Títol:	Desenvolupament d'un motor de recomanació social per a un web de tendències
Volum:	1/1
Alumne:	Anna Sánchez Inglada
Director/Ponent:	Luís Antonio Belanche Muñoz
Departament:	LSI
Data:	30/11/2009

Índex

Índex de figures

Capítol 1

Introducció

La introducció d'aquesta memòria té com a finalitat definir les motivacions i l'origen del projecte així com els objectius d'aquest. Aquest apartat acaba amb una breu presentació de l'estructura i continguts de la memòria.

1.1 Situació

Des dels inicis d'Internet i de l'aparició de la *World Wide Web*, la xarxa ha evolucionat de forma vertiginosa passant a convertir-se en el centre neuràlgic de la informació i, en gran part, dels negocis i les relacions entre les persones. Any rere any augmenta el nombre d'usuaris globals d'Internet - 1.000 milions a inicis de l'any en curs[1] - i les necessitats i usos de la Xarxa es van tornant més sofisticats.

Tot i que les primeres xarxes ciutadanes remunten a principis de la dècada dels 70, l'autèntic *boom* social d'Internet no va néixer fins a l'arribada de l'anomenada *Web 2.0*. Amb aquesta nova fornada del món de la web, s'ha aconseguit un espai global de col·laboració i intercanvi àgil d'informació entre els usuaris.

No obstant, encara que tant els buscadors com aquestes noves eines socials faciliten a l'usuari l'accés a la informació, el fet d'obtenir una informació enfocada als nostres interessos està esdevenint una necessitat i una molt bona oportunitat per les empreses que llancen els seus negocis per Internet. És per això que cada cop més les aplicacions

[1] Dada publicada el 23/01/2009 per la firma de anàlisis comScore

Idealment, m'hauria agradat treballar amb una xarxa social ja establerta, encara que fos una *start-up* amb pocs usuaris i poc tràfic, per incorporar-hi un sistema de recomanació i treballar la part social, per veure els efectes que podia tenir. No obstant, davant la impossibilitat d'aquesta opció s'ha optat per centrar esforços en l'estudi i disseny d'un sistema de recomanació genèric que pugui treballar de forma eficaç tant en contextos de gran densitat d'usuaris i d'informació com a la inversa. Finalment, per tal de dur a la pràctica les idees pensades al llarg del projecte, s'ha creat un pilot, una petita xarxa social que incorpora com a eix del seu model el recomanador ideat.

1.3 Objectius generals

Aquest projecte es pot resumir en dos grans objectius:

- **Estudiar i dissenyar un sistema de recomanació social.**
- **Desenvolupar un pilot a partir de les conclusions extretes en l'estudi previ.**

Pel que fa a l'estudi i al disseny del sistema s'han identificat les següents fites:

- Realització d'un estudi sobre els sistemes de recomanació, les tècniques utilitzades i les problemàtiques que poden sorgir.
- Realització d'un estudi sobre la relació entre les xarxes socials i els sistemes de recomanació per detectar requeriments del sistema a dissenyar.
- Realització del disseny del sistema a partir de les idees extretes dels dos punts anteriors.
- Creació d'un model de dades òptim per al posterior ús en el sistema de recomanació.

Per altra banda, durant la construcció del pilot s'ha posat especial èmfasis en:

- Pel que fa al desenvolupament del sistema de recomanació:
  - Eficiència
  - Portabilitat a altres entorns

detallada del sistema de recomanació pensat, sense entrar en detalls de com es pot construir.

Construcció

En la fase de construcció intervindran diferentes subfases comuns de la enginyeria del software que són les següents:

- Especificació
- Disseny
- Implementació
- Proves

Documentació

En aquesta fase s'engloba tant algunes fases de la redacció de la memòria (objectius assolits, resultats, annexos, etc.) com la creació d'un document guia del sistema per l'usuari.

1.4.2 Recursos

Per tal de dur a terme aquest projecte, s'han destinat molts pocs recursos:

- **Humans**: 1 persona amb una jornada de treball de 4 hores
- **Software**: tot el software utilitzat és de codi lliure per minimitzar el cost del projecte.

1.4.3 Temps

Al no tenir un client final i una data d'entrega concreta, el temps per dur a terme aquest projecte és bastant lliure. No obstant, s'ha fet una estimació del temps a invertir que es detalla a continuació:

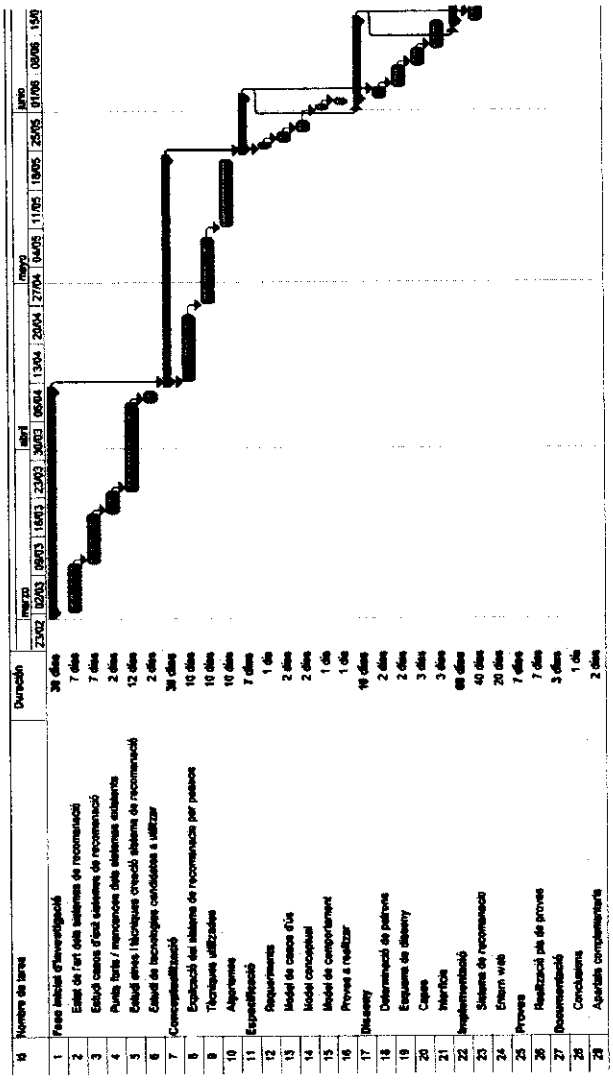


Figura 1.1: Diagrama de Gantt Part 1

1.5 Organització de la memòria

La memòria s'ha organitzat de la següent manera:

Capítol 1. Introducció

En el capítol que acabem de veure s'ha volgut situar al lector, explicant les motivacions, els objectius generals del projecte, les fases que es tindran en compte i la planificació establerta.

Capítol 2. Sistemes de recomanació

En aquest apartat s'argumenta tot l'estudi previ a la conceptualització i posterior desenvolupament del sistema de recomanació d'aquest projecte. L'objectiu és conèixer tots els elements que intervenen en un sistema de recomanació així com altres eines de recomanació existents per extreure requeriments del sistema a desenvolupar.

Capítol 3. Conceptualització del sistema de recomanació

S'exponen les decisions preses a partir de les conclusions extretes de l'apartat anterior i es descriu conceptualment com funcionarà el sistema de recomanació ideat i quines eines utilitzarà.

Capítol 4. Desenvolupament del sistema de recomanació

En aquest apartat es defineix el projecte software a desenvolupar, les tecnologies que s'utilitzaran i s'explica cada una de les fases del procés de desenvolupament del software (Especificació, Dissey, Implementació, Proves).

Capítol 5. Conclusions

S'exposen els objectius aconseguits, s'estima el cost del projecte i es dóna una opinió personal de tot el procés.

Altres:

- **Guia de l'usuari**:
  S'inclou la guia d'iniciació al sistema que es troba en el entorn web com a guia de primers passos, que intenta explicar a l'usuari tot el que pot fer en la web i com fer-ho.

Capítol 2

Sistemes de Recomanació

En els darrers anys, ha augmentat considerablement la interacció dels usuaris en la xarxa i aquest fet ha comportat un augment considerable de la informació disponible. Cada cop més, l'usuari necessita suport a l'hora d'aconseguir la informació que requereix i és per aquest motiu que els sistemes de recomanació, els motors de cerca personalitzada i altres sistemes intel·ligents han agafat força en el món d'Internet.

> Un sistema de recomanació és una aplicació que té com a objectiu donar suport al usuari a l'hora de prendre decisions mentre interactua amb una massa important informació. Aquests sistemes formulen recomanacions que poden interessar a l'usuari basant-se en les preferències que ha expressat de forma implícita o explícita [Ala09]

En altres paraules, el seu objectiu és millorar l'experiència d'usuari ajudant a superar la sobrecàrrega d'informació existent en la web i oferint, d'altra banda, una sensació de novetat, sorpresa i rellevància. En aquest capítol es pretén comentar els fonaments dels sistemes de recomanació així com contextualitzar alguns dels sistemes existents per entendre el seu funcionament i els avantatges i inconvenients que ofereixen.

Anem a veure com influeixen aquests paràmetres en les entranyes del sistema de recomanació. És fàcil entendre la importància del domini i de l'entorn. Sense un coneixement acurat del domini en que es treballa, sense informació sobre els elements susceptibles de ser recomanats, el sistema seria una eina sense valor. Per exemple, si es volgués recomanar llibres, s'hauria de tenir suficient informació sobre autors, editorials, títols, temàtiques, etc. per poder generar recomanacions de forma satisfactòria. Partint d'una temàtica com podria ser "*Contes per adults*"es podrien treure autors com "*Roald Dahl*", "*Edgar Allan Poe*", etc.

No obstant, amb aquests dos paràmetres no s'aniria més enllà del filtratge d'una base de dades, de la simple cerca de continguts. Per això és interessant parar atenció als altres dos paràmetres.

El punt de partida, i un dels passos més importants a l'hora de crear un sistema de recomanació, és la **representació dels perfils d'usuari**. És necessari conèixer el màxim possible d'informació referent a l'usuari per tal de poder-li proporcionar uns resultats adequats. Quan es parla però d'informació de l'usuari no s'engloben només les seves dades sinó també les seves interaccions. En molts dominis pot influir, per exemple, l'edat o el sexe de l'usuari. No obstant, el que acaba influint més, són les coincidències entre els comportaments d'aquests. Quan un usuari interacciona amb un sistema, proporciona una gran quantitat d'informació referent a les seves predileccions, ja sigui de forma implícita o explícita. Entenem de forma implícita saber les pàgines que visita, les cerques que realitza, els altres perfils d'usuari que consulta, etc. Per altra banda, proporciona informació explícitament quan afegeix etiquetes a elements, vota o opina sobre un tema, etc. Aquesta informació de l'usuari és utilitzada per generar i mantenir el seu perfil i, per tant, el sistema ha de saber **captar les interaccions** i aprendre dels seus gustos i interessos. Aquesta adquisició de coneixement per part del sistema es realitza fent **ús de tècniques d'aprenentatge** o d'altres mètodes que, a partir del perfil d'usuari, extreuen informació i la estructuren segons la representació del perfil que s'ha definit.

sistema de recomanació ja pot començar a explotar la informació per tal de recomanar productes o accions a l'usuari. Com que pot existir un gran nombre d'informació disponible, un punt molt important és escollir la informació adequada per realitzar les recomanacions, pel que un **mètode de filtratge d'informació** és imprescindible. No s'ha d'oblidar però que també seran necessàries tècniques de correspondència entre perfils d'usuari o entre preferències d'usuari i elements. Entenem per **tècniques de correspondència** aquells algorismes que buscaran relacions existents entre el contingut o entre els usuaris.

