Fortaleses i debilitats de les bases de dades SQL i NoSQL en aplicacions web

Memòria de Projecte Final de Màster
Màster Universitari d'Enginyeria Informàtica
Desenvolupament d'aplicacions web

Autor: Antoni Llusà Sala

Consultor: Ignasi Lorente Puchades
Professor: César Pablo Córcoles Briongos

Gener del 2016

Dedicatòria/Cita

Donar gràcies a la meva família, la Núria i els nostres tres fills, en Toni, la Jana i en Pau, ja que han tingut molta paciència, ja que els he deixat sols més d'una vegada per poder realitzar els treballs de les assignatures que he anat fent durant el Màster. També donar gràcies els meus pares, pel suport que m'han donat.

Abstract

Aquest projecte està dins l'àrea del desenvolupament d'aplicacions web. S'ha realitzat un estudi on el seu objectiu principal és poder decidir, en el cas d'implementar una aplicació web, saber resoldre si és millor opció escollir una base de dades SQL o bé NoSQL. Per poder-ho deduir punts a favor i en contra de cada una de les opcions, s'han plantejat diferents objectius, que són, desenvolupar un administració on es puguin gestionar les dades pels diferents sistemes, realitzar comparacions de rendiment en les operacions d'inserció, d'actualització, d'eliminació i cerques de registres en les bases de dades. També en creacions d'index en camps de les bases de dades. També s'ha pretès veure com s'interacciona amb les bases de dades amb el framework seleccionat. Per realitzar l'aplicació web i les proves s'han escollit les bases de dades MySQL per part de l'opció SQL i MongoDB, per la part de NoSQL.

Les dades que s'han tractat per poder realitzar els testos són del web https://www.librarything.com/ S'han realitzat diferents procediments, el primer ha estat crear un procediment per recollir la informació del web comentat, d'on s'han aconseguit 8 GB de dades per importar. Un cop obtingudes les dades s'han analitzat i s'ha escollit una estructura per emmagatzemar-les en cada base de dades. Llavors s'han importat les dades i s'ha guardat fitxers de logs per poder analitzar el seu rendiment. Paral·lelament s'ha desenvolupat l'aplicació web amb accés restringit amb usuari i contrasenya, el Back-Office, on es poden realitzar cerques, altes, baixes i modificacions de registres. S'ha creat una gestió d'usuaris on hi ha diferents perfils administrador, editor i visualitzador.

Un cop realitzada l'aplicació i les dades han estat carregades a la base de dades, s'han realitzat tests empírics de cerca, actualitzacions i eliminacions de registres i creació d'index en les taules, aquests testos s'han realitzat directament interactuant amb les bases de dades, per la MySQL des de la seva consola, i pel MongoDB des del programa RoboMongo. S'ha anat anotant els resultats per poder-los comparar i poder realitzar les gràfiques comparatives.

S'ha pogut observar que alhora d'inserir les dades ha anat més ràpid el procediment d'importació en MongoDB que el d'MySQL, ja que amb l'estructura escollida del MySQL s'han hagut de fer validacions per tal de mantenir l'estructuració i integritat de les dades que demana una opció SQL. Alhora de crear els index en la taula de SQL i en la col·lecció a MongoDB els resultats han estat molt anivellat. Alhora de cercar la informació el MongoDB ha estat més ràpid, però alhora de modificar i eliminar els registres, quan hi ha hagut index l'SQL ha estat superior.

A nivell del desenvolupament de l'aplicació el fet d'utilitzar un framework que permet treballar amb les dues bases de dades, ha facilitat li anivellat les dues opcions, gràcies al seu ORM (object-relational mapping). S'ha hagut d'adaptar la metodologia segons les eines escollides. Les eines pel desenvolupament d'aquest projecte han estat PHP, MySQL, MongoDB, AngularJS, Bootstrap i el Fat Free Framework.

Paraules clau: Fortaleses, Debilitats, MySQL, MongoDB, Back-Office, CRUD, Anàlisi de dades, Model de dades.

Abstract (english version)

This project is in the area of web application development. A comparative analysis has been performed with the main objective of deciding the best option to choose, SQL or NoSQL, for the management of the application data. Several main objectives have been planned in order to find conclusions. The planned tasks have been the development of a back-office where you can manage the data with both systems, also performance comparisons when doing inserts, updates, deletes and searches to the database records. Also creating index fields in the database. Furthermore a development framework has been selected in order to see how it interacts with databases. To develop and test the web application MySQL for SQL, and MongoDB for NoSQL have been chosen.

The data which have been chosen to perform the tests were taken from the website https://www.librarything.com/ . In order for the author of the project to get the results and conclusions different procedures have been executed. The first procedure has been the collecting and importing of 8GB of data from the site. Then the data has been analysed and then the best storing structure for each database has been chosen. Next steps have been importing the data and saving log files that will make it possible to analyse their performance. In parallel a web application have been developed; the Back-Office, with username and password restricted access, in which users can search, insert, delete and modify records. Also a management area has created which enables different profiles such as administrator, editor and viewer.

Once the web application has been completed and the information has been loaded into the databases, empiric tests have been performed on searches, updates and deletions of records and also on indexes creation. These tests have been performed directly interacting with databases; the MySQL's from the console, and the MongoDB's from RoboMongo program. The results have been saved to compare them and to make comparative charts.

It has been observed that, when inserting data MongoDB import procedures has been faster than MySQL, because of the chosen MySQL structure. Due to SQL relational basis, for MySQL imports, existing records validations have been necessary. There haven't been relevant differences on the behaviour of the index creation procedure between MySQL and MongoDB. Searching information on MongoDB has been faster. Also modifying and deleting records. However SQL has been faster when index exist.

In the development of the application a framework with an ORM (Object-relational mapping) and that can work with both databases has been used. The author of the project has adapted the methodology according to the chosen tool. The tools used for the development of this projecte have been PHP, MySQL, MongoDB, AngularJS, Bootstrap and Fat Free Framework.

Keywords: Strengths, Weaknesses, MySQL, MongoDB, Back-Office, CRUD, data analysis, Data Model.

Agraïments, Notacions i Convencions

Títol 1
Títol 2
Títol 3
Peu de Figures i Taules
`Codi Font`

Índex

Figures i taules

Llistat d'imatges, taules, gràfics, diagrames, etc., numerades, amb títols i les pàgines on apareixen.

Índex de figures

Índex de taules

1. Introducció/Prefaci

El projecte neix a partir de l'interès personal d'entrar en el món de les bases de dades NoSQL, ja que durant els estudis em va cridar l'atenció. Actualment està molt acceptat que tant les bases de dades SQL com les NoSQL s'usen amb èxit en diversitat d'entorns. Possiblement les bases de dades relacionals han estat les més utilitzades en aplicacions de propòsit general. Les bases de dades relacionals basen el seu èxit en el fet que el dissenyador de l'aplicació i de la base de dades, els desenvolupadors i també l'usuari, fan un gran esforç per adaptar, estructurar, filtrar incongruències, evitar duplicats, ... etc. Amb això s'aconsegueix una gran qualitat de les dades i una gran capacitat d'extracció d'informació útil per ser analitzada. El temps ha donat oportunitat a maneres de gestionar la informació amb menys restriccions, mes adaptables, menys estructuració, amb més capacitat d'adaptar-se a les incongruències i falta de rigor inherents a la comunicació humana. Les bases de dades NoSQL treuen partit del fet de poder-se incorporar en entorns més àgils, més adaptables, amb menys esforç d'estructuració i integritat, perdent algunes bones característiques de les bases de dades SQL, però guanyant-ne d'altres de gran interès.

En la professió en què treballo, el desenvolupament d'aplicacions web, cada vegada més ens trobem que quan el client ens planteja l'aplicació desitjada es tenen dubtes de quina base de dades escollir, també pot passar que dins d'un projecte podria haver la possibilitat de utilitzar les dues. M'agradaria tenir una visió més profunda alhora de decidir quan et plantegen el projecte, en quines parts utilitzaries un cas o l'altre.

Un moment on la programació és molt propera a la base de dades és el BackOffice. És per això que en el BackOffice és on podem veure un millor aprofitament de la decisió, SQL o NoSQL. Amb la decisió correcta quan es dissenya l'aplicació, l'usuari de l'aplicació pot sortir-ne molt beneficiat.

Per dur a terme el projecte, es construirà un BackOffice com a part d'una aplicació web i s'implementaran accions bàsiques per les diferents base de dades SQL i NoSQL, prenent en cada moment la decisió del tipus de BBDD després d'analitzar-ne pros i contres.

Donat que es tractarà d'un entorn privat, s'hi haurà d'accedir mitjançant un usuari i contrasenya. Dins de l'entorn hi haurà la opció de canviar la contrasenya on hi haurà validacions per aconseguir una contrasenya robusta.

L'interès que em porta a realitzar el projecte, són les ganes d'introduir-me en les base de dades NoSQL i de conèixer millor les eines que utilitzaré en el desenvolupament.

Amb aquest projecte, es pretén concloure en quines opcions la millor opció és usar SQL i en quines NoSQL.

2. Descripció/Definició/Hipòtesi

El TFM està dins de l'àrea de desenvolupament d'aplicacions web del Màster Universitari d'Enginyeria Informàtica.

El projecte tracta de veure les fortaleses i debilitats de les base de dades SQL i NoSQL en aplicacions web, per poder-les veure, es realitzarà un desenvolupament d'un BackOffice per realitzar les funcions bàsiques. El BackOffice serà un entorn privat on per accedir-hi s'haurà d'entrar amb un usuari i contrasenya. Amb la construcció del BackOffice llavors es podrà corroborà les fortaleses i debilitats de les base de dades SQL i NOSQL en les aplicacions web.

Per la realització del projecte s'utilitzaran les següent eines:

- Interfície gràfica: HTML5, CSS, Bootstrap i Angularjs.
- Base de Dades: MySQL (per la part SQL) i MongoDB (per la part NoSQL).
- Per la interacció amb les Base de Dades s'utilitzarà una API REST en PHP.

Per la maquetació de l'administració s'utilitzaran les eines d'interfície gràfica. Amb el Bootstrap aconseguirem que es pugui veure en qualsevol dispositiu. Amb l'Angularjs aconseguirem que els filtres de la pantalla de veure els resultats siguin flexibles.

Per la interacció amb les base de dades utilitzarem una API REST en PHP, ja que es el llenguatge que estic a basat a utilitzar. La API en qüestió s'haurà de buscar quina es la que s'adapta més a les necessitats i pensant en un futur per ampliació.

Les base de dades escollides són MySQL i MongoDB. MySQL és un sistema de gestió de bases de dades relacional i multiusuari, que usa el llenguatge SQL (Structured Query Language), i el MongoDB es un sistema de creació i gestió de base de dades orientada a documents, escalable i d'alt rendiment.

Amb l'estudi es vol aconseguir els punts forts i dèbils en la definició de les dades, en la seva tipologia, validacions de les dades, com afronten els valors a null, camps a multi-avaluats. També veure com es realitzen les operacions CRUD a les diferents base de dades.

3. Objectius

L'objectiu principal del projecte és poder concloure en quines opcions es millor gestionar les dades amb SQL i en quines NOSQL, per aconseguir-ho, s'ha separat els objectius amb dos aparts:

3.1 Principals

Els Objectius claus pel desenvolupament del TFM són els següents:

- Implementar un BackOffice com a part d'una aplicació web amb les accions bàsiques de CRUD per les diferents base de dades SQL i NOSQL.
- Cercar i escollir dades per importar en les base de dades.
- Escollir els motius per poder saber les fortaleses i debilitats dels dos sistemes.
- Utilitzar els coneixements adquirits durant el Màster.

Implementar un BackOffice, és la part principal de l'aplicació ja que és l'entorn on es podrà testejar el comportament amb les dues bases de dades, ja que es podran realitzar Cerques / Filtratges, Altes, Modificacions i Baixes de productes.

S'han d'escollir dades que permetin portar a terme les comparacions dels punts forts i dèbils, en la seva definició de les dades, en la seva tipologia, com s'han de validar les dades, que passa amb els valors nuls, camps que puguin ser multi-avaluats i com es realitzen les operacions CRUD, també es tindrà en compte les insercions de l'importació inicial per veure com es comporta cada base de dades. S'haurà de crear l'estructura de les dues bases de dades i fer una importació inicial de les dades.

Per poder realitzar els objectius, s'haurà d'utilitzar coneixements adquirits durant el Màster.

3.2 Secundaris

Els Objectius secundaris que es deriven de la creació del projecte són els següents:

- L'accés al BackOffice que sigui segur, tant alhora d'entrar-hi com alhora de gestionar les contrasenyes.
- L'interfície sigui usable per l'usuari.
- Configuració de les base de dades i del repositori.
- Escollir la API REST a utilitzar.

Com que es tracta de desenvolupar un entorn privat, l'accés a ell serà mitjançant un usuari i contrasenya. Hi haurà diferents perfils d'usuaris: Administradors, Editors i Observadors. L'Administrador podrà crear usuaris amb els perfils Editor i Observador. Les contrasenyes seran encriptades.

L'interfície ha de ser usable perquè els diferents perfils d'usuaris puguin utilitzar-la fàcilment.

S'han de configurar les bases de dades, el MongoDB per defecte no hi ha usuari i contrasenya per accedir-hi, s'haurà de configurar. També es configurarà un repositori GIT per poder treballar de manera segura a nivell de la programació.

S'ha de cercar un framework que permeti treballar conjuntament les dues bases de dades SQL i NoSQL (SQL i MongoDB).

4. Marc teòric/Escenari

Per la realització d'aquest projecte, degut que es desenvoluparà una aplicació web, s'ha escollit per les bases de dades SQL el MySQL, i per les NoSQL el MongoDB.

4.1. SQL

SQL, significa (Structured Query Language). S'utilitza per crear, transformar i recuperar informació de RDBMS (Sistema de gestió de base de dades relacionals). És un llenguatge de programació declarativa dissenyat per a la creació i consulta dels sistemes de gestió de base de dades relacionals. És un llenguatge simple i alhora potent. Pot inserir dades en taules de bases de dades SQL, pot modificar i eliminar dades existents, també pot crear, modificar i eliminar taules i altres objectes de base de dades.[13]

MySQL, és un sistema de gestió de bases de dades relacional. La seva llicència d'ús és GPL. Es un component de l'arquitectura LAMP (Linux, Apache, MySQL i PHP). Pertany a l'empresa Oracle.[10]

El sistema de gestió de bases de dades relacional, dóna suport a la definició de les dades del model relacional, que té en compte tres aspectes:

- L'estructura, que ha de permetre representar la informació que ens interessa del món real.
- La manipulació, ha de permetre operacions d'actualització i consulta de les dades.
- La integritat, és facilitada mitjançant l'establiment de regles d'integritat.

El principal objectiu del model de dades relacional és facilitar que la BD sigui vista per l'usuari com una estructura lògica que consisteix en un conjunt de relacions, i no com una estructura d'implementació. Aquesta estructura lògica hauria de ser simple i uniforme. [12]

La base de dades SQL està composta de taules i aquestes taules tenen files.

4.2 NoSQL

NoSQL (Not Only SQL), abasta una amplia varietat de diferents tecnologies de base de dades que es van desenvolupar en resposta a un augment en el volum de dades emmagatzemades sobre els usuaris, objectes i productes, la freqüència d'accés i les necessitat de rendiment i processament.

Les bases de dades NoSQL, són més escalables i proporcionen un rendiment superior a les base de dades relacionals. Ofereixen:

- Gran volum de dades estructurades, semi-estructurades i no estructurades
- "Agile Sprints", iteració ràpida i "frequent code pushes"
- Programació orientada a objectes fàcils i flexibles d'usar.
- Arquitectura eficient, escalabilitat horitzontal i arquitectura monolítica.

Hi ha diferents tipus de base de dades NoSQL[14]:

- Key/Value, promouen escalabilitat i creixement a ràpida velocitat. El concepte bàsic és que una taula hash distribuïda té claus que condueixen als diferents servidors de bases de dades repartides arreu. Cada element de dades es converteix en una clau utilitzant una fórmula única que s'emmagatzema a la taula de cerca o directori. Quan es necessiten les dades, la clau es converteix en la ubicació de les dades i en conseqüència es recuperen les dades.

- BigTable, es coneixen com orientat a registres o bases de dades tabulars que consta de diverses taules, cadascun amb un conjunt de files direccionables.

- Columnar databases, són un híbrid entre NoSQL i bases de dades relacional. Proporcionen una estructura de files i columnes, però no tenen estrictes les regles de les bases de dades relacionals.

- Document databases, les dades s'emmagatzemen com a documents. Un conjunt de documents s'anomenen col·leccions. Les col·leccions poden contenir qualsevol nombre de documents de qualsevol tipus. Aquesta és la característica clau de bases de dades orientades a documents. Es centren en l'emmagatzematge i l'accés a l'optimització dels documents en lloc de files o registres. Els documents tenen un format JSON.

- Graph databases, emmagatzemen informació en tuples múltiples atributs que reflecteixen les relacions d'una manera diferent.

MongoDB, és un sistema de gestió de bases de dades orientades a documents de codi obert, proporciona un alt rendiment, alta disponibilitat i escalable automàticament. La seva llicència d'ús és GNU AGPL v3.0 (drivers: llicència Apache).[11]

La Base de dades de MongoDB està composta per col·leccions i cada col·lecció té documents.

El format dels documents és BSON, semblant al JSON, poden emmagatzemar: string, number, boolean, null, array i objectes.

4.3 Teorema CAP

El teorema CAP o teorema Brewer, estableix que en sistemes distribuïts es impossible garantir alhora: consistència, disponibilitat i tolerància a particions.[15] [16] [17][18]

- Consistència: Tots els nodes veuen la mateixa informació al mateix temps.
- Disponibilitat: Garanteix que tots els clients puguin llegir i escriure, encara que hagi caigut un dels nodes.
- Tolerància a particions: Els sistemes distribuïts poden estar dividits en particions. Així que el sistema ha de seguir funcionant encara que existeixin errors o caigudes parcials que divideixin el sistema.

MongoDB és CP en el teorema CAP, consistència i tolerància als errors. Es pot configurar el nivell de consistència triant el número de nodes als quals es replicaran les dades, també es pot configurar si es poden llegir dades del nodes secundaris. Si triem llegir a nodes secundaris obtindrem disponibilitat però disminuirem consistència.

SQL és CA en el teorema CAP, garanteixen la consistència i disponibilitat, però tenen problemes amb la tolerància a particions.

5. Continguts

El projecte consta d'un gestor de continguts on l'administrador podrà gestionar les dades de les diferents bases de dades, SQL (MySQL) i NoSQL (MongoDB). En les dues bases de dades s'importaran articles de la pàgina web https://www.librarything.com/ , és una comunitat d'usuaris on cadascú d'ells entra articles com: llibres, música, pel·licules, poadcasting, etc. En el web existeixen APIS on pots exportar dades de la comunitat. S'agafaran varis perfils d'usuaris públics i s'importaran les seves dades a MySQL i MongoDB. Així tindrem dades omplertes per poder testejar les dues bases de dades amb gran quantitat de dades. Llavors es podran modificar, eliminar i crear registres de les dues bases de dades. S'han extret 8 GB d'arxius JSONS (3.000 arxius) dels quals hi ha uns 1.000 usuaris que tenen llibres i 500 usuaris que tenen la resta (música, pel·licules, poadcasting, etc.) Amb la importació s'ha agafat per cada usuari 5.000 llibres ordenats alfabèticament creixent i 5.000 llibres ordenats alfabèticament decreixent així s'aconsegueixen 10.000 llibres per usuari. S'ha de tenir en compte que si un usuari te menys de 10.000 llibres, hi haurà llibres repetits per l'usuari, per tant alhora de la importació s'haurà de tenir en compte de no inserir repetits pels mateixos usuaris i pels altres. Pels procés d'importació, es gravaran arxius de log per poder fer un seguiment en les insercions, per veure el nombre d'insercions que s'aconsegueixen per cada grup d'arxius que s'importin, nombre de consultes per veure si existeixen en les bases de dades. I poder mostrar comparatives de rendiment, per aquestes proves es realitzaran proves creant index a les taules i sense index.

L'aplicació constarà de varis apartats: "Pàgina d'identificació", "Gestionar articles", "Canvi de contrasenya" i "Gestió d'usuaris". Hi haurà 3 perfils d'usuaris: Administrador, Editor i Observador.

L'administrador del sistema podrà gestionar les bases de dades SQL i NoSQL, podrà realitzar cerques / filtres, veure el detall de les dades, donar d'alta, modificar i donar de baixa articles. També podrà gestionar el usuaris del sistema, podrà donar-los d'alta, modificar-los i donar-los de baixa.

L'editor, podrà realitzar cerques / filtres, veure el detall de les dades, donar d'alta, modificar i donar de baixa articles.

L'observador només podrà visualitzar els articles, podrà fer cerques / filtres per localitzar els articles i visualitzar en detall l'article.

Tots els tres perfils podran canviar-se la seva contrasenya.

La pàgina d'identificació, servirà perquè l'usuari d'identifiqui posant el seu usuari, la seva contrasenya i un codi captcha per tal d'evitar els robots. Un cop identificat podrà entrar al panell de control.

L'apartat de gestionar articles (productes), constarà de dues pantalles, la pàgina de cerca / filtratge, on es podran cercar / filtrar articles, podràs accedir a ells per veure més informació, modificar les

dades del registre, podràs eliminar articles i podràs donar d'alta un article. Llavors hi haurà la pantalla del detall de la informació de l'article, on veuràs la resta d'informació, la podràs modificar i eliminar.

La pàgina de Canvi de contrasenya, l'usuari podrà canviar-se la contrasenya d'accés al sistema. El sistema mirarà que la contrasenya sigui segura, comprovant que hi hagi majúscules, minúscules, números i caràcters especials. També verificarà que la contrasenya que posin nova, no l'hagin entrat els últims 3 cops. La contrasenya es guardarà encriptada.

L'apartat de la gestió d'usuaris, l'usuari administrador, podrà gestionar els diferents usuaris i donar permisos d'editor o observador.

Per la gestió dels usuaris, s'utilitzarà la base de dades MySQL. Per la gestió dels articles, s'utilitzaran les dues bases de dades, per separat, o s'utilitzarà MySQL o MongoDB. En les pantalles de gestió dels articles es podrà escollir en quina base de dades interaccionar.

Per dur a terme les comparacions dels punts forts i dèbils s'analitzarà la seva definició de les dades, en la seva tipologia, com s'han de validar les dades, que passa amb els valors nuls, camps que puguin ser multi-avaluats i com es realitzen les operacions CRUD, també es tindrà en compte les insercions de l'importació inicial per veure com es comporta cada base de dades.

S'utilitzaran diferents eines per poder realitzar les diferents pantalles:

Per la interacció amb les bases de dades s'utilitzarà un micro framework anomenat "Fat Free Framework" de PHP, aquesta eina la utilitzaré ja que permet interaccionar tant amb base de dades MySQL com MongoDB.

Per la interacció de l'aplicació amb l'usuari s'utilitzarà HTML, CSS amb el framework Bootstrap i el framework Angularjs.

D'aquestes eines mencionades, les que he utilitzat alguna vegada són HTML, CSS, PHP i MySQL. Les que no he utilitzat cap vegada són: MongoDB i el Fat Free Framework.

Per la realització dels Mockups s'utilitzarà el web http://www.ninjamock.com.

6. Metodologia

Per la realització del projecte, s'utilitzarà la metodologia SDLC (Cicle de vida de desenvolupament d'un sistema d'informació), basada en el diagrama de Gantt, així es podrà coordinar i es podrà fer un seguiment mitjançant el calendari, i es podran definir totes les entregues d'obligat compliment en ell.

Les fases del cicle de vida del desenvolupament d'un sistema d'informació són:

- Planificació, on s'indiquen els objectius del producte i s'estableixen els processos, les activitats, les eines, els mètodes de treball i recursos, dins del temps.
- Anàlisi, on es prenen els requisits de les parts interessades i s'estableix un model lògic de funcionament futur.
- Disseny, a partir de l'anàlisi i del model lògic, s'estableix l'arquitectura de maquinari, base de dades, comunicacions, interfícies d'usuaris i l'arquitectura del programari
- Construcció, es duu a terme el desenvolupament tècnic, les proves i la documentació del sistema.
- Manteniment, es duu a terme correccions d'errors i suport a usuaris durant un temps acordat.

En cada entrega es revisarà la planificació establerta per detectar possibles desajustaments que puguin sorgir i poder adaptar la planificació a cada moment.

Per desenvolupar el BackOffice, es realitzaran les proves en un servidor web local, amb les diferents base de dades configurades, així podrem desenvolupar el projecte sense haver de treballar amb un servidor extern. Llavors per poder provar el projecte, s'instal·larà en un servidor d'internet.

7. Arquitectura de l'aplicació/sistema/servei

Degut que el projecte és una aplicació web, l'arquitectura del sistema es basa en el paradigma l'arquitectura client-servidor.

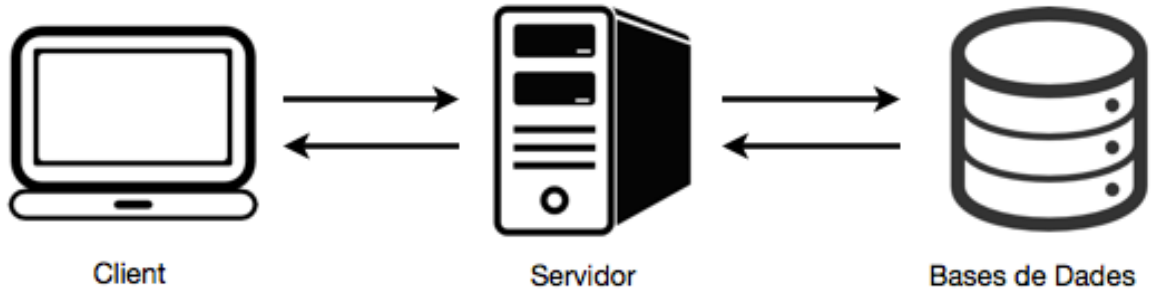

Figura 1: Arquitectura de l'aplicació

7.1 Client

El client és el navegador web de l'usuari. L'usuari de l'aplicació ha d'accedir a ella a través d'un dispositiu: ordinador, mòbil o tauleta, amb connexió a internet, mitjançant el navegador web, aquest ha de permetre reproduir arxius HTML, CSS i Javascript.

