ha estat inclosa al diagrama d'interaccions entre classes degut a la gran densitat d'elements que ja conté la capa d'usuari en aquesta figura. Per a simplificar-ne la representació i enteniment, s'ha representat aquest menú com a un mètode de la classe **listview**.

L'últim element de l'**Explorador** és l'**infoview**. Aquest bloc mostra una vista prèvia més en detall del fitxer, aprofitant un major ample de pantalla que el bloc **listview**. Tots els usuaris veuran el mateix resultat.

A la *figura 30* es mostra el resultat de sel.leccionar un fitxer d'imatge pel qual s'ha pogut crear una copia reduïda de 128x128 píxels. Aquesta informació és la mateixa que s'utilitza per a les icones del **listview.** En cas que es tracti d'un fitxer PDF, també es mostra una vista prèvia utilitzant la mateixa interfície, en aquest cas amb una resolució de 250x300 pixels.

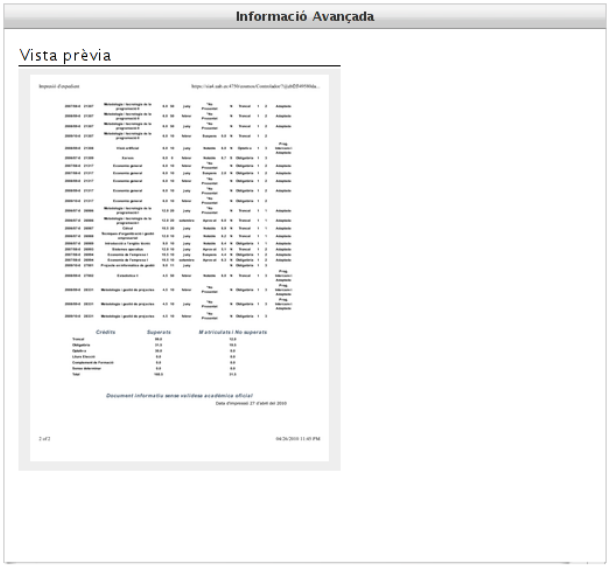


Figura 33. Vista prèvia de la primera pàgina d'un fitxer PDF.

Pel que fa als fitxers de text, veurem el seu contingut carregat en un **frame**, tal i com es mostra a la figura 34.

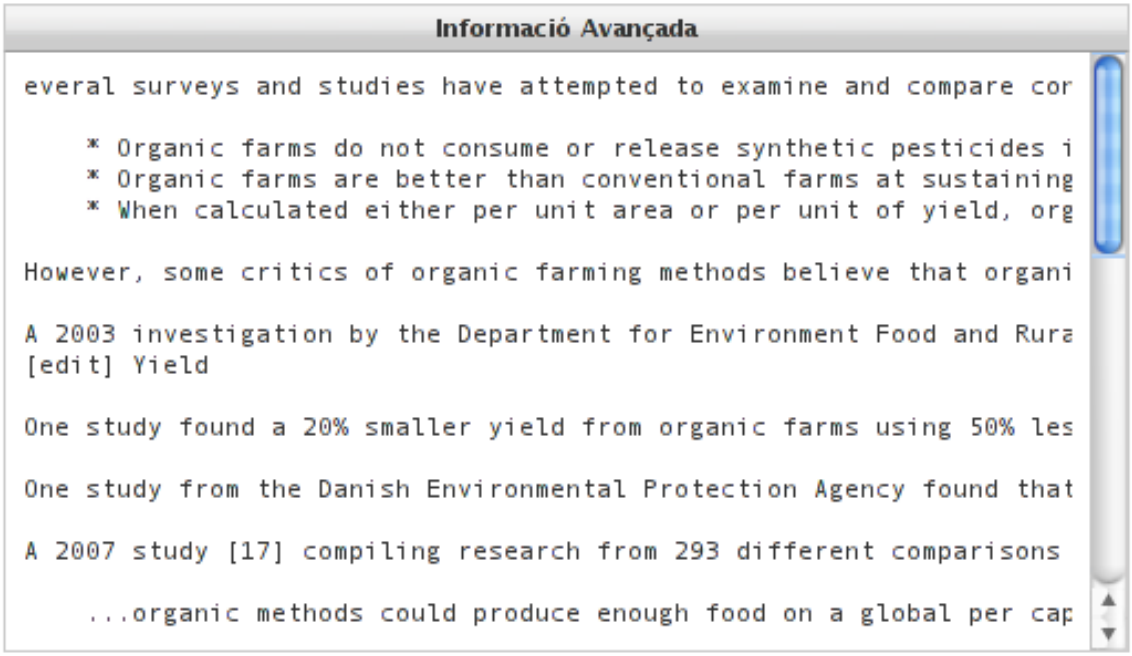


Figura 34. Vista d'un fitxer de text.

En cas que el fitxer seleccionat no tingui cap tipus d'informació addicional que extreure de l'índex, ja sigui perquè el format no és reconegut o degut a que ha fallat l'extracció del contingut per part de la capa motor, es mostrarà el missatge d'error vist a la *figura 35.*

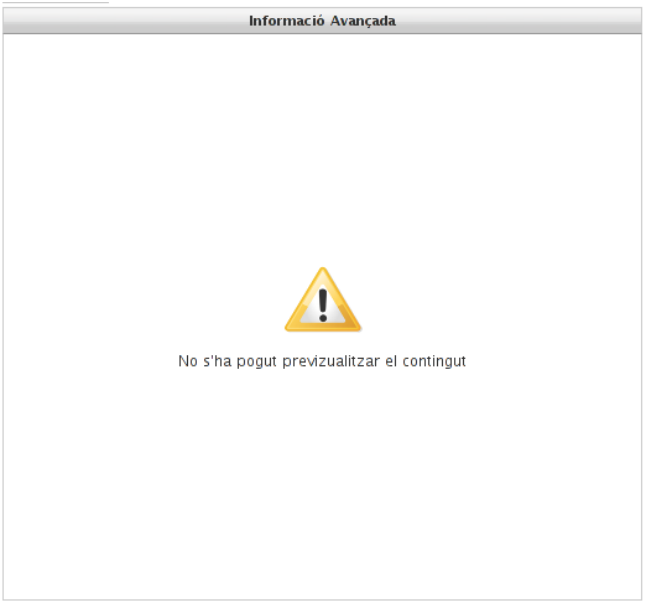


Figura 35. Missatge d'error en cas que no es pugui extraure informació addicional.

4.6.4. Cercador

Per a la cerca de fitxers hem pensat en un disseny bastant minimalista, que ens permeti aprofitar el bloc **listview** de l'explorador per a mostrar els resultats de forma directa en pantalla sense haver de carregar una pàgina nova. Podríem haver utilitzat una nova instància de la interfície de **listview** en una pàgina diferent. En canvi, hem preferit incrustar el bloc de cerca de fitxers a on aniria el **treeview** i aprofitar la interfície de l'explorador per a interactuar amb els resultats.

A la *figura 36* podem observar els diferents camps de cerca possibles. Cada camp pot ser exclós amb la casella de sel·lecció corresponent, però si realitzem una cerca sense cap casella marcada rebrem un error. Per a la sel·lecció d'una data, hem utilitzat un calendari en javascript activat al fer clic sobre la casella corresponent (*figura 37)*. Hem atorgat la possibilitat de limitar els resultats obtinguts degut a la limitació de la tecnologia HTML a l'hora de dibuixar en pantalla un gran nombre d'elements. El valor per defecte d'aquest camp és de 500 resultats en pantalla, però l'usuari el pot modificar segons la capacitat del seu sistema.