Així doncs, els dos requeriments finals d'un sistema de recomanació són:

6. Utilitzar un mètode de filtratge d'informació
7. Utilitzar tècniques de correspondència

Resumint, des del punt de vista funcional del sistema, podem distingir dos grans grups, la generació i manteniment del perfil i l'explotació del perfil. Dins d'aquests grups hem identificat els punts que es troben resumits en la Figura 2.3 i que veurem amb més profunditat en els següents apartats.

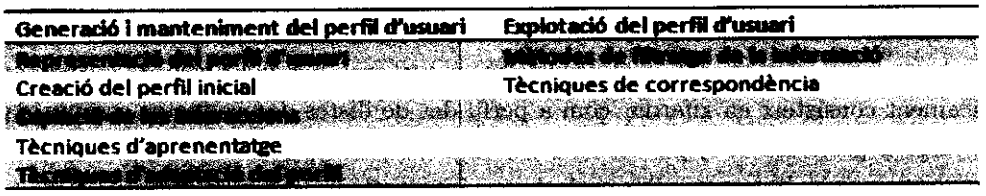


Figura 2.3: Elements d'un sistema de recomanació

2.2 Els sistemes de recomanació per passos

En l'apartat anterior s'ha estudiat el funcionament a alt nivell dels sistemes de recomanació i s'ha extret un seguit de requisits que un sistema de recomanació ha d'acomplir. En les següents pàgines, s'estudiarà cada pas presentant avantatges i inconvenients que es poden trobar al llarg del disseny d'un sistema com aquest.

Model d'espai vectorial

El model d'espai vectorial és un model algebraic molt utilitzat en la representació de documents de text però aplicable a qualsevol altre tipus d'objecte d'estudi. La representació consisteix en vectors d'identificadors que tenen un valor associat. Aquest valor representa la freqüència en que un objecte apareix i, per tant, mesura la rellevància que té. Tot i que aquest model es pot encabir en el camp de la recomanació, s'utilitza especialment en indexació i en filtratge de la informació.

Per exemple, es podria representar el contingut dels llibres en forma de vectors i serviria per trobar altres llibres de temàtica similar.

Models de classificació

Els sistemes que utilitzen un classificador com a tècnica d'aprenentatge (secció 2.2.4) dels perfils d'usuari, retenen l'estructura del classificador com a perfil. Aquest és el cas de les xarxes neuronals, els arbres de decisió, les regles d'inducció o les xarxes Bayesianes.

Una **xarxa neuronal** és una model matemàtic basat en el model biològic de les xarxes neuronals. Consisteix en la interconnexió d'un grup de cel·les o neurones artificials d'entrada i de sortida que s'utilitzen per processar informació. Les xarxes neuronals creen una representació sòlida que dóna una resposta ràpida a les consultes. No obstant, són difícils d'entrenar.[Mon03]

Un **arbre de decisió**, consisteix en un conjunt de nodes i un conjunt de connexions que segueixen una estructura d'arbre. Els nodes interns representen preguntes sobre els paràmetres i les connexions possibles respostes. Les decisions finals es troben a les fulles de l'arbre.[Alu]

Les **regles d'associació** són una eina molt utilitzada en màrketing per crear patrons de preferències envers un producte i per recomanar productes als clients. Aquesta representació pot comportar avantatges a nivell d'eficiència, d'emmagatzematge i rendiment. [cla]

Les **xarxes Bayesianes** són grafs acíclics on cada node representa una variable i cada arc una dependència. Per tant, cada node es computa en funció dels nodes dels quals depèn. En el camp de la recomanació per exemple, es podria intentar predir com de probable és que li agradi un element a un usuari a partir d'aquestes xarxes.[Rei06]

Per posar un exemple d'un d'aquests tipus de classificadors, i seguint amb la temàtica

Perfil Buit

Un sistema de recomanació pot començar a treballar amb un model buit, és a dir, que inicialment no tingui en compte les interaccions dels usuaris i que es valgui d'altres eines que li permetin generar recomanacions. L'avantatge d'utilitzar aquest model inicial és que no es necessita un entrenament que a vegades pot resultar costós. No obstant, el gran inconvenient és que limita el sistema de recomanació.

Manual

Un sistema manual per generar el perfil inicial d'usuari és un sistema que requereix la col·laboració dels usuaris demanant, per exemple, la introducció dels seus interessos. Per una banda, l'avantatge d'aquest mètode és la transparència cap a l'usuari i la qualitat de la informació obtinguda, ja que és directa i no predita. Per altra banda, l'inconvenient és que molts cops l'usuari no està disposat a col·laborar i li pot suposar una feina tediosa. Per aquest motiu, si s'escull aquest enfocament s'ha de posar esforços en fer aquesta col·laboració fàcil per l'usuari.

Entrenament

Es pot generar el perfil inicial utilitzant un conjunt d'interaccions d'exemple per tal d'entrenar el sistema. Un cas podria ser demanar als usuaris que valorin un seguit d'elements escollits com exemples representatius del context i, posteriorment, processar les dades obtingudes amb alguna tècnica d'aprenentatge. Aquest mètode pot aportar una bona base pel sistema de recomanació però, per tal de que tingui sentit, s'ha d'escollir una mostra d'exemples representativa i sovint no és una tasca fàcil.

2.2.3 Captació de les interaccions d'usuari

Ja hem vist que una de les claus per obtenir un bon sistema de recomanació és, per una banda la representació del perfil i, per altra banda, com mantenim la informació d'aquest. Podem considerar el coneixement, és a dir, la informació rellevant sobre els usuaris, de tres formes diferents:[Ala09]

- **Coneixement explícit**

  Aquest grup reuneix tota aquella informació que l'usuari proporciona a l'aplicació de forma explícita. Per exemple, l'etiquetatge o votacions d'articles. És una eina molt potent per la qualitat de la informació que s'obté, directa i, en principi,

Els mecanismes d'adquisició implícits acostumen a ser els preferits ja que tenen un impacte molt reduït o inexistent en la interacció habitual de l'usuari amb l'aplicació. Per exemple, el monitoratge no intrusiu de l'activitat dels usuaris al llarg d'un període de temps determinat, ajuda a descobrir dades rellevants de comportament de l'usuari. Aquestes dades es poden utilitzar per inferir en les seves preferències més comuns. Altres tècniques com l'heurística també es poden utilitzar per crear hipòtesis a partir de les dades existents. No obstant, les tècniques d'adquisició implícites requereixen un grau d'interpretació per part de l'expert per entendre els objectius reals de l'usuari i per tant, es genera un possible error que afectarà a la creació dels perfils d'usuari.

En canvi, l'adquisició de coneixement a partir de tècniques explícites redueix aquest marge d'error ja que la informació s'obté directament de la resposta de l'usuari. Tot i així, aquestes tècniques requereixen l'atenció de l'usuari i per tant la interrupció de la seva activitat habitual en l'aplicació, fet que a vegades pot resultar un inconvenient.

Vistos els avantatges i inconvenients de cada tècnica d'adquisició de coneixement, resulta raonable que molts sistemes acabin utilitzant un sistema híbrid d'ambdós tècniques. D'aquesta manera, s'aconsegueix no abusar de la col·laboració de l'usuari i es fa ús de la seva interacció per ratificar la informació que s'hagi pogut extreure implícitament. Per exemple, demanant la valoració de possibles suggeriments que un sistema pot fer a l'usuari.

2.2.4 Tècniques d'aprenentatge i altres mètodes

Les tècniques d'aprenentatge són necessàries per construir els perfils d'usuari a partir de les dades que es disposen.Tot i que existeixen un gran nombre de tècniques, ens centrarem en els enfocaments més comuns: *clustering*, classificadors i els sistemes que no necessiten tècniques d'aprenentatge.[Ala09][AB]

Tècniques d'aprenentatge

Clustering

Un sistema que utilitza una tècnica de *clustering*, intenta agrupar la informació similar dels usuaris en grups o *clusters* basats en aquestes dades. D'aquesta manera, quan apareix un nou element, aquests grups són comparats amb la informació de l'element actual per tal d'extreure conclusions respecte a si és o no interessant per l'usuari.

usuaris fet que pot comportar una manca d'informació o acabar tenint la informació desactualitzada si els requeriments per part de l'usuari canvien molt ràpidament. No obstant, si es planteja de forma atractiva, usable i ràpida, pot ser la manera més fàcil per aconseguir que els usuaris obtinguin recomanacions més similars a les preferències que estan demanant en un moment determinat.

Afegir nova informació

Aquesta tècnica és la més utilitzada pels sistemes de recomanació. Es tracta d'afegir les noves preferències dels usuaris, obtingudes a partir de la recuperació de la informació dels seus perfils, per així actualitzar-los. No obstant, aquest sistema té un inconvenient que és que té informació nova rellevant però continua tenint informació antiga que potser ja no interessa als usuaris.

Funcions que esborrin la informació antiga

Per tal de solucionar la problemàtica de la tècnica anterior, alguns sistemes utilitzen algorismes que utilitzen la informació més recent i, obliden progressivament la informació més antiga per així garantir que sempre es manega la informació que en l'actualitat és rellevant pels usuaris.

No es podrà evitar la participació dels usuaris per tal d'actualitzar la seva informació, però si que s'ha d'intentar que sigui de la manera menys tediosa possible. També és important que el sistema sàpiga adaptar-se a l'actitud canviant dels usuaris.

2.3 Explotació del perfil d'usuari

Un cop definida la manera com es crea i es manté el perfil, tan sols queda explicar els mètodes i sistemes que s'utilitzen per explotar la informació continguda en aquests perfils.

2.3.1 Sistemes de filtració d'informació

Per tal de trobar relacions entre els usuaris i el contingut i, també, relacions entre els mateixos usuaris, necessitem un llenguatge comú per computar la rellevància entre aquests elements i usuaris.

Existeixen dos enfocaments per realitzar aquesta filtració d'informació: la filtració basa-

poesia, un llibre es tracta com una entitat sense tenir en compte el que conté. Per tant, la mateixa infraestructura pot ser utilitzada en diferents dominis i llenguatges.

- En els sistemes basats en el contingut, els algorismes no tenen cap noció de la qualitat dels elements. En canvi, els sistemes que utilitzen filtració col·laborativa, tenen una gran quantitat d'informació respecte la qualitat dels elements.
- Al llarg d'un període de temps concret, els resultats obtinguts mitjançant un sistema basat en el contingut no canvien massa. En canvi, els sistemes que utilitzen filtració col·laborativa, al basar-se en les interaccions de l'usuari, obtindran resultats més variats al llarg del mateix període de temps.

Avantatges de la filtració basada en el contingut

- Els sistemes basats en la filtració col·laborativa, confien en la informació proporcionada per l'usuari per tal de trobar usuaris similars i recomanar elements. Si no existeix una quantitat suficient de dades, aquests sistemes pot ser que realitzin unes recomanacions molt pobres. Per a un nou usuari, que hagi interactuat poc amb el sistema, no existirà suficient informació per trobar usuaris similars. En la filtració de contingut, si existeix contingut representable, sempre es donaran resultats.
- El sistemes basats en la filtració col·laborativa, no recomanaran nous elements que hagin pogut ser introduïts al sistema recentment fins que un nombre substancial d'usuaris els hagin valorat. En canvi, en la filtració de contingut, un cop computat el vector de termes o mots, es podrà recomanar instantàniament.

Així doncs, es pot veure que tant un com l'altre tenen les seves limitacions i els seus punts forts, fet que la opció d'un sistema híbrid pot interessar per intentar cobrir les mancances de cada un. Els sistemes híbrids intenten explotar les característiques de ambdós sistemes, ja que són complementaris. Aquests sistemes són molt més eficaços que els que centren tot el seu potencial únicament en una de les dues vessants explicades anteriorment.