7.2 Servidor

El Servidor web, té com a sistema operatiu un Linux (Debian 8) i s'han instal·lat els mòduls de Apache, PHP, MySQL i MongoDB per poder executar l'aplicació web.

Pel desenvolupament del projecte, es treballarà amb una màquina virtual amb les següents prestacions:

- **SO** : Debian 8 64bit
- **CPU**: 1 de 1GB
- **RAM**: 2 GB
- **HD**: 40 GB

Per fer les proves amb un servidor real, he contractat un servidor dedicat a www.arubacloud.es amb les mateixes prestacions que la màquina virtual.

La versió de php és 5.6
La versió de l'apache és 2.2
La versió MySQL és 5.5
La versió de MongoDB és 3.0.7

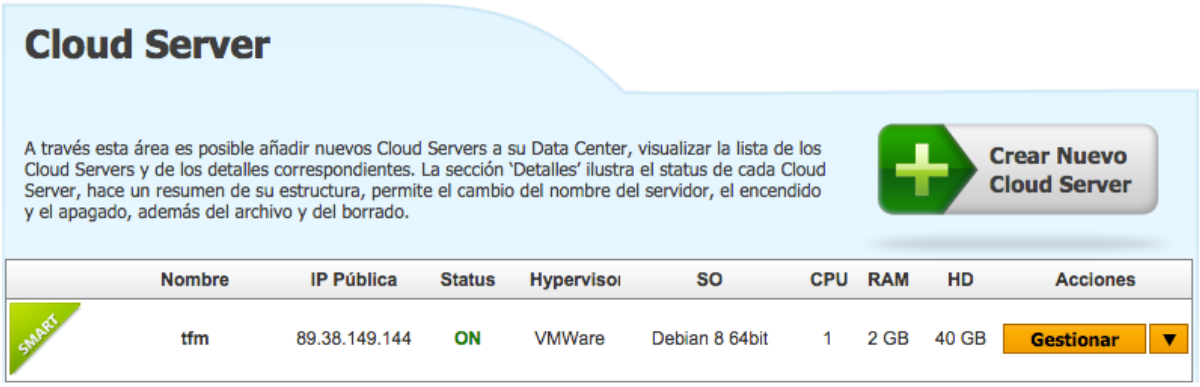

Figura 2: Informació extreta del servidor web de www.arubacloud.es

7.3 Base de dades

A continuació es mostren les diferents bases de dades:

7.3.1 MySQL

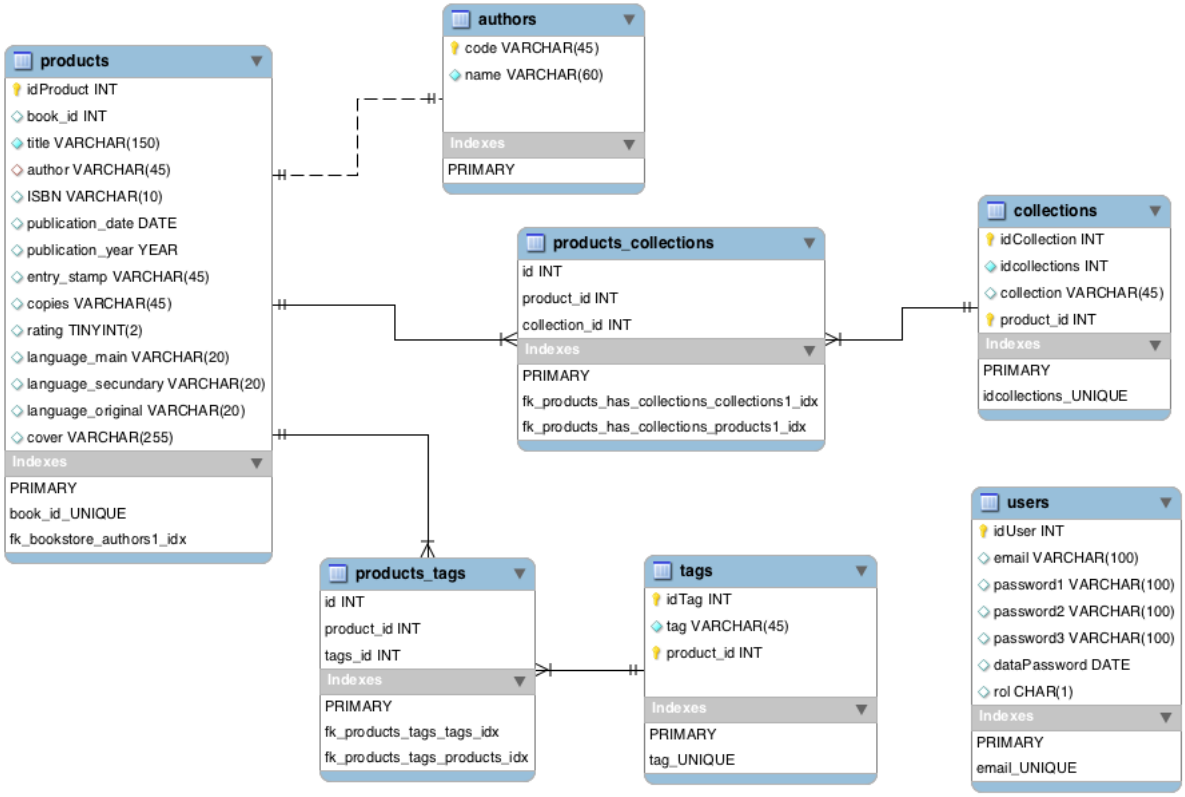

Figura 3: Relacions de la base de dades MySQL

La base de dades bookstore (tfm), contindrà les següents taules:

- Products: contindrà tots els articles registrats / importats.
- Authors: contindrà tots els autors disponibles, que s'hagin importat / creat, codificats amb un codi alpha-numèric, que és el que es registrarà a la taula de productes per relacionar l'autor amb l'article.
- Collections: contindrà totes les col·leccions disponibles, que s'hagin importat / creat. Hi ha un idCollection autoincrement, un idcollections que servirà per identificar-lo alhora de les importacions, ja que es un identificador importat de l'API. I una descripció.
- Tags: contindrà tots els tags disponibles, que s'hagin importat / creat. Hi ha un idTag auto-increment i una descripció.
- Users: contindrà tots els usuaris que poden accedir a l'aplicació. Hi haurà 3 tipus de rol 'A' Administrador, 'E' Editor i 'O' Observador.

Existeixen varies relacions 1:1 i N:M :

La relació 1:1 la taula products conté un camp author, que és el que el relaciona amb la taula authors.code.

Les relacions N:M, s'utilitzarà taules inter-mitges per relacionar: productes amb tags i productes amb col·leccions. Ja que Un producte pot tenir varis tags. Un tag pot estar dins de varis productes. I Un producte pot estar dins de varies col·leccions. I Una col·lecció pot estar dins de varis productes.

MongoDB

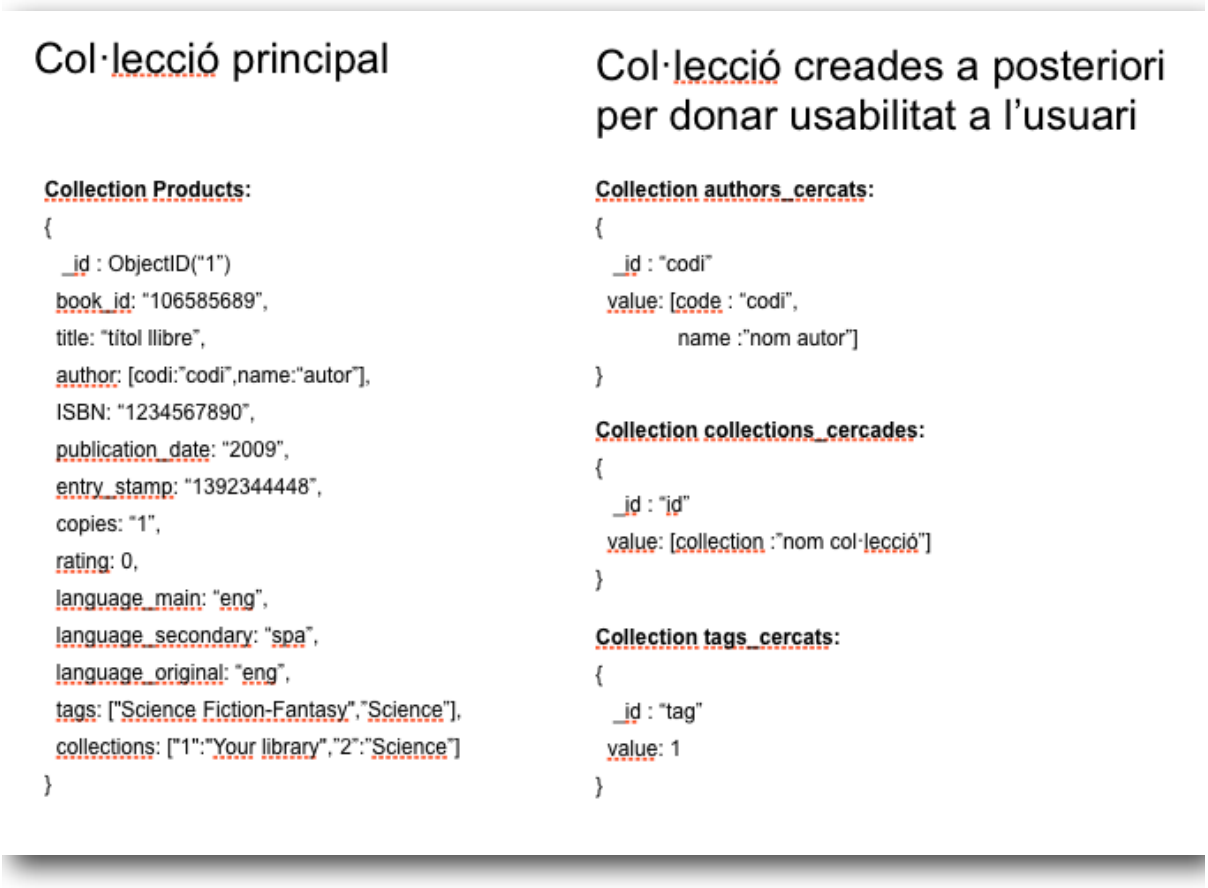

Figura 4: Estructura de les col·leccions del MongoDB

La base de dades de la bookstore (tfm) contindrà la col·lecció Products, l'estructura de la Figura 4, és l'estructura bàsica de la col·lecció. Aquestes estructures podran variar segons el contingut que s'importi / registri. Ja que si un atribut de l'objecte no hi és, aquest no es registrarà.

La col·lecció Products, contindrà tots els productes amb tota la seva informació. Es pot veure que els tags i les "collections" son arrays.

Les col·leccions authors_cercats, collections_cercades i tags_cercats, s'han creat a posteriori un cop la importació de les dades estava feta, durant el desenvolupament s'ha volgut donar usabilitat als usuaris que hi haguessin camps autocomplete i com que no es podien aconseguir registres únics des de la col·lecció products per aconseguir els autors, col·leccions i tags, s'ha hagut de realitzar tres map-reduce per aconseguir noves col·leccions per ajudar a sol·lucionar el problema.

8. Plataforma de desenvolupament

Per desenvolupar l'aplicació utilitzaré varis recursos tecnològics, explicats a continuació:

8.1 Angularjs

Angularjs és un framework 100% javascript de codi obert, s'executa en el client i es compatible amb els navegadors d'escriptori i mòbils, és compatible amb les últimes versions de Chrome, Firefox, Safari, Safari per IOs, internet explorer 9-11. La llicència es MIT. Usa el MVC (Model Vista Controlador) [19]

8.2 Fat-Free Framework

He escollit el Fat-Free Framework, perquè el recomanen al web de mongodb per desenvolupar aplicacions amb PHP, MongoDB i MySQL, a la seva pàgina web es defineixen com a un micro-framework de PHP dissenyat per ajudar a construir aplicacions web dinàmiques i robustes ràpidament. Porta un conjunt d'eines amb totes les funcions, codi base ocupa 65Kb, és fàcil d'aprendre, utilitzar i ampliar. La seva filosofia darrera el framework es tenir una estructura minimalista, evitant complexitat a l'aplicació i aconseguir un equilibri entre el codi elegant, rendiment de les aplicacions i la productivitat del programador. Es de llicència GPL v3. [20]

8.3 Bootstrap

És un framework creat per twitter, que permet crear interficies web amb CSS i Javascript, està preparat perquè el resultat final sigui adaptable a tots els dispositius "responsive design". Els dissenys amb bootstrap són simples, nets i intuïtius. Té varies llibreries amb estils predefinits fàcilment configurables: botons, menús desplegables, etc.

8.4 Sequel Pro

És un programa per Mac OS X gratuït per gestionar bases de dades MySQL. L'utilitzaré per crear i gestionar la base de dades MySQL, crear i gestionar les seves taules i camps. Per accedir a la base de dades es necessitaran les dades de connexió del MySQL. [21]

8.5 Robomongo

És un programa per Mac OS X gratuït per gestionar bases de dades MongoDB. Lútilitzaré per crear la base de dades, col·leccions i els camps per defecte. També l'utilitzaré per fer consulta sobre les col·leccions per veure si les operacions de gestionar les dades des de el PHP funcionen correctament. [22]

8.6 Repositori GIT

És un software de control de versions pensat en l'eficiència i con-fiabilitat de manteniment de versions d'aplicacions. Utilitzaré una sol·lució per us personal que és gratuïta que s'anomena BitBucket.

8.7 PhpStorm

És un IDE de programació desenvolupat per JetBrains, és compatible amb Windows, Linux i Mac OS X. Permet la gestió de projectes fàcilment, proporciona un fàcil auto-completat del codi. I hi ha un mòdul per ajudar amb el desenvolupament del framework Fat Free Framework. [33]

9. Planificació

9.1 Dates Clau

A continuació es mostren les dates claus del projecte, que corresponen a les diferents PACs i a l'entrega final del projecte:

Activitat	Data
PAC1	30/09/2015
PAC2	28/10/2015
PAC3	15/12/2015
Lliurament Final	08/01/2015

Taula 1: Dates clau

9.2 Diagrama de Gantt

A continuació es mostra el diagrama de Gantt, on es pot veure que està classificat per cinc tasques principals, que corresponen a les activitats principals: (PAC1, PAC2, PAC3,Lliurament Final i el desenvolupament de l'aplicació). Llavors cada tasca principal, hi ha les subtasques per poder portar a terme el projecte.

ID	Nom	Inici	Finalització	Durada	Relació
0	**PAC1**	16/09/2015	30/09/2015	11	
15	Definició del projecte	16/09/2015	21/09/2015	4	
16	Planificació del projecte	22/09/2015	24/09/2015	3	
18	Redacció de la memòria	22/09/2015	28/09/2015	5	
22	Revisió de la PAC1	28/09/2015	29/09/2015	2	
23	Entrega de la PAC1	30/09/2015	30/09/2015	1	
2	**PAC2**	01/10/2015	28/10/2015	20	
114	Definició de l'arquitectura de l'aplicació	01/10/2015	21/10/2015	15	
116	Realització dels diagrames UML i casos d'ús	01/10/2015	21/10/2015	15	
118	Realització del disseny centrat a l'usuari (Prototips, Perfils d'usuaris, Usabilitat)	01/10/2015	21/10/2015	15	
35	Redacció de la memòria	01/10/2015	21/10/2015	15	
26	Revisió de la memòria i de la PAC2	26/10/2015	27/10/2015	2	
27	Entrega de la PAC2	28/10/2015	28/10/2015	1	

ID	Nom	Inici	Finalització	Durada	Relació
95	**Desenvolupament de l'aplicació**	05/10/2015	01/01/2016	65	
96	Instal·lació i configuració del sistema (apache i bases de dades)	05/10/2015	09/10/2015	5	
98	Escollir la API REST PHP per desenvolupar la part de comunicació amb la base de dades	07/10/2015	07/10/2015	1	
97	Crear una API per fer les importacions i CRUDs per aplicar-los en les dues situacions, SQL i NOSQL	08/10/2015	06/11/2015	22	98
101	Descarregar JSONS de https://www.librarything.com/ per poder-los importar	16/10/2015	22/10/2015	5	
100	Desenvolupar un programa per importar els JSONS i guardar log d'informació de les importacions	23/10/2015	30/10/2015	6	101
99	Importar els JSONS, i validar la generació de logs pel seu funcionament	02/11/2015	04/12/2015	25	100,101
102	Crear la inferficie d'usuari per poder fer ús dels CRUDs i poder fer les proves	28/10/2015	04/12/2015	28	
110	Crear l'entorn privat, amb accés amb usuari i contrassenya	23/11/2015	01/12/2015	7	
109	Fer comparacions de com ha anat amb les diferents base de dades	07/12/2015	30/12/2015	18	97,102, 110
106	Testejar l'aplicació	07/12/2015	15/12/2015	7	
3	**PAC3**	29/10/2015	15/12/2015	34	2
37	Redacció de la memòria	29/10/2015	13/12/2015	32	
30	Revisió de la memòria i de la PAC3	14/12/2015	14/12/2015	1	
31	Entrega de la PAC3	15/12/2015	15/12/2015	1	
5	**Lliurament Final**	16/12/2015	08/01/2016	18	3
39	Revisió la programació i BBDD	16/12/2015	29/12/2015	10	
40	Revisió dels testos	16/12/2015	29/12/2015	10	
42	Revisió de la Memòria Final	16/12/2015	29/12/2015	10	
43	Crear el vídeo de la prestació	30/12/2015	05/01/2016	5	42
44	Revisió de la presentació	06/01/2016	07/01/2016	2	43
46	Entrega de la memòria i del vídeo	08/01/2016	08/01/2016	1	44

Taula 2: Diagrama de Gantt

ID	Nom	Inici	Finalització	Durada	Relacions
0	PAC1	16/09/15	30/09/15	11	
15	Definició del projecte	16/09/15	21/09/15	4	
16	Planificació del projecte	22/09/15	24/09/15	3	
18	Redacció de la memòria	22/09/15	28/09/15	5	
22	Revisió de la PAC1	28/09/15	29/09/15	2	
23	Entrega de la PAC1	30/09/15	30/09/15	1	
2	PAC2	01/10/15	28/10/15	20	0
114	Definició de l'arquitectura de l'aplicació	01/10/15	21/10/15	15	
116	Realització dels diagrames UML i casos d'ús	01/10/15	21/10/15	15	
118	Realització del disseny centrat a l'usuari (Prototips, Perfils d'usuaris, Usabilitat)	01/10/15	21/10/15	15	
35	Redacció de la memòria	01/10/15	21/10/15	15	
26	Revisió de la memòria i de la PAC2	26/10/15	27/10/15	2	
27	Entrega de la PAC2	28/10/15	28/10/15	1	
95	Desenvolupament de l'aplicació	05/10/15	18/12/15	55	
96	Instal·lació i configuració del sistema (apache i bases de dades)	05/10/15	09/10/15	5	
98	Escollir la API REST PHP per desenvolupar la part de comunicació amb la base de dades	07/10/15	07/10/15	1	
97	Crear una API per fer les importacions i CRUDs per aplicar-los en les dues situacions, SQL i NOSQL	08/10/15	06/11/15	22	98
101	Descarregar JSONS de https://www.librarything.com/ per poder-los importar	16/10/15	22/10/15	5	
100	Desenvolupar un programa per importar els JSONS i guardar log d'informació de les importacions	23/10/15	30/10/15	6	101
99	Importar els JSONS, i validar la generació de logs pel seu funcionament	02/11/15	04/12/15	25	100,101
102	Crear la inferficie d'usuari per poder fer ús dels CRUDs i poder fer les proves	28/10/15	04/12/15	28	
110	Crear l'entorn privat, amb accés amb usuari i contrassenya	23/11/15	01/12/15	7	
109	Fer comparacions de com ha anat amb les diferents base de dades	07/12/15	18/12/15	10	97,102,110
106	Testejar l'aplicació	07/12/15	15/12/15	7	
3	PAC3	29/10/15	15/12/15	34	2
37	Redacció de la memòria	29/10/15	11/12/15	32	
30	Revisió de la memòria i de la PAC3	14/12/15	14/12/15	1	
31	Entrega de la PAC3	15/12/15	15/12/15	1	
5	Lliurament Final	16/12/15	08/01/16	18	3
39	Revisió la programació i BBDD	16/12/15	29/12/15	10	
40	Revisió dels testos	16/12/15	29/12/15	10	
42	Revisió de la Memòria Final	16/12/15	29/12/15	10	
43	Crear el vídeo de la prestació	30/12/15	05/01/16	5	42
44	Revisió de la presentació	06/01/16	07/01/16	2	43
46	Entrega de la memòria i del vídeo	08/01/16	08/01/16	1	44

Figura 5: Diagrama de Gantt

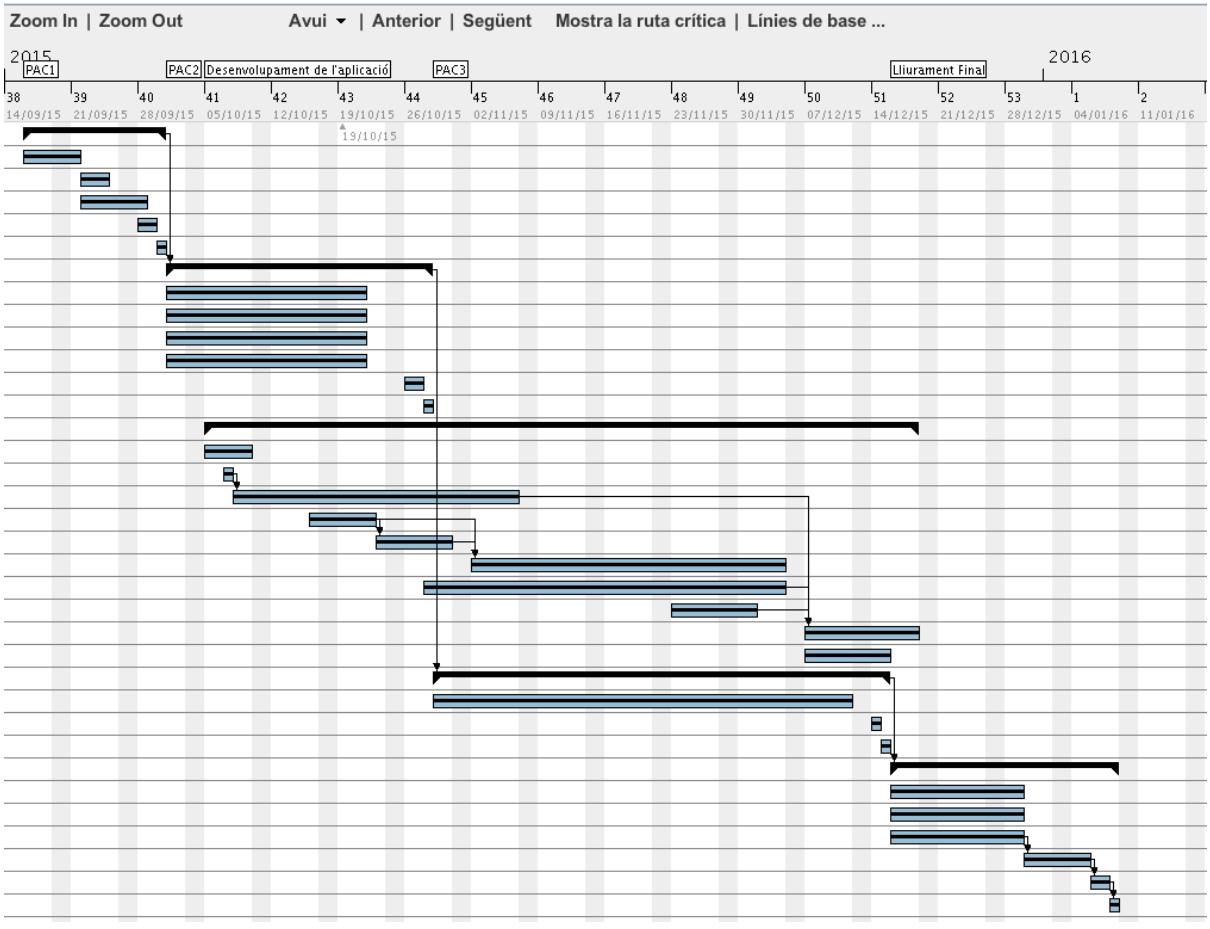

Figura 6 Diagrama de Gantt

9.3 Riscos

Durant la realització del projecte podrien sorgir els següents riscos:

Fallades del sistema / infraestructura:

Durant el desenvolupament del projecte, ens podríem trobar que hi hagués un problema amb el servidor / màquina en que es desenvolupa l'aplicació i es perdes la informació.

Les contingències que realitzaria seria guardar el codi de l'aplicació en un repositori i en un pendrive. Llavors podríem restaurar el codi font des de l'última còpia. També es guardaria una còpia de les bases de dades (estructura i dades) en el pendrive, per poder-les restaurar. També haurem de tenir en compte que s'haurà de restablir el sistema (re-configurar el servidor/màquina de treball) per poder continuar treballant.

Fallades del software:

Durant el desenvolupament del projecte, ens podem trobar que el desenvolupador, per accident, malmeti l'aplicació i deixi de funcionar l'aplicació.

Les contingències que realitzaria seria usar un control de versions del software per tal de recuperar i poder revisar les transaccions.

Problemes amb les eines escollides:

Durant el desenvolupament del projecte, ens podem trobar que el desenvolupador, desconeix-hi o bé no estigui habituat amb les eines escollides pel desenvolupament web.

Les contingències que realitzaria seria formar el desenvolupador per poder resoldre els contra-temps.

Errors del servei:

Durant el desenvolupament del projecte, ens podem trobar que en un moment determinat el servidor de PHP, MySQL i MongoDB, deixin de funcionar.

Les contingències que realitzaria seria restablir el servei que doni problemes. Per poder saber si el servei ha fallat, necessitarem instal·lar un sistema de monitoratge d'equips i serveis de xarxa, com per exemple el Nagios [8] [9]. També es podria aprofitar per analitzar el MySQL, el phpMyAdmin que ofereix una opció per mostrar els processos obert. I amb l'instrucció top del linux poder saber els processos si estan encallats. I depenent de l'anàlisi llavors s'hauria de detenir els processos penjats.