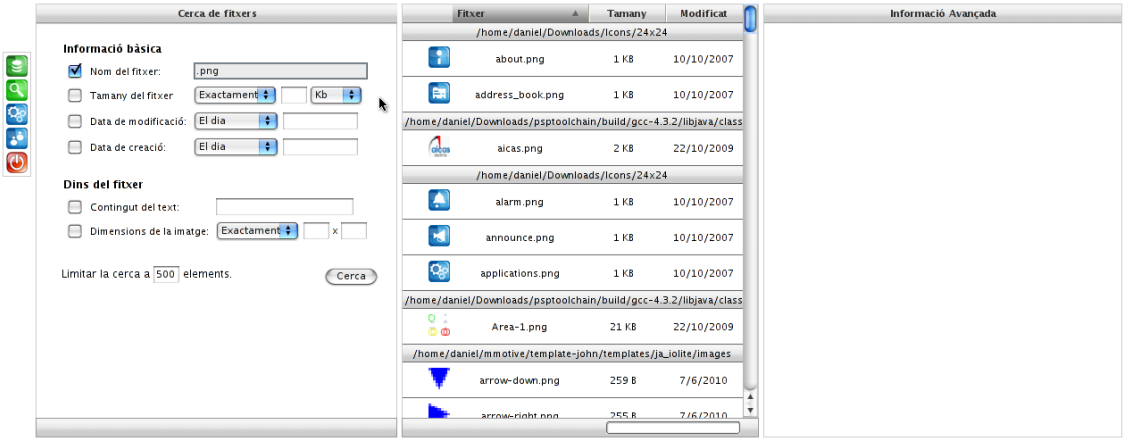


Figura 36. Mòdul de cerca en acció, usant el listview per a mostrar els resultats.

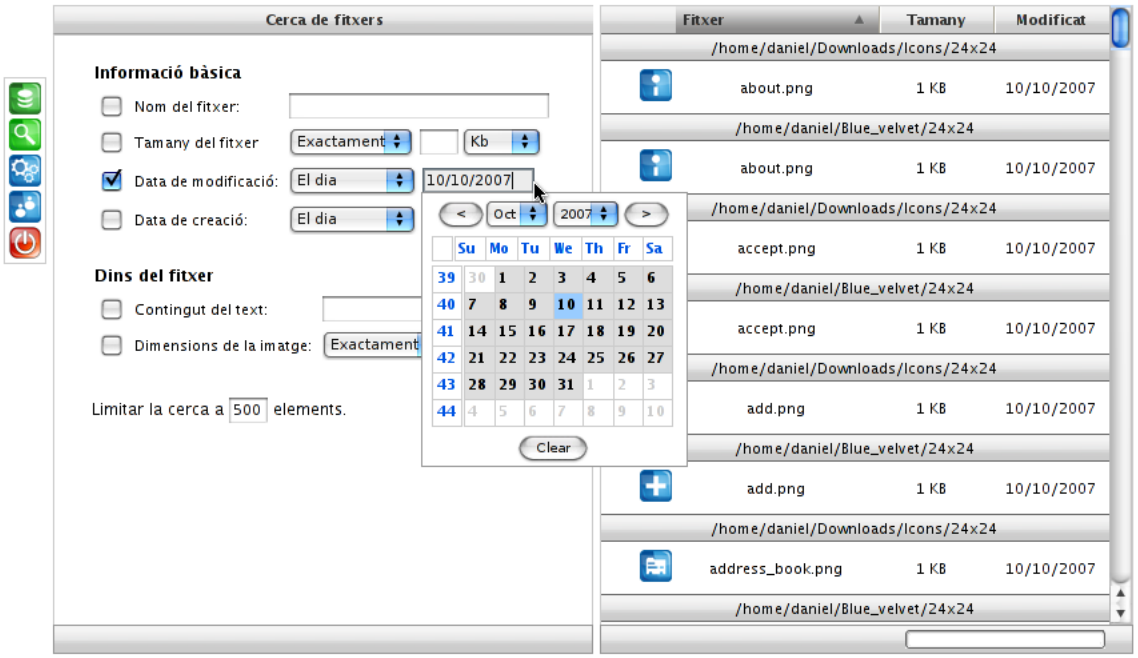


Figura 37. Cerca per data de modificació, amb sel·lecció de la data a través d'un calendari.

Una possibilitat interessant d'aquest mòdul és la cerca per paraules clau. Com hem explicat anteriorment, cada fitxer de text conté una llista de paraules clau sobre les qual l'usuari pot realitzar cerques de contingut. Totes aquestes cerques són quasi instantànees, ja que es realitzen sobre la cache en memòria.

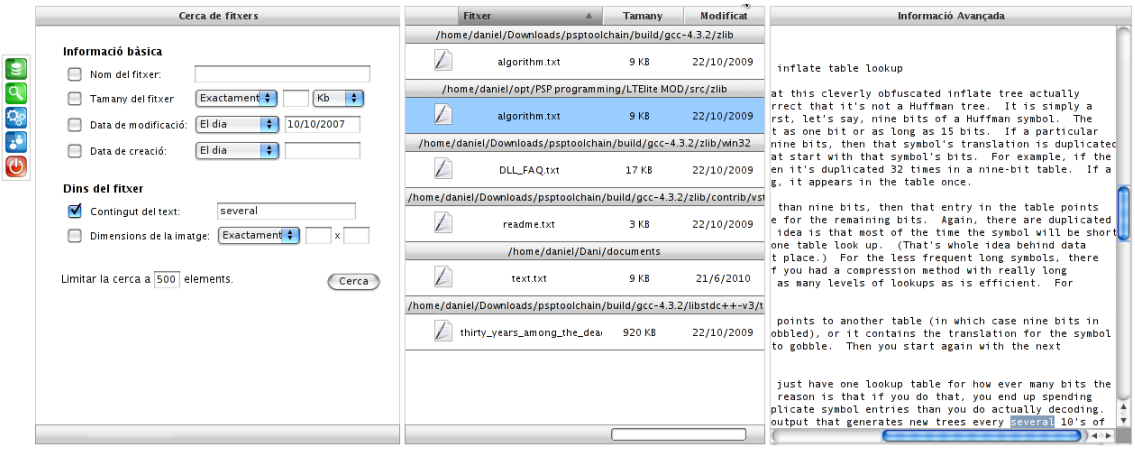


Figura 38. Mostra d'una cerca per paraules clau.

Aquest mòdul mostra la mateixa interfície per a tots els usuaris. El bloc de **listview** i **contentview** mantenen la mateixa funcionalitat que tenen per a l'explorador, aplicada als resultats trobats. Així, es pot clicar sobre qualsevol element trobat i observar la seva vista prèvia, o bé descarregar-lo.

4.6.5. Gestor d'usuaris

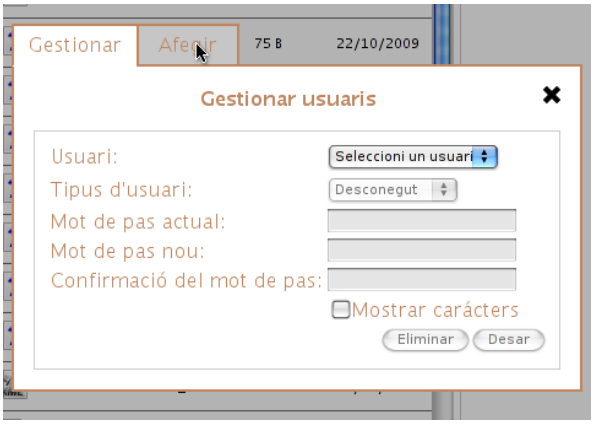


Figura 39. Pestanya per a gestionar usuaris.

El gestor d'usuaris separa la seva funcionalitat en dues "pestanyes". La primera (*figura 39)* permet a l'usuari Gestor modificar el tipus d'usuari o mot de pas d'aquells usuaris existents (excepte l'usuari mateix, per a evitar que no hi hagi cap gestor al sistema). Això a banda, amb aquest menú es poden eliminar altres usuaris (no a un mateix).