2.3.2 Tècniques de correspondència

A l'hora de trobar correspondències per generar les recomanacions, el sistema es pot basar o bé en els elements del context o bé en els usuaris.[Ala09]

de perfils segons les característiques particulars de l'usuari (sexe, edat, nacionalitat, etc.).

En el primer cas, la relació entre elements es pot fer tal i com s'ha vist a partir de les tècniques de filtració basades en el contingut o bé, si es té una base de dades amb informació relacional, es pot fer ús de la simple filtració de la base de dades.

En el cas de la relació entre usuaris, si es tracta de filtració col·laborativa, es fa ús de tècniques com la cerca de veïns més propers o altres tècniques de cerca per similituds. La cerca de veïns més propers (*Nearest Neighbour*) es basa en obtenir la distància del element que ens interessa respecte a la resta d'elements o de classes d'elements del perfil d'usuari. D'aquesta manera, s'acaben creant "veïnats"on els elements o usuaris d'un mateix veïnat són els que es tindran en compte. Un cas particular de l'aplicació del *Nearest Neighbour* és el *Case-Based Reasoning* que s'explicarà en més detall en el proper capítol.

2.4 Alguns sistemes de recomanació en la xarxa

Un cop s'ha presentat una visió abstracta dels sistemes de recomanació, resulta interessant observar el que trobem a la xarxa per veure quines són les tècniques que s'utilitzen. Observant diferents webs s'ha anat trobant que la majoria utilitzen patrons similars. Per exemple, la majoria utilitzen filtració col·laborativa en comptes de filtració basada en el contingut. En canvi, segons si es tracta d'una xarxa social o d'una web amb altres finalitats com *e-commerce* o agregació de contingut, les tècniques de correspondència varien.

Per tal de resumir les experiències obtingudes de l'observació de la xarxa, a continuació s'analitza per passos dos exemples, una xarxa social basada en el model de filtració col·laborativa com és *Last.fm* i una web de *e-commerce* emblemàtica i pionera en utilitzar la recomanació en el seu model com és *Amazon*.

Last.FM

Last.fm és una xarxa social de música que utilitza un sistema de recomanació anomenat *Audioscrobbler* per tal de construir un perfil detallat de les preferències musicals de cada usuari, enregistrant els temes que cada usuari escolta tant a la radio que la mateixa web proporciona com en el seu ordinador personal o altres dispositius. La informació capturada és transmesa a la base de dades de *Last.fm* i posteriorment utilitzada per ser

manualment o començar escoltant directament música. Si la cançó arriba fins al final, el sistema la marca com a bona. Però si l'usuari no li agrada la cançó i prem el botó de canviar de cançó, és marcada com a no bona. D'aquesta manera, al llarg del temps, es va construint un perfil d'usuari amb les seves preferències [Las]. Finalment, comparant els perfils de cada usuari que escolta música a *Last.fm*, el sistema pot predir quines cançons li poden agradar particularment a un usuari a partir de les que altres usuaris semblants ja han escoltat i han marcat com a bones. Per tant, utilitza un mètode de correspondència basada en els usuaris. Un punt fort de *Last.fm* és que pot captar molta informació de les preferències de l'usuari de forma implícita i, per tant, transparent a aquest. Si l'usuari té connexió a la xarxa, el sistema enregistra tota la música que aquest escolta al seu ordinador i, per tant, es poden arribar a conèixer, per exemple, les seves bandes preferides. També capta informació explícitament, com ara si un usuari marca una cançó com a preferida.

Així doncs, *Last.fm* disposa d'un volum d'interaccions molt elevat. No obstant, hi ha certes recomanacions que són acurades degut a aquest volum d'informació però per altra banda, hi ha altres recomanacions que acaben sent no tan acurades com es desitjaria per manca de coneixement derivat. Per exemple, les recomanacions de bandes acostumen a ser molt encertades ja que un usuari pot generar molt volum de temes en poc temps. No obstant, la recomanació d'esdeveniments acostuma a ser més fluixa ja sigui perquè l'usuari no indica a la plataforma tots els esdeveniments a que ha anat,o bé pel volum d'esdeveniments als que un usuari pot assistir, que són molt inferiors al nombre de cançons escoltades en el mateix període de temps.

Observant doncs *Last.fm*, i el fet comentat unes línies amunt sobre la precisió en les recomanacions de música que no tenen els esdeveniments, fa pensar que un sistema basat purament amb la filtració col·laborativa pot portar aquest tipus de problemes: davant un volum petit d'interaccions amb el sistema, aquest no és capaç de generar una recomanació de qualitat. També s'observa que la captació explícita de les interaccions de l'usuari pot limitar la qualitat de informació de l'usuari, ja que aquest sovint no perd el temps introduint certa informació que no considera rellevant.

També s'ha de tenir en compte que no utilitza aparentment cap tècnica d'adaptació del perfil. En usuaris que porten molt de temps en contacte constant amb la plataforma, i que els seus gustos han evolucionat, segueix proporcionant cançons similars a preferències del passat. L'adaptació del perfil es dur generalment a terme afegint nous elements. Tot i així, de manera no gaire directe, l'usuari també pot eliminar preferències de la llista o ressetejar el seu perfil. Aquestes opcions no són massa visibles, cosa que fa pensar que

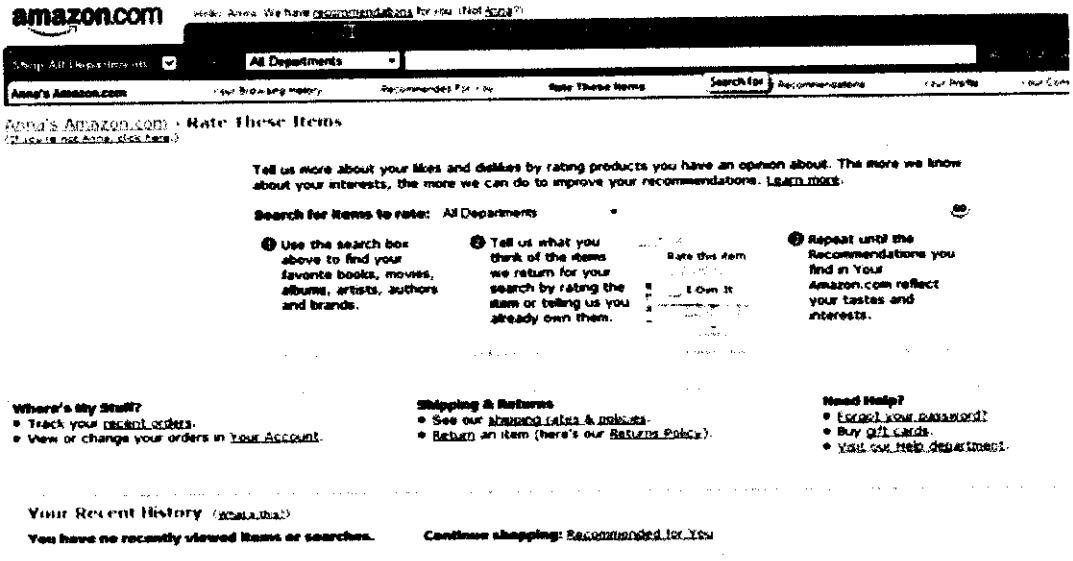


Figura 2.11: Amazon, captació preferències usuari

Un cop el sistema té aquesta informació podem explicar com funciona el sistema de recomanació. El sistema pot generar recomanacions només basant-se amb la part d'historial de cerca i de navegació. No obstant, les recomanacions obtingudes sense valorar o comprar cap element són molt dolentes. Com més elements votes i més elements compres, millors són les recomanacions. Aquest fet s'explica perquè, com més informació d'aquest tipus proporciones al sistema, més capaç és de comparar els teus elements amb els de la resta dels usuaris. Així doncs de nou s'utilitza la filtració col·laborativa però aquest cop basada en els elements i no en els usuaris. És a dir, Amazon intenta aconseguir regles d'associació de l'estil "Qui ha comprat aquest element, també ha comprat aquest altre".

llocs s'han extret les següents conclusions:

1. La filtració col·laborativa enriqueix les recomanacions que es poden generar. Per tant, s'ha de tenir en compte com a una eina molt vàlida sempre que es complementi amb altres mètodes si el volum de interaccions dels usuaris és baix.

2. S'ha d'intentar aconseguir el màxim nombre d'interaccions dels usuaris. La captació de coneixement implícit és interessant però aporta poc valor informatiu. S'ha d'intentar captar també informació explícita de l'usuari però de manera que sigui fàcil i amigable.

3. En un sistema de recomanació social, serà més interessant utilitzar correspondència entre els usuaris, ja que serà molt més fàcil explotar la informació dels usuaris similars.

4. S'ha observat que la generació manual del perfil inicial és la tècnica més utilitzada. És la manera més ràpida d'obtenir informació de valor de l'usuari.

5. Les tècniques d'adaptació del perfil que més comunament s'utilitzen són les acumulatives. Tot i així, també és interessant pensar en eliminar cert comportament del passat de l'usuari que pot ser que ja no l'interessi.

Un cop es tenen en compte aquestes consideracions i tot el què s'ha tractat, ens proposem a descriure el sistema de recomanació pensat en aquest projecte.

Figura 3.1: Visió d'un sistema de recomanació com una caixa negra

3.1.1 Domini

Com ja s'ha vist, el domini és l'element bàsic per on començar un sistema de recomanació. Per això cal una bona base de dades amb molta informació sobre els elements que es volen recomanar.

Primer, es va intentar utilitzar la base de dades d'una xarxa social anomenada *Trendtation*, una *start-up* catalana de moda. No obstant, ràpidament es va descartar aquesta opció perquè el volum d'informació disponible era reduït i no s'hauria pogut desenvolupar un motor de recomanació sota els paràmetres que es volien tenir en compte, i que més endavant s'explicaran.

La següent opció va ser buscar recursos disponibles a la xarxa de forma gratuïta. No és fàcil, però, trobar bases de dades amb un gran volum d'informació que es puguin descarregar, instal·lar i fer-ne ús lliurement. Així doncs, la decisió de sota quin domini treballar ha vingut condicionada per aquesta dificultat. Finalment, després de realitzar una cerca a la xarxa en busca de bases de dades, es va trobar una base de dades de música molt completa, utilitzada i actualitzada per una comunitat d'usuaris i, també, per grans corporacions com les que hem parlat abans, *Last.fm* i *Amazon*. Aquesta comunitat i base de dades s'anomena *Musicbrainz*.

Musicbrainz és una projecte de la fundació *MetaBrainz* que pretén crear una base de dades musical de contingut obert. Emmagatzema informació d'artistes, de discs, de cançons i la relació entre aquests. Es pot fer ús dels serveis de *Musicbrainz* per etiquetar

3.1.4 Motor del sistema de recomanació

Aquest ha estat el punt central del projecte al qual s'han dedicat majors esforços. Independentment del domini del que s'acaba de parlar, s'han desenvolupat una sèrie d'hipòtesis i raonaments per tal de crear un sistema de recomanació genèric que pugui ser utilitzat en l'entorn d'una xarxa social però sota qualsevol domini.

Després d'estudiar les formes d'encarar un sistema de recomanació i tenir un primer contacte amb altres sistemes existents, s'han observat una sèrie de pros i contres que han fet que al final s'optés per la solució que en els propers apartats es comenta.

El primer punt important a determinar ha estat quina tècnica de filtratge utilitzar, la tècnica basada en el contingut o la filtració col·laborativa. Com ja hem vist, per crear un bon recomanador resulta quasi indispensable fer ús de la filtració col·laborativa. No obstant, els dubte que ha sorgit ha estat:

Existiran suficients interaccions dels usuaris per realitzar filtració col·laborativa?