10. Procés de treball/desenvolupament

10.1 Planificació del projecte

La planificació del projecte estar orientada segons les entregues que es realitzaran durant el seu curs. Hi ha quatre dates claus d'entrega comentades a la planificació. Es farà tot el possible per seguir els terminis establerts per cada punt per aconseguir la realització del projecte. Hi hauran fites que són punts claus que s'hauran d'assolir per dur a terme l'aplicació. Per acabar d'assolir el projecte hi ha la tasca del desenvolupament de l'aplicació on a dins d'ella hi ha sub-tasques per poder-la desenvolupar.

10.2 Allotjament / Servidor web

Per poder desenvolupar el projecte, necessitarem un servidor LAMP (Linux Apache MySQL PHP) i amb MongoDB. S'ha llogat un servidor web a http://www.arubacloud.es per la realització del projecte, i s'ha instal·lat tot el sistema (LAMP i MongoDB) per poder funcionar l'aplicació. Llavors també s'ha configurat una màquina virtual amb el programa VirtualBox amb les mateixes característiques que el servidor web. El servidor web s'hi accedirà mitjançant la seva IP pública, no hi haurà cap domini associat. Per accedir a la màquina virtual s'hi accedirà mitjançant la seva IP de xarxa. Per realitzar les importacions dels fitxers JSONS, es realitzaran des de la configuració amb local, un XAMPP amb MongoDB.

10.3 Desenvolupament de l'aplicació

Per dur a terme el desenvolupament de l'aplicació, la part del servidor PHP per realitzar les operacions de CRUD, s'utilitzarà un micro-framework anomenat Fat-Free Framework, que és compatible amb varies bases de dades MySQL i MongoDB, aquest framework per mi es desconegut, ja que no hi he treballat mai, per tant hauré d'habituar-me en ell per dur a terme l'aplicació a desenvolupar. Hi haurà diferents funcions, per retornar la cerca de l'usuari, per modificar un registre, per donar d'alta un registre i per eliminar un registre.

En la part del client, es realitzarà amb el framework Angularjs, aquest el conec una mica, ja que en el lloc de treball, estic col·laborant amb projectes on s'utilitza aquesta eina. S'utilitzarà per realitzar la interactivitat amb l'usuari, per les operacions de cerca / filtratge.

El primer que es realitzarà serà crear l'estructura de les bases de dades, i importar dades a la base de dades per així poder fer testos amb un gran volum de dades per poder tenir un joc de proves. Les dades s'importaran de les APIS públiques del web http://www.librarything.com. Llavors, es durà a terme el disseny per veure quines funcionalitats necessitarem per programar en l'angularjs, en l'apartat 10.4 explico més detalladament el disseny. Un cop tinguem clara l'estructura prepararem l'API perquè ens retorni resultats per poder acabar de preparar la part interactiva amb l'usuari de fer

filtratges / cerques. Un cop tinguem funcionant les cerques llavors prepararem la part de altes, modificacions i eliminació de registres, per les dues bases de dades.

10.4 Disseny

El primer disseny que es realitzarà serà de baix nivell, creant wireframes tenint en compte la seva usabilitat. Un cop tinguem els wireframes es portarà a terme la seva maquetació amb HTML, CSS amb el suport del framework Bootstrap, aquest framework es compatible amb l'Angularjs. Un cop tinguem el disseny fet llavors hi posarem la programació javascript d'Angularjs per realitzar la interacció amb l'usuari: filtratges / cerca de dades.

10.5 Entorn Privat

L'aplicació que es realitza és un BackOffice, aquest serà un entorn privat on per accedir-hi s'haurà d'entrar amb un usuari, contrasenya i un captcha. Es guardaran cookies per la identificació i la seguretat. El BackOffice permetrà canviar la contrasenya de l'usuari, aquesta contrasenya es encriptada i es guardaran les 3 últimes, així quan es canvii la contrasenya no podrà ser cap de les tres anteriors.

10.6 Proves

Es crearan jocs de prova per poder realitzar l'estudi que es vol aconseguir que són: els punts forts i dèbils en la definició de les dades, en la seva tipologia, validacions de les dades, com afronten els valors a null, camps a multi-avaluats. També veurem com es realitzen les operacions CRUD a les diferents base de dades. Per aconseguir el jocs de proves, farem importacions de registres a les diferents bases de dades, per poder fer filtratges, cerques, modificacions i eliminacions de registres.

Es portaran a terme els jocs de proves per poder veure si el funcionament de l'aplicació es el correcte.

11. APIs utilitzades

11.1 LibraryThing APIs

Librarything.com és una web col·laborativa on els seus usuaris pengen el seu catàleg de llibres, música, pel·licules, poadcasting, etc. En aquest portal hi ha una API on retorna arxius JSON. Aquests JSONs s'utilitzaran per alimentar les bases de dades Mysql i MongoDB. Els JSONs retornats, són d'usuaris específics que en vols extreure les seves dades. S'agafaran varis usuaris i es carregarà la seva informació dins del sistema, cada usuari tindrà una varietat de col·leccions (llibres, música, pel·licules, poadcasting, etc.) La informació que s'aconseguirà serà diferent depenent del seu contingut.

Es pot veure un exemple a:
http://www.librarything.com/api_getdata.php?userid=steelprimate&showstructure=1&max=1&showTags=1&showCovers=1&showAuthors=1&showTitles=1&showRatings=1&showCollections=1&showReviews=1&booksort=title_REV

Les dades obtingudes son de l'usuari "steelprimate".

Es pot veure com funciona la API a: https://www.librarything.com/wiki/index.php/LibraryThing_JSON_Books_API

A continuació mostro informació extreta de la seva pàgina web on es pot veure els paràmetres que retornen els arxius JSON:

book_id	The unique id of every book
title	The title of your book
author_lf	Author (last, first)
author_fl	Author (first, last)
author_code	The author code that gets the author page, eg., twainmark produces the author URL http://www.librarything.com/author/twainmark
ISBN	Book's ISBN as entered
ISBN_cleaned	Book's ISBN with extraneous dashes and other data removed, converted to a 10-digit ISBN if a 13-digit 978-based ISBN. Omitted if the ISBN is not valid.

publicationdate	The publication date, as entered
entry_stamp	Unix timestamp of entry
entry_date	The date of entry (US format)
copies	The "copies" field, as entered
language_main	Main language
language_secondary	Secondary language
language_original	Original language
hasreview	Does the book have a review?
rating	Rating of book. 0 is unrated.
bookreview	The review itself, limited by reviewmax (above)
bookreview_stamp	Unix timestamp of review date
bookreview_date	Date of review (US format)
tags	An array of all tags per book, in display order

Taula 3: Variables del JSON

12. Diagrames UML

12.1 Casos d'ús

Títol: Cercar un usuari.
Actors: Administrador, Editor, Observador.
Descripció: Cercar un usuari per poder visualitzar les seves dades.
Resultat: Aconseguir la informació dels usuaris que compleixen les regles de la cerca.

Títol: Crear un usuari.
Actors: Administrador.
Descripció: Donar d'alta un usuari amb un rol, perquè pugui entrar dins de l'aplicació i poder realitzar les tasques que se l'hi doni permís segons el perfil d'usuari que se l'hi assigni.
Resultat: Donar d'alta un usuari.

Títol: Veure un usuari.
Actors: Administrador, Editor, Observador.
Descripció: Seleccionar un usuari per poder veure la seva informació.
Resultat: Aconseguir la informació de l'usuari que s'ha seleccionat.

Títol: Modificar un usuari.
Actors: Administrador.
Descripció: Modificar les dades del registre de l'usuari per canviar la informació registrada.
Resultat: Modificar un usuari.

Títol: Eliminar un usuari.
Actors: Administrador.
Descripció: Eliminar un usuari escollit de la base de dades.
Resultat: Donar de baixa un usuari.

Títol: Cercar un producte.
Actors: Administrador, Editor, Observador.
Descripció: Cercar un producte per poder visualitzar les seves dades.
Resultat: Aconseguir la informació dels usuaris que compleixen les regles de la cerca.

Títol: Crear un producte.
Actors: Administrador, Editor.
Descripció: Donar d'alta un producte amb totes les seves dades disponibles.
Resultat: Donar d'alta un producte.

Títol: Veure un producte.
Actors: Administrador, Editor, Observador.

Descripció: Seleccionar un producte per poder veure la seva informació.
Resultat: Aconseguir la informació del producte que s'ha seleccionat.

Títol: Modificar un producte.
Actors: Administrador, Editor.
Descripció: Modificar les dades del registre del producte per canviar la informació registrada.
Resultat: Modificar un producte.

Títol: Eliminar un producte.
Actors: Administrador, Editor.
Descripció: Eliminar un producte escollit de la base de dades.
Resultat: Donar de baixa un producte.

12.2 Diagrama de classes

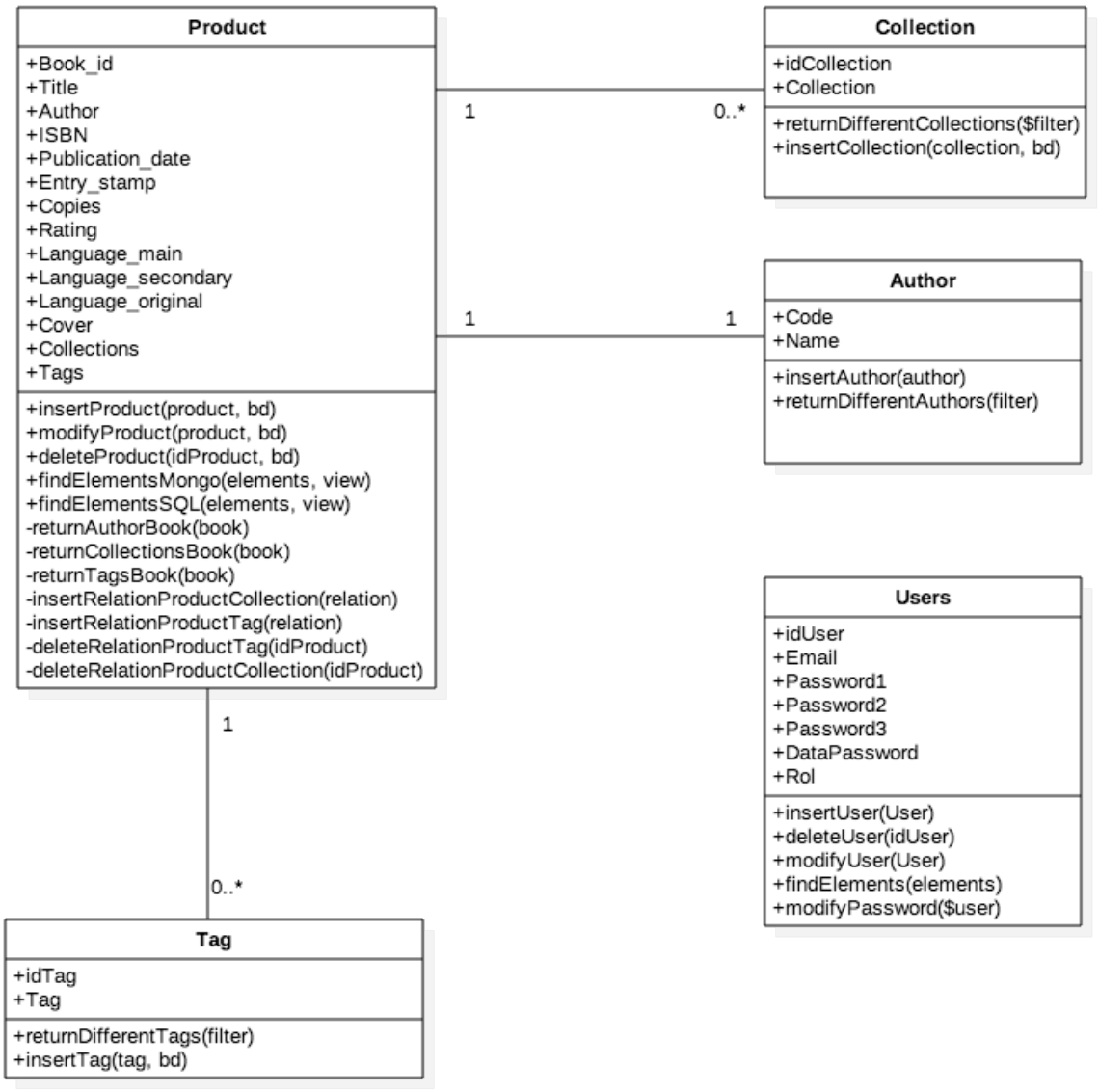

Figura 7: Diagrama de Classes

Product: Classe que utilitzarem per gestionar els diferents productes.

Collection: Classe que utilitzarem per gestionar les col·leccions, inserir i consultar els valors

Tag: Classe que utilitzarem per gestionar els tags, inserir i consultar els valors

Author: Classe que utilitzarem per gestionar els autors, inserir i consultar els valors

Users: Classe que utilitzarem per gestionar els diferents usuaris.

13. Prototips

A continuació es mostren els prototips de baixa fidelitat:

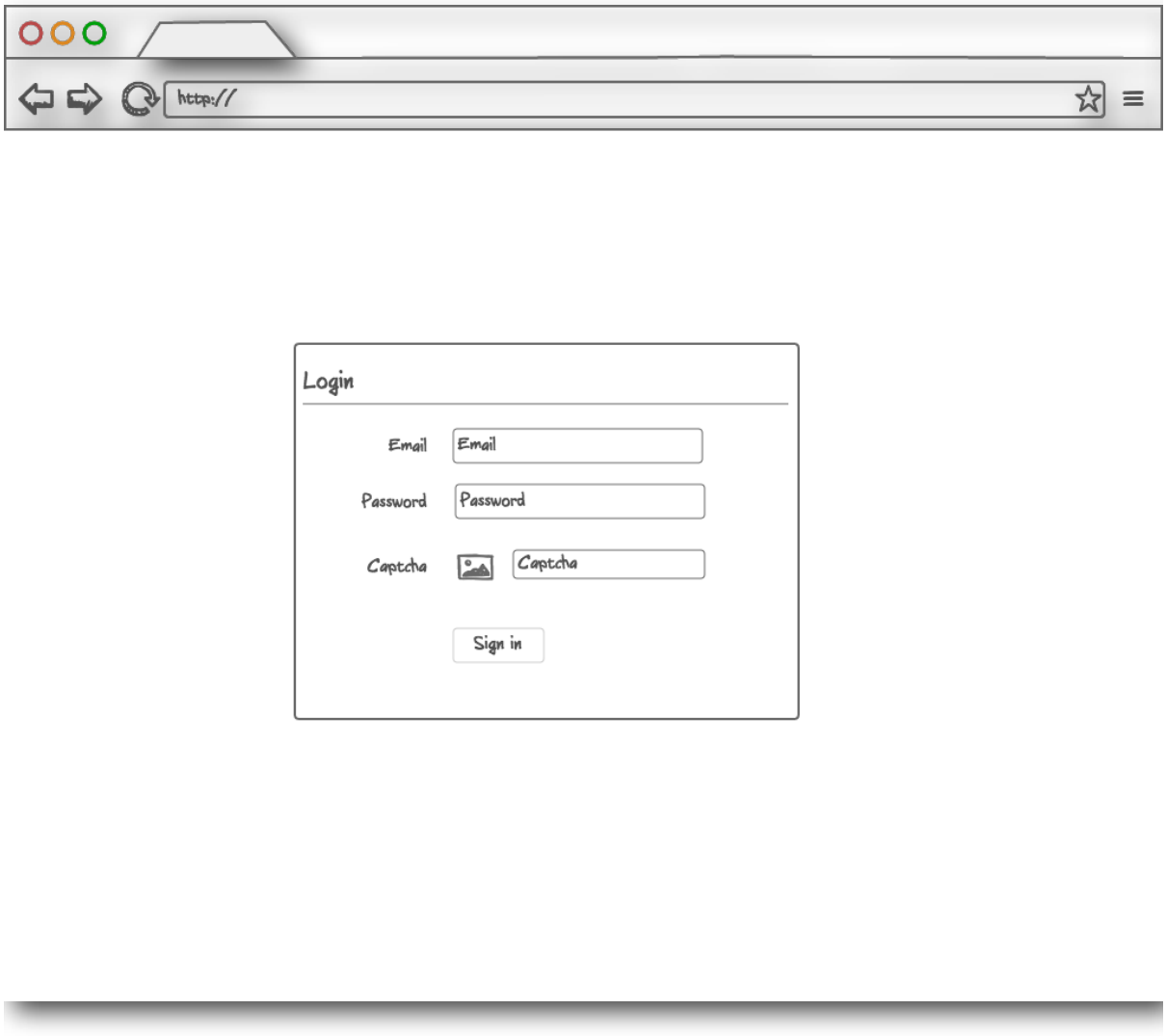

Figura 8: Wireframe pantalla de login

Aquesta pantalla és la pantalla inicial, es la pantalla d'identificació de l'usuari. Aquí l'usuari ha d'entrar les seves credencials per entrar al sistema. Per evitar robots, hi ha el captcha.

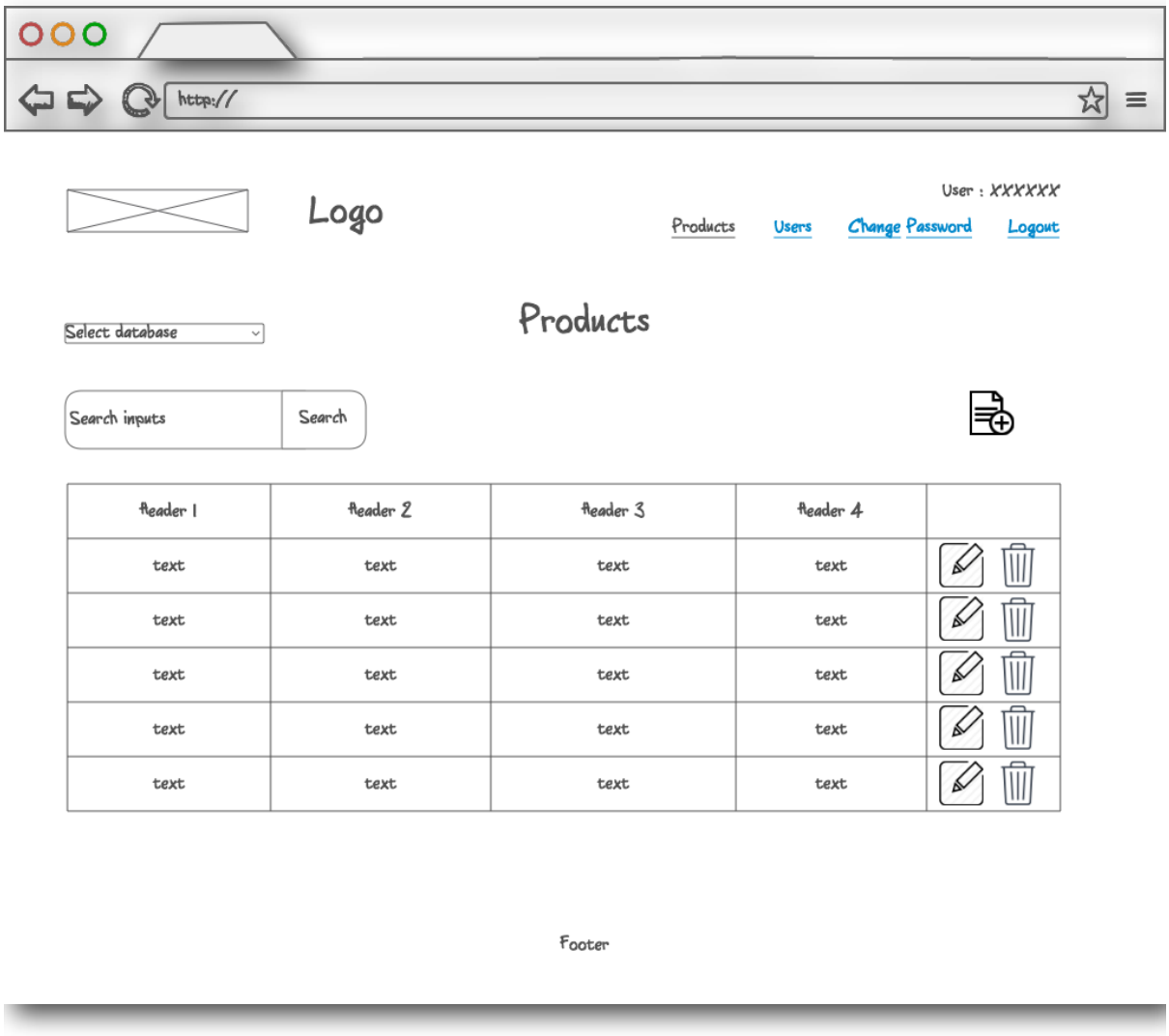

Figura 9: Wireframe pantalla cerca de productes

Un cop l'usuari està identificat l'hi apareixerà la pàgina de productes on l'hi apareixeran els últims productes entrats de la base de dades per defecte. Llavors podrà cercar productes amb la bases de dades desitjada. Pel selecciona-ble "Select database" seleccionarà la base de dades que vol interaccionar. I pels "Search inputs" seleccionarà els diferents filtres per per la cerca. Per fer la cerca s'ha de clicar sobre el botó Search.
En la part superior dreta hi haurà el menú on es veuen els diferents apartats, Productes, Users, Canviar la contrasenya i sortir de l'aplicació.
Es pot veure una taula, on contindrà els registres de la cerca, es pot observar que a cada fila hi ha dos botons, un llapis i una paperera. El llapis servirà per visualitzar la informació detallada de l'article i poder-lo modificar. La paperera servirà per eliminar el registre. Quan es clica a sobre el llapis es visualitzarà el següent wireframe. Quan es clica sobre la paperera sortirà una finestra emergent per confirmar l'eliminació.
Per donar d'alta un nou article s'ha de clicar sobre el paper amb el símbol +. Llavors apareixerà el següent wireframe.

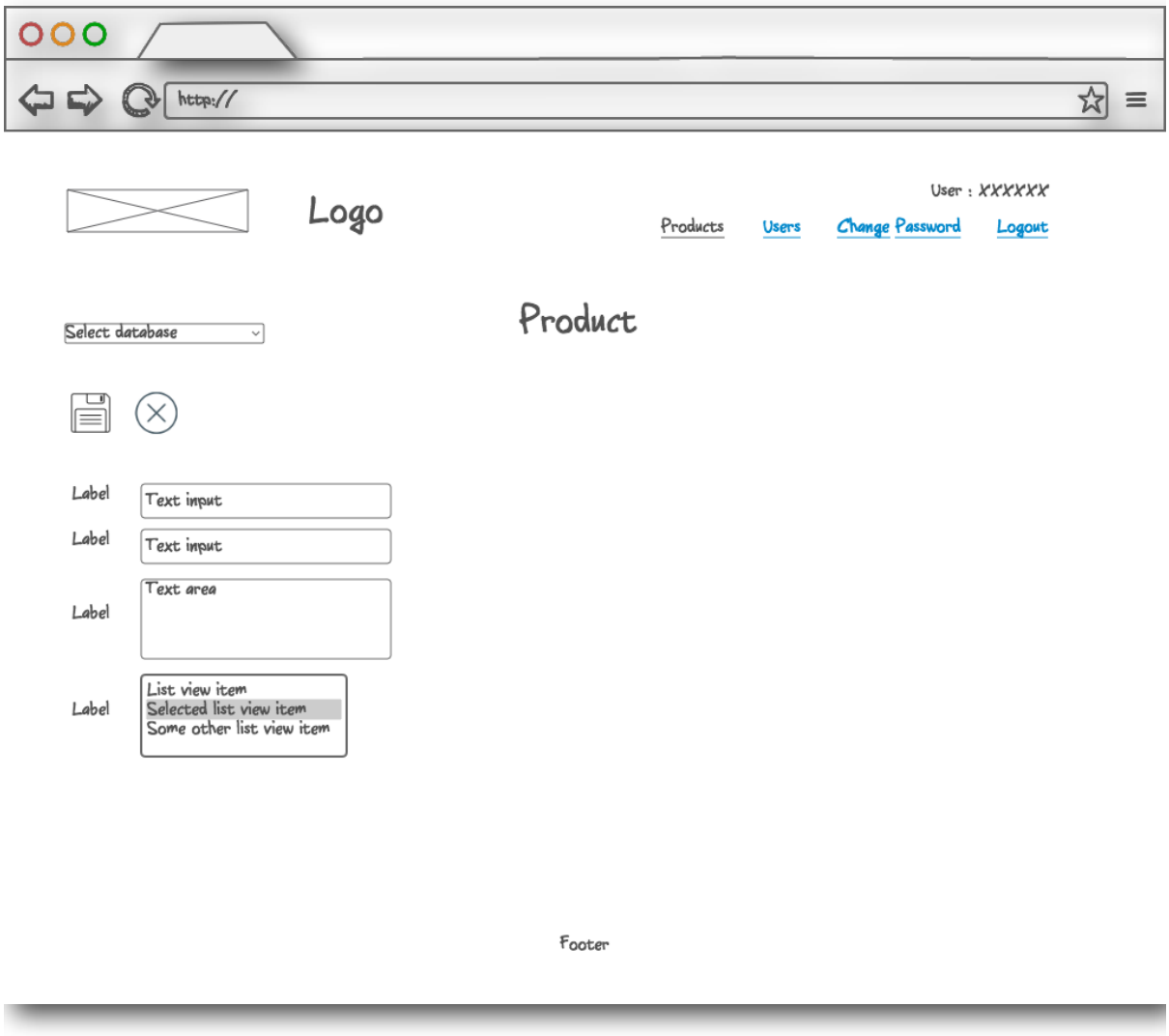

Figura 10: Wireframe pantalla detall del producte

Aquesta pantalla té varies funcionalitats: pantalla de donar d'alta un producte, pantalla de visualització de més informació de producte i pantalla de modificar un producte.
Per guardar la informació tant en modificar com alta, s'haurà de clicar el disquet. Quan es clica el disquet, llavors apareixerà una finestra emergent per confirmar que es vol guardar. Per cançel·lar la operació o bé tornar enrere (a la pantalla de cerca de productes) clicar la creu.