La segona pestanya del Gestor d'Usuaris permet afegir un usuari nou, escollint un nom, tipus d'usuari i mot de pas.

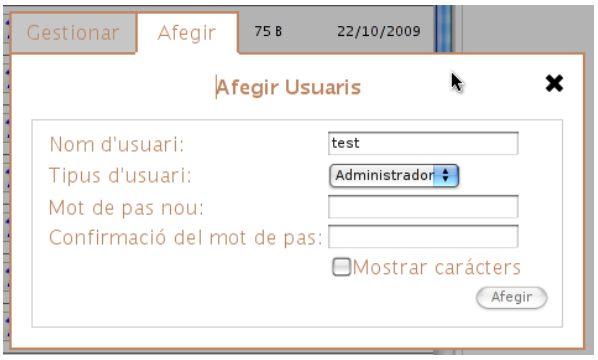


Figura 40. Pestanya per a afegir usuaris.

Cal notar varis aspectes de la interfície d'aquest mòdul. El més aparent és la forma en que es mostra en pantalla, utilitzant la classe creada per a generar diàlegs modals (veure capítol **4.6.7. Efectes varis**).

D'altra banda, a les captures es mostren els botons desactivats (afegir, desar canvis, eliminar usuari tots apareixen desactivats). Això es deu a que cada cop que es modifica un camp dels formularis, es realitzen comprobacions de seguretat per a activar o desactivar la opció de desar els canvis. Així, el botó de desar canvis no s'activarà fins que no es modifiquin els valors originals de l'usuari, mentre que el botó d'eliminar usuaris es desactivarà si es té l'usuari actual sel·leccionat.

Finalment, la casella marcada com a "mostrar caràcters" permet veure els camps de mot de pas, normalment amagats amb asteriscs.

4.6.6. Preferències d'usuari

Aquest mòdul permet que qualsevol usuari modifiqui aspectes de comportament del programa o les seves dades personals i mot de pas. Hem separat aquests dos grups d'opcions en dues pestanyes diferents, per tal d'oferir una representació clara de les possibilitats de personalització.

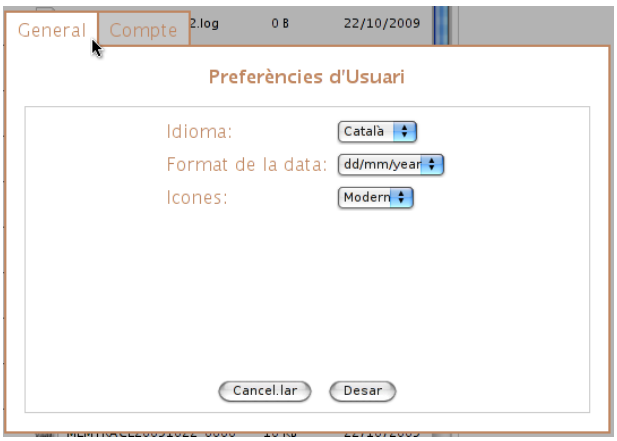


Figura 41. Primera pestanya de les preferències d'usuari.

A la primera pestanya observem les opcions de programa. Mitjançant una capça de selecció, l'usuari pot seleccionar el seu idioma preferit, a més del format de la data mostrada al **listview** de l'explorador. Les icones actualment només permeten una opció, però el programa està pensat per a poder permetre diferents sets d'icones segons la sel·lecció de l'usuari.

Figura 42. Segona pestanya de preferències d'usuari.

A la segona pestanya hi trobem les opcions de configuració de dades personals. Al igual que al gestor d'usuaris, trobem una casella per a mostrar els caràcters amagats del mot de pas.

4.6.7. Efectes Varis

Hi ha certs elements del sistema que no formen part d'una classe en concret. Es tracta de funcions auxiliars utilitzades per a varis objectius, la major part d'aquests per a aconseguir efectes visuals atractius de cara a l'usuari. Tot seguit veiem una llista d'aquestes funcions o classes auxiliars junt amb un exemple d'ús dins del programari.

4.6.7.1. Fade

Aquest efecte l'hem fet servir per a mostrar o amagar l'explorador i sobreposar el cercador a sobre del **treeview**. Es tracta d'un increment (**fadeIn)** o decrement (**fadeOut**) de la opacitat d'un element de forma gradual, de forma que dongui un efecte "d'aparició" molt aconseguit. Per a intentar entendre el resultat d'aquest efecte, observem tot seguit una sèrie de captures en diferents estats de temps al utilitzar aquest efecte.

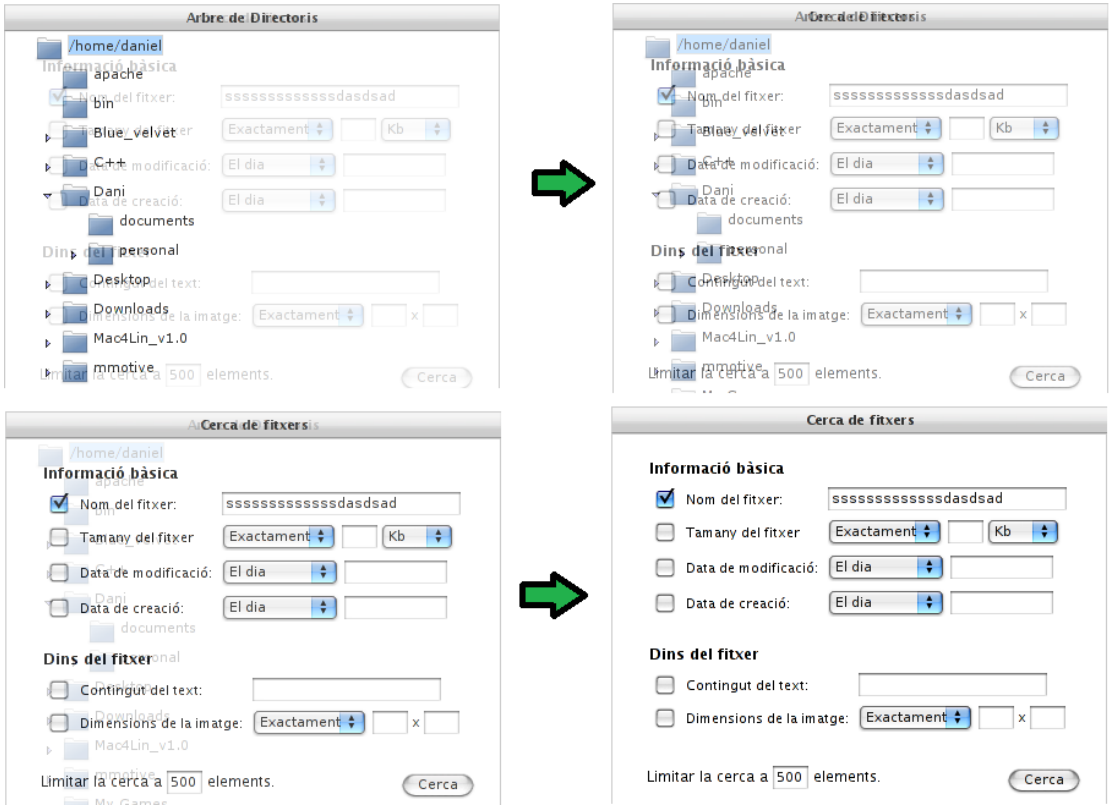


Figura 43. Efecte de transició de fade In per al mòdul de cerca.

El codi utilitzat es pot veure als annexes (**8.1.4**) i és realment senzill. Es tracta simplement de l'ús de la funció **setTimeout()** de Javascript, que ens permet executar funcions de forma retardada. Així, s'executa un canvi de la opacitat de l'element HTML en diferents moments de temps, de forma que passi d'opacitat 0 (transparent) a 1 (opac) en un periode de temps determinat.