Per tal de trampejar la possible manca d'interaccions en les fases inicials del funcionament del recomanador, s'ha optat per prendre la següent decisió:

- Ús d'una **visió híbrida del sistemes de filtració on, en el millor dels casos ambdues parts cooperin**. És a dir, si no existeix suficient informació dels usuaris per fer ús de la filtració col·laborativa, el sistema serà capaç de retornar recomanacions basant-se en la correspondència entre elements de la base de dades. En canvi, si existeix suficient informació dels usuaris, s'utilitzarà la filtració col·laborativa per obtenir elements d'usuaris similars i enriquir així la part basada en la cerca de contingut.

Aquesta és la idea embrionària però ja es poden començar a desvelar alguns detalls. El component basat en el contingut és bastant clar, utilitzarà la correspondència entre elements de la base de dades per tal d'obtenir els elements que permetran generar les recomanacions. Com s'acaba de comentar, cal sofisticar el sistema i aprofitar la potència del factor social. D'aquí neix la idea de crear una component social que utilitzi la correspondència entre els usuaris per generar recomanacions i enriquir el component basat en el contingut. Per realitzar aquest component social, s'ha decidit fer ús del raonament basat en casos (CBR).

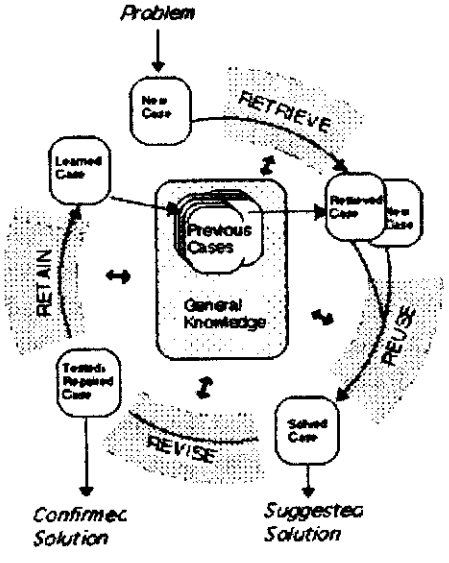


Figura 3.2: Cicle del CBR

Per tant, quan sorgeix un nou problema a resoldre, aquest és comparat amb els casos que trobem en la base de casos i llavors es recuperen un o més casos semblants. Una solució suggerida dintre els casos obtinguts és reutilitzada i testejada per tal de saber si és vàlida pel nou problema. Excepte que el cas recuperat sigui molt similar al problema, la solució molt possiblement hagi de ser revisada. Per últim, el nou cas solucionat és retingut a la base de casos.

Com utilitzar CBR en els sistemes de recomanació?

El CBR pot donar suport als sistemes de recomanació de moltes maneres diverses. La manera més simple consisteix en, donada una entrada configurada amb les preferències de l'usuari, el sistema retorna un conjunt d'elements que coincideixen amb aquestes preferències. En aquest procés es poden identificar tres fases: l'entrada de requeriments de l'usuari, la recuperació dels elements similars i la sortida amb les recomanacions per a l'usuari. Per exemple, una aplicació senzilla del CBR en la recomanació, és el cas d'un usuari que busca nous productes per comprar. L'usuari introdueix els requeriments sobre el producte i el sistema busca en la base de casos productes similars que encaixin amb els requeriments introduïts. El sistema retorna un conjunt de casos des de la base de casos i aquests són els que són susceptibles de ser retornats com a recomanació a

s'observa la importància que tindrà pel CBR les tècniques que s'utilitzin per mantenir el perfil de l'usuari, que s'explicaran en següents apartats.

3.2 Descripció del sistema de recomanació

En aquest apartat s'explicarà a fons el sistema de recomanació ideat. Primerament, s'explicarà com es construeix el perfil de l'usuari i seguidament s'argumentarà el motor de recomanació, que és la part més important. Finalment es completarà l'explicació amb les demés parts que conformen el sistema.

3.2.1 Perfil d'usuari

Un dels elements clau de l'èxit d'un sistema de CBR és la representació de la seva base de casos ja que els mecanismes de raonament que s'utilitzen a posteriori depenen directament d'aquesta estructura i, l'eficiència es pot veure perjudicada si no es té en compte la manera en com emmagatzemar aquesta informació o no s'emmagatzema la informació adequada.

En el sistema de recomanació que s'està estudiant,**la base de casos està formada per el conjunt de perfils d'usuari que interactuen amb el sistema**. Cada cas és doncs un perfil que conté dues parts, les dades socioeconòmiques de l'usuari i les seves preferències. Les preferències de l'usuari s'han d'entendre com un conjunt de llistes d'elements que ell ha introduït o que han derivat de recomanacions amb una resposta positiva i representativa. D'una manera més formal tenim que,

$$L = \{e_1, e_2, ..., e_n\}$$

on cada e_x correspon a un identificador d'un element en una taula de la base de dades.

Així doncs, les preferències d'usuari es poden expressar com un conjunt de llistes de la següent manera,

$$Pref = \{L_1, L_2, ..., L_m\}$$

En el cas que ens incumbeix, el de la música, la representació de les preferències de l'usuari està definida de la següent manera:

En aquest projecte s'ha considerat important afegir al perfil d'usuari tres extres per controlar el tipus de recomanacions que vol rebre l'usuari. Aquest, pot escollir entre rebre recomanacions amb contingut popular, és a dir, contingut molt votat per altres usuaris o bé contingut menys popular així com també pot escollir si centrar-se en poques recomanacions o si prefereix rebre un volum més gran d'aquestes. Finalment, si es decanta més per una època de la música que per una altra, pot escollir una franja musical.

Així doncs, un perfil de l'usuari acabarà tenint la següent forma,

$$P = \{DP, Pref, Extres\}$$

I en el cas que es tracta en aquest projecte tenim que el perfil de l'usuari es representarà com mostra la figura 3.4

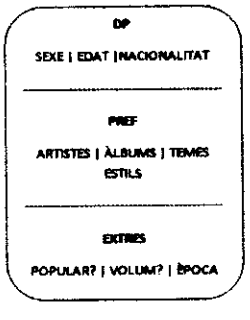


Figura 3.4: Perfil de l'usuari del sistema

3.2.2 Generació del perfil inicial

La decisió de com generar el perfil inicial molts cops pot venir determinada pels recursos que hom pot disposar. La manera més fàcil, directe i precisa és la generació manual per part de l'usuari introduint les seves preferències. Recordem que el problema que pot comportar aquest enfocament és la poca predisposició dels usuaris a gastar el seu temps proporcionant informació a un sistema.

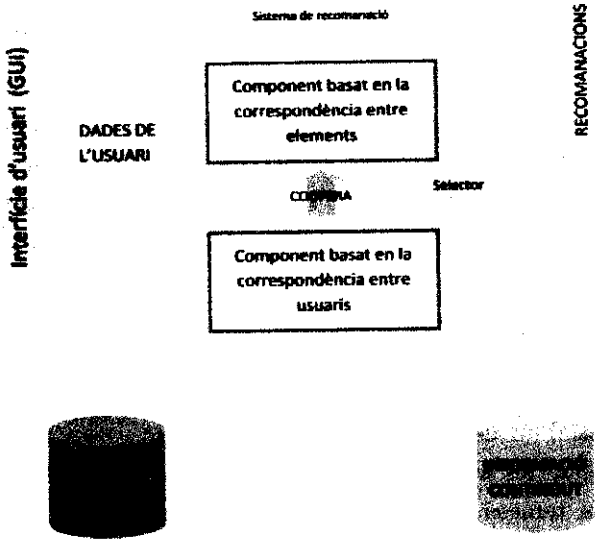


Figura 3.5: Esquema de funcionament del sistema

Aquesta figura pretén donar una visió simple de com funciona el sistema. Un usuari interacciona amb el sistema a partir d'una GUI, proporcionant informació sobre les seves preferències així com les seves dades personals. Aquesta informació és utilitzada per ambdues components per tal de trobar recomanacions per l'usuari. Si existeix un volum suficient de informació d'altres usuaris, el component de correspondència entre usuaris intervé en el component de correspondència entre elements, per tal de millorar els resultats d'aquest. Si no és així, la major part de la feina la realitza el component de correspondència entre elements. D'aquesta manera, si el sistema té un volum baix d'usuaris, es poden seguir donant recomanacions.

Un cop s'han obtingut un seguit de possibles recomanacions, un component selector segons els paràmetres extres proporcionats per l'usuari , acaba fent el treball per tal de retornar-li a l'usuari les recomanacions. L'usuari pot o no donar *feedback* al sistema

molt similars, possiblement els hi semblin bé el mateix tipus de recomanacions. A partir d'ara, el conjunt d'usuaris més similars a un altre usuari l'anomenarem "veïnat".

Reutilitzar: donats els casos recuperats i, per tant, les seves llistes de preferències i de recomanacions, es vol obtenir un subconjunt de preferències i/o recomanacions que puguin servir com a recomanacions per l'usuari nou. Al llarg d'aquest procés, es computa la popularitat dels elements i, d'aquesta fase es podria extreure una possible solució tenint en compte, per exemple, la popularitat. No obstant, recordem que el component de correspondència entre usuaris col·labora amb el de correspondència entre elements per enriquir així la possible solució.

Aquest són els quatre primers punts a considerar. El component basat en la correspondència entre usuaris treballarà amb el conjunt de perfils del usuaris emmagatzemats per tal d'aconseguir un veïnat pel nou problema o perfil d'usuari. Per tal de recuperar aquest veïnat, el component es basa en els llistats de preferències dels usuaris. Tot i que podria fer ús de tots els llistats per tal de dur a terme la seva funció (fase de recuperar), les hipòtesis detectades han estat les següents:

En qualsevol domini existeix un element que no és ni massa específic ni massa generic que el converteix en representatiu per determinar la similitud. Per exemple, en el cas de la música, observem que l'estil és molt genèric i poc representatiu. De grups d'estil rock n'existeixen milers però és ben diferent un grup de rock dels 60 que un d'actual. Per altra banda, els temes són elements massa específics i poc vinculants, et pot agradar la *Sonata Claro de Luna* de *Beethoven* però no implica que siguis amant de la música clàssica. En canvi, els artistes engloben un terme mig. Si t'agrada un artista, és probable que t'agradin artistes similars. Per tant, els artistes es converteixen en un element representatiu que, a més, ajuden a acotar la cerca.

Així doncs la hipòtesi (H1) plantejada és

H1 : La similitud entre usuaris es computa a partir de la seva llista de preferències d'artistes.

A continuació es descriu en llenguatge natural els passos que es realitzen en cada fase.

Fase RECUPERAR[1]

1. Obtenir els usuaris que tenen algun artista preferit igual als de l'usuari:

[1]Veure pseudocodi capítol disseny 4.4.3

Recuperar

Imaginem que un cop s'ha dut a terme el punt 1 obtenim els següents usuaris que tenen algun element en comú amb l'usuari U1, que a partir d'ara s'anomenarà simplement U1:

U2	{Home, 32, US}	Artistes: {1,2,3,7,8}	Popular/Poc contingut
		Àlbums: {$}	
		Estils: {A,C,D}	
		Cançons: {-}	
U3	{Dona, 24, FR}	Artistes: {1,4,7,9}	No popular/Poc contingut
		Àlbums: {$}	
		Estils: {C,D}	
		Cançons: {m}	
U4	{Dona,24, UK}	Artistes: {1,2,3,4,5,8}	Popular/Molt contingut
		Àlbums: {%,#}	
		Estils: {-}	
		Cançons: {l,j}	
U5	{Home, 21, FR}	Artistes: {8,9,10}	Popular/Molt contingut
		Àlbums: {@}	
		Estils: {A,B}	
		Cançons: {w,x}	

Taula 3.1: Exemple de configuració de 4 usuaris

Per la premissa del punt 1 tenim que U2 , U3 i U4 passen a ser candidats a veïns.