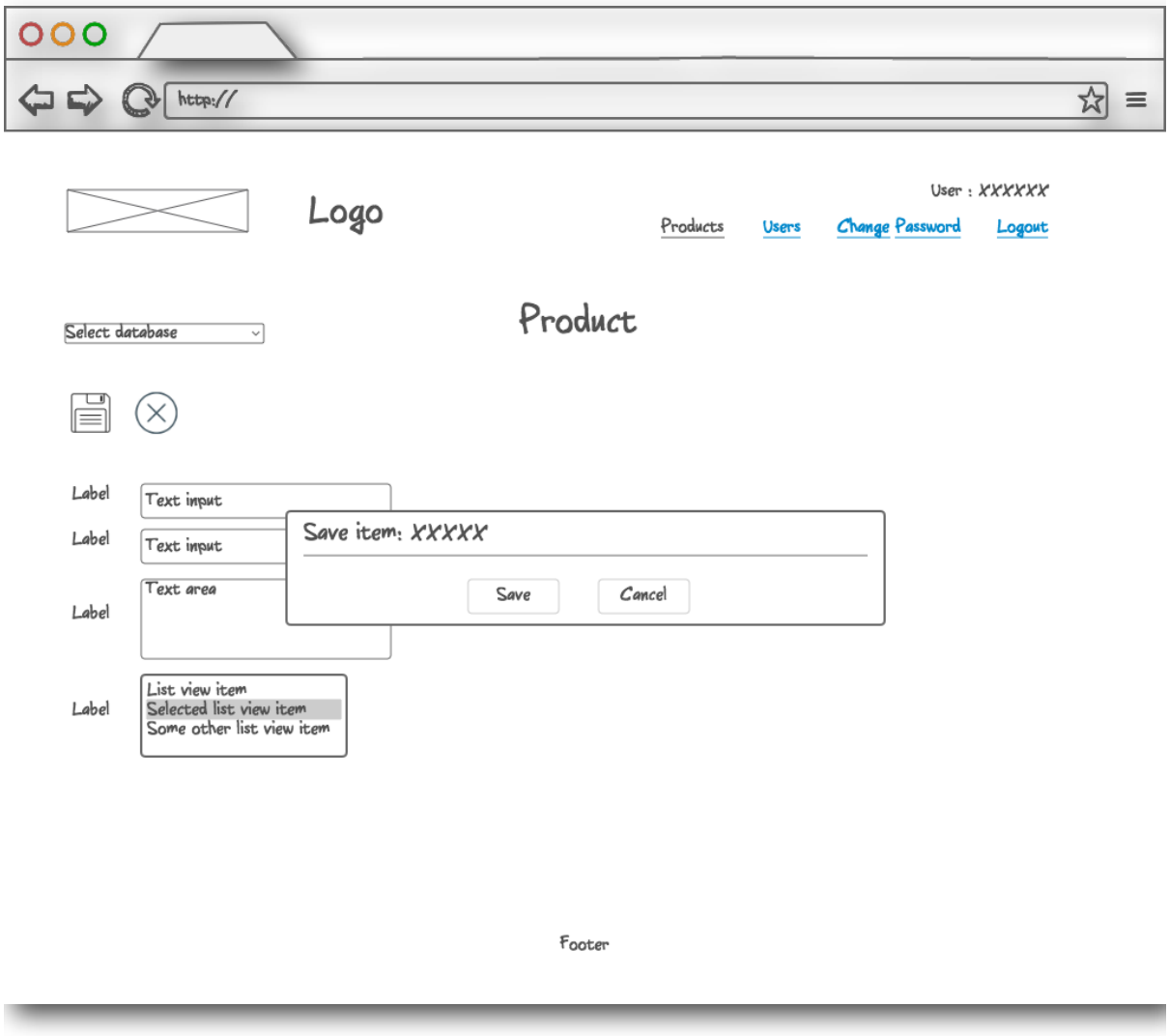

Figura 11: Wireframe pantalla detall del producte, amb finestra emergent per confirmar si vols guardar o no el producte.

Un cop cliques gravar, la següent pantalla que apareix es la de cerca de productes. El boto cancel·lar continues a la mateixa pantalla.

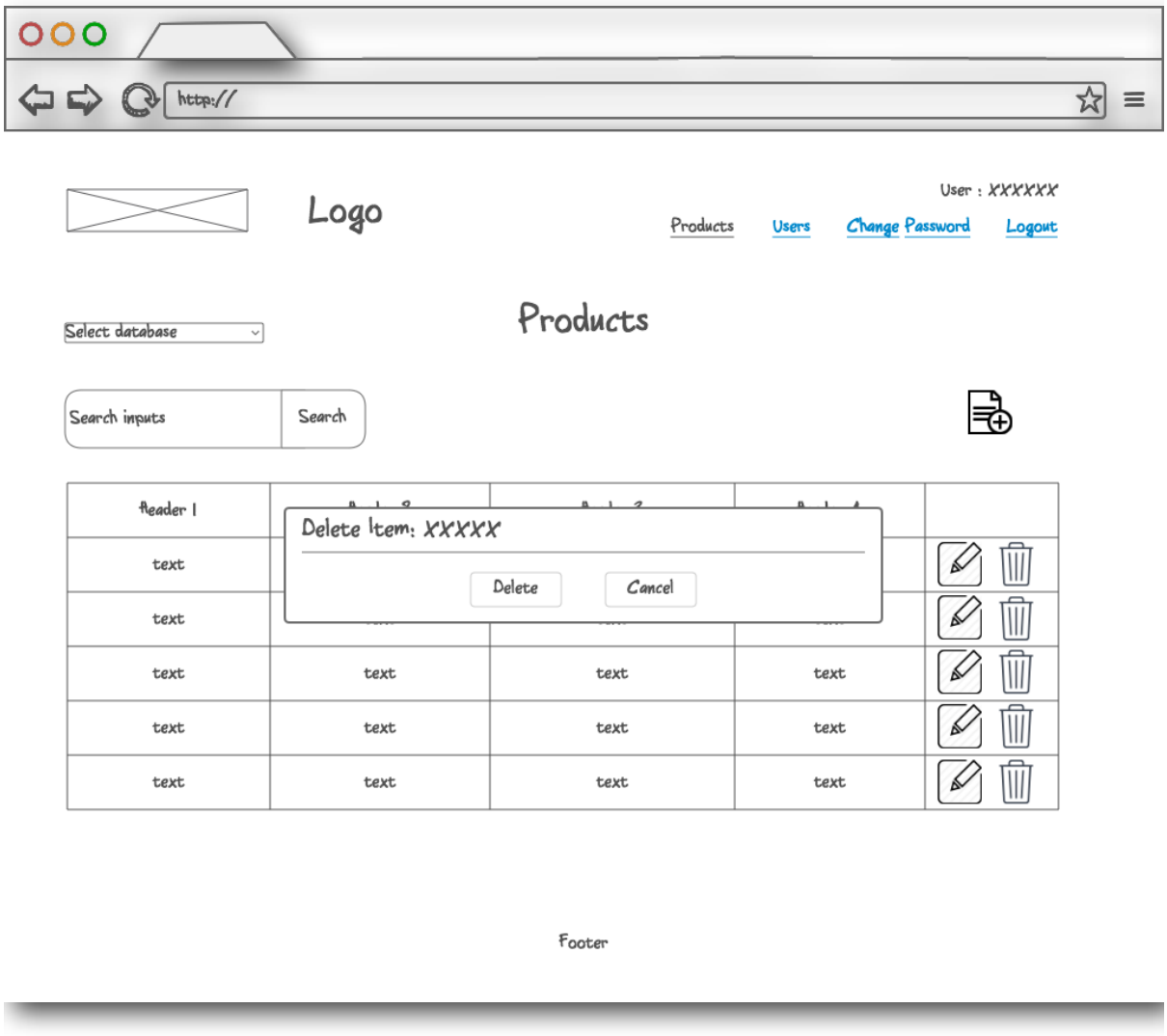

Figura 12: Wireframe pantalla de cerca del producte, amb finestra emergent per confirmar si vols eliminar o no el producte.

Qualsevol botó que es cliquí estaràs a la mateixa pantalla.

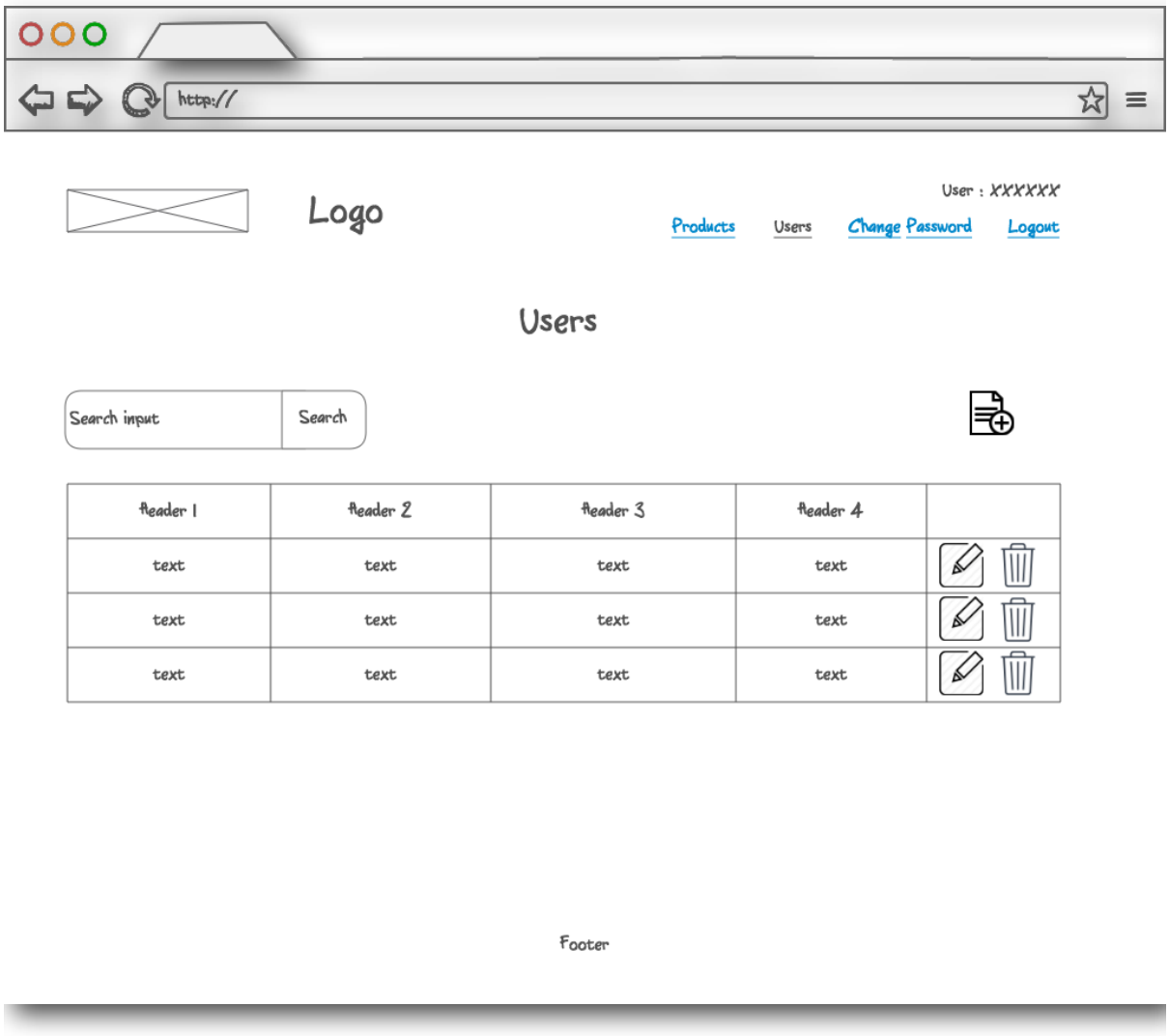

Figura 13: Wireframe pantalla cerca de usuaris

Aquesta pantalla serveix per l'administrador del sistema per realitzar cerques dels usuaris que hi ha al sistema, poder donar d'alta, eliminar usuaris i accedir al detall de l'usuari per veure'n més informació. Si cliques al llapis aniràs a la pantalla que es mostrarà a continuació. Si cliques a la paperera et sortirà una finestra emergent per confirmar l'eliminació.

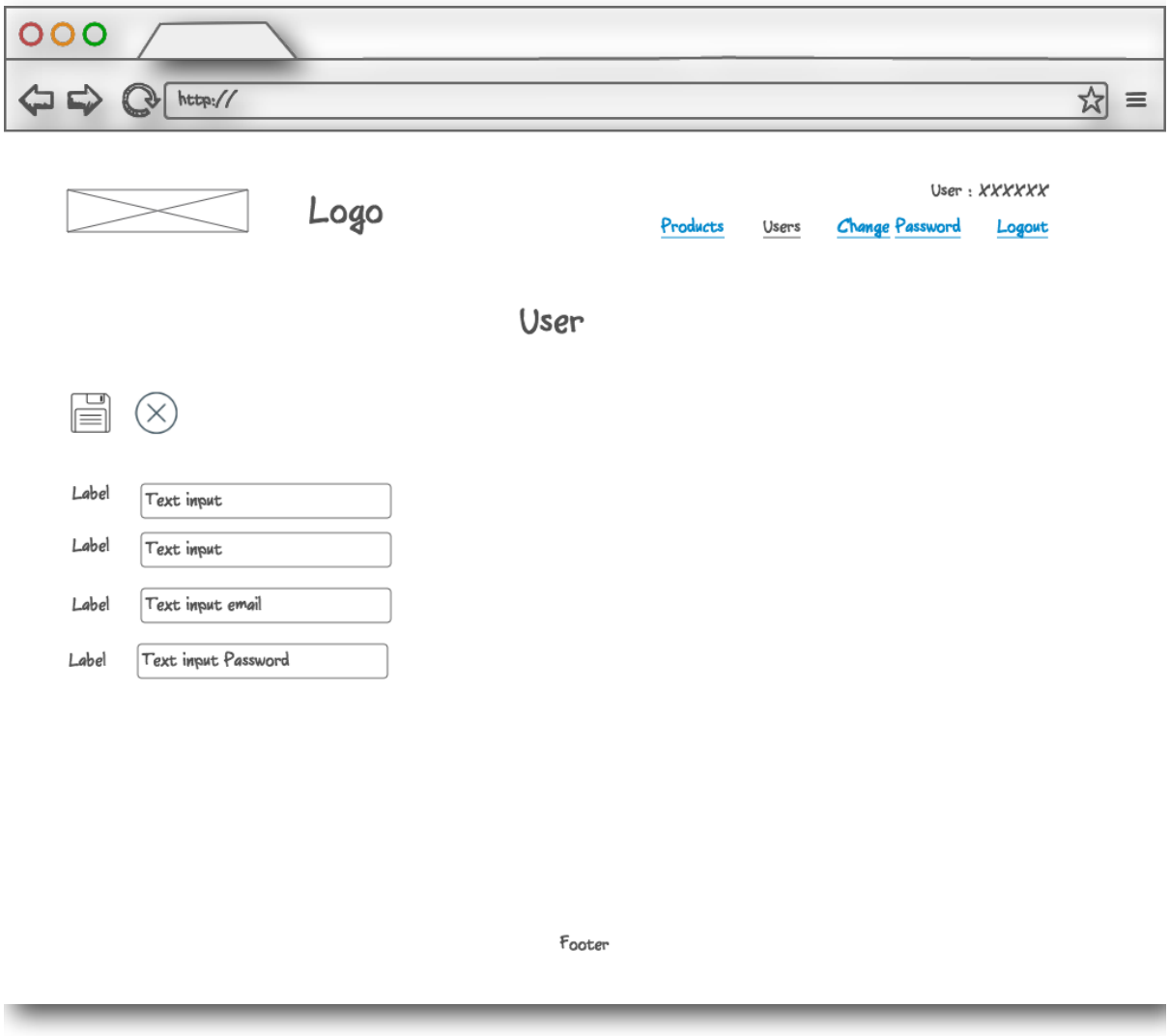

Figura 14: Wireframe pantalla detall d'usuari

Aquesta pantalla serveix per l'administrador del sistema per poder veure el detall de l'usuari i poder-ne modificar les dades. Llavors podrà guardar o bé cancel·lar la operació.

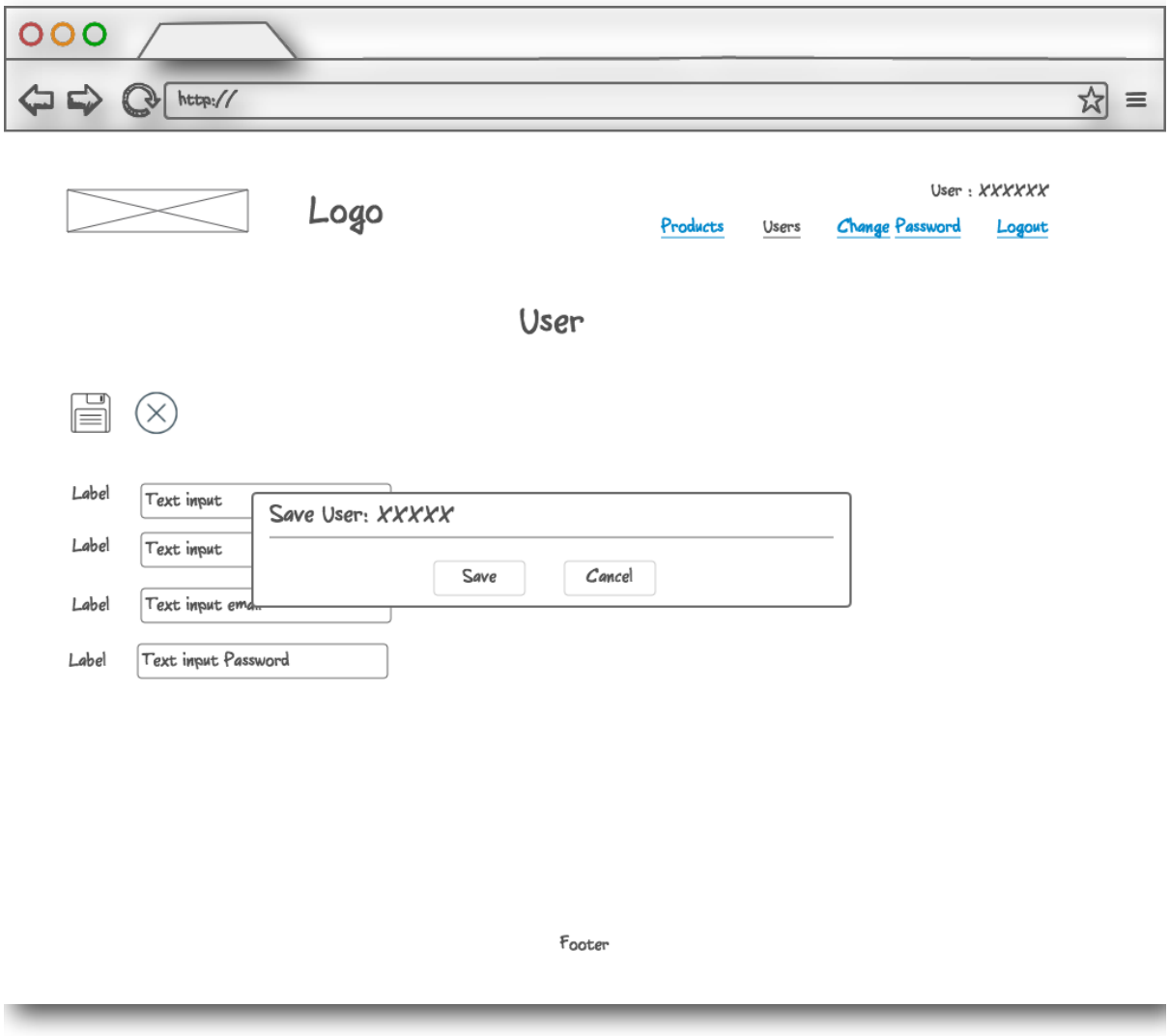

Figura 15: Wireframe pantalla detall d'usuari amb finestra emergent per gravar l'usuari.

Si es clica el boto gravar aniràs al a pantalla de cerca d'usuaris, si cliques el boto cancel·lar et mantindràs a la mateixa pantalla.

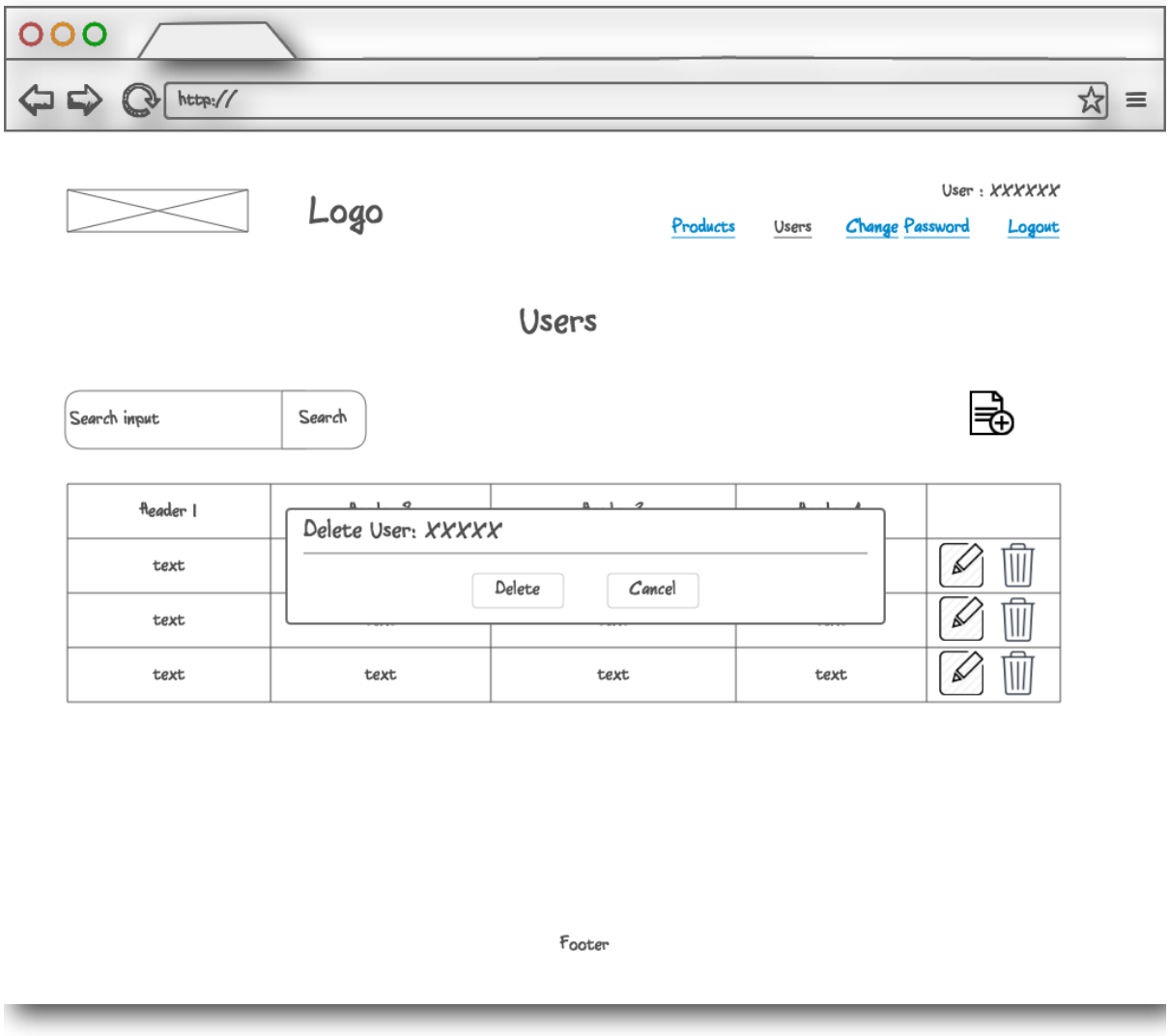

Figura 16: Wireframe pantalla cerca d'usuari amb finestra emergent per eliminar l'usuari.

Qualsevol botó que es cliqui et mantindràs a la mateixa pantalla.

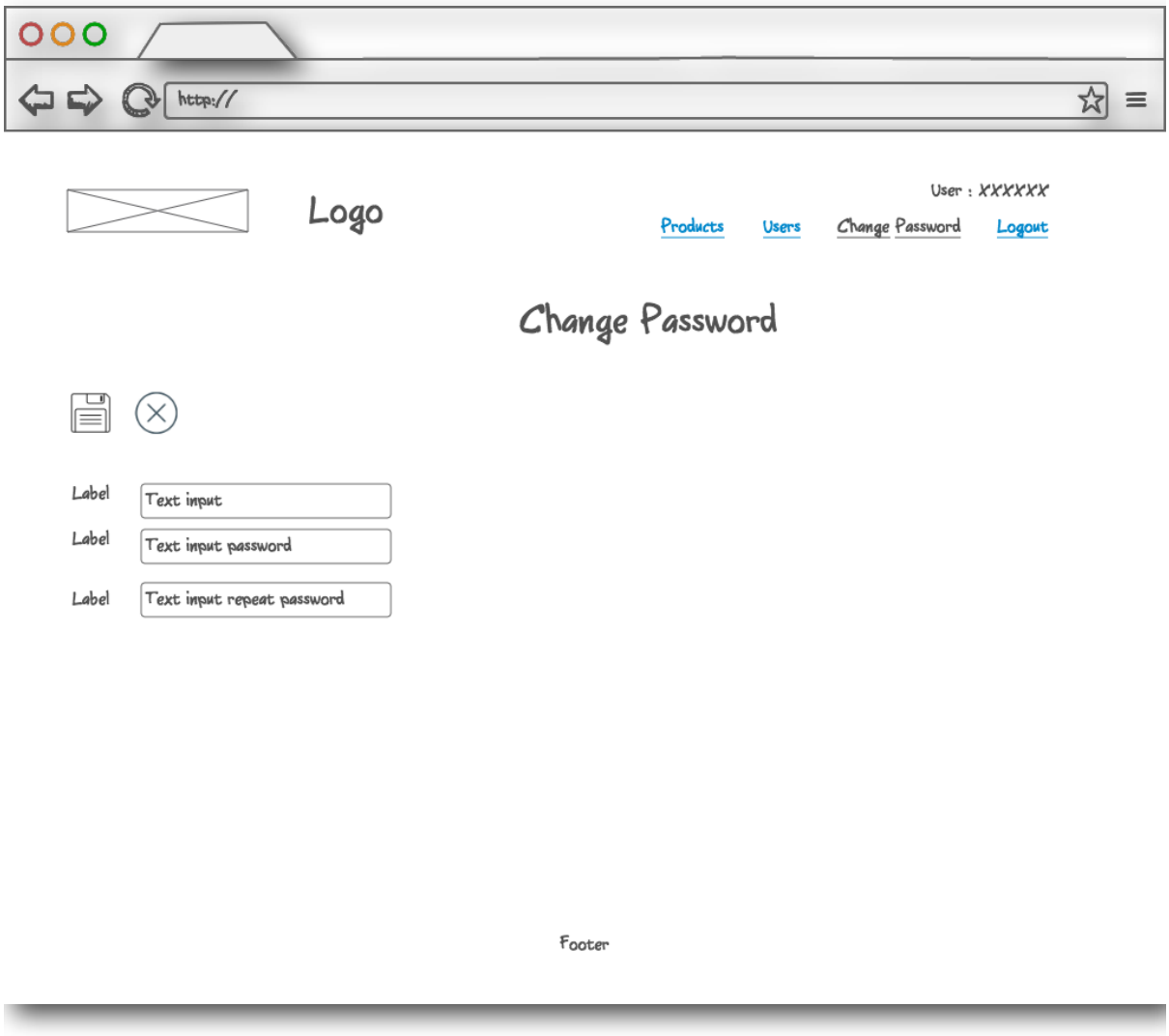

Figura 17: Wireframe pantalla de modificació de password

Aquesta pantalla serveix per tots els perfils d'usuaris per canviar la seva contrasenya. A la pantalla informarà si la contrasenya es segura.