4.6.7.2. Sliders

Dins de la interfície gràfica hi trobem varis elements que poden esser redimensionats. Tant el **listview, infoview** o **treeview** com les diferents columnes de la llista de fitxers poden ser redimensionades a gust de l'usuari. Això és possible gràcies a la classe auxiliar "draggables" (**annex 3**). Aquesta classe col·loca un "listener" per a l'event de clic del ratolí a cada element desitjat (les superfícies que separen cada bloc o cada columna). Aquest "listener" reacciona al fer clic i deixar anar el botó del

mouse, capturant la posició actual i redimensionant els elements pertinents.

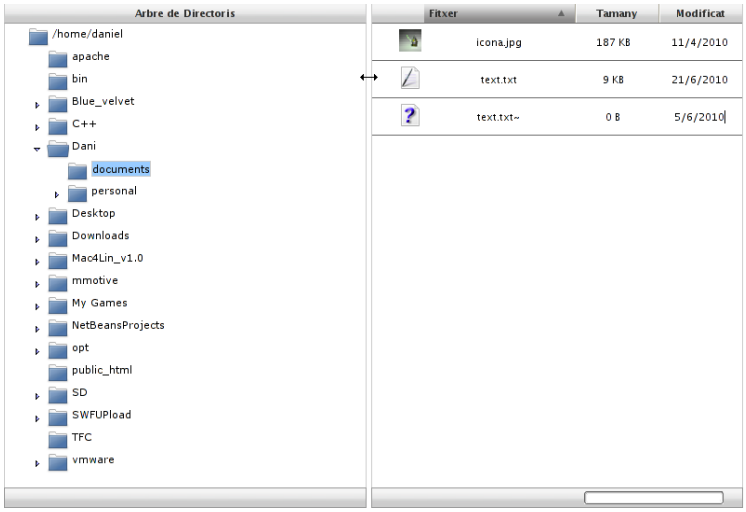


Figura 44. Exemple de redimensió horitzontal.

4.6.7.3. Diàleg modal

Tant el Gestor d'Usuaris com les Preferències d'Usuari es mostren sobreposats a la pàgina actual, amb un fons gris de fons i deshabilitant la interacció amb altres elements. Això es coneix com a **Diàleg Modal**. Alguns navegadors ja incorporen la possibilitat de generar un element d'aquest estil. No obstant, per a mantenir la compatibilitat amb tots els exploradors hem hagut de crear la nostra propia classe.

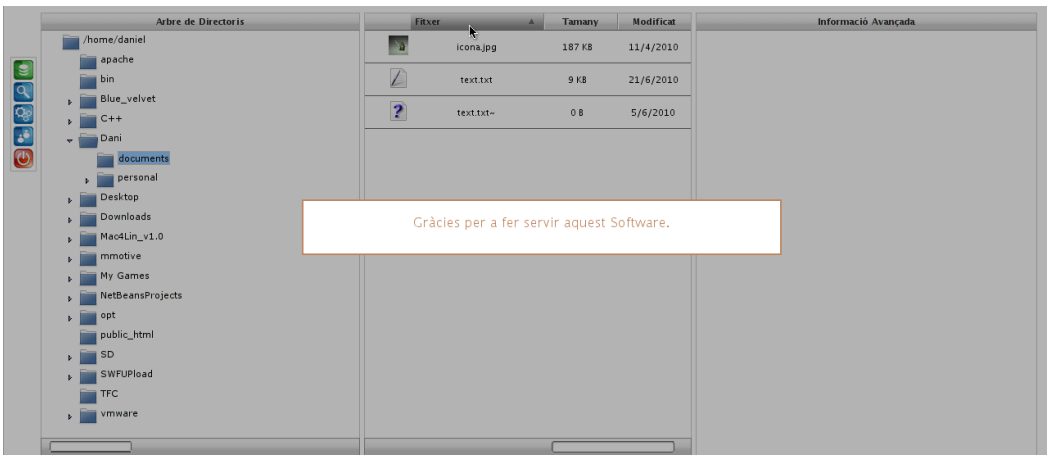


Figura 45. Missatge mostrat al sortir del programa, usant un diàleg modal.

Altres llocs de la pàgina on es pot trobar aquest tipus de missatge poden ser missatges d'alerta, el missatge de **logout** o missatges de càrrega al iniciar el programa.

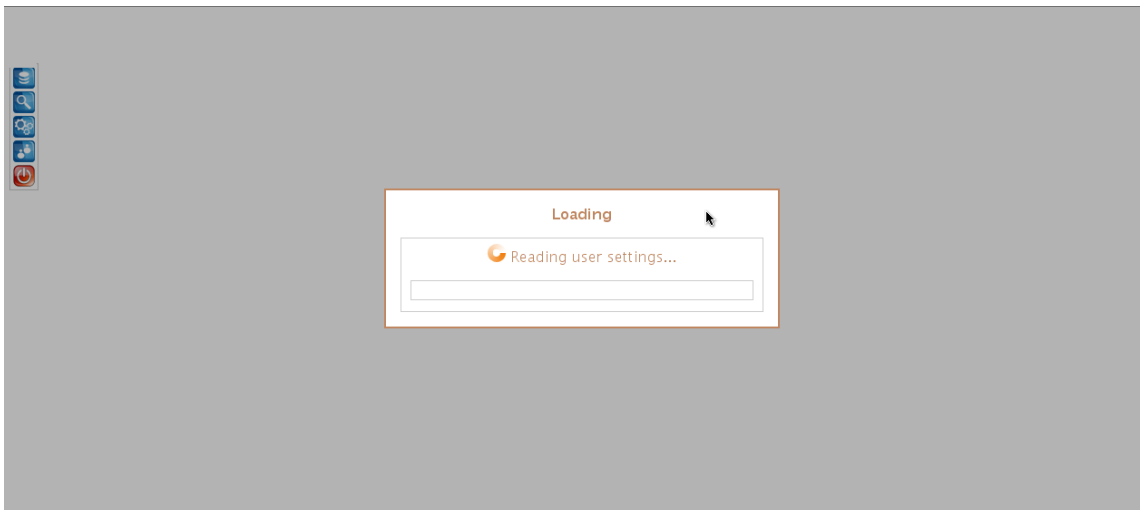


Figura 46. Missatge de càrrega mostrat a l'inici del programa, usant un diàleg modal.

4.6.7.4. Barra i missatges de progrés

A vegades realitzem algunes operacions que requereixen consultes al servidor o a la caché, com ara cerques de fitxers o consultes de contingut. En aquests casos volem informar al usuari d'alguna forma que el programa està ocupat amb aquests processos. Això proporciona una millor experiència d'usuari i alhora evita que el client es pregunti per què una operació no està mostrant cap resultat.

Per a aconseguir aquest objectiu hem utilitzat dos recursos. D'una banda mostrem una barra de càrrega a sota del **listview** amb una imatge animada a l'estil de la barra de l'explorador **Mozilla Firefox**. Aquesta barra també pot anar acompanyada d'un text explicatiu sobre el procés actual.

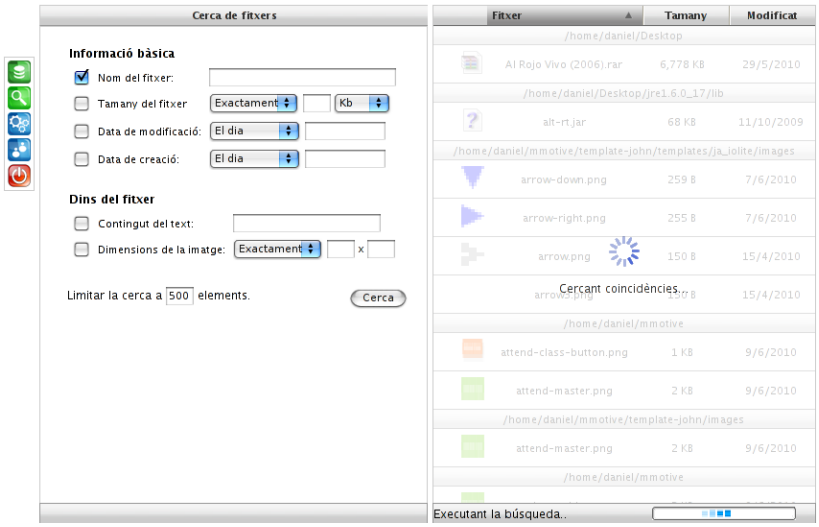


Figura 47. Una cerca activa la barra de progrés i un missatge semi-transparent al listview.