Seguidament, es computaria quin d'aquests usuaris formarà part del veïnat seguint les premisses del pas 2. Es calcula la similitud seguint la fórmula prèviament indicada.

$$sim(U1, U2) = \frac{4}{8} = 0'75$$
$$sim(U1, U3) = \frac{3}{8} = 0'375$$
$$sim(U1, U4) = \frac{5}{8} = 0'625$$
$$sim(U1, U5) = \frac{0}{10} = 0$$

Segons la hipòtesis H2, no es considera representatiu un usuari si no aconsegueix una

manera:

Pop.	Artistes	Pop.	Albums	Pop.	Estils	Pop.	Cançons
		3	5				
		1	%	3	C	1	L
		1	#	3	D	1	J
2	8						
2	9						
1	6						
1	10						
1	11						

Figura 3.8: Exemple computacio reutilitzar un cop s'han tingut en compte les recomanacions dels veïns

Les caselles grogues són les actualitzades mentre que també podem veure que certes recomanacions que a l'usuari U4 li van servir, a U1 no servirien perquè les té en les seves llistes de preferències.

Les recomanacions obtingudes que no estan en color gris podrien ser retornades com a possible solució si no es tingués en compte el component de correspondència entre elements. Com s'ha comentat, interessa que les dues parts cooperin i per aquest motiu, en aquest punt entre en joc el segon component d'aquest motor de recomanació.

Component basat en la correspondència entre elements

Component que utilitzarà el filtratge del contingut per tal de proporcionar recomanacions a l'usuari.

Aquest a diferència de l'anterior, no és conscient en cap moment de la informació proporcionada pels usuaris i les seves interaccions. L'objectiu d'aquest component és trobar recomanacions a partir de la relació que existeix entre els elements. Recuperem d'apartats anteriors la següent imatge:

No obstant, per realitzar aquest filtratge en la base de dades de música, ha calgut trobar l'element que vinculés cada un dels conjunts amb la resta de conjunts d'elements diferents. En aquest cas l'element connector ha estat l'estil. Per exemple, a partir de la llista d'artistes preferits, s'han extret els estils d'aquests artistes i, a partir d'aquests estils, s'ha filtrat la base de dades obtenint un subconjunt d'artistes, àlbums, temes i altres estils relacionats.

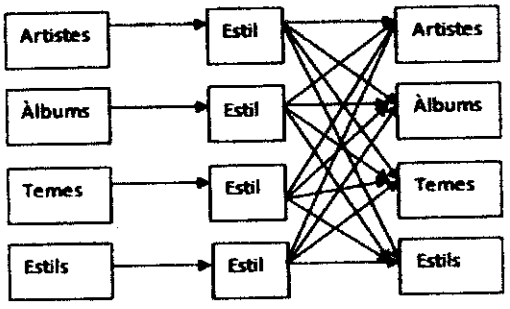


Figura 3.10: A partir de cada element, es busquen els estils i seguidament els altres elements

Coneixent ja els detalls de com realitza el filtratge aquest component de correspondencia entre elements, es detalla com en el cas anterior el seu comportament per passos:

1. A partir de les llistes de les preferències dels usuaris es busca la col·lecció d'estils que hi estan relacionats.
2. A partir del conjunt d'estils, computa quins són els estils més popular:
   a. Si existeix un veïnat, llavors afegeix els estils del veïnat i computa la popularitat
3. Filtratge de la base de dades a partir dels estils
4. Els llistats d'artistes, cançons, àlbums i estils obtinguts, es guarden com a possibles recomanacions.
   a. Si ja existeix perquè el component de correspondència entre usuaris també l'ha considerat, incrementa la popularitat.

A - 4 / B - 3 / C - 5 / D - 6 / E - 2

Per tant, en aquest cas, el estil més popular seria el D amb el qual es buscarien la resta d'artistes, àlbums, cançons i estils de la base de dades.

Si suposem que el resultat obtingut del filtratge de la base de dades a partir de D, és

Artistes: {3,5,7,9,11,12}
Àlbums: {$,%,&}
Estils: {A,C,B}
Cançons: {k,l,j}

El conjunt de possibles recomanacions un cop executat el motor de recomanació seria

Pop.	Artistes	Pop.	Àlbums	Pop.	Estils	Pop.	Cançons
		$	$				
		2	%	4	C	2	L
		1	#	4	D	2	J
		1	&				
						1	K
2	8						
3	9						
1	6						
1	10						
2	11						
1	12						

Un cop dut a terme el procés d'aquest component el component selector seria l'encarregat de decidir quines recomanacions retornar a l'usuari.

Component selector

En el moment en que s'han generat les recomanacions un component selector s'encarrega de decidir quines són les recomanacions que se li mostraran a aquest usuari.

Aquest component pot fer ús de diverses variables o criteris de filtratge els quals poden ser determinats directament per l'usuari o, per altra banda, poden ser producte d'una decisió de disseny del sistema de recomanació.

Retenció: Si l'usuari dóna *feedback* ja sigui positiu o negatiu, es passa a la última fase del cicle del CBR. Si la resposta de l'usuari ha estat positiva, aquest es guarda en la base de casos i podrà ser reutilitzat en posteriors problemes. En contra, si la resposta ha estat negativa, es descarta i no serà reutilitzada.

3.2.5 Manteniment del perfil

Per acabar l'explicació del sistema de recomanació, queda tenir en compte quina tècnica d'adaptació del perfil s'utilitza. Tal i com s'ha vist en el punt Tècniques d'adaptació del perfil (punt 2.2.5), és molt necessari mantenir el perfil amb la informació rellevant per tal de poder garantir que el perfil s'adapta a la situació actual de l'usuari.

Hem vist que es podia fer una adaptació manual, és a dir, que l'usuari sigui l'encarregat de modificar lliurement les seves preferències, esborrant i afegint de noves sempre que ho consideri necessari. Una altra opció consistiria en afegir informació de forma transparent a l'usuari, com pot ser el cas de *Last.fm*, que a partir de la música que l'usuari escolta, va actualitzant el seu perfil. El problema, és que d'aquesta manera, existeix informació nova però segueix existint informació antiga que potser ja no s'ha de tenir en compte. Aquest fet es podria corregir fent ús de funcions que esborrin la informació antiga, ja sigui de forma manual o de forma intrínseca.

En aquest projecte s'ha decidit fer ús de l'adaptació manual, per ser coherents amb la decisió de que la captació de les interaccions de l'usuari també s'ha fet de forma manual. Es vol aconseguir donar a l'usuari tot el control sobre el seu perfil.

D'aquesta manera, l'usuari podrà modificar les seves preferències lliurement. Les recomanacions generades sobre aquelles preferències s'esborraran cada cop que es torni a sol·licitar noves recomanacions excepte que l'usuari hagi proporcionat un *feedback*, llavors es mantindran en base de dades fins que aquest decideixi començar de zero i eliminar tot el seu perfil.

3.3 Resum de l'apartat

En aquest apartat tercer, s'ha detallat la conceptualització del sistema de recomanació híbrid que s'ha pensat en aquest projecte així com tots els elements que intervenen en el seu funcionament.

- Les dades personals de l'usuari s'han decidit no utilitzar per inferir en les recomanacions a diferència de les seves preferències que són la clau de l'èxit del recomanador.

- S'ha decidit fer ús d'un enfocament híbrid de recomanació, fent tant ús de la part social com de la relació entre el contingut. D'aquesta manera s'aconsegueix cobrir les mancances que cada part té per separat. Així, si existeix un nombre d'usuaris que ha interaccionat amb el sistema ambdues components cooperen i, si es dóna el cas que el sistema no té usuaris, pot seguir recomanant gràcies a la component basada en el contingut.

- Mantenir el perfil és important per tal de que l'usuari, si canvia de preferències, pugui seguir rebent recomanacions del seu gust. És per aquest motiu que s'ha donat tot la llibertat a l'usuari d'afegir i eliminar qualsevol de les seves preferències i recomanacions que hi hagi en sistema.

música perquè els usuaris puguin escoltar tant les seves preferències com les recomanacions que pugin rebre. També, l'usuari pot conèixer informació sobre els artistes sense haver de sortir de la web, oferint a més, un espai per intercanviar opinions amb els altres usuaris. La xarxa social resideix en el vincle entre usuaris que tenen preferències similars. Un usuari s'anomena veí quan té amb un altre una similitud característica. A través de la web, es pot visitar els perfils dels usuaris veïns per poder també establir conversacions o, simplement, per intentar conèixer altres preferències musicals d'una persona similar. No es pretén obtenir un espai social per crear relacions interpersonals entre usuaris, com altres xarxes socials del mercat ofereixen. Es vol mantenir el focus en la música, per aquest motiu no s'ha considerat adequat oferir funcionalitats com crear amistats, fer-se fan d'una banda, etc. S'ha volgut presentar la web com un espai de descobriment musical on poder compartir opinions amb altres persones interessades en la música.

Pel que fa al recomanador, detallat en el punt 3, sols es recordarà en aquest apartat que es tracta d'un sistema que pretén oferir recomanacions tan si existeix un volum d'usuaris important com si es tracta d'una web emergent amb pocs usuaris actius. Per aquest motiu, s'ha formulat un sistema que generi les recomanacions a partir de dues components, una social i una de contingut, que es complementen i fan que es pugui donar sempre un resultat al usuari. L'usuari pot configurar a través de la web, alguns paràmetres perquè les recomanacions s'ajustin més al que ell està buscant.

4.1.2 Oportunitat de negoci

Actualment, al mercat existeixen un nombre considerable de pàgines web d'*streaming* de música així com xarxes socials. No obstant, el seu model de negoci acostuma a estar centrat en la captació de molts usuaris per guanyar beneficis a través de la subscripció o la publicitat. Recmusic utilitza els serveis i APIs d'algunes d'aquestes pàgines que ofereixen als desenvolupadors. D'aquesta manera, si recmusic aconseguís tràfic es podria arribar a pactes amb les demés companyies ofertant-se com un servei complementari.

4.1.3 Anàlisis de requisits

A continuació es llisten el conjunt de requisits funcionals i no funcionals que s'han extret de la conceptualització del conjunt del sistema i dels objectius que es volien aconseguir.

2. El sistema tindrà accés a la base de dades de música de la qual podrà consultar informació però no modificar-la ni eliminar-la.

3. El sistema tindrà accés a la part relacionada amb els usuaris de la base de dades i la podrà consultar i modificar.

4. El sistema ha de ser capaç de generar recomanacions a partir de la informació de la base de dades.

5. El sistema ha de ser capaç de mostrar a un usuari altres usuaris similars a ell.

6. Les recomanacions es realitzaran sota demanda de l'usuari.

7. El sistema emmagatzemarà les recomanacions realitzades als usuaris per tal de tenir un historial de les recomanacions fetes.

8. El sistema ha de ser capaç de mostrar les recomanacions generades al usuari corresponent.

9. El sistema ha de permetre que les recomanacions mostrades a l'usuari siguin votades en un dels següents estats: recomanació encertada o recomanació rebutjada.