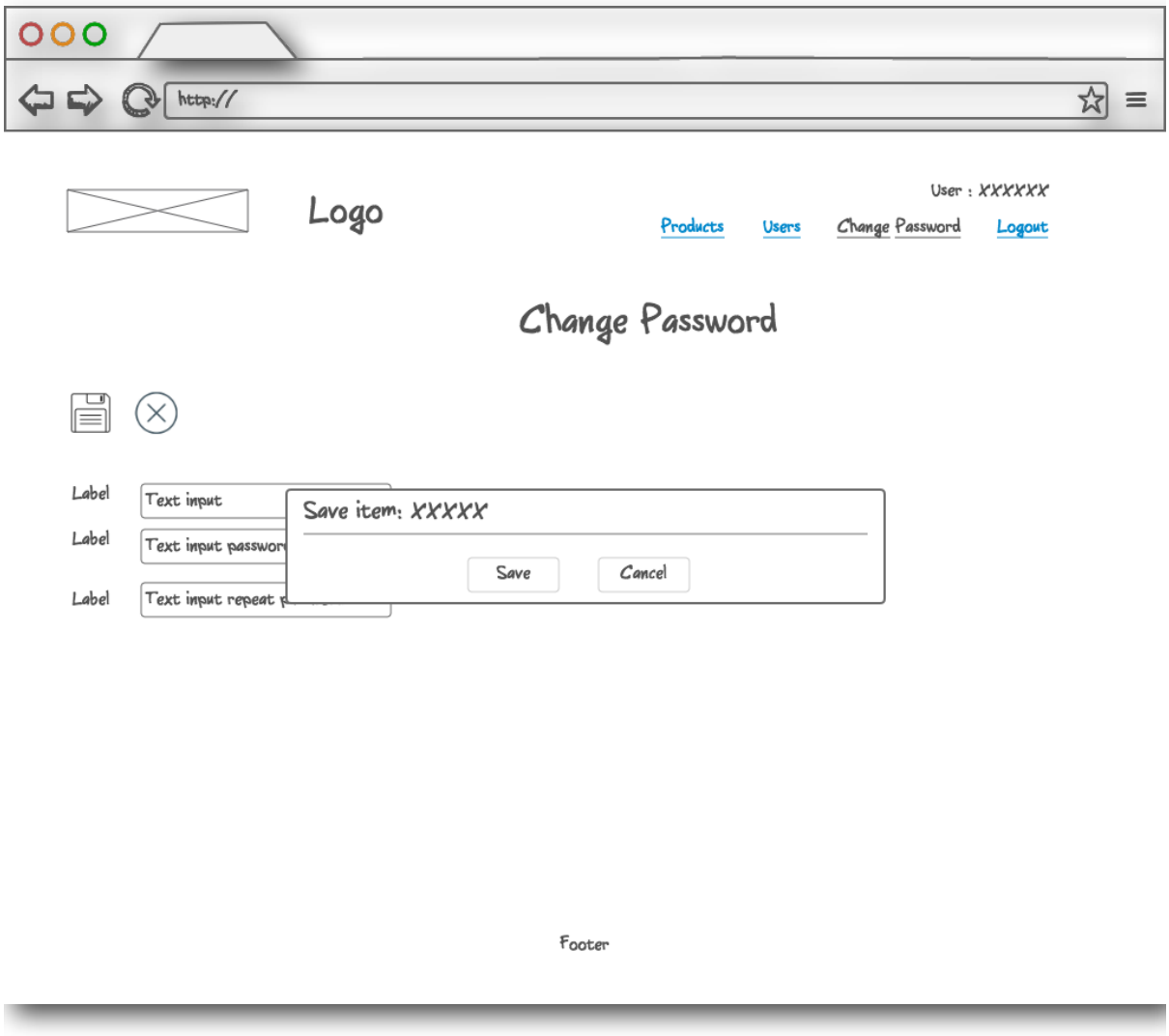

Figura 18: Wireframe pantalla de modificació de password amb finestra emergent per guardar la informació.

Quan cliques el botó gravar aniràs a la pantalla de cerca de productes. Si cliques el boto cancel·lar et mantindràs a la mateixa pàgina.

14. Perfils d’usuaris

Al ser una àrea privada hi haurà tres perfils d’usuaris, l’administrador del sistema, l’editor i l’observador. Tots aquests perfils d’usuaris podran realitzar diferents tasques per assolir l’objectiu principal.

L’administrador del sistema podrà gestionar les bases de dades SQL i NoSQL, podrà realitzar cerques / filtres, veure el detall de les dades, donar d’alta, modificar i donar de baixa articles. També podrà gestionar el usuaris del sistema, podrà donar-los d’alta, modificar-los i donar-los de baixa.

L’editor, podrà realitzar cerques / filtres, veure el detall de les dades, donar d’alta, modificar i donar de baixa articles.

L’observador només podrà visualitzar els articles, podrà fer cerques / filtres per localitzar els articles i visualitzar en detall l’article.

Tots els tres perfils podran canviar-se la seva contrasenya.

15. Usabilitat/UX

15.1 Usuaris i context d'ús

El mètode d'indignació utilitzat pel procés del DCU, és el de l'anàlisi competitiva, s'analitzaran productes similars que gestionen informació per veure les funcionalitats bàsiques d'una àrea privada per gestionar articles d'una llibreria (bookstore). Els productes similars que es volen analitzar són l'opencart (sistema de comerç electrònic opensource), plantilles d'administració com per exemple http://themeforest.net/item/materialism-angular-bootstrap-admin-template/11322821

Segons el perfil d'usuari haurà de poder fer cerques / filtratges dels productes de la llibreria, haurà de permetre crear articles, modificar-los i eliminar-los. També ha de permetre gestionar els usuaris. El menú web dependrà la seva informació del perfil d'usuari que sigui.

15.2 Disseny conceptual

A continuació es descriuran diferents escenaris d'ús:

Nom: Escenari 1.
Perfil usuari: Administrador, Editor, Observador.
Context: L'usuari necessita entrar dins de l'aplicació.
Objectiu: Entrar dins de l'aplicació.
Descripció: L'usuari haurà d'entrar dins un navegador web i entrar la url de l'aplicació, un cop carregada la pàgina, haurà d'entrar les seves credencials i el codi captcha per identificar-se i poder entrar dins de l'aplicació.

Nom: Escenari 2.
Perfil usuari: Administrador, Editor, Observador.
Context: L'usuari necessita saber la informació d'un producte.
Objectiu: Cercar un producte en concret per saber la seva informació detallada.
Descripció:Primer haurà de fer ús de l'escenari 1, L'usuari haurà d'entrar dins a la gestió dels productes i dins la pantalla de cerca, haurà d'utilitzar els filtres disponibles per localitzar-lo. Els filtres que te disponibles són títol, autor, ISBN, data de publicació, rating, idioma principal, idioma secundari, idioma original, tag i col·lecció. També haurà de seleccionar la base de dades en que vol treballar.

Nom: Escenari 3.
Perfil usuari: Administrador, Editor.
Context: L'usuari necessita editar la informació d'un producte.
Objectiu: Modificar un producte en concret.
Descripció: Primer haurà de fer ús des l'escenaris 1 i 2, i un cop localitzat el producte, haurà de clicar al botó d'edició d'aquell producte, per poder entrar dins la pantalla per modificar les seves

dades. Un cop dins de la pantalla d'edició, podrà modificar les dades que necessita i llavors clicar el boto per guardar la modificació.

Nom: Escenari 4.
Perfil usuari: Administrador, Editor.
Context: L'usuari necessita eliminar un producte.
Objectiu: Eliminar un producte en concret.
Descripció: Primer haurà de fer ús des l'escenaris 1 i 2, i un cop localitzat el producte, haurà de clicar al botó per eliminar el producte, i un cop apareix la pantalla de confirmació llavors haurà d'acceptar l'eliminació del producte.

Nom: Escenari 5.
Perfil usuari: Administrador, Editor.
Context: L'usuari necessita crear un producte.
Objectiu: Crear un nou producte.
Descripció: Primer haurà de fer ús des l'escenaris 1, un cop a dins a la gestió dels productes, haurà de clicar sobre el boto d'alta, per crear un nou producte. Un cop hagi entrat totes les dades del producte, haurà de clicar el boto per guardar la informació.

Nom: Escenari 6.
Perfil usuari: Administrador, Editor, Observador.
Context: L'usuari necessita canviar la seva contrasenya.
Objectiu: Modificar la seva contrasenya.
Descripció: Primer haurà de fer ús de l'escenari 1, llavors haurà de clicar sobre l'enllaç modificar contrasenya, i allà canviar la contrasenya. La contrasenya haurà de ser segura, comprovarà que hi ha caràcters de lletres minúscules, majúscules, números i caràcters especials. També comprovarà que les tres últimes vegades hagi escrit la contrasenya diferent.

16. Seguretat

En el desenvolupament del projecte s'han afrontat varis riscos de seguretat, que s'explicaran a continuació.

Al tractar-se d'un entorn privat, el seu accés serà mitjançant un usuari (email) i una clau d'accés. Per augmentar la seguretat en el formulari d'accés hi haurà un captcha, així s'habitarà accessos massius de robots. La sessió de l'usuari expira al cap de 24 hores.

Política de contrasenyes:

Com a política de seguretat, els usuaris estan obligat a canviar la contrasenya cada 6 mesos, un cop passat aquest 6 mesos, l'usuari no podrà accedir a l'aplicació fins que es canvii la contrasenya.
La longitud de la contrasenya ha de ser mínim de 8 caràcters, ha de contenir com a mínim una lletra amb minúscules, una amb majúscules, un número i un caràcter especial.
No es pot repetir cap de les 3 contrasenyes anteriors que s'hagin usat.
Les contrasenyes estan encriptades amb una llavor per augmentar-ne la seguretat.

Política de seguretat:

Al tractar-se d'una aplicació web, s'ha de tenir en compte varies vulnerabilitats de les pàgines web [28]:

1. Cross Site Scripting (XSS), és un tipus d'injecció HTML que permet a una persona mal intensionada executar codi en el navegador de la víctima i pot realitzar atacs de phising, robar sessions, etc. El framework que he utilitzat, el Fat Free Framework, per defecte amb el seu sistema de gestió de plantilles, escapa totes les variables utilitzades, per protegir-se del atacs XSS i d'injecció de codi.
2. Injecció de codi, es basa en l'enviament de dades per part del client a l'aplicació amb contingut maliciós i utilitzats directament, per exemple en sentencies SQL, atacs de SQL Injection, HTML Injection, XPath, XML. S'ha utilitzat un data mapper (ORM - Object-relational mapping), del Fat Free Framework, que protegeix la injecció SQL.
3. Execució maliciosa de codi, aquesta vulnerabilitat es basa en l'us d'una variable procedent del client per incloure el nostre codi en un fitxer determinat, per exemple pel paràmetre $_GET. PER corregir aquest punt s'ha realitzat una configuració de rutes, només accepta rutes que hi ha dins del fitxer de rutes.
4. Mostratge d'errors, quan l'aplicació estigui en mode producció, es desactivaran els errors, warnings, notices, perquè donen a l'usuari final informació de possibles errors del sistema i es poden trobar vulnerabilitats.

Seguretat en les bases de dades:

En la base de dades MongoDB quan la instal·les per defecte ve configurada sense contrasenya per accedir-hi. S'han estat fent proves seguint el tutorial de web de mongodb [32]. S'han creat l'usuari admin, s'han realitzat les proves d'accés. I llavors per donar la facilitat d'accés a la base de dades per poder visualitzar-la des de qualsevol IP, no s'ha restringit el seu accés.

17. Tests

Per poder testejar l'aplicació web que s'estar creant, realitzarem uns tests segons els diferents perfils d'usuaris que s'han descrit anteriorment.

Test d'usabilitat:

A continuació es plantegen diferents accions que els usuaris hauran de dur a terme per poder analitzar les dificultats i millorar els punts més negatius:

A continuació es mostren tasques que hauran de realitzar els usuaris Administradors, Editors i Observadors, per tal de saber si la cerca de productes es intuïtiva i senzilla d'utilitzar:

Testos a realitzar per les dues bases de dades: MongoDB i SQL

1. Cercar els productes que tinguin per autora l'Elizabeth Levy
2. Anomenar el títol del llibre amb l'ISBN "8472029689"
3. Anomenar l'autor del llibre "1776 Census of Maryland"
4. Anomenar els llibres publicats el 1967
5. Anomenar els llibres que l'idioma original sigui l'espanyol (spa)
6. Anomenar els tags del llibre amb l'ISBN "9781596986046"
7. Anomenar quants llibres té la col·lecció "Your Library"
8. Anomena els llibres que tinguin el tag "fiction"
9. Anomena els llibres que tinguin els tags "fiction" i "kids" com a mínim.
10. Anomena quants llibres tenen només el tag "to-read"

A continuació es mostren tasques que hauran de realitzar els usuaris Administradors, Editors per tal de saber si es poden gestionar els productes fàcilment:

1. Modificar les dades del producte amb l'ISBN "07607734672" de la base de dades MongoDB
2. Eliminar el producte amb el títol "1 is One" de la base de dades SQL
3. Crear un nou producte amb les dades d'un títol d'un llibre que l'hi ha agradat per la base de dades MongoDB.

A continuació es mostren tasques que hauran de realitzar l'usuari administrador:

1. Crear nou usuari
2. Modificar el perfil de l'usuari creat, si l'ha creat com a editor posar-lo com a observador, si nó viceversa.
3. Eliminar l'usuari creat.

A continuació es mostren tasques per a realitzar per qualsevol perfil d'usuari:

1. Canviar la seva contrasenya d'accés.
2. Fer un logout (Sortir de l'aplicació)
3. Entrar al sistema amb la nova contrasenya.

Tests de seguretat:

Es validarà que les accions realitzades als testos d'usabilitat s'hagin realitzat correctament a la base de dades corresponent.

18. Projecció a futur

A continuació es mostra una serie de millores que es podrien fer en pròximes versions:

Relacionat amb les dues base de dades el MySQL i el MongoDB, es podrien configurar amb Sharding, consistiria en tenir varis servidors amb les bases de dades i s'utilitzaria aquesta tècnica per gestionar la carrega dels servidors, es distribuirien les dades entre els diferents shards (conjunts de servidors que emmagatzemen part de les dades), per repartir les tasques de consulta i insercions entre els servidors.

Es podria optimitzar la importació amb SQL, perquè amb importacions llargues que es redueixi la pèrdua de dades importades i millorar el temps d'insercions. S'hauria d'analitzar el framework utilitzat per intentar de corregir els errors produïts.

Es podria implementar el fet de quan s'insereixen nous productes en MongoDB, afegir a les respectives col·leccions els nous tags, col·leccions i autors.

Es podria canviar la política de les contrasenyes i augmentar una mica més la seguretat de l'aplicació, es tractaria de bloquejar l'usuari si erra tres cops seguits la contrasenya d'accés. Llavors s'hauria de posar en contacte amb l'administració del sistema per restablir la contrasenya.

Es podria millorar la paginació de la pantalla de cerca dels productes, ja que si es retornen molts resultats, pot donar errors. I es podria implementar una paginació des del servidor i no com estar actualment des del client.

19. Conclusió/-ns

19.1 Conclusions Personals

Personalment, estic satisfet de la realització d'aquest treball, ja que l'he trobat molt interessant. Perquè al realitzar-lo, he pogut endinsar-me una mica en el MongoDB i conèixer una mica com es realitzen les consultes, les actualitzacions, l'eliminació i l'inserció de registres, he pogut conèixer una mica en detall les eines que he escollit pel seu desenvolupament, que m'han facilitat la realització del treball. M'ha agradat molt el fet de construir un CRUD per dues bases de dades diferents al mateix temps, ja que penso que m'ha ajudat a conèixer les estructures de les diferents bases de dades que s'han escollit i saber com tractar les dades per unificar-les per poder-les mostrar a l'usuari.

Treballar amb el Fat Free Framework, m'ha agradat, l'he trobat interessant perquè facilita el seu ús de cara al programador, les APIs les he trobat ben estructurades per trobar la informació, l'he trobat pràctic per desenvolupar el projecte, degut al fet de tenir un ORM (object-relational mapping) de base de dades, m'ha facilitat la comunicació amb les bases de dades, i les diferencies entre les dues bases de dades, amb aquesta eina, gairebé bé no les he notat.

Utilitzar el Bootstrap, m'ha ajudat amb la construcció de l'html i amb la seva visualització responsive, m'he basat amb les estructures bàsiques de la seva pàgina oficial [30].

L'Angularjs m'ha ajudat amb la interacció amb les pantalles i la base de dades, també alhora de mostrar la informació en la pantalla de cerca dels productes utilitzant una llibreria de angular smart-table [31] ja que m'ha facilitat la realització del filtratge en les columnes, la paginació a nivell del client.

19.2 Procés de treball

Per poder conèixer les fortaleses i debilitats de les bases de dades MySQL i MongoDB he realitzat diferents proves, comentar que totes s'han executat des de el robomongo per el MongoDB i per consola amb la comanda "mysql" pel MySQL, excepte les d'inserció que s'han realitzat mitjançant una classe que s'ha desenvolupat (importacio.php).

19.2.1 Insercions

Per poder valorar el comportament en les insercions de registres a les diferents bases de dades , s'ha realitzat una importació de 8 GB d'arxius JSONS per cada una de les dues bases de dades. Aquesta informació s'ha processat en els servidors comentats en l'apartat "7.2 Servidor".

Les importacions es realitzen mitjançant una classe que s'ha desenvolupat (importacio.php) on llegeix els fitxers, els processa i els insereix a la base de dades corresponents. El programa registra el temps d'inserció per cada fitxer.
En la importació a la base de dades de MongoDB, s'han processat els fitxers en execucions de 500 fitxers cada procés, per poder realitzar aquesta tasca s'ha ampliat el temps d'execució del servidor a 7.200 segons. Cada proces aproximat ha tingut una durada d'un temps aproximat de 1'5 hores.
Per la importació de les dades a la base de dades SQL, s'han realitzat diferents testos, es va començar a importar paquets de 50 fitxers, però el servidor no ho suportava, donava error de memòria, la memòria del servidor està configurada a 2.000 Mb, el limit del servidor, i fins i tot així no ho suportava, i vaig decidir a crear un cron, que executa el programa d'importació, i processar 1 fitxer cada 3 minuts, que són 20 fitxers cada hora, un cop importats vora 1.000 fitxers, l'execució de l'importació del fitxer durava més de 3 minuts i s'ha incrementat el cron a 7 minuts per no col·lapsar tant el servidor.

Un cop importats tots els registres, s'ha arribat als següents resultats: en MongoDB hi havia 12.519.541 de registres i en MySQL hi havia 11.195.842. Aquesta diferència es deguda a errors inesperats en el procés d'importació derivats per manca de memòria.

La diferència d'importar les dades amb les diferents estructures de la base de dades, és bastant important. Amb el MongoDB ha anat més ràpid, ja que l'estructura que s'ha pensat en MySQL hi ha taules relacionades de Autors, Tags i Col·leccions que tenen registres únics per tant ha de verificar que no existeix el registre abans d'inserir-lo. A canvi llavors ja hi haurà les dades apunt per cercar els camps d'auto-completar de les pantalles.

S'ha de tenir en compte que en la base de dades MongoDB, un cop importades les dades, s'ha hagut de realitzar 3 MapReduce, per poder fer consultar els autors, col·leccions i tags diferents pels llocs del web on hi ha un text d'auto-completar , la realització del MapReduce incrementa el temps de manipulació de les dades. Aquesta manipulació es necessària cada cop que s'inserissin nous registres d'autors, col·leccions i tags.

Penso que hi ha 3 estratègies a seguir pel tractament dels autors, col·leccions i tags en MongoDB. La primera tractaria d'importar tota la informació i llavors realizar un MapReduce i cada vegada que s'insereixin nous productes, tornar a generar un MapReduce per cadascú d'ells. La segona seria importar tota la informació i llavors realitzar un MapReduce i cada vegada que s'insereixin nous productes, verificar a cada col·lecció generada pel MapReduce si el registre existeix en ella i si no existeix s'hauria d'inserir. I la tercera seria en el moment de fer la importació tenir les col·leccions d'autors , tags i col·leccions i anar verificant si existeix cada element en les respectives col·leccions. Aquesta última l'he desestimat, ja que penso que augmentaria significativament el resultat final de les importacions. El que s'ha implementat ha estat la primera part de realitzar el MapReduce per cada variable.

A continuació mostro gràfiques extretes de fitxers de log que s'han extret amb les importacions realitzades d'un conjunt de 100 arxius escollits a l'atzar. Aquestes proves les he realitzat tenint en compte varies casuístiques:

- Comprovant si existeix el registre del camp book_id amb el camp book_id sent index i sense ser index.
- Sense comprovar si existeix el registre del camp book_id amb el camp book_id sent index i sense ser index.

La següent gràfica és d'insercions de grup de 5 fitxers a la base de dades MySQL amb la base de dades sense registres, s'han realitzat 2 cops per cada execució i s'ha realitzat la mitjana, un cop executats tots els fitxers s'ha anat buidant la base de dades per poder seguir amb següents proves. En les execucions comparant book_id (book_id sense index) només s'han realitzat 10 execucions ja que ja s'observava uns valors molt grans).

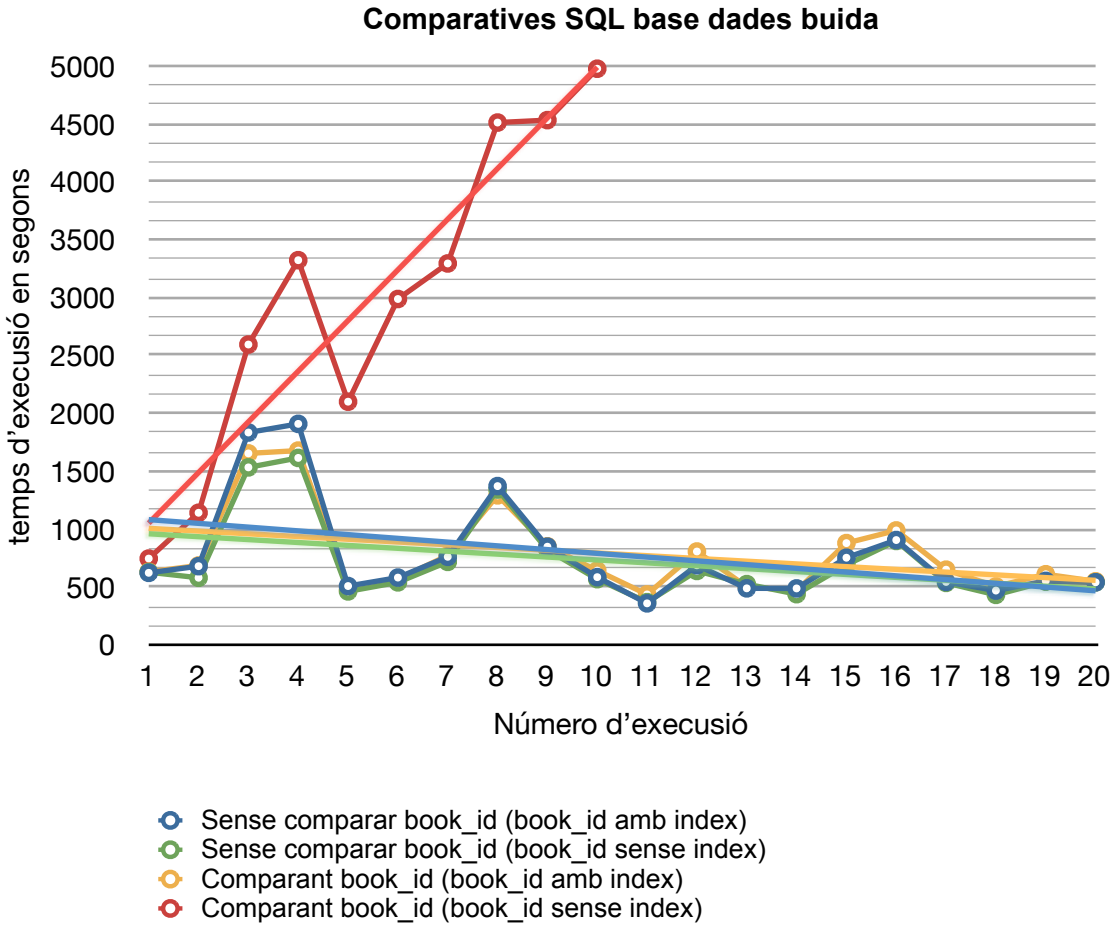

Figura 19: Gràfica comparativa SQL inserció registres.

Es pot observar en la gràfica anterior que accepta el “comparant book_id (book_id sense index)” tots segueixen un temps similar alhora de fer les insercions. El fet de comprar el book_id , sense que tingui index penalitza ja que en el pitjor dels casos ha de recórrer tota la taula de la base de dades.