D'altra banda, utilitzem una classe que ens permet insertar un missatge de progrés sobre qualsevol element de la pàgina (en el nostre cas majoritàriament sobre del **listview** o **infoview**). Aquest missatge anirà acompanyat d'un fons blanc semi-transparent, que obfuscarà els elements a sota seu. Els elements més emprats són el missatge de progrés (cercant elements, creant base de dades...) junt amb una imatge animada que simbolitza la càrrega (*figura 47*) i un missatge d'alerta per a mostrar errors o avisos.

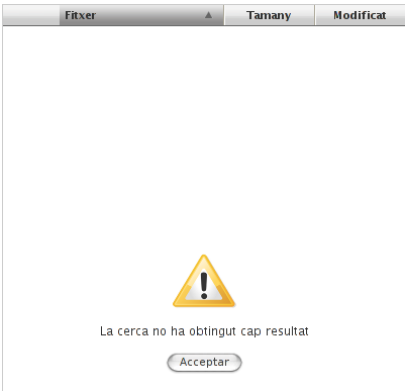


Figura 48. Missatge d'avís superposat al listview.

4.6.7.5. Traductor

Per tal de poder atorgar a l'usuari la possibilitat de sel·leccionar una interfície gràfica en diferents idiomes (anglés, castellà i català), hem implementat una classe en Javascript, utilitzada a l'hora de mostrar qualsevol text en pantalla. Aquesta classe, anomenada babel, cerca el text demanat dins d'el fitxer de diccionari corresponent i en retorna el valor trobat. Si no hi ha cap coincidència en retorna el text original (per defecte en anglès).

4.7. Implementació de la capa motor

La capa motor és una part del sistema el funcionament de la qual és dificil de descriure. Mentre que la capa d'usuari té una interfície gràfica ben definida, la capa motor realitza un conjunt de funcions que són transparents a l'usuari. De fet, com més transparents siguin aquestes funcions, més ben implementada estará aquesta part del programari.

Com hem vist en el disseny de la interacció de classes, el major gruix de funcions de la part motora realitzen funcions d'indexació. En concret, la creació de l'índex de fitxers i la seva actualització periòdica.

Per tal d'agilitzar al màxim aquestes funcions, hem utilitzat la tècnica de **multi-threading** amb PHP. Utilitzant el codi vist als Annexes (**8.1.1**), es crea la base de dades bàsica per a poder iterar els elements amb l'explorador. Després, es crear un procés fill que genera les icones en 24x24 i cerca les paraules clau dins dels fitxers de text, utilitzant l'algorisme corresponent. Cal dir que l'usuari pot començar a utilitzar el programa un cop acabada la primera fase de creació de la Base de Dades, però no veurà icones en petit de les imatges, ni podrà realitzar cerques de contingut fins que no acabi la segona fase de la indexació.

Un cop s'hagi creat la Base de Dades per primera vegada, caldrà mantenir-la actualitzada. Per tal d'aconseguir aquest objectiu es realitzaran tres tipus de manteniment. Podríem classificar aquest manteniment com a automàtic, semi-automàtic i manual.

4.7.1. Manteniment Automàtic

En el moment en que qualsevol usuari engegui el programa, s'engegarà un procés que s'encarregarà de realitzar una indexació de la Base de Dades cada 10 minuts. Aquest procés només s'engegarà una vegada i només si no es troba ja en execució (es comproven els processos actius abans d'executar-lo). Mentre es realitzin aquestes actualitzacions transparents a l'usuari es podrà fer servir el programa sense cap problema.

4.7.2. Manteniment Semi-Automàtic

Cada vegada que un usuari itera sobre un directori, el servidor executa un procés de fons que comprova la sincronia de la base de dades amb el sistema de fitxers. Si es troben elements obsolets, aquests s'actualitzen automàticament, mostrant una animació al client per a que l'usuari no vegi aparèixer i desaparèixer directoris i fitxers sense saber perquè passa.

Es mostren els directoris o fitxers esborrats amb una creu, i els directoris afegits amb un signe de suma. A més, mentre es realitza el procés d'actualització apareix una icona animada de refresc sobre els directoris implicats (*figura 49*).

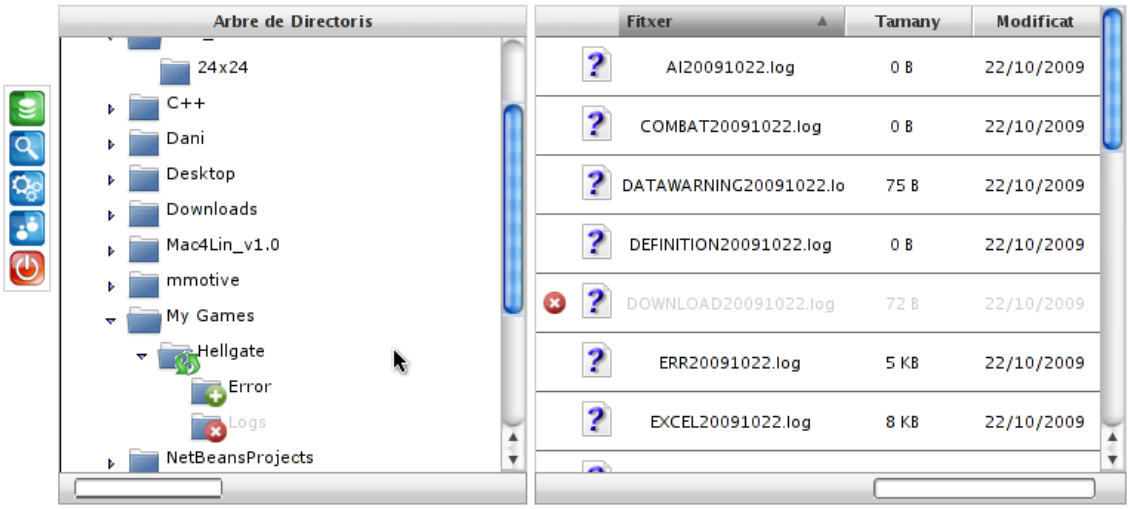


Figura 49. Representació d'una actualització semi-automàtica.

4.7.3. Manteniment manual

En cas que es vulgui realitzar una actualització sota petició, l'usuari administrador pot realitzar-la si fa clic sobre el botó corresponent del seu menú lateral (*figura 29*). En aquest cas es mostrarà un missatge de progrés a l'explorador. Aquest usuari no podrà seguir amb l'ús del programa fins que no acabi l'actualització, però els altres usuaris no es veuran afectats.

4.8. Estructura de fitxers

A l'hora d'emmagatzemar el codi font, hem seguit una jerarquia determinada per tal de facilitar el procés de desenvolupament, testeig i millora del sistema.

Hem separat en diferents directoris elements com els recursos (imatges), pàgines d'estil **CSS**, codi **javascript** i codi **php**. A més a més, trobem les diferents classes en subdirectoris. Dins de cada subdirectori de classe, conviuen els fitxers PHP i javascript per a generar les pàgines HTML necessàries.