10. L'usuari ha de poder resetejar les seves recomanacions guardades.

Altres serveis

1. L'usuari ha de poder escoltar els artistes, àlbums i cançons presents en el sistema.

2. L'usuari ha de poder consultar biografies dels artistes presents en el sistema.

3. L'usuari ha de poder visitar els perfils d'usuaris similars.

4. L'usuari ha de poder deixar comentaris a altres usuaris.

5. L'usuari ha de poder deixar comentaris a les biografies dels artistes.

Requisits no funcionals

1. La manera d'introduir les preferències del usuaris ha de ser molt fàcil, en un màxim de tres passos.

- **Codi lliure.**

  Tan sols per tractar-se d'un projecte de final de carrera sense capital invertit, la opció d'utilitzar codi lliure ha estat la millor opció. No obstant, les raons per fer ús del software, *frameworks* i llibreries de codi lliure van més enllà de la simple evidència. S'ha de tenir en compte que algunes de les eines més robustes del mercat són de codi lliure així com també, que algunes de les empreses més influents del sector d'Internet estan darrere de totes aquestes eines, fet que convida a fer-ne ús.

- **Un trencaclosques de tecnologies punteres.**

  Qui s'embarca en el món del desenvolupament web es troba amb un ventall de possibilitats enorme de tecnologies a utilitzar. Per prendre les decisions tecnològiques d'aquest pilot, s'ha valorat notòriament que les tecnologies utilitzades fossin punteres, intentant construir l'aplicació modularment, com un trencaclosques on anar encaixant peces amb l'objectiu de millorar el resultat final.

4.2.2 Decisions preses

Un cop plantejada la filosofia utilitzada, es poden presentar ja les tecnologies emprades pel desenvolupament del pilot així com els motius que han portat a utilitzar-les.

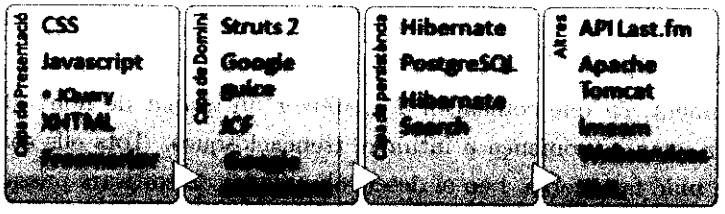


Figura 4.1: Resum de tecnologies utilitzades

Ús de tecnologies Java

Com que es volia treballar amb el patró de disseny d'orientació a objectes (OO)[1], el llenguatge de programació per antonomàsia d' aquest patró que s'adapta millor al desenvolupament web és el Java. Tot i així, no ha estat l'únic motiu per escollir Java davant altres llenguatges populars en la web com el Ruby o el PHP , que també té OO

[1]Veure patrons de disseny, apartat 4.4.1

- **Struts 2**

  Apache Struts 2 és el *framework* més nou i punter del ventall enumerat anteriorment. No és tan sols una nova versió del conegut Struts sinó un *framework* completament nou basat en el framework WebWork de OpenSymphony. Cobreix perfectament les necessitats de domini i, a part, aporta més valor en la part d'interfície d'usuari (UI) que el seu competidor més proper, Spring MVC[DB08].

  Fent una cerca a Google Trends , es pot veure clarament la creixent evolució de llocs web que parlen sobre Struts 2, fet que també denota que, tot i ser un *framework* bastant nou, ha tingut una acceptació important.

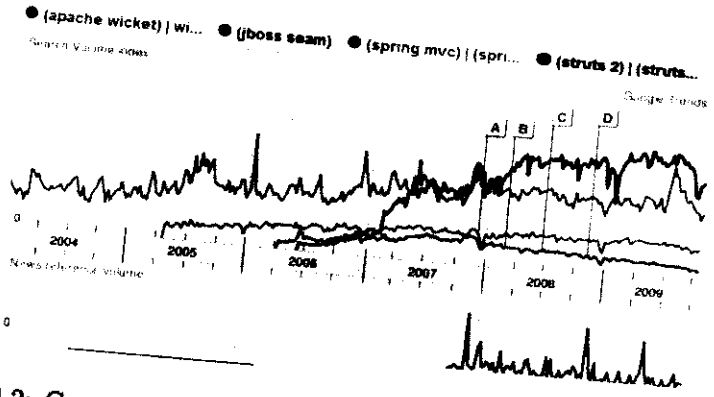


Figura 4.2: Comparativa dels frameworks java a partir de Google Trends

- **Hibernate**[2]

  Hibernate és una potent eina de gestió de persistència que permet mapejar el model de domini orientat a objectes a una base de dades relacional. Com a *framework*, té nombrosos avantatges on cal destacar la sorprenent senzillesa d'ús un cop configurat, tant en el mapeig com en la portabilitat. Amb pocs canvis en el fitxer de configuració d'Hibernate, es pot migrar ràpidament una base de dades.[CB07]

  Aquest *framework* ha estat una decisió important perquè ha facilitat molt el tractament del model de dades. Al dissenyar un sistema que ha depès d'una estructura establerta per la base de dades de *MusicBrainz*, ha sigut molt còmode treballar amb Hibernate per la rapidesa en el desplegament dels canvis del model de domini a la base de dades. També, ha sigut un encert per les opcions de consulta

[2]Veure més en apartat de Disseny, punt 4.4.2

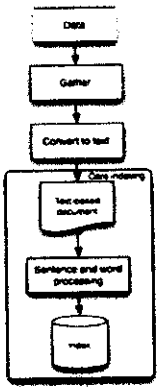


Figura 4.4: Components d'Hibernate Search

Bàsicament, el que proporciona és una eina d'indexació a mida d'atributs i entitats de la capa de domini, aconseguint una gran millora en el temps de resposta de les cerques.

Amb l'ús d'Hibernate Search, s'ha aconseguit millorar el cercador de música, permeten una cerca molt més eficient i un codi molt més net.

- **Google Guice**

  Google Guice, tot i no ser pròpiament una tecnologia Java, hi està estretament relacionada i és per aquest motiu que s'inclou en aquest apartat.

  Es tracta d'un framework creat per Google que implementa el patró d'Injecció de dependències (DI) . Fent ús d'aquest framework, s'ha pogut treure avantatge de totes les facilitats que dóna el patró DI [3]:

  a. Desaparició de codi repetit

  b. Millora en la concurrència

  c. Cessió de la responsabilitat de buscar els objectes necessaris al framework Google Guice, podent així centrar més esforços de nou en tasques de no tant baix nivell.[GG:b]

[3]Veure patró de disseny Injecció de Dependències (DI), apartat 4.4.1

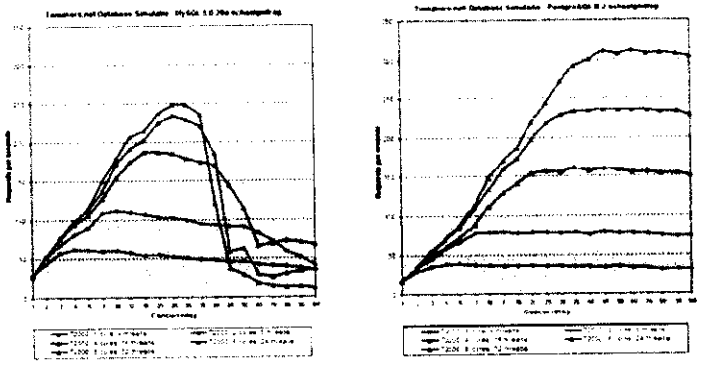


Figura 4.5: Comparativa entre els SGBDs PostgresSQL i MySQL.

Capa de presentació: centrar esforços en la reutilització

Les tecnologies de la capa de presentació, són també una part important del sistema. Una aplicació web pot tenir moltes vistes i, per tant, un desenvolupament darrere que pot arribar a ser molt costós. No obstant, actualment existeixen tecnologies pensades per centralitzar necessitats de *look & feel* , estructura i comunicació amb la capa de domini que agilitzen molt el procés del desenvolupament web. A continuació s'expliquen les escollides.

- **Estils: CSS**

  Es tracta de l'acrònim de *Cascading Style Sheets*, és a dir, fulles d'estil en cascada. Aquest llenguatge s'utilitza per definir el *look & feel* de documents HTML, XML o d'altres llenguatges de marcatge i està regulat per la W3C[w3c] .

  El CSS és la millor opció actualment per la maquetació web pels següents motius:

  - Permet la separació del contingut del document de la capa de presentació.
  - Dóna més flexibilitat i control en l'especificació de la capa de presentació.
  - Permet que diferents pàgines comparteixin el mateix format i redueix la complexitat i repetició del contingut estructurat.

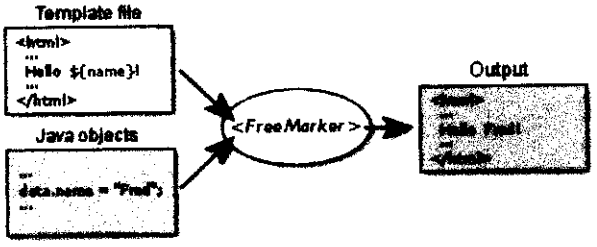


Figura 4.6: Esquema de funcionament de Freemarker

- **Codi client: Javascript / JQuery**

  JavaScript és una tecnologia web que s'executa en el navegador de l'usuari i pretén controlar certs aspectes de la pàgina que l'usuari està visualitzant. Fent ús no accessiu de JavaScript s'aconsegueix que les pàgines siguin més interactives, intuïtives i atractives per l'usuari.

  Per tal de treballar més còmodament amb JavaScript, s'ha decidit escollir de nou un framework amb la finalitat d'agilitzar el desenvolupament. Un framework JavaScript, és bàsicament un fitxer js de poca grandària - al voltant de 30KB - que es descarrega a la màquina de l'usuari com qualsevol JavaScript comú.

  De frameworks de JavaScript n'hi ha molts, però malgrat que ofereixen funcionalitats i característiques semblants, varien considerablement en eficiència, portabilitat entre navegadors i facilitat d'ús.

  De tots els disponibles, s'ha valorat ter us o bé del Dojo o bé del JQuery. Finalment, el que més ha destacat per la seva eficiència és JQuery motiu pel qual és el framework Javascript de l'aplicació desenvolupada.

  JQuery

  A part de destacar per la seva portabilitat i per la seva velocitat, destaca també per la seva facilitat d'ús i perquè es tracta d'un framework fàcilment escalable amb plugins que estenen la seva funcionalitat.[Coma]

  JQuery (versió 1.3) per defecte, incorpora les següents funcionalitats bàsiques:

  - Creació, eliminació, modificació i cerca d'elements HTML i de les seves propietats.

l'usuari pugui escoltar la música que ha escollit com a preferida i la que se li pugui recomanar. Per altra banda, els serveis web s'utilitzaran per recuperar les URL's de les cançons a reproduir en el reproductor i la biografia dels artistes.

Els serveis web de *Imeem* també són de tipus REST, fet que serà necessari tractar la resposta que el sistema rebrà via XML. Per aquest motiu, el sistema també implementarà un *parser* de XML.

manació donant una validació que pot ser OK si li ha agradat, KO si no li ha agradat o bé deixar-la buida (EMPTY). A part, cada usuari té uns usuaris veïns, que són els que s'assemblen més a ell.

Per últim, l'usuari pot deixar comentaris a altres usuaris o deixar opinions sobre artistes concrets. El sistema enregistra aquestes opinions i comentaris i la seva data corresponent.

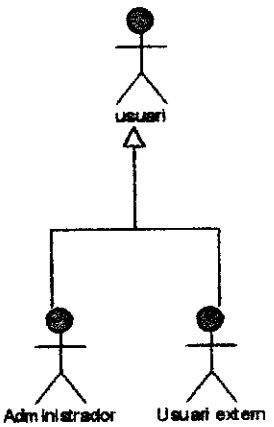
4.3.2 Model de casos d'ús

En el model de casos d'ús es pretén organitzar els requeriments segons les funcions que el sistema ha de realitzar, delimitant així l'abast del sistema.