La següent gràfica és d’insercions de grup de 5 fitxers a la base de dades MongoDB amb la base de dades sense registres, s’han realitzat 2 cops per cada execució i s’ha realitzat la mitjana, un cop executats tots els fitxers s’ha anat buidant la base de dades. En les execucions comparant book_id (book_id sense index) només s’han realitzat 10 execucions ja que ja s’observava uns valors molt grans).

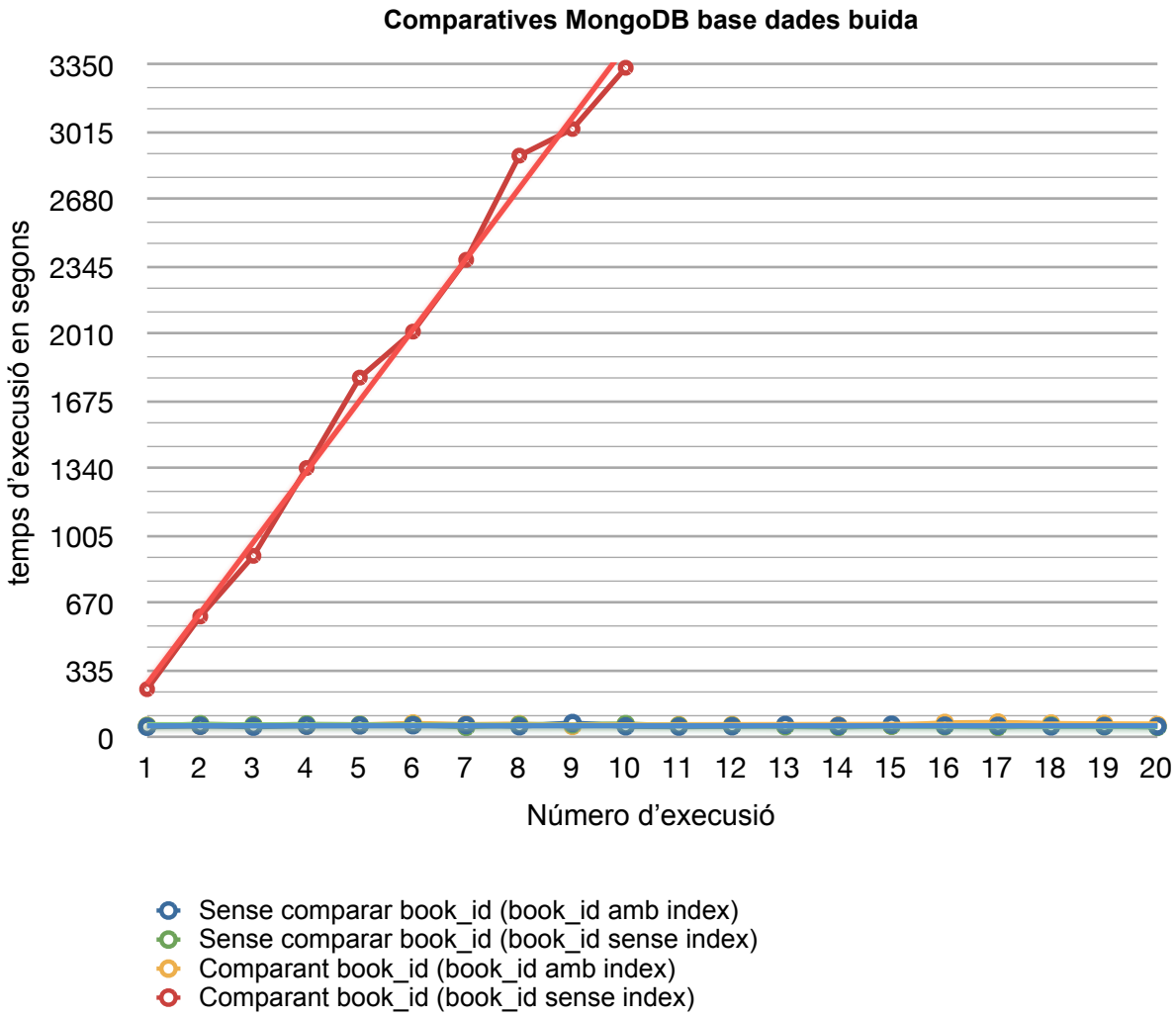

Figura 20: Gràfica comparativa MongoDB inserció registres.

En l’anterior gràfica es pot observar que el “comprovant book_id (book_id sense index) es diferencia molt amb els altres. Creix d’una manera molt ràpida el temps d’insersió. A canvi la resta els temps són similars.

La següent gràfica és d'insercions d'un grup de 100 fitxers a la base de dades MySQL i MongoDB, es pot observar la relació de número de registres inserits amb el temps d'execució en segons.

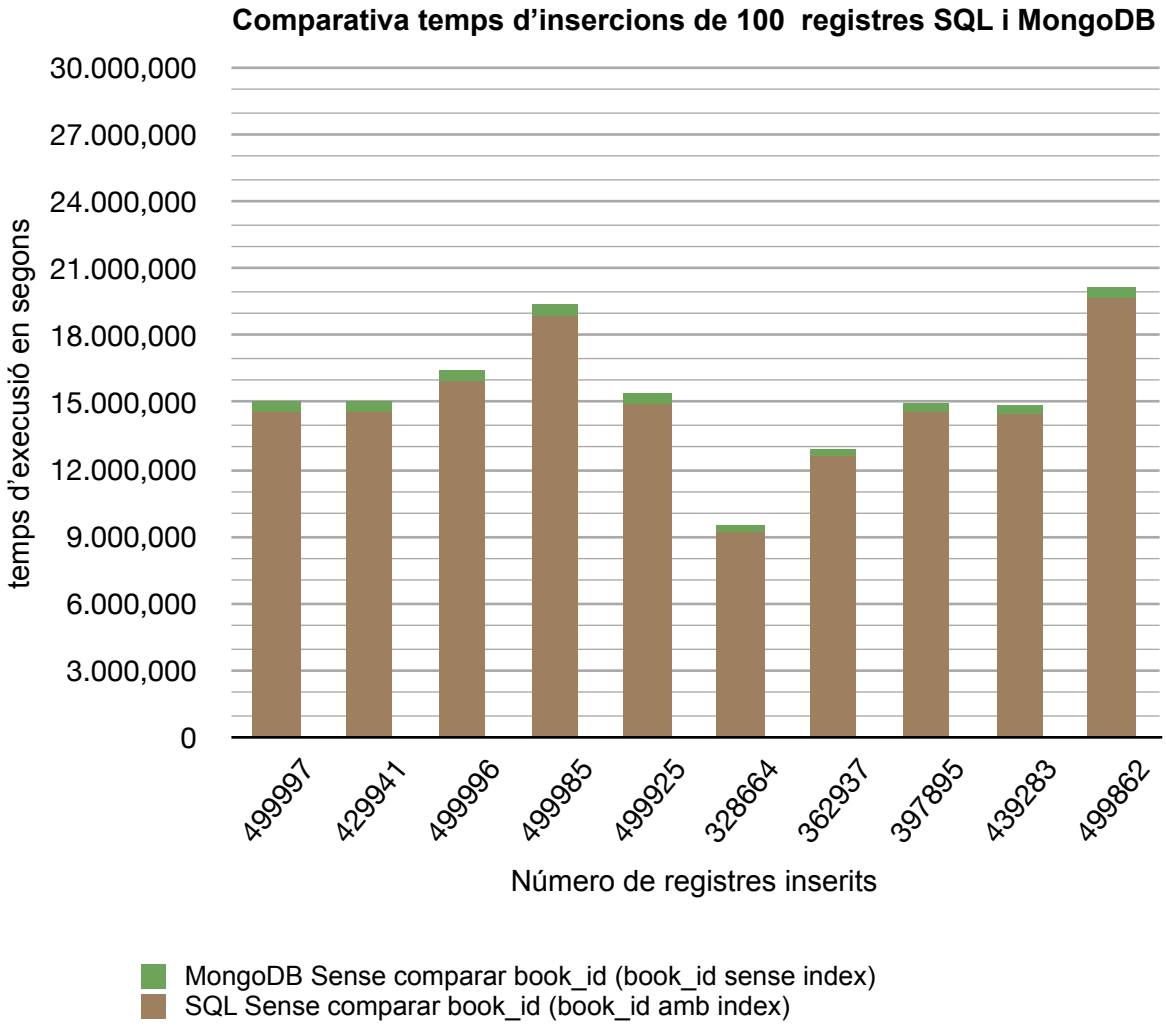

Figura 21: Gràfica comparativa SQL i MongoDB

En la gràfica anterior es pot observar que el temps d'inserció es molt més ràpid el MongoDB que l'SQL. Per cada base de dades el temps es similar, es pot observar que segueix un patró segons el nombre de registres inserits.

La següent gràfica mostra l'espai ocupat segons les insercions de 100 fitxers a la base de dades MySQL i MongoDB. Es pot veure la relació del registres que hi ha en la base de dades i el seu espai a disc.

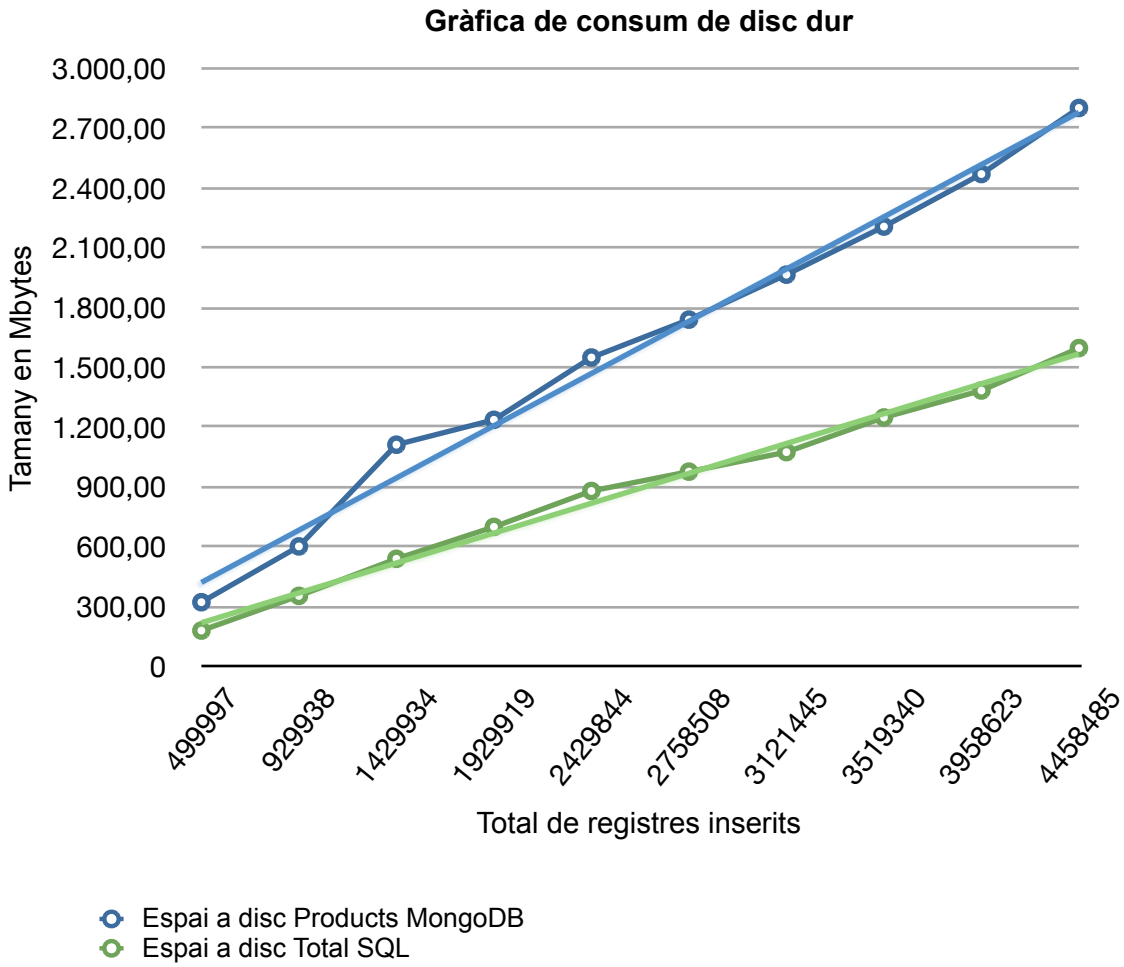

Figura 22: Gràfica comparativa d'ocupació d'espai a disc de SQL i MongoDB.

Observant la gràfica anterior es pot comprovar que l'espai a disc consumit pel MongoDB creix molt ràpidament comparat amb l'SQL. En el punt final, quasi bé arriba a duplicar el seu espai a disc. Abans d'arribar al milió de registres podríem dir que no hi ha diferència, però llavors si.

19.2.2 Consultes

A continuació es mostren diferents mostres de temps de consultes realitzades sobre les bases de dades MongoDB i MySQL:

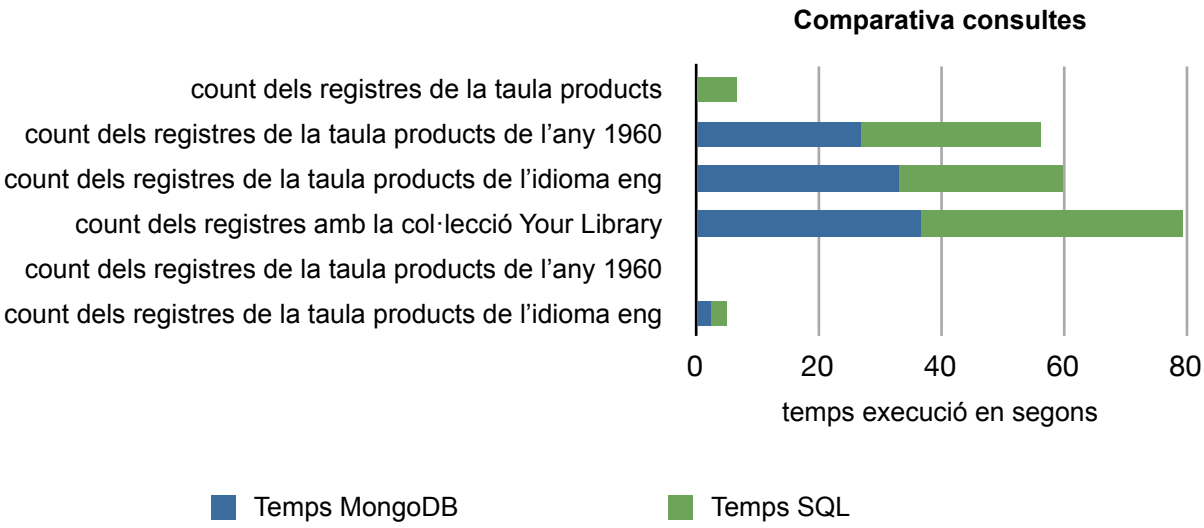

Figura 23: Gràfica comparativa del temps d'execució alhora de cercar SQL i MongoDB

Comentar que les quatre primeres files són consultes sense index, i les dues darreres amb index. Com es pot veure a la gràfica anterior en pràcticament en tots els casos el MongoDB ha estat més ràpid en diferència.

A continuació es mostra com s'han realitzat les consultes

MongoDB:

- db.getCollection('Products').find({}).count()
- db.getCollection('Products').find({'publication_date':'1960'}).count()
- db.getCollection('Products').find({'language_main':'eng'}).count()
- db.getCollection('Products').find({'collections.collection': {"$in":['Your library'], "$exists":true}}).count()
- db.getCollection('Products').find({'publication_date':'1960'}).count()
- db.getCollection('Products').find({'language_main':'eng'}).count()

MySQL:

- select count(*) from products;
- select count(*) from products where publication_date = '1960';
- select count(*) from products where language_main = 'eng';
- select count(product_id) from products_collections pc where pc.`collection_id` = '1';
- select count(*) from products where publication_date = '1960';
- select count(*) from products where language_main = 'eng';

19.2.3 Creació d'index

A continuació es mostren diferents mostres de temps de creació d'index, quan la base de dades ha estat plena.

S'han realitzat tres proves de creació d'index, crear un index dels camps: publication_date, language_main i ISBN:

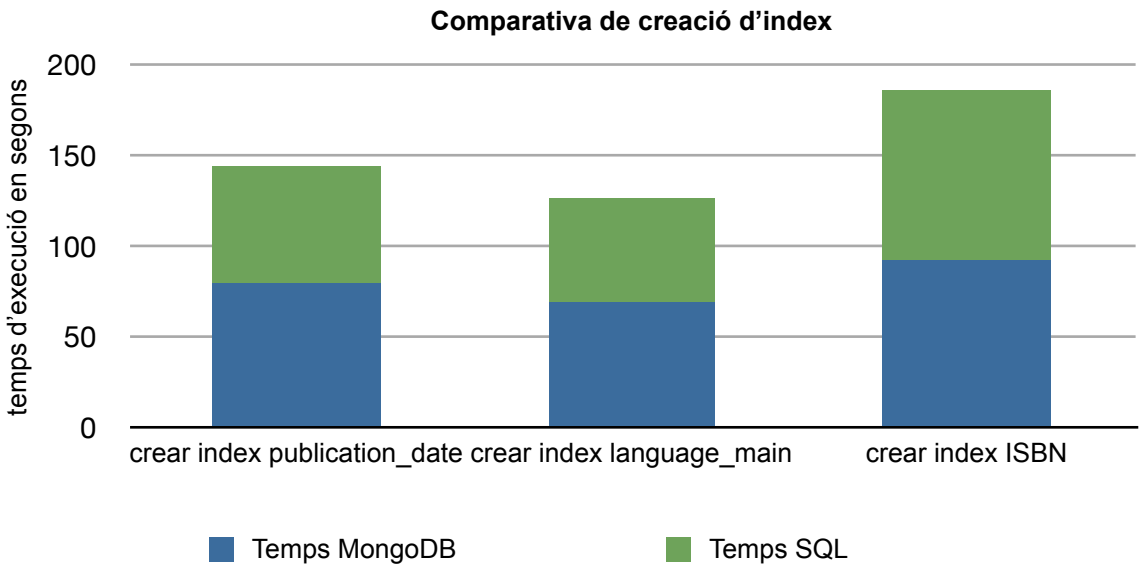

Figura 24: Gràfica comparativa del temps d'execució alhora de crear index SQL i MongoDB.

Com es pot veure en la gràfica anterior no hi ha gaire diferencia, el MySQL ha estat més ràpid en les dues primeres columnes i el mongoDB en la tercera ha estat més ràpid però en tots els casos per poca diferència.

A continuació es mostra com s'han creat els diferents index:

MongoDB:

- db.Products.ensureIndex({"publication_date":1})
- db.Products.ensureIndex({"language_main":1})
- db.Products.ensureIndex({"ISBN":1})

MySQL

- ALTER TABLE `products` ADD INDEX `publication_date` (`publication_date`);
- ALTER TABLE `products` ADD INDEX `language_main ` (`language_main `);
- ALTER TABLE `products` ADD INDEX `ISBN` (`ISBN`);

19.2.4 Modificacions

A continuació es mostren diferents mostres de temps d'actualitzacions de registres:

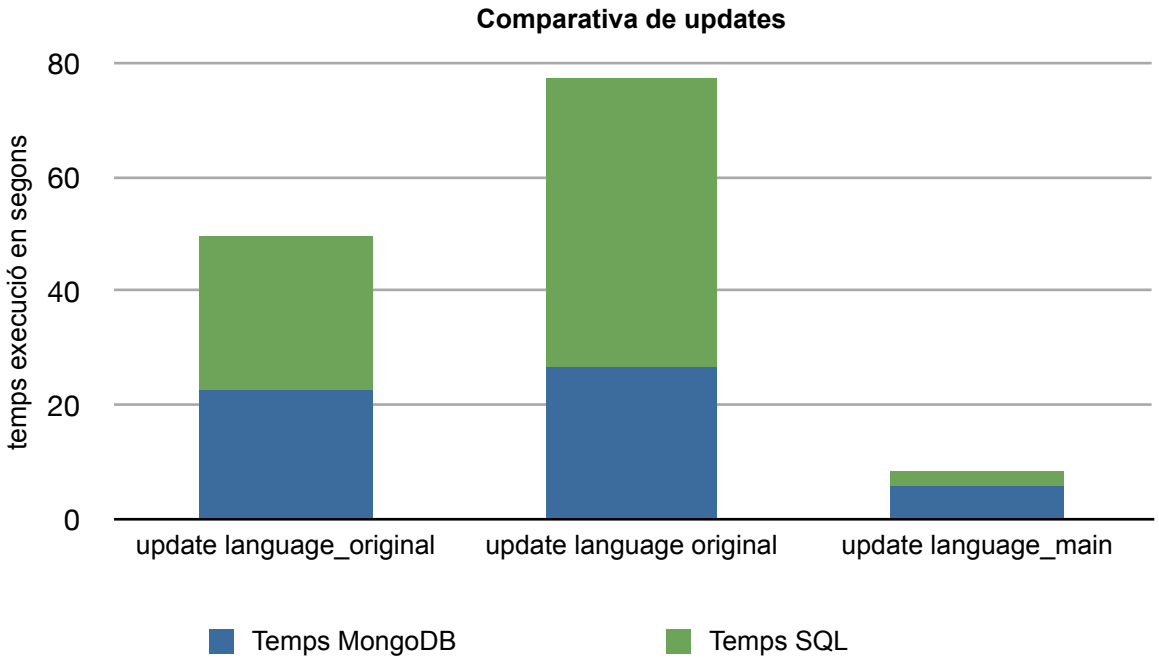

Figura 25: Gràfica comparativa del temps d'execució alhora de fer updates SQL i MongoDB.

Cal comentar que language_original no estar indexada, i language_main si que té index.
Com es pot veure a la gràfica anterior a les dues primeres columnes on no hi ha index, el MongoDB ha anat molt més ràpid. En canvi quan el camp en que es modifica segon un camp indexat el MySQL ha anat més ràpid

A continuació es mostra les instruccions amb que s'han realitzat les actualitzacions:

MongoDB:

- db.Products.update({'language_original':'ita'},{$set:{'language_original':'tia'}},{multi:true})
- db.Products.update({'language_original':'tia'},{$set:{'language_original':'ita'}},{multi:true})
- db.Products.update({'language_main':'ita'},{$set:{'language_original':'tia'}},{multi:true})

MySQL:

- update products set language_original='tai' where language_original='ita';
- update products set language_original='ita' where language_original='tai';
- update products set language_original='tia' where language_main='ita';

19.2.5 Eliminacions

A continuació es mostren diferents mostres de temps d'eliminacions de registres:

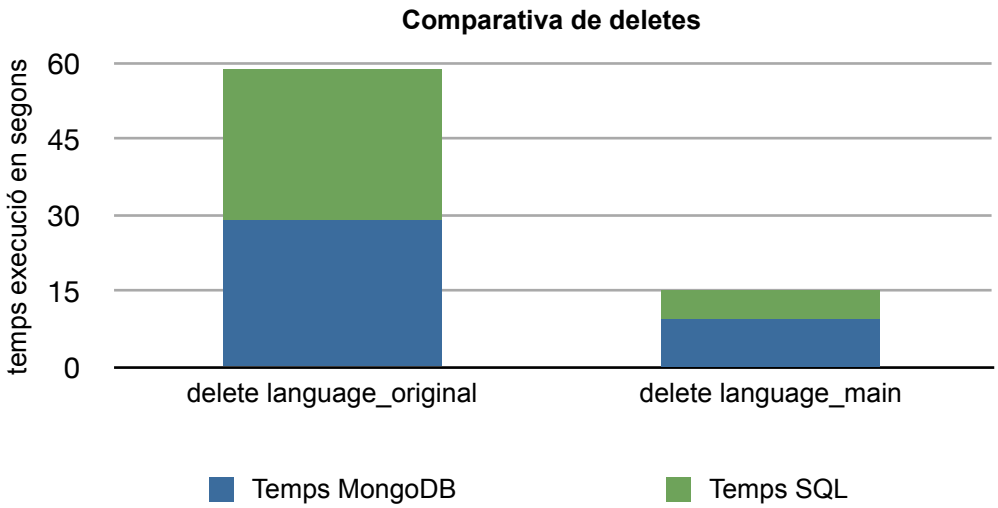

Figura 26: Gràfica comparativa del temps d'execució alhora de fer deletes SQL i MongoDB.

Comentar que language_original no té index en canvi language_main sí.
Com es pot observar a la gràfica anterior que quan s'elimina tenint en compte un camp que no té index, el Mongo ha anat més ràpid, per molt poc. A canvi amb l'index, el MySQL ha anat molt més ràpid.

A continuació es mostra les instruccions amb que s'han realitzat les eliminacions:

MongoDB:

- db.Products.remove({'language_original':'ita'})
- db.Products.remove({'language_main':'ita'})

MySQL:

- delete from products where language_original = 'ita';
- delete from products where language_main = 'ita';

19.3 Resultats obtinguts

En les diferents bases de dades, alhora d'emmagatzemar la informació hi ha hagut diferents plantejaments per tractar la informació, en l'apartat 7.2 Bases de dades, es pot veure les estructures que s'han utilitzat en les diferents bases de dades. Els punts que destaquen més són:

L'estructura de la col·lecció en el MongoDB és dinàmica, els camps buits no s'han importat. A canvi a SQL no ho permet i el camp es queda buit, sense valor. En el MongoDB, les col·leccions, tags i autors són arrays dins el registre del producte, en canvi en SQL, hi ha taules a part per les col·leccions, tags i autors, i llavors taules que relacionen el producte amb tags i amb col·leccions. Això comportaria que si es decidís que la informació fos unificada, si es volgués actualitzar la descripció d'algun d'ells, en MySQL, només s'hauria de canviar un registre, en canvi en MongoDB, depenent de si els camps fossin index, s'haurien de recorrer tots els registres de la col·lecció i anar-ho actualitzant. Alhora d'eliminar un registre dins d'un array en MongoDB, s'hauria de recorrer tots els documents per anar eliminar el registre, a canvi en MySQL, s'hauria d'eliminar el registre de la taula corresponent i llavors els registres de la taula relacionada que contenen la clau del registre eliminat.

Com s'ha pogut veure en les proves realitzades mostrades en les gràfiques anteriors, el MongoDB en el punt en que destaca més és en l'inserció de les dades, penso que es degut a l'estructura de les diferents bases de dades i el fet que en el MySQL es comparen els autors, col·leccions i tags, i això fa allargar el temps. A Canvi en el MongoDB no hi ha cap comprovació i insereix directament.