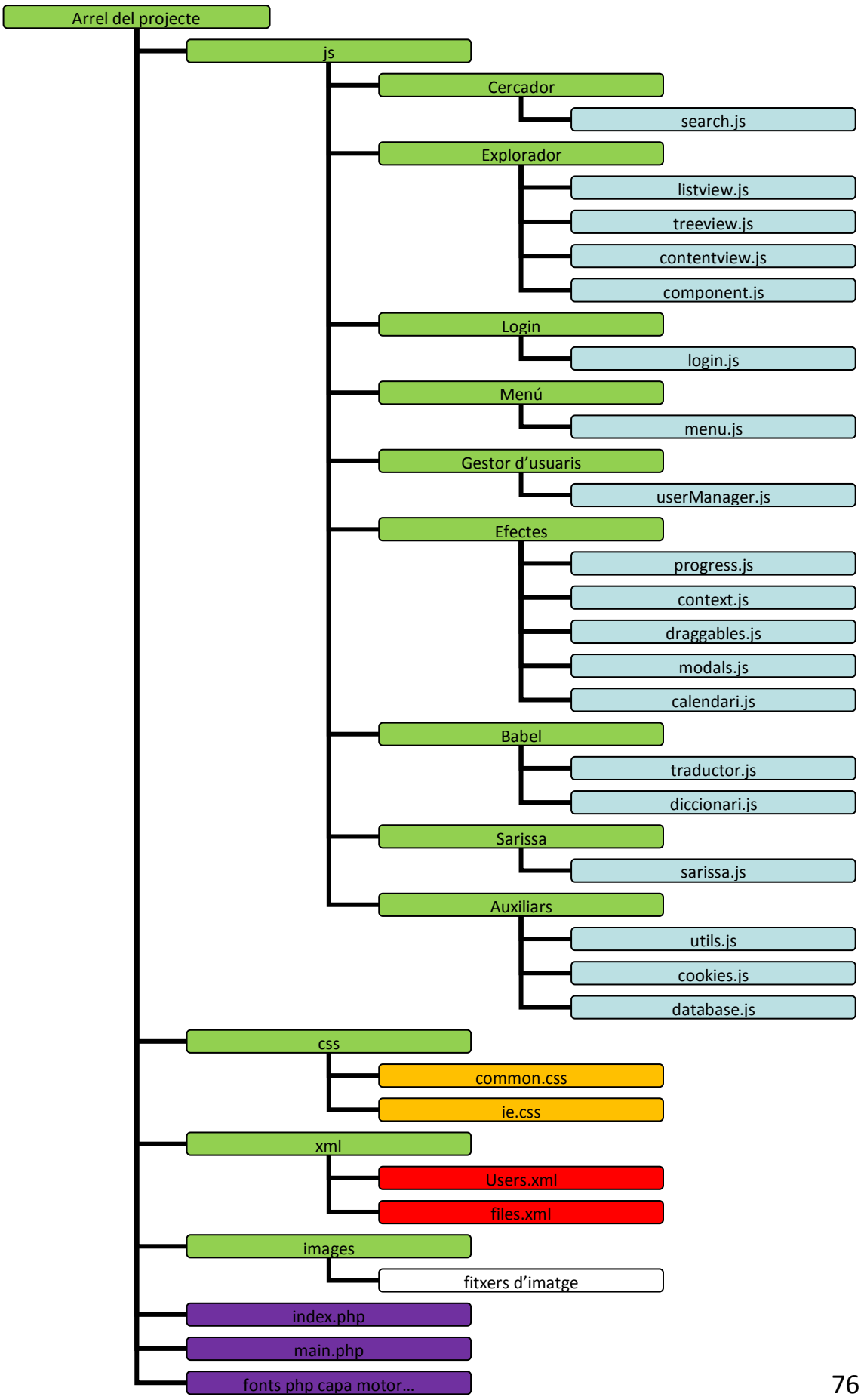


Figura 50. Estructura de fitxers del sistema.

A la *figura 50* veiem una representació d'aquesta estructura. Els nodes verds representen directoris, mentre que els nodes blaus i lila representen fitxers amb codi font en Javascript i PHP respectivament. Els nodes en color taronja representen codi CSS per a estilar la pàgina. Finalment, els nodes en vermell son fitxers d'índex i els grups de varis fitxers son de color blanc.

5. Proves

5.1. Introducció

Un cop implementat cada mòdul del projecte, caldrà realitzar un conjunt de proves per tal d'assegurar el bon funcionament del sistema i solucionar la majoria d'errors que hi pugui haver al sistema.

Parlar d'eliminar tots els possibles bugs o errors del programa seria ignorar la realitat. Per molt de temps que es dediqui a la caça de *bugs*, sempre hi haurà petits errors que no detectarem. Així, intentarem localitzar el major nombre d'errades possibles o, almenys, tots aquells que no permetin un correcte funcionament del programa.

5.2. Proves de compatibilitat

Al tractar-se d'una aplicació executada sota una pàgina web, hem basat les nostres proves de compatibilitat en l'ús de diferents navegadors i la observació de diferències en la representació del mateix codi. Hem utilitzat els següents navegadors:

- Internet Explorer 7.0 i 8.0
- Mozilla Firefox 3.6
- Seamonkey

Com que el desenvolupament de la aplicació es va dur a terme sota Firefox, les diferències principals s'han trobat al utilitzar Internet Explorer. Aquest navegador té bastantes incompatibilitats amb algunes definicions CSS reconegudes de forma diferent per Mozilla i Seamonkey. A més a més,

s'han trobat petits problemes en l'ús del sistema DOM sota javascript que han estat corregits posteriorment.

Per a solucionar els problemes d'interpretació de fulles d'estil, s'ha utilitzat un fitxer CSS exclusiu per a IE, que només es llegeix al arrencar el programa sota aquest navegador.

5.3. Proves de seguretat

Degut a l'existència de diferents nivells d'accés a la nostra aplicació, les proves de seguretat s'han centrat en detectar possibles forats que permetessin accedir a contingut restringit per a cada tipus d'usuari.

Al tractar-se d'una aplicació en Javascript, on no hi ha enllaços URL visibles a l'usuari, la seguretat del programa ja és bastant elevada en aquest aspecte. Això es deu a que un usuari que no conegui el codi font del programa, no sabrà quines funcions o crides al servidor s'estan executant en cada moment.

Per si de cas, s'han provat varies crides a la capa motor de forma directa a través d'scripts PHP i s'ha concluït que la seguretat del sistema en aquest aspecte és suficient.

5.4. Proves d'unitat

La major part d'aquestes proves, basades en la inspecció de cada mòdul per separat, s'han realitzat durant el mateix desenvolupament del programa. Així, la quantitat d'errors trobats durant la fase de proves ha estat mínima.

Per a definir els possibles escenaris en que ens podem trobar, s'ha utilitzat la llista de casos d'ús definida al **Capitol 3**. Després, s'han provat cada una d'aquestes accions per a assegurar que produeixen el resultat esperat. S'ha corregit el comportament quan ha fet falta.

5.5. Proves de integració

L'últim grup de proves realitzat ha estat el de integració. En aquesta fase s'ha testejat el comportament dels mòduls treballen conjuntament.

Degut a que s'havia modificat la integració de tots els mòduls en una darrera etapa de la fase de desenvolupament, aquestes han estat les proves on s'han detectat més errors. Després de varies iteracions en un procés de correcció de tots aquests problemes, podem dir que ja no hi ha errors en la interacció de cada part del sistema.

5.6. Conclusió dels resultats

Després de tot aquest conjunt de proves i correccions aplicades al codi, podem assegurar que el sistema funciona de forma estable.

6. Conclusions

6.1. Consecució d'objectius

En aquest apartat fem repàs dels objectius llistats al **Capítol 2** per a constatar si s'han complert o no.

El primer dels nostres objectius era poder indexar el contingut de diverses estacions de treballs en una sola base de dades. Hem aconseguit aquest propòsit utilitzant una combinació del sistema NFS i un índex en format XML creat per la capa motor.