Un cas d'ús és un document que descriu una seqüència d'esdeveniments que realitza un actor o agent extern que utilitza el sistema per dur a terme algun procés que té algun valor per a ell. Per aquest motiu, primerament es determinen els actors i seguidament el conjunt de casos d'ús que aquests realitzaran.

Actors

En el sistema que s'està descrivint existeixen dos actors, l'usuari i l'administrador. Tant l'usuari com l'administrador poden interaccionar de la mateixa manera amb el sistema. La diferència bàsica és que l'administrador tindrà accés a una part de *back-office* per tal de poder gestionar aspectes de la plataforma. Per altra banda, qualsevol persona que vulgui interaccionar amb la plataforma, se'l considera dintre el grup usuari ja que tots podran dur a terme les mateixes funcionalitats.



En aquest apartat es detallen els diagrames de casos d'ús junt al seu detall. S'ha dividit l'explicació aproximadament per blocs de funcionalitats que es realitzen en espais similars.

En el primer bloc trobem les funcionalitats bàsiques per accedir o sortir a l'aplicació.

4a. Les dades introduïdes per l'usuari no són correctes:

4a1. El sistema mostra de nou la pantalla de login i indica els errors a l'usuari

4a2. L'usuari introdueix de nou els camps erronis

4a3. El cas d'ús segueix en el punt 4

2. Tancar sessió

Actor: Usuari

Precondició: L'usuari ha d'haver iniciat sessió al sistema

Descripció: L'usuari vol tancar sessió en el sistema

Curs típic d'esdeveniments

1. L'usuari prem el botó de sortir del sistema
2. El sistema desconnecta l'usuari del sistema
3. El sistema esborra la sessió

3. Alta Usuari

Actor: Usuari

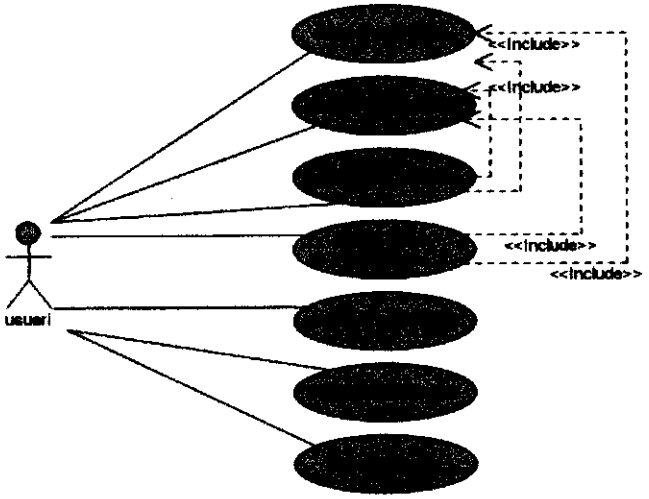
Precondició: -

Descripció: L'usuari vol donar-se d'alta al sistema

Curs típic d'esdeveniments

1. L'usuari prem el botó de registrar
2. El sistema mostra un formulari amb els camps nom d'usuari, correu electrònic i contrasenya

A banda, per aprofitar la informació existent en altres sistemes i per facilitar així la introducció de preferències, s'ha pensat en donar la opció d'importar el perfil des de una altra aplicació que també treballi sota el domini de la música. En aquest cas, s'ha fet ús de *Last.fm*.



5. Afegir Preferència

Actor: Usuari

Precondició: L'usuari ha d'haver iniciat sessió al sistema

Descripció: : L'usuari vol afegir una preferència al sistema

Curs típic d'esdeveniments

1. L'usuari escull un grup, cançó, àlbum o estil a afegir
2. El sistema afegeix a la llista de preferències de l'usuari el nou element

8. Editar Altres Preferències

Actor: Usuari

Precondició: L'usuari ha d'haver iniciat sessió al sistema

Descripció: L'usuari vol modificar les seves preferències

Curs típic d'esdeveniments

1. L'usuari escull la opció d'editar un artista, àlbum o cançó preferits
2. El sistema mostra un buscador i les seves preferències
3. L'usuari vol eliminar un preferència i la selecciona
4. **Eliminar Preferència**

   Extensions:

   2a. L'usuari vol afegir una nova preferència:

   2a1. L'usuari introdueix en el camp de cerca la música que vol buscar i prem buscar

   2a2. El sistema busca la informació proporcionada per l'usuari

   2a3. El sistema mostra el llistat de coincidències

   2a4. **Afegir Preferència**

9. Mostrar Preferències

Actor: Usuari

Precondició: L'usuari ha d'haver iniciat sessió al sistema

Descripció: L'usuari vol veure les seves preferències

Curs típic d'esdeveniments

1. L'usuari prem el link de mostrar preferències

2a. El nom d'usuari no existeix o és incorrecte:

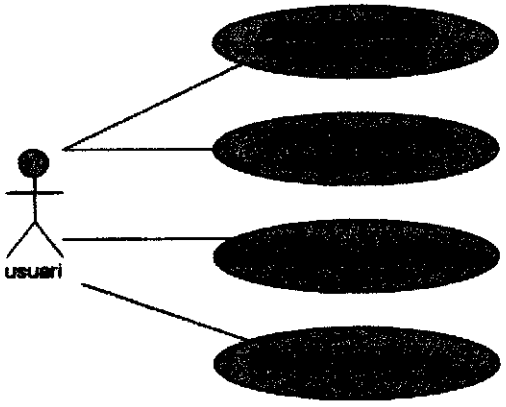
2a1. L'usuari torna a introduir el nom d'usuari

2a2. El cas d'ús segueix en el punt 2

3a. Es produeix un error durant el procés d'importació:

3a1. El sistema mostra a l'usuari un error

El tercer bloc, comprèn tot aquelles funcionalitats relacionades en modificar el perfil personal de l'usuari així com els paràmetres del recomanador que faran que les recomanacions obtingudes siguin unes o altres.



12. Modificar Informació Personal

Actor: Usuari

Precondició: L'usuari ha d'haver iniciat sessió al sistema

Descripció: L'usuari vol modificar la seva informació personal

Curs típic d'esdeveniments:

1. L'usuari indica que vol modificar la seva informació personal

14. Canviar Preferències Recomanador

Actor: Usuari

Precondició: L'usuari ha d'haver iniciat sessió al sistema

Descripció: L'usuari vol canviar les seves preferències de recomanació

Curs típic d'esdeveniments:

1. L'usuari selecciona la quantitat de recomanacions que vol rebre
2. L'usuari selecciona la popularitat de les recomanacions que vol rebre
3. L'usuari selecciona l'època de les recomanacions que vol rebre
4. El sistema enregistra les preferències

15. Eliminar Historial Recomanacions

Actor: Usuari

Precondició: L'usuari ha d'haver iniciat sessió al sistema

Descripció: L'usuari vol eliminar el seu historial de recomanacions

Curs típic d'esdeveniments:

1. L'usuari escull eliminar l'historial
2. El sistema elimina l'historial de recomanacions

I per últim, tenim el bloc relacionat amb les funcionalitats més importants del sistema i que li donen més valor. Es tracten de la recomanació, l'obtenció d'informació d'artistes, la capacitat d'opinar sobre gustos musicals o artistes i la possibilitat d'escoltar la música relacionada amb el perfil de l'usuari, ja siguin recomanacions o preferències.

17. Mostrar Biografia

Actor: Usuari

Precondició: L'usuari ha d'haver iniciat sessió al sistema

Descripció: L'usuari vol veure la biografia d'un artista

Curs típic d'esdeveniments:

1. L'usuari indica que vol veure una biografia d'un artista
2. El sistema recupera la biografia
3. El sistema recupera els comentaris que altres usuaris han deixat sobre l'artista
4. El sistema mostra a l'usuari la informació recuperada

18. Escoltar Música

Actor: Usuari

Precondició: L'usuari ha d'haver iniciat sessió al sistema

Descripció: L'usuari vol escoltar música

Curs típic d'esdeveniments:

1. L'usuari indica que vol escoltar música de les seves preferències o recomanacions
2. El sistema busca la música demanada per l'usuari
3. El sistema reprodueix la música demanada

   Extensions:

   2a. La música demanada no existeix en el sistema:

   i. El sistema mostra un missatge a l'usuari

Descripció: L'usuari vol consultar les seves recomanacions

Curs típic d'esdeveniments:

1. L'usuari indica que vol veure les seves recomanacions
2. El sistema mostra les recomanacions que encara no han estat votades

22. Mostrar Recomanacions bones

Actor: Usuari

Precondició: L'usuari ha d'haver iniciat sessió al sistema

Descripció: L'usuari vol consultar les seves recomanacions que li han agradat

Curs típic d'esdeveniments:

1. L'usuari indica que vol veure quines són les seves recomanacions que li han agradat
2. El sistema mostra les recomanacions que han estat votades com a OK

23. Mostrar Recomanacions no bones

Actor: Usuari

Precondició: L'usuari ha d'haver iniciat sessió al sistema

Descripció: L'usuari vol veure les seves recomanacions que no li han agradat

Curs típic d'esdeveniments:

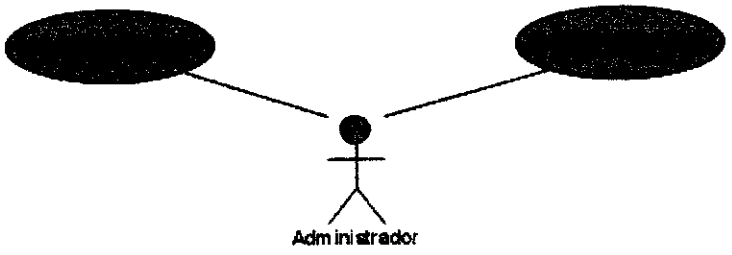
1. L'usuari indica que vol veure les seves recomanacions que no li han agradat
2. El sistema mostra les recomanacions votades com a KO

Descripció: L'usuari vol marcar un comentari com a abús

Curs típic d'esdeveniments:

1. L'usuari marca un comentari com a ofensiu
2. El sistema enregistra que el comentari no es adequat
3. El sistema avisa al administrador

Pel que fa a les funcionalitats que només podrá dur a terme l'administrador tenim les següents:



27. Eliminar Comentari

Actor: Administrador

Precondició: L'administrador ha d'haver fet login al sistema

Descripció: L'administrador vol eliminar un comentari d'un usuari

Curs típic d'esdeveniments:

1. L'administrador indica quin és el comentari a esborrar
2. El sistema elimina el comentari

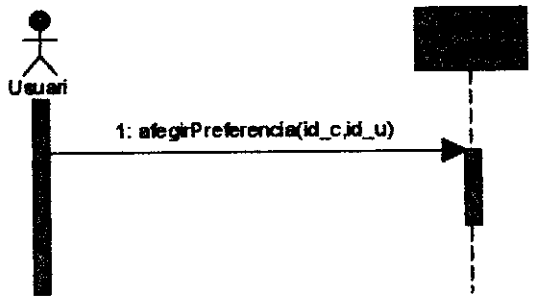
Nom	iniciarSessio(username,pwd) : session
Responsabilitat	Crear una sessió per l'usuari que vol entrar al sistema
Excepcions	L'usuari no introdueix correctament el seu nom d'usuari o contrasenya
Precondició	-
Postcondició	S'ha creat una nova sessió per l'usuari que ha entrat al sistema
Sortida	sessió nova, error si l'usuari s'ha equivocat

Tancar Sessió

Nom	tancarSessio()
Responsabilitat	Eliminar la sessió actual de l'usuari
Excepcions	-
Precondició	Ha d'existir una sessió per l'usuari
Postcondició	S'ha eliminat la sessió de l'usuari
Sortida	-