Cal destacar l'espai a disc consumit amb el MongoDB, opino que es degut que a cada document es guarda tota l'estructura, ja que cada document pot ser diferent.

En les consultes s'ha pogut veure que quan no hi ha index, el MongoDb ha estat més ràpid, a canvi quan hi ha index, les consultes són similars.

En la creació d'index es pot considerar que s'han comportat de la mateixa manera.

En les actualitzacions i les eliminacions s'ha pogut veure que sense index el MongoDB era més ràpid i amb index el superava el MySQL.

Hi ha hagut un punt que s'ha trobat, que no s'esperava, i que s'ha hagut d'implementar, ha estat alhora d'oferir usabilitat a l'usuari en l'apartat d'omplir els autors, tags i les col·leccions, i que l'usuari pugui saber quins registres diferents existeixen de cadascú d'ells a les bases de dades, s'ha optat de que aquests camps tinguessin la propietat d'auto-completar-se. Per poder-ho dur a terme, amb els

plantejament de les estructures de les bases de dades, s'ha hagut d'implementar diferents estratègies en les diferents bases de dades, que estan explicades en l'Annex 2, amb codi font inclòs.

Alhora de retornar els registres a la pantalla de cerca de productes, s'ha hagut de posar un limit de retorn de registres, ja que col·lapsava la memòria del servidor. S'ha posat un limit de 50.000 registres, per tant a les cerques que es retornen aquest nombre de registres, no es cert, estar limitat. Per tant quan hi ha una gran quantitat de dades, s'hauria de fer una paginació mitjançant la programació en el servidor i no en el client, ja que sinó l'aplicació pot donar un mal rendiment.

Annex 1. Lliurables del projecte

Dins la carpeta tfm hi ha els arxius de l'aplicació web, dins d'ella hi trobarem les carpetes app, jsons, lib, tmp, ui. A la carpeta app hi ha classes de php i arxius de configuració. A la carpeta jsons hi ha exemples d'arxius que s'han importat. A la carpeta lib, hi ha les llibreries del framework utilitzat. A la carpeta tmp, es generen arxius temporals d'execució de l'aplicació i arxius de log. A la carpeta ui, hi ha els html i les carpetes css i js on hi ha que s'han utilitzat en l'aplicació.

tfm:

composer.json	index.php

app:

admin.php	collection.php	import.php	tag.php
author.php	config.ini	product.php	user.php
cms.php	controller.php	routes.ini	

jsons:

lib:

CHANGELOG	basket.php	image.php	session.php	web
COPYING	bcrypt.php	log.php	smtp.php	web.php
audit.php	code.css	magic.php	template.php	
auth.php	db	markdown.php	test.php	
base.php	f3.php	matrix.php	utf.php	

tmp:

ui:

admin.htm	images	products_edit.htm	users_edit.htm
change_password.htm	js	products_list.htm	
css	layout.htm	user.htm	
fonts	login.htm	users_list.htm	

ui/css:

bootstrap.css	bootstrap-theme.css.map
bootstrap.css.map	bootstrap-theme.min.css
bootstrap.min.css	select.css
bootstrap-theme.css	style.css

ui/js:

angular-spinner.js	bootstrap.min.js	tfm.js
angular-spinner.min.js	npm.js	
bootstrap.js	select.js	

Cal destacar que l'arxiu app/config.ini es l'arxiu de configuració del sistema, on dins d'ell es configuren si es volen els logs, es configuren les configuracions de la base de dades, variables del sistema, variable globals del framework.

Dins la carpeta extres hi ha els arxius de log que he anat generant per la realització de les gràfiques que s'han mostrat en el treball, també hi ha la fulla de càlcul amb que he realitzat les gràfiques.

Annex 2. Codi font (extractes)

A continuació s'explica les diferents estratègies per dur a terme el camp autocomplete per millorar l'usabilitat de cara l'usuari:

En la base de dades SQL, s'ha realitzat de la següent manera:
Com que la informació a cercar, estar cadascuna d'elles (autors, tags i col·leccions) en taules diferents a la dels productes, s'ha procedit com l'exemple que mostro a continuació:

```
public function returnDifferentTags($filter)
{
    $arr_IDBooksTags = $this->dbSQL->exec("SELECT `tag` FROM tags  WHERE
`tag` LIKE ?",$filter);
    if(count($arr_IDBooksTags) >0)
    {
        $arr_Tags = array();
        foreach($arr_IDBooksTags as $keys)
        {
            $arr_Tags[] = $keys['tag'];
        }
        return $arr_Tags;
    }
    else{
        return false;
    }
}
```

Exemple codi de retornar els diferents tags.

En la base de dades MongoDB, s'ha realitzat de la següent manera:
Com que la informació a cercar, estar dins de la col·lecció Products, i els atributs autors, tags i col·leccions estan dins de la col·lecció com arrays, no es pot utilitzar directament la mateixa tècnica com l'SQL, ja que si vols cercar com a "distinct" i fer la cerca d'auto-completar, el seu retorn es erroni. I per això s'ha optat a realitzar un mapreduce per aconseguir tres col·leccions "tags_cercats", "collections_cercades" i "authors_cercats" i llavors poder operar de manera similar que l'SQL mitjançant el framework utilitzat.

MapReduce consta de dues fases, com diu el seu nom la fase Map i la Reduce. La fase Map duu a terme la recollida de totes les dades d'entrada, el seu filtratge i la seva classificació. Llavors es realitza la fase Reduce, que s'encarrega d'agrupar les parts i combinar-les segons les claus iguals per realitzar les operacions que calgui. [27]

A continuació es mostra el MapReduce per el Tags:

```
var m = function() {
        if(this.tags)
        {
                for(var i = 0;i< this.tags.length;i++)
                {
                        emit(this.tags[i],1);
                }
        }
};
var r = function(keySKU, countObjVals) {
        return 1;
};
printjson(db.Products.mapReduce(m, r, {out: {replace: "tags_cercats"}}));
```

A continuació es mostra el MapReduce per les Col·leccions:

```
var m = function() {
  if(this.collections)
  {
        for(var i = 0;i< this.collections.length;i++)
        {
                emit(this.collections[i].idCollection,
                {idCollection:this.collections[i].idCollection,
                collection:this.collections[i].collection});
        }
  }
};
var r = function(key, values) {
        result = {collection:values[0].collection};
        return result;
};
printjson(db.Products.mapReduce(m,r,{out: {replace:
"collections_cercades" } }));
```

A continuació es mostra el MapReduce pels Autors:

```
var m = function() {
                     if(this.author)
                     {
                        if(this.author.code != "")
                        {
                            emit(this.author.code,{code:this.author.code,
                                                   name:this.author.name});
                        }
                     }
                    };
var r = function(key, values) {
                        result = values[0];
                        return result;
                   };
printjson(db.Products.mapReduce(m, r, {out: {replace:
"authors_cercats" } } ) );
```

Un cop s'ha realitzat el MapReduce i s'han creat les diferents col·leccions llavors per aconseguir la informació per l'autocomplete es fa una cerca la base dades, a continuació en mostro un exemple:

```
$dbMongoCollectionTags=new DB\Mongo\Mapper($this->dbMongo,'tags_cercats');
return json_encode($dbMongoCollectionTags->distinct('_id',$filter));
```

Exemple codi de retornar els diferents tags.

A continuació adjunto un exemple de codi extret de la pàgina oficial del framework[29] per veure la diferenciació alhora de cercar amb el mapper a les diferents bases de dades:

SQL:

```
$frequentUsers=$user->find(array('visits>?',
3),array('order'=>'userID'));
```

MongoDB:

```
$frequentUsers=$user-
>find(array('visits'=>array('$gt'=>3 ) ),array('userID'=>1 ));
```

A continuació es mostra un exemple de com es prepara la informació per les diferents bases de dades:

```
public function findProducts($f3)
{
        $book = $f3->get('GET');
        $arr_search = array();
        if($book['ddbb'] == 'M') {

                if (isset($book['_id']) && strlen(trim($book['_id']))) {
                        $arr_search['_id'] = new \MongoId($book['_id']);
                }
                if (isset($book['title']) && strlen(trim($book['title'])))
                {
                        $arr_search['title'] = $book['title'];
                }
                if (isset($book['ISBN']) && strlen(trim($book['ISBN']))) {
                        $arr_search['ISBN'] = $book['ISBN'];
                }
                if (isset($book['author']) &&
                                        strlen(trim($book['author']))) {
                        $arr_search['author.name'] = new \MongoRegex('/' .
$book['author'] . '/');
                }
                if (isset($book['year']) && strlen(trim($book['year']))) {
                        $arr_search['publication_date'] = new
\MongoRegex('/' . $book['year'] . '/');
                }
                if (isset($book['language_p']) &&
strlen(trim($book['language_p']))) {
                        $arr_search['language_main'] = $book['language_p'];
                }
                if (isset($book['language_s']) &&
strlen(trim($book['language_s']))) {
                        $arr_search['language_secundary'] =
$book['language_s'];
                }
                if (isset($book['language_o']) &&
strlen(trim($book['language_o']))) {
                        $arr_search['language_original'] =
$book['language_o'];
                }
                if (isset($book['copies']) &&
strlen(trim($book['copies']))) {
                        $arr_search['copies'] = $book['copies'];
                }
                if (isset($book['rating']) &&
strlen(trim($book['rating']))) {
                        $arr_search['rating'] = $book['rating'];
                }
                if (isset($book['tags']) && is_array($book['tags']) &&
```

```
count($book['tags'])>0) {
                        $inTag = array();

                        foreach($book['tags'] as $key)
                        {
                                $inTag[]=$key;
                        }

                        $arr_search['tags'] = array('$in' => $inTag);
                }
                if (isset($book['collections']) &&
is_array($book['collections']) && count($book['collections'])>0) {
                        $inCollection = array();
                        foreach($book['collections'] as $key)
                        {
                                $inCollection[]=$key;
                        }

                        $arr_search['collections.collection'] = array('$in'
=> $inCollection);
                }

        }
        elseif($book['ddbb'] == 'S')
        {
                $cerca = "";
                $arr_search = array();
                if (isset($book['idProduct']) &&
strlen(trim($book['idProduct']))) {
                        $cerca .= "idProduct = ? AND ";
                        array_push($arr_search,$book['idProduct']);
                }
                if (isset($book['title']) && strlen(trim($book['title'])))
{
                        $cerca .= "title = ? AND ";
                        array_push($arr_search,$book['title']);
                }
                if (isset($book['ISBN']) && strlen(trim($book['ISBN']))) {
                        $cerca .= "ISBN = ? AND ";
                        array_push($arr_search,$book['ISBN']);
                }
                if (isset($book['year']) && strlen(trim($book['year']))) {
                        $cerca .= "publication_date LIKE ? AND ";
                        array_push($arr_search,$book['year']);
                }
                if (isset($book['language_p']) &&
strlen(trim($book['language_p']))) {
                        $cerca .= "language_main = ? AND ";
                        array_push($arr_search,$book['language_p']);
                }
                if (isset($book['language_s']) &&
strlen(trim($book['language_s']))) {
```

```
                        $cerca .= "language_secundary = ? AND ";
                        array_push($arr_search,$book['language_s']);
                }
                if (isset($book['language_o']) &&
strlen(trim($book['language_o']))) {
                        $cerca .= "language_original = ? AND ";
                        array_push($arr_search,$book['language_o']);
                }
                if (isset($book['copies']) &&
strlen(trim($book['copies']))) {
                        $cerca .= "copies = ? AND ";
                        array_push($arr_search,$book['copies']);
                }
                if (isset($book['rating']) &&
strlen(trim($book['rating']))) {
                        $cerca .= "language_rating = ? AND ";
                        array_push($arr_search,$book['rating']);
                }
                if (isset($book['author']) &&
strlen(trim($book['author']))) {

                        $arr_IDBooksAuthors = $this->dbSQL->exec("SELECT
idProduct FROM products p LEFT JOIN authors a ON a.`code` = p.`author`
WHERE a.`name` LIKE ?",$book['author']);
                        foreach($arr_IDBooksAuthors as $keys)
                        {
                                $arr_IDBooks[] = $keys['idProduct'];
                        }

                }
                if (isset($book['tags']) && is_array($book['tags']) &&
count($book['tags']) >0) {

                        $inTag = array();
                        foreach($book['tags'] as $key)
                        {
                                $inTag[]=$key;
                        }

                        $elsTags = implode("','",$inTag);
                        $elsTags = "'".$elsTags."'";


                        $arr_IDBooksTags = $this->dbSQL->exec("SELECT
product_id FROM products_tags p LEFT JOIN tags t ON t.`idTag` = p.`tag_id`
WHERE t.`tag` IN (".$elsTags.")");

                        foreach($arr_IDBooksTags as $keys)
                        {
                                $arr_IDBooks[] = $keys['product_id'];
                        }
```

```
                }
                if (isset($book['collections']) &&
is_array($book['collections']) && count($book['collections'])>0) {

                        $inCollection = array();
                        foreach($book['collections'] as $key)
                        {
                                $inCollection[]=$key;
                        }

                        $lesCollections = implode("','",$inCollection);
                        $lesCollections = "'".$lesCollections."'";

                        $arr_IDBooksCollections = $this->dbSQL->exec("SELECT
product_id FROM products_collections p LEFT JOIN collections c ON
c.`idCollection` = p.`collection_id` WHERE c.`collection` IN (".
$lesCollections.")");
                        foreach($arr_IDBooksCollections as $keys)
                        {
                                $arr_IDBooks[] = $keys['product_id'];
                        }
                }
                if(count($arr_IDBooks)>0)
                {
                        $llistatIds =
implode("','",array_unique($arr_IDBooks));
                        $cerca .= "idProduct IN ('".$llistatIds."') AND ";
                        //array_push($arr_search,stripslashes($llistatIds));
                }

                if(strlen($cerca))
                {
                        $cerca = substr($cerca,0,-4);
                        array_unshift($arr_search,$cerca);
                }
        }
        $product = new Product();

        if($book['ddbb'] == 'M')
        {
                $list = $product->findElementsMongo($arr_search,
$book['view']);
        }
        elseif($book['ddbb'] == 'S')
        {
                $list = $product->findElementsSQL($arr_search,
$book['view']);
        }

        echo json_encode($list);
        exit;
}
```

Annex 3. Llibreries/Codi extern utilitzat

Pel desenvolupament de l'aplicació s'han utilitzat varies llibreries de tercers, en l'apartat 8. Plataforma de desenvolupament, s'expliquen els recurs tecnològics.

- Bootstrap i Jquery:

El Bootstrap i el Jquery s'han utilitzat per la maquetació de l'aplicació, ja que faciliten eines per fer una maquetació responsive.

```
<!-- Bootstrap -->
<link href="css/bootstrap.min.css" rel="stylesheet">
<script src="js/bootstrap.min.js"></script>
<link href="css/bootstrap.min.css" rel="stylesheet">
<!--[if lt IE 9]>
<script src="https://oss.maxcdn.com/html5shiv/3.7.2/
html5shiv.min.js"></script>
<script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></
script>
<![endif]-->

<!-- Jquery -->
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/
jquery.min.js"></script>
```

- Angulajs:

S'ha utilitzat angularjs, per fer la interacció de l'usuari amb el sistema, interacció amb l'api per llavors mostrar la seva informació a l'usuari. S'han utilitzat varies eines realitzades amb angularjs: smart-table, spinner, passwordStrength i el select.

L'smart-table, m'ha ajudat alhora de mostrar la taula de cerca de productes i autors, perquè l'usuari pugui filtrar el resultat extret de la base de dades. Facilita el filtre i l'ordre de les columnes.

L'spinner, és el pinyonet que dona voltes, per alertar a l'usuari que el sistema està treballant, s'executa en operacions que poden durar varis segons, en consultes a les bases de dades.

PasswordStrength, està extret de http://plnkr.co/edit/oZhqny?p=preview

El select, s'utilitza per la realització dels selects múltiples de tags i col·leccions. També per fer el seu autocomplete.

```
<!-- Angularjs -->
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.7/angular.min.js"></script>
<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.2.18/angular-sanitize.js"></script>

<!-- smart-table -->
<script src="https://cdnjs.cloudflare.com/ajax/libs/angular-smart-table/2.1.4/smart-table.min.js"></script>

<!-- spinner -->
<script type="text/javascript" src="http://fgnass.github.io/spin.js/spin.min.js"></script>
<script src="js/angular-spinner.js"></script>
<!-- select -->
<script src="https://cdnjs.cloudflare.com/ajax/libs/angular-ui-select/0.12.0/select.js"></script>
<link rel="stylesheet" href="http://cdnjs.cloudflare.com/ajax/libs/select2/3.4.5/select2.css">
```

Annex 4. Captures de pantalla

Captures de pantalla tant del treball/servei/aplicació realitzat com del procés de treball. Aquest annex també es pot utilitzar per recopilar les captures mostrades en altres seccions, en mida més gran per a la seva millor visualització, o no ser necessari el seu ús pel tipus de treball realitzat.

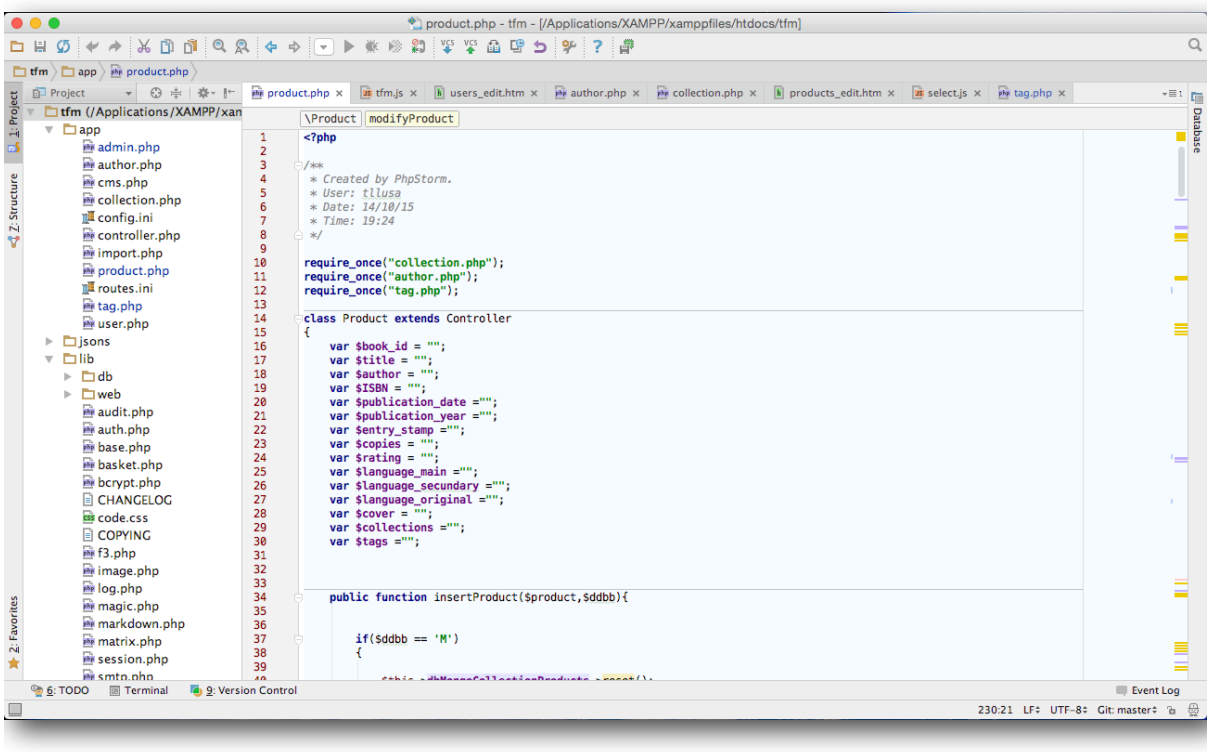

Figura 27: Captura de pantall del programa PHPStorm, que es el que s'ha utilitzat per fer el desenvolupament.

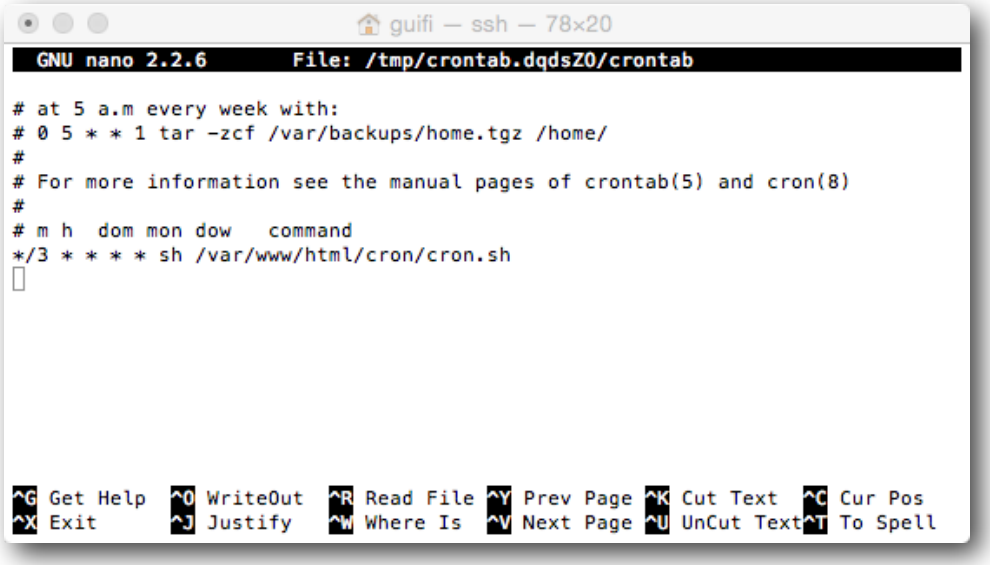

Figura 28: Captura de pantalla del cron utilitzat per carregar els productes a la base de dades MySQL

Un cop hem tingut les dades importades a les bases de dades MongoDB i MySQL, i havent esborrat l'els jsons per alliberar espai per la realització de proves a posteriori, entre el sistema operatiu i les dades s'ha ocupat un 92% de disc:

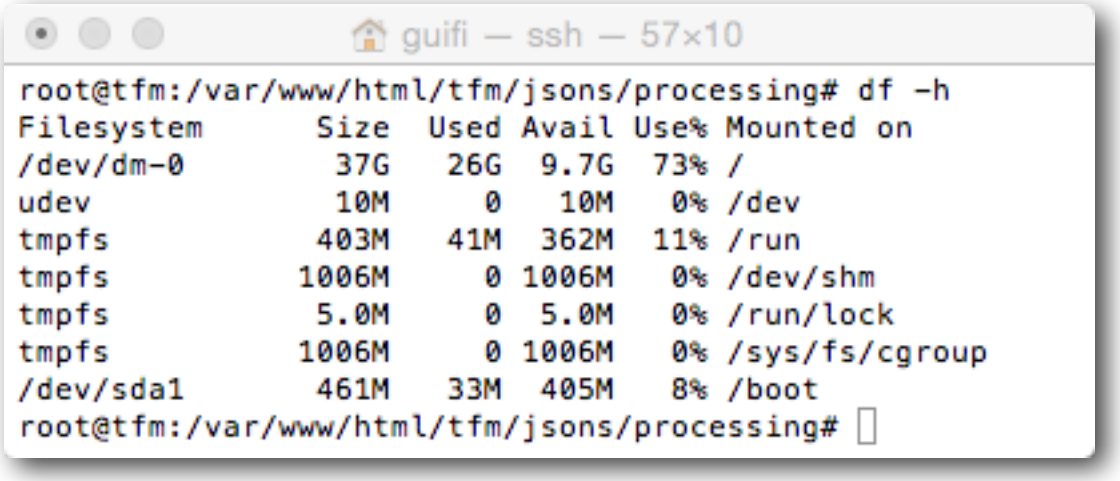

Figura 29: Captura de pantalla de l'espai a disc disponible un cop importades totes les dades

Figura 30: Captura de pantalla del phpMyAdmin de l'espai a disc consumit pel MySQL

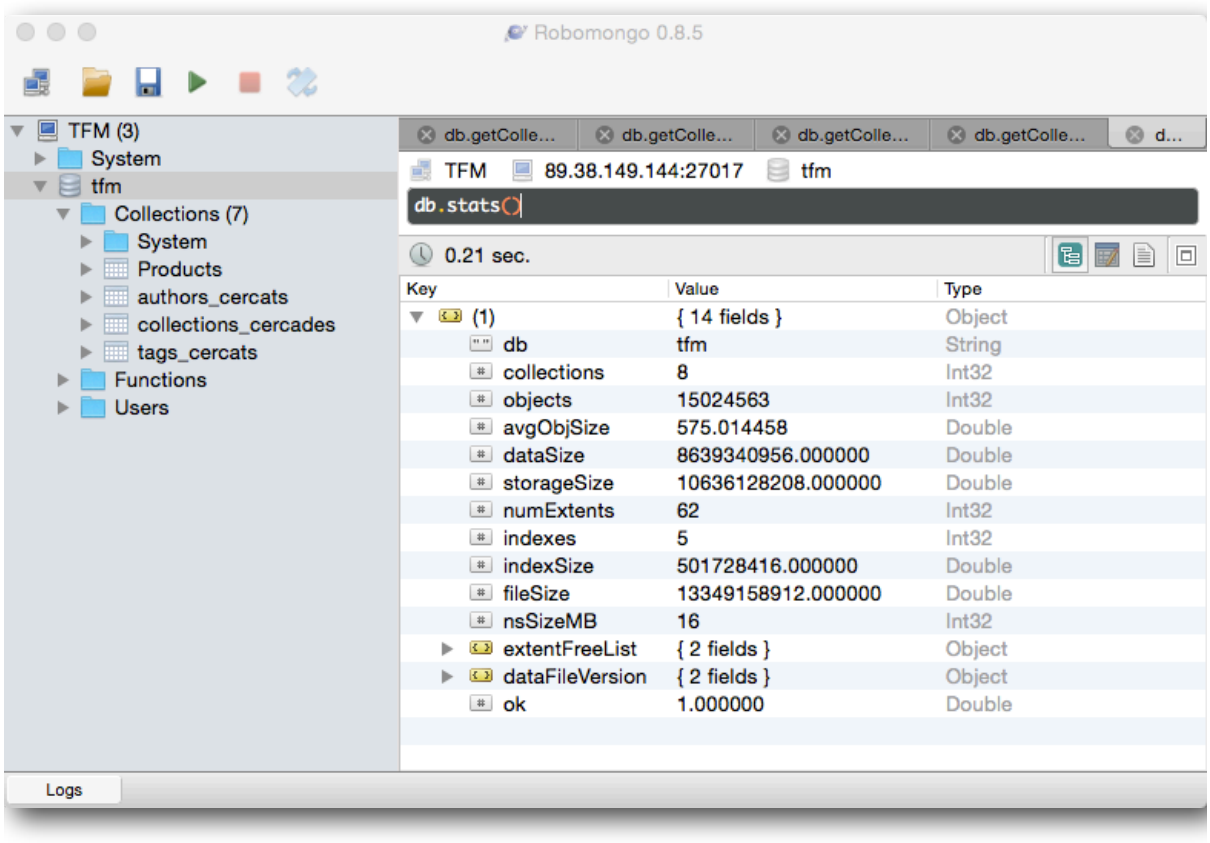

Figura 31: Captura de pantalla del robomongo de l'espai a disc consumit pel MongoDB