L'objectiu de crear un aplicatiu web per a poder realitzar cerques de diferent tipus ha estat aconseguit gràcies als mòduls cercador i explorador.

S'han aconseguit permetre cerques sobre el contingut de fitxers com es volia, gràcies als algorismes d'indexació per paraules clau utilitzats a la capa motor.

També gràcies als algorismes d'indexació de la capa motor, s'ha fet possible veure imatges en tamany més petit dels fitxers d'imatge o PDF sense haver d'obrir-los.

Volíem crear un sistema d'actualització de base de dades dinàmic. A més a més de l'actualització "en viu" a mida que iterem els directoris, s'ha aconseguit implementar un sistema de refresc automàtic de forma periòdica que no interfereix en l'ús del programa. Així doncs, aquest objectiu ha estat assolit i millorat.

Ens varem proposar crear una interfície d'usuari agradable i creiem que s'ha aconseguit la fita, gràcies a l'ús d'un menú que ens permet accedir a tots els mòduls en una sola pàgina. A més a més, la inclusió de varis efectes visuals a la pàgina augmenten l'experiència d'usuari.

Un altre objectiu era el de crear tres jerarquies d'usuari i atorgar un control de l'aplicació diferent per a cada una. Això ha estat possible gràcies a l'ús de les *Cookies* i el mòdul de Gestió d'Usuaris.

Finalment, varem planejar maximitzar la compatibilitat amb els diferents navegadors. Això ha estat assolit gràcies a les diferents proves de compatibilitat realitzades, on hem assegurat el funcionament del programa sota els navegadors més utilitzats (Internet Explorer 7 i 8 i Mozilla Firefox).

En conclusió, podem afirmar que hem assolit tots els objectius proposats, superant en alguns casos aquestes expectatives.

6.2. Desviacions observades

Durant el transcurs d'aquest projecte no hi ha hagut problemes majors que hagin provocat un retràs important en el temps planificat a l'estudi previ. No obstant, el temps dedicat a desenvolupar el nostre codi ha estat molt superior a l'esperat. Això s'ha degut principalment a les ganes d'optimitzar el funcionament del programa per part de l'estudiant i a la complexitat de les tasques que ens havíem proposat. Tot i així, el resultat final ha estat l'assoliment de tots els objectius inicials en el marge de temps establert. Per tant, no podem anotar desviacions importants en aquest aspecte.

6.3. Línees d'ampliació

Durant el desenvolupament d'aquesta aplicació han sorgit algunes idees que no s'han pogut implementar degut a la necessitat de complir unes dates d'entrega determinades.

Tenint en compte que el programa ja indexa informació de paraules clau per a fitxers de text, es podria realitzar la mateixa tasca per a altres tipus de document, com ara DOC o altres formats pertanyents a paquets d'ofimàtica. En aquesta mateixa línea, es podria augmentar la quantitat de informació mostrada en la vista prèvia de fitxers, indexant informació per a formats de varis tipus (vídeo, àudio, etc...).

Mentre implementàvem el cercador varem pensar en la possibilitat de mostrar els resultats en "temps real", a mida que es vagi teclejant. Això hagués estat possible de forma molt fàcil si els navegadors no tinguessin limitació a l'hora de renderitzar una gran quantitat d'elements en pantalla.

Una possibilitat que varem tenir en compte durant el desenvolupament del programa és la utilització de varis grups de icones. Aquests es podrien escollir des del menú de configuració d'usuari. Dins de la nostra aplicació ja existeix un camp per a poder escollir les icones a utilitzar, però només podem escollir un grup (modern).

Finalment, aprofitant l'entorn multi-usuari del programa, semblaria interessant implementar una petita aplicació que permetés als diferents treballadors xatejar entre ells si estan connectats a la base de dades.

7. Referències

Tant durant el desenvolupament del software com en l'anàlisi del projecte, hem tingut a mà les pàgines web amb documentació sobre els llenguatges de programació utilitzats. A més, hem cercat la xarxa per a trobar llibreries que ens han estat molt útils. Tot seguit llistem algunes d'aquestes pàgines.

Pàgina	Descripció
http://www.w3schools.com/	La pàgina més global que coneixem, amb informació sobre Javascript, HTML, XML, SQL o PHP entre altres.
http://www.php.net/manual/en/	Documentació completa sobre PHP, llenguatge utilitzat per al software al servidor.
http://www.w3schools.com/ajax/default.asp	També a la pagina de w3c, trobem explicacions molt clares sobre la tecnologia AJAX, una tecnologia utilitzada a l'hora de programar la interfície web.
http://www.w3.org/Style/CSS/	Pàgina amb molt contingut i exemples de disseny en CSS, que hem utilitzat per donar un toc més agradable a la interfície web.
http://mygnet.net/	Pàgina amb moltes ajudes per al programador i fòrums d'ajuda, d'aquí s'ha extret un manual de PHP molt complert.
http://roundcube.net/	Un exemple del tipus de disseny implementat. Aquesta pagina utilitza DHTML (javascript, CSS i HTML) per crear efectes molt interessants.
http://en.wikipedia.org/wiki/Spotlight_(software)	Explicació d'una tecnologia que representa l'ideal de funcionalitat a obtenir a nivell client, amb una cerca casi instantània i visualment molt agradable.
http://dev.abiss.gr/sarissa/	Pàgina d'una interessant API XML per a javascript, que permet implementar AJAX mantenint compatibilitat amb varis navegadors.

http://www.javascriptkit.com/script/script2/epoch/index.shtml	Calendari en JavaScript utilitzat per a entrar les dates al formulari de cerca

Taula 3. Relació de pàgines web de referència.

A banda d'aquests recursos, s'han realitzat consultes periòdiques amb el tutor de projecte, que ha aportat la seva opinió sobre el software en desenvolupament. A més a més, gràcies al seu ajut es recopilarà documentació que d'altra forma podria haver passat desapercebuda.

8. Annexes

8.1. Codi font

En aquest apartat veurem el codi font que fa possible la realització d'algunes tasques dins del nostre programa. Mostrarem principalment parts de la capa motor, ja que aquestes no son visibles fàcilment.

8.1.1. Creació de l'índex

```
function buildXML( $path , $dom, $currentNode, $level = 0, $pid, $output, $thefile = null,$partial = false)
{
    try
    {
        $iterator = new DirectoryIterator($path);
    } catch(Exception $e) {
        error_log("Ignored $path directory");
        return;
    }

        while($iterator->valid())
        {
                $file = $iterator->current();


                $fileName = $file->getFilename();

                if( ($fileName[0] != '.') && ($fileName != "..") && !$file->isLink())
                {
                    $newNode = add_node_from_spl($file, $dom, $output);

                    $currentNode->appendChild($newNode);

                    if ($file->isDir())
                    {
                        $newPath = $file->getPath();
                        buildXML( "$newPath/$fileName", $dom, $newNode, ($level+1), $pid, $output, $thefile, $partial );
                    }
                }
                $iterator->next();
        }
}
```

Figura 51.Codi pertanyent a la creació de l'índex de fitxers.

Gràcies a l'execució d'aquesta funció recursiva, es crea l'índex de fitxers. La recursió va augmentant a mida que troba directoris, mentre que desa la informació de cada element en un node XML. Al final es desa l'arbre en memòria a un fitxer en disc dur.

Després d'aquesta iteració, es creen les icones en miniatura i les paraules clau dels documents. Això es fa amb la tècnica de **multi-threading**.

```
function reindex($mode = "async",$dom = null)
{
    function shutdown() {
        posix_kill(posix_getpid(), SIGHUP);
    }
    $pid = pcntl_fork();

    if($pid == -1) {
        error_log("Could not split for keywords.");
        die("");
    } elseif($pid == 0) {
        // this is the first child. Not used
    } else {
        // This part is only executed in the parent
        exit();
    }
    // this part is executed as the daemon we wanted. The parent is dead.
    ob_end_clean(); // Discard the output buffer and close

    register_shutdown_function('shutdown');

    if (posix_setsid() < 0)
        return;

    // Now running as a daemon. This process will even survive
    // an apachectl stop.

    sleep(10);

    /* and here it ends. Below is the child */

    require_once("keyworder.php");
    doGenerateKeyWords();
    doGenerateThumbs();
}
```

Figura 52. Part multi-thread de la creació de la BBDD.

8.1.2. Actualització automàtica

El servei d'actualització automàtica engegat al arrancar el programa és bastant simple. Es tracta d'un bucle que, amb una parada de 10 minuts, reescriu la base de dades cada vegada. Abans de començar, l'script comprova que no hi hagi una altra instància executant-se, de forma que només existeixi un actualitzador automàtic al servidor.

```php
require_once("variables.php");
require_once("files.php");
require_once("xmlWrite.php");

//header("Content-Type: text/html");
set_time_limit  ( 0 );

$loop = True;

$pid = getmypid();

if (file_exists($lock_update))
{
    $current_pid = readFileText($lock_update);
    if (pid_exists($current_pid))
    {
        error_log("currently already running updater");
        exit();
    }
    else
        error_log("we have detected that updater is not running anymore");
}

createFile($lock_update, $pid);

while($loop == True){
    clearstatcache();
    if (!file_exists($xml_file))
    {
        continue;
    }
    else
    {
        writeXML($p_pattern,$p_pattern, $xml_file, $pid, "admin", $numFiles, "false","sync");
    }

    /* Bucle executat cada X segons */

    usleep(1000000 * $updateSeconds);
}

function pid_exists($pid)
{
    $command = "ps -e | grep $pid";
    exec($command." 2>&1", $output);

    foreach ($output as $line)
    {
        if ( !strpos($line,"apache") === FALSE || !strpos($line,"httpd") === FALSE)
        {
            return true;
        }
    }

    return false;
}
```

Figura 52. Algorisme d'actualització automàtica de BBDD.

8.1.3. Codificació del fitxer d'usuaris

Hem comentat que l'índex d'usuaris es codifica amb una barreja de base 64 i md5. No obstant, no hem especificat l'algorisme utilitzat per a tal fi. Tot seguit en veiem el codi.

```
function __encode($string)
{

    $string = 'STT576PWZA'.$string;

    $result = '';
    for($i=0; $i<strlen($string); $i++)
    {
        $char = substr($string, $i, 1);
        $keychar = substr($this->key, ($i % strlen($this->key))-1, 1);
        $char = chr(ord($char)+ord($keychar));
        $result.=$char;
    }
    return base64_encode($result);
    //return base64_encode($string);
}

function __decode($string)
{

    $result = '';
    $string = base64_decode($string);

    for($i=0; $i<strlen($string); $i++)
    {
        $char = substr($string, $i, 1);
        $keychar = substr($this->key, ($i % strlen($this->key))-1, 1);
        $char = chr(ord($char)-ord($keychar));
        $result.=$char;
    }

    $result = substr($result,10,strlen($result));

    return $result;
}
```

Figura 53. Algorisme d'encriptació per al fitxer d'usuaris.

8.1.4. Efecte de fade-in i fade-out

Per tal de mostrar un dels elements més senzills i a la vegada efectistes del programa, hem adjuntat la part del codi que fa possible aquesta transició. Pot esser aplicat a qualsevol element de la pàgina.

```
function fadeIn(id)
{
    var browser = navigator.appName;

    if (browser.indexOf("Netscape")!=-1 && parseInt(navigator.appVersion) >=5 )
    {
        for (var i = 1;i < 9; i++)
                setTimeout("$('" + id + "').style.MozOpacity = 0." + i ,i*50);

        setTimeout("$('" + id + "').style.MozOpacity = 1" ,500);
    }

    if (browser == "Microsoft Internet Explorer")
    {
        for (var i = 1;i < 9; i++)
                setTimeout("$('" + id +"').filters.alpha.opacity = " + i*10 ,i*50);

        setTimeout("$('"+ id +"').filters.alpha.opacity = 100" ,500);
    }
    else
    {
        for (var i = 1;i < 9; i++)
                setTimeout("$('"+ id +"').style.opacity = 0." + i ,i*50);

        setTimeout("$('"+ id +"').style.opacity = 1" ,500);
    }

}
```

Figura 54. Algorisme utilitzat per a crear l'efecte de Fade In.

8.1.5. Algorisme de cerca

La nostra cerca es realitza utilitzant el llenguatge XPATH. Es fa a nivell de client utilitzant la caché que ja té el navegador, raó per la qual és realment ràpida. Tot seguit en veiem el codi.

```
if (anything)
{
        var string = "//file[";
        for (var i=0;i < conditions.length;i++)
        {
            if (i == 0)
                string += conditions[i]
            else
                string += " and " + conditions[i];
        }
        string += "]";
        var nodes = window.explorer.xmlDoc.selectNodes(string);
}
else
        var nodes = null;
```

Figura 55. Cerca de la caché amb una expressió XPATH, depenent de les condicions seleccionades.

8.1.6. Petició XMLHttp Asíncrona

```
function getXmlRequest()
{
        try {
                XmlHttp = new ActiveXObject("Msxml2.XMLHTTP");
        } catch (e) {
                try {
                        XmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
                } catch (E) {
                        XmlHttp = false;
                }
        }

        if (!XmlHttp && typeof XMLHttpRequest!='undefined') {
                try {
                        XmlHttp = new XMLHttpRequest();
                } catch (e) {
                        XmlHttp=false;
                }
        }

        if (!XmlHttp && window.createRequest) {
                try {
                        XmlHttp = window.createRequest();
                } catch (e) {
                        alert("This browser does not support AJAX applications.")
                        XmlHttp=false;
                }
        }

    return XmlHttp;

}
```

Figura 56. Funció per a utilitzar un objecte XMLHttp en varis navegadors.

És la base de tot AJAX. L'objecte XMLHttp o ActiveX en cas d'Internet Explorer. Per tal de poder utilitzar una sola crida per a tots els

navegadors, hem utilitzat una funció auxiliar que retorna l'objecte pertinent segons el navegador utilitzat.

8.2. Manual d'usuari

Aquest programari és molt senzill i intuïtiu d'utilitzar, així que qualsevol usuari podrà començar a utilitzar-lo sense cap coneixement previ necessari. No obstant, calen un seguit de passos per a poder configurar-lo en un ordenador servidor i crear els usuaris pertinents. Aquests passos normalment els realitzarà l'administrador de l'empresa.

És possible realitzar aquesta instal·lació sota el sistema operatiu Windows, però per la naturalesa del software recomanem utilitzar un sistema operatiu Linux per a l'ordenador servidor.

Instal·lar servidor apache:

Caldrà instal·lar el software Apache a l'ordenador servidor. Les llibreries utilitzades pels scripts PHP són les següents:

- PHP-dom
- PHP-pcntl
- PHP-posix
- Imagick

Configurar la pàgina servidor:

Un cop instal·lat Apache, caldrà configurar un servidor o subdomini que apunti a on es troba el directori amb els codis font. Cal configurar l'usuari apache com a propietari d'aquest directori.

Configurar directoris NFS:

Es muntaran els directoris remots de la xarxa utilitzant la utilitat NFS. Tot seguit, es configurarà el fitxer “variables.php” modificant el valor de la variable $p_pattern pel que vulguem fer servir com a directori arrel de l’explorador.

Crear gestor:

Arribats a aquest punt ja es pot crear el primer usuari gestor. Aquest pas el pot realitzar l’usuari Gestor, que crearà els subseqüents usuaris a partir d’aleshores.

Engegar el programa:

El sistema ja és totalment funcional I es pot utilitzar sense problemes.