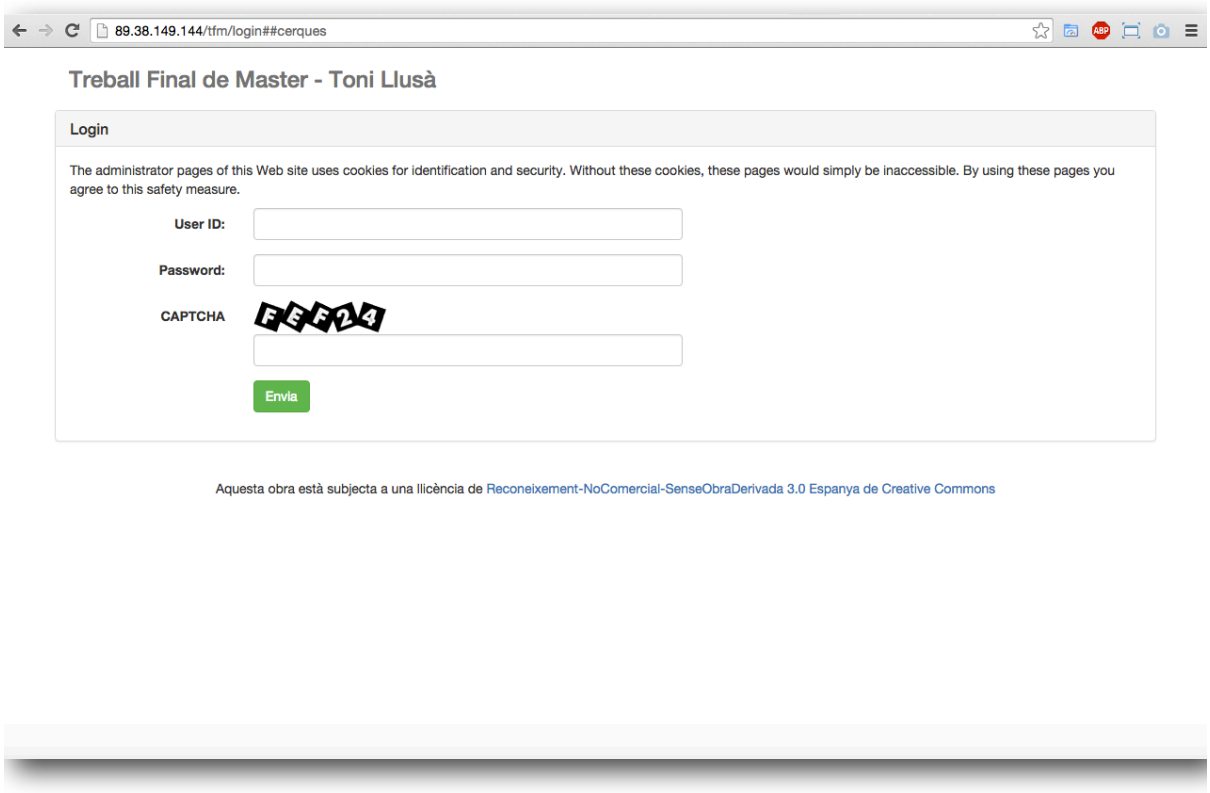

Figura 32: Captura de pantalla de la pantalla d'entrada de l'aplicació

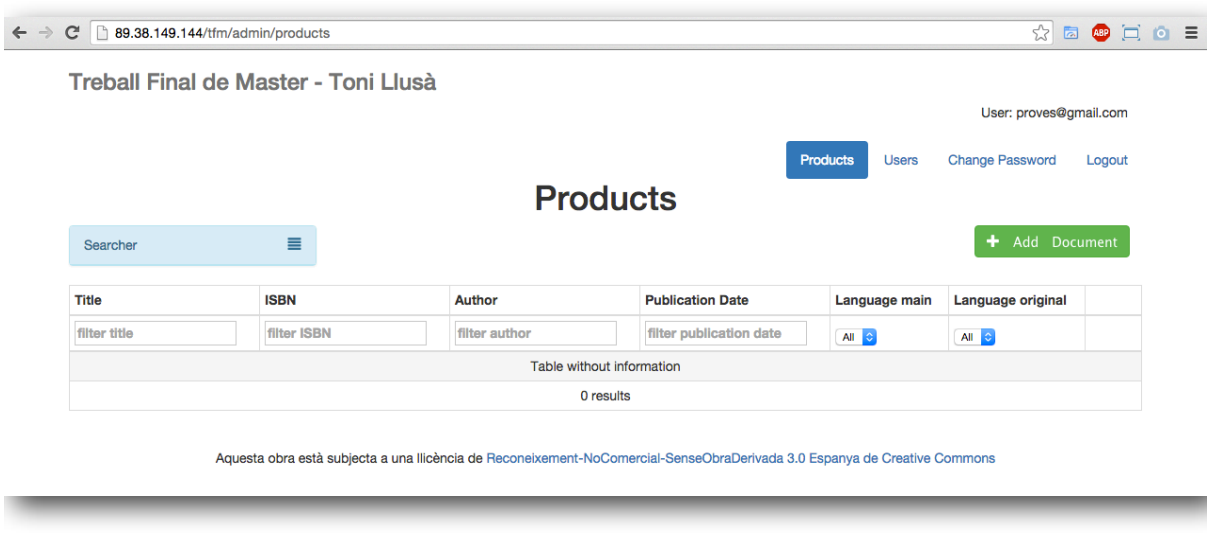

Figura 33: Captura de pantalla de la pantalla de cerca de productes

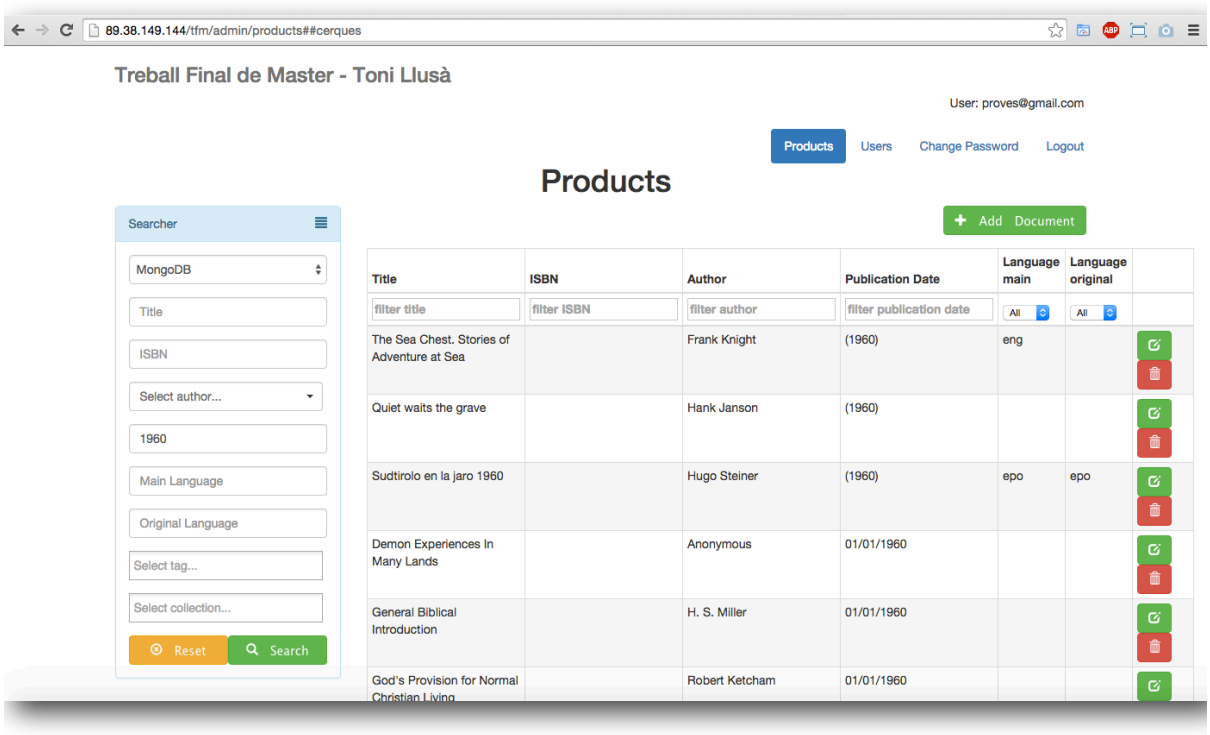

Figura 34: Captura de pantalla de la pantalla de cerca de productes, cercant llibres del 1960 amb MongoDB.

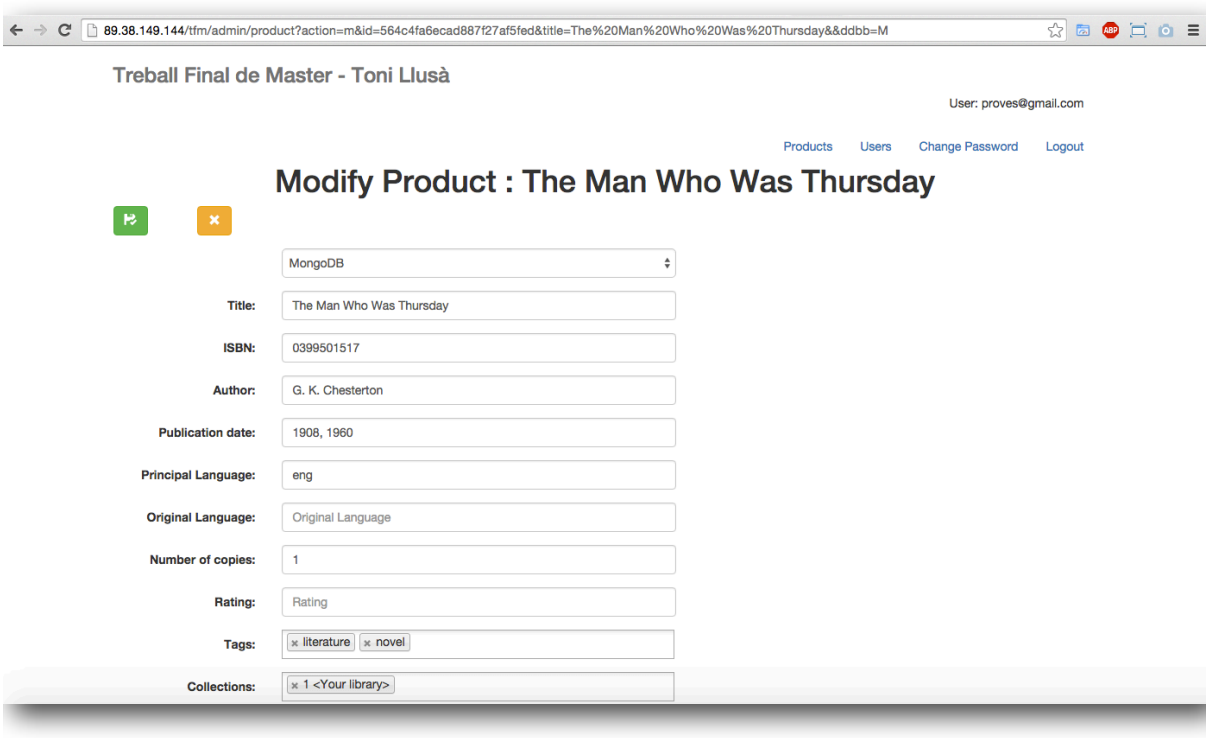

Figura 35: Captura de pantalla de la pantalla de detall de productes, del llibre "The Man Who Was Thursday" amb MongoDB.

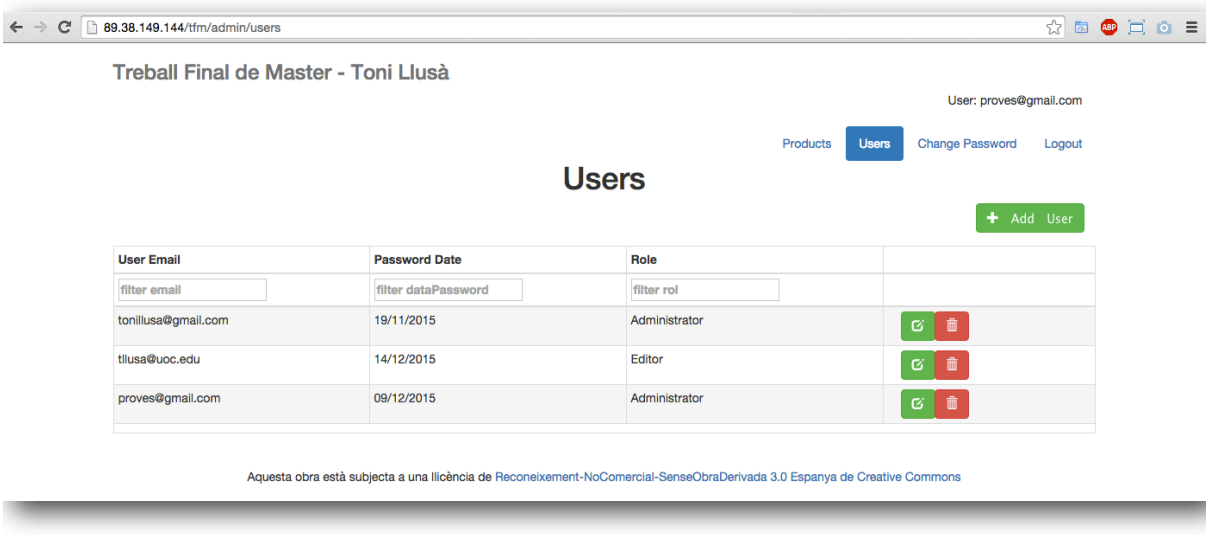

Figura 36: Captura de pantalla de la pantalla de llistat dels usuaris.

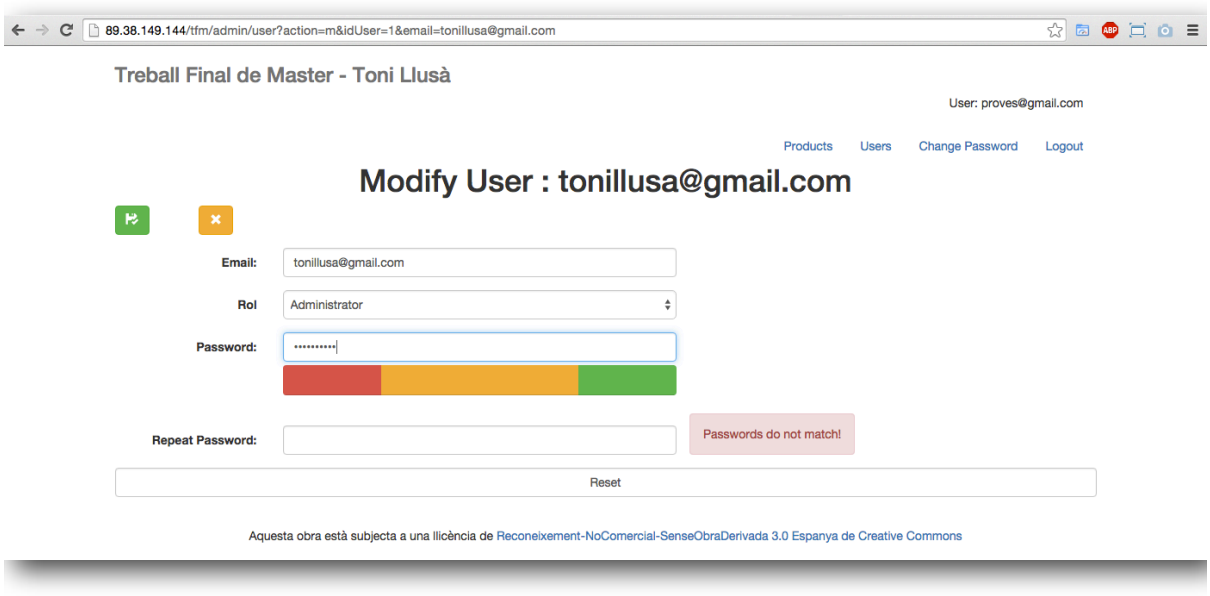

Figura 37: Captura de pantalla de la pantalla de modificar l'usuari.

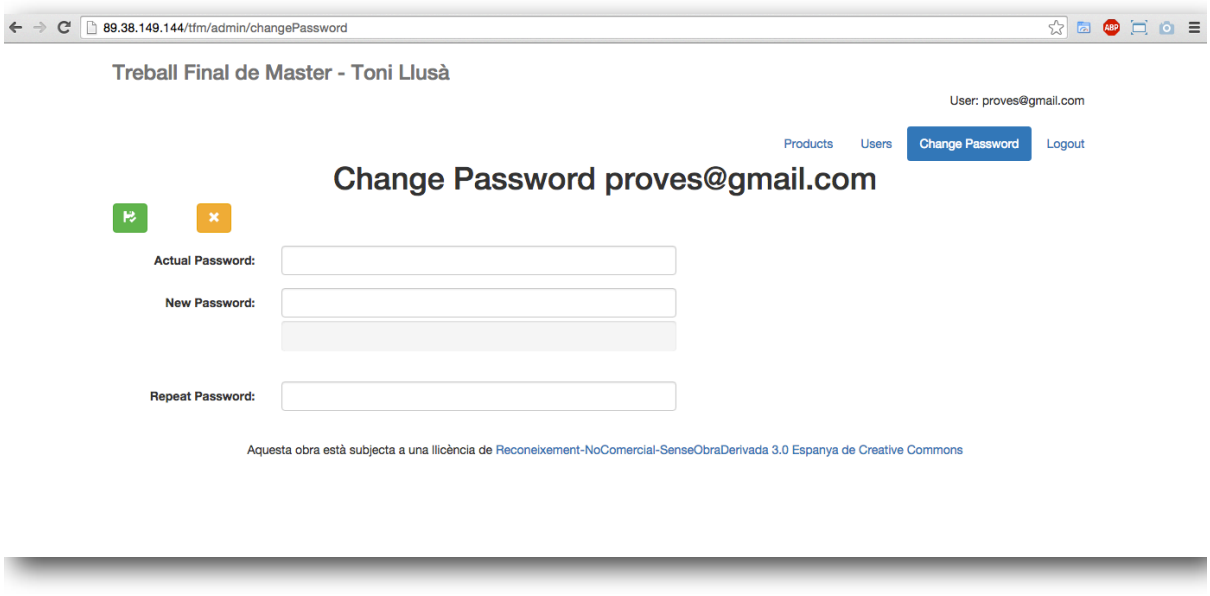

Figura 38: Captura de pantalla de la pantalla de modificar la contrasenya de l'usuari connectat.

Annex 5. Bibliografia

[1] https://www.apachefriends.org/download.html [23/09/2015]
[2] https://www.mongodb.org/downloads [23/09/2015]
[3] http://robomongo.org/ [23/09/2015]
[4] http://www.sequelpro.com/ [23/09/2015]
[5] PID_00153527 - La gestió de projectes. Conceptes bàsics.
[6] https://ca.wikipedia.org/wiki/MySQL [25/09/2015]
[7] https://ca.wikipedia.org/wiki/MongoDB [25/09/2015]
[8] https://ca.wikipedia.org/wiki/Nagios [30/09/2015]
[9] https://www.nagios.org/ [30/09/2015]
[10] https://ca.wikipedia.org/wiki/MySQL [2/10/2015]
[11] https://ca.wikipedia.org/wiki/MongoDB [2/10/2015]
[12] PID_00212075 - El model relacional i l'àlgebra relacional
[13] http://www.databasedir.com/what-is-sql/ [3/10/2015]
[14] BASE analysis of NoSQL database [3/10/2015]
[15] Perspectives on the CAP Theorem [3/10/2015]
[16] http://blog.mongodb.org/post/475279604/on-distributed-consistency-part-1 [4/10/2015]
[17] http://www.genbetadev.com/bases-de-datos/nosql-clasificacion-de-las-bases-de-datos-segun-el-teorema-cap [4/10/2015]
[18] http://blog.rodrigopuente.com/bases-de-datos-teorema-cap-cual-base-de-datos-debo-usar/ [4/10/2015]
[19] https://docs.angularjs.org [durant el desenvolupament de l'aplicació]
[20] http://fatfreeframework.com [durant el desenvolupament de l'aplicació]
[21] http://www.sequelpro.com/ [5/10/2015]
[22] http://robomongo.org/ [5/10/2015]
[23] http://lorenzofox3.github.io/smart-table-website/ [durant el desenvolupament de l'aplicació]
[24] http://github.com/angular-ui/ui-select [durant el desenvolupament de l'aplicació]
[25] https://github.com/urish/angular-spinner [durant el desenvolupament de l'aplicació]
[26] http://plnkr.co/edit/oZhqny?p=preview [durant el desenvolupament del l'aplicació]
[27] https://ca.wikipedia.org/wiki/MapReduce [28/11/2015]
[28] http://www.sergiquinonero.net/diferentes-tipos-de-ataques-en-paginas-y-aplicaciones-web.html [28/11/2015]
[29] http://fatfreeframework.com/databases [durant el desenvolupament de l'aplicació]
[30] http://getbootstrap.com/ [durant el desenvolupament de l'aplicació]
[31] http://lorenzofox3.github.io/smart-table-website/[durant el desenvolupament de l'aplicació]
[32] https://docs.mongodb.org/manual/tutorial/enable-authentication/ [durant el desenvolupament de l'apliació]
[33] https://www.jetbrains.com/phpstorm/ [durant el desenvolupament de l'aplicació]

Annex 6. Manual per provar / instal·lar l'aplicació

Per poder provar l'aplicació, s'ha instal·lat en un servidor web, explicat en l'apartat 7.2, per accedir a l'aplicació, s'ha d'anar al navegador web hi accedir a http://89.38.149.144/tfm/ veureu que demana un usuari i una clau d'accés, podeu accedir amb:

Usuari: proves@gmail.com
Contrasenya: Hola!15pwd

Un cop connectat, ja es podrà veure l'aplicació.

Per veure la base de dades Mysql: s'ha d'accedir a l'url: http://89.38.149.144/phpmyadmin/index.php

Usuari: toni
Contrasenya: Tfm!15pwd

Per veure la base de dades MongoDB des de l'exterior s'ha de connectar mitjançant el RoboMongo per exemple: amb l'adreça 89.38.149.144 i el port 27017.

Si es volgués instal·lar l'aplicació en un servidor web:

Aquest servidor web hauria de tenir les característiques comentades en l'apartat 7.2 i tenir instal·lat l'Apache , el MySQL , el PHP i el MongoDB.

Un cop configurat el servidor, s'haurien de crear les bases de dades SQL i MongoDB, segons el fitxer de configuració app/config.ini. Es pot trobar un arxiu bd.sql a dins la carpeta extres, amb la configuaració inicial de la base de dades SQL. Per la base de dades MongoDB, no es necessari, ja que en la propia importació si no hi ha la col·lecció creada ja la crea i afegeix els documents.

Es recomana ampliar el temps d'execució del servidor a 2 hores i la memòria màxima a 2Gb.

Llavors s'hauria de descarregar els arxius JSONS a http://89.38.149.144/tfm/jsons.tar.gz , un cop descarregat s'ha de descomprimir dins de l'arrel del projecte, llavors s'haurà d'executar els programes per importar en les diferents bases de dades:

Per exemple: http://localhost/tfm/importSQL i http://localhost/tfm/importMongo

Des de el fitxer de configuració app/config.ini hi ha la variable howManyFilesImport on es configuren els arxius a importar.

L'arxiu que s'ha acabat de processar va a parar a la carpeta jsons/processed, llavors s'han de moure altre cop a jsons/ per poder-los importar amb l'altre bases de dades.

Un cop la base de dades ja estar plena ja es pot executar l'aplicació amb http://localhost/tfm

Annex 7. Vitae

L'Antoni Llusa Sala, és Enginyer Tècnic d'Informàtica de Gestió per la Universitat de Vic, actualment treballa a Tuatara Aplicacions TIC SL, com a desenvolupador web realitzant tasques d'anàlisi, gestió de projectes i de programació web.

Per realitzar el Treball Final de Màster, s'han adquirit les següents competències:

Capacitat per a la integració de tecnologies, aplicacions, serveis i sistemes propis de l'enginyeria informàtica, amb caràcter generalista i en contexts més amplis i multidisciplinaris.

Capacitat per a modelar, dissenyar, definir l'arquitectura, implantar, gestionar, operar, administrar i mantenir aplicacions, xarxes, sistemes, serveis i continguts informàtics.

Capacitat per a assegurar, gestionar, auditar i certificar la qualitat dels desenvolupaments, processos, sistemes, serveis, aplicacions i productes informàtics.

Capacitat per a dissenyar, desenvolupar, gestionar i avaluar mecanismes de certificació i garantia de seguretat en el tractament de la informació i l'accés a aquesta informació en un sistema de processament local o distribuït.

Capacitat per a conceptualitzar, dissenyar, desenvolupar i avaluar la interacció persona-ordinador de productes, sistemes, aplicacions i serveis informàtics.

L'ús i aplicació de les TIC en l'àmbit acadèmic i professional.

Capacitat per a l'aprenentatge continu, autodirigit i autònom.

Realització, presentació i defensa, una vegada obtinguts tots els crèdits del pla d'estudis, d'un exercici original fet individualment davant d'un tribunal universitari, consistent en un projecte integral d'enginyeria informàtica de naturalesa professional en el qual se sintetitzin les competències adquirides en els ensenyaments.