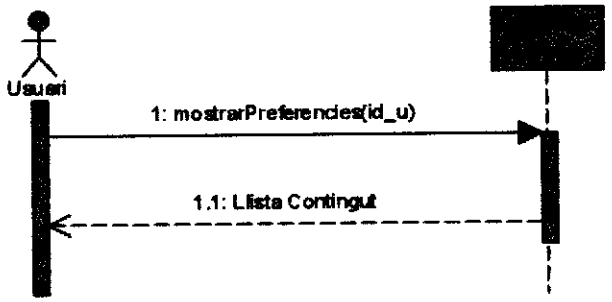
Nom	baixaUsuari(id_usuari)
Responsabilitat	Eliminar del sistema la instància d'Usuari amb id=id_usuari. Eliminar associacions d'aquesta instància.
Excepcions	-
Precondició	L'usuari u ha iniciat sessió en el sistema
Postcondició	S'ha eliminat la instància u d'Usuari amb id=id_usuari i les seves associacions amb les classes Contingut, Recomanacio, Artista i Comentari
Sortida	-

Afegir Preferència



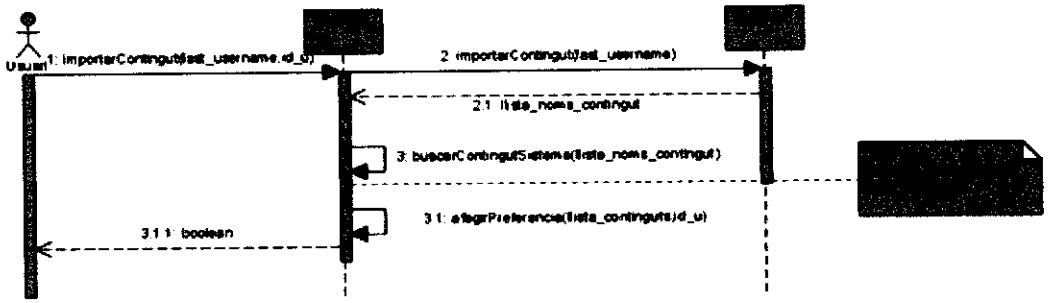
Nom	afegirPreferencia(id_c, id_u)
Responsabilitat	Iniciar l'enregistrament d'una preferència de l'usuari
Excepcions	La perferència amb id_c ja està enregistrada
Precondició	L'usuari u ha iniciat sessió en el sistema. La preferència amb id_c no està enregistrada com a preferència de u
Postcondició	S'ha creat una nova instància de l'associació 'prefereix' que associa l'Usuari u amb id=id_u i la instància de Contingut c amb id=id_c
Sortida	-

Mostrar Preferències / Ocultar Preferències



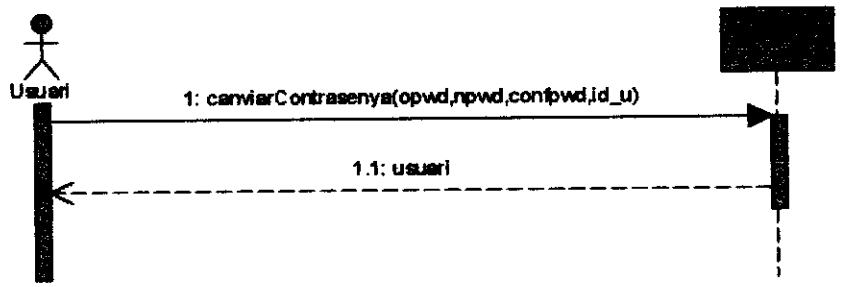
Nom	mostrarPreferencies(id_u) / ocultarPreferències(id_u)
Responsabilitat	Mostrar/Ocultar les instàncies de Contingut relacionades amb l'Usuari u amb id=id_u
Excepcions	-
Precondició	L'usuari u ha iniciat sessió al sistema. Existeixen instàncies de la relació "prefereix"entre l'Usuari u i la classe Contingut.
Postcondició	El sistema activa un flag si es mostren les preferències, es desactiva el flag si s'oculten
Sortida	Mostra/Oculta el llistat de les preferències de u

Importar Perfil



Nom	modificarInfoPersonal(nsex,nbirth,nname,nfoto,ncountry,id_u)
Responsabilitat	Canviar els atributs data_naix, pais, nom, sexe i foto de l'Usuari u amb id=id_u
Excepcions	Els paràmetres nsex, nbirth, nfoto i ncountry no tenen el format correcte
Precondició	L'usuari u ha iniciat sessió en el sistema. Els paràmetres nsex, nbirth, nfoto i ncountry tenen el format correcte
Postcondició	L'usuari u ha estat modificat, data_naix=nbirth , pais=ncountry, nom=nname, sexe=nsexe i foto=nfoto
Sortida	L'usuari amb els atributs canviats

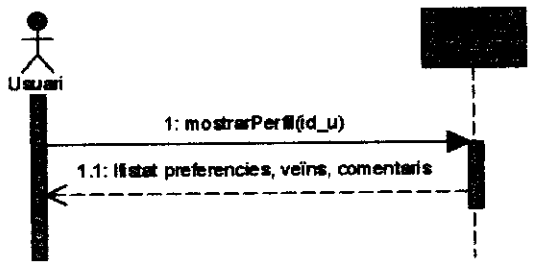
Canviar Contrassenya



Nom	canviarContrasenya(opwd,npwd,confpwd,id_u)
Responsabilitat	Modificar l'atribut pwd de la instància d'Usuari u amb id=id_u
Excepcions	opwd no és correcte o bé npwd <> confpwd
Precondició	L'usuari u ha iniciat sessió en el sistema. Els paràmetres opwd, npwd i confpwd són correctes
Postcondició	L'usuari u ha estat modificat, pwd=npwd
Sortida	L'usuari amb l'atribut canviat

Nom	eliminarHistorial(id_u)
Responsabilitat	Eliminar les instàncies de l'associació "rep"entre la instància d'Usuari u amb id=id_u i les de la classe Recomanacio on validacio<>EMPTY
Excepcions	-
Precondició	L'usuari ha iniciat sessió al sistema
Postcondició	S'han eliminat les instàncies de l'associació "rep"entre u i les instàncies de Recomanació amb validacio<>EMPTY
Sortida	Booleà confirmant l'eliminació

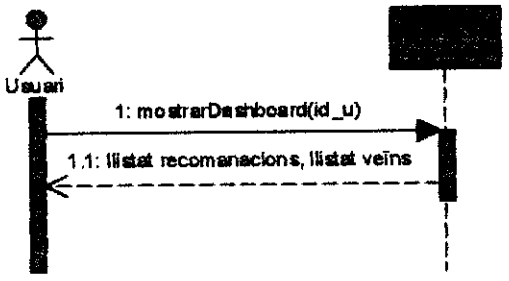
Mostrar Perfil



Nom	mostrarPerfil(id_u)
Responsabilitat	Mostrar les preferències de l'Usuari u amb id=id_u, els comentaris que altres usuaris han deixat a u i la seva informació personal
Excepcions	-
Precondició	L'usuari ha iniciat sessió al sistema
Postcondició	-
Sortida	Llistat de les preferències, dels comentaris i de la informació personal

Nom	escoltarMusica(nnom)
Responsabilitat	Obtenir la url de la cançó amb nom=nnom o cançons relacionades amb l'artista amb nom=nnom o àlbum amb nom=nnom i reproduir-ia
Excepcions	No es troben urls per el paràmetre nnom
Precondició	L'usuari ha iniciat sessió i ha pulsat el botó *play*
Postcondició	-
Sortida	Url a reproduir

Mostrar Dashboard

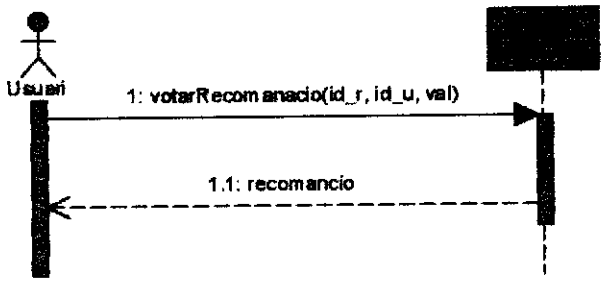


Nom	mostrarDashboard(id_u)
Responsabilitat	Mostar a l'Usuari u amb id=id_u la pantalla dashboard amb el seu llistat de recomanacions de artistes, àlbums, cançons, estils i veïns
Excepcions	-
Precondició	L'usuari u ha iniciat sessió en el sistema
Postcondició	-
Sortida	Llistats de recomanacions i veïns

Nom	mostrarRecomanacions(id_u)
Responsabilitat	Mostar un llistat de les recomanacions de l'Usuari u amb id=id_u i validacio = EMPTY
Excepcions	-
Precondició	L'usuari u ha iniciat sessió en el sistema.
Postcondició	-
Sortida	Llistat de recomanacions

Nota: *Mostrar Recomanacions bones i Mostrar Recomanacions no bones actuen de manera similar tenint en compte el parametre validacio, que en el primer cas sera validacio=OK i en el segon validacio=KO*

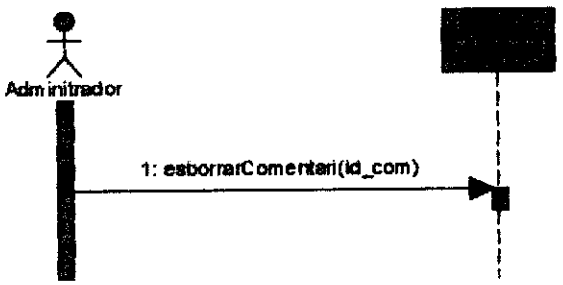
Votar Recomanació



Nom	votarRecomanacio(id_r, id_u, val)
Responsabilitat	Modificar l'atribut validacio de l'instància de Recomanació r amb id=id_r
Excepcions	-
Precondició	L'usuari ha iniciat sessio en el sistema
Postcondició	L'atribut validacio de r passa a tenir el valor validacio=val
Sortida	Recomanció r modificada

Nom	marcarAbus(id_c)
Responsabilitat	Modificar la instància de Comentari c amb id=id_c i canviar l'estat a estat=ABUSE
Excepcions	-
Precondició	L'usuari ha iniciat sessió al sistema
Postcondició	L'instància de Comentari c amb id=id_c passa a tenir un estat=ABUSE
Sortida	Comentari c

Eliminar Comentari



Nom	eliminarComentari(id_com)
Responsabilitat	Eliminar del sistema la instància de Comentari c amb id=id_com
Excepcions	-
Precondició	L'usuari que dur a terme l'acció és l'administrador i ha d'haver iniciat sessió
Postcondició	S'ha esborrat la instància de Comentari c
Sortida	-

4.4 Disseny

La fase de disseny és l'activitat d'aplicar diferents tècniques i principis amb el propòsit de definir un sistema amb el suficient detall per permetre la seva construcció física, és a dir, la seva implementació. [Pen03a]

El disseny parteix dels capítols anteriors, especialment de l'especificació i de les tecnologies que s'ha decidit utilitzar. En aquest apartat, es desenvoluparà l'arquitectura del sistema definint quins patrons arquitectònics i de disseny s'utilitzaran per realitzar la implementació del sistema final. No obstant, s'ha de tenir en compte que la majoria de patrons de disseny utilitzats han vingut marcats per la decisio d'emprar certes tecnologies explicades en el punt 4.2, ja que aquestes tecnologies implementen aquests patrons.

4.4.1 Decisions preses

Abans de determinar les decisions preses cal veure els requisits de disseny que s'han tingut en compte:

- Es vol permetre que el sistema sigui capaç d'atendre peticions simultànies independentment de la càrrega que tingui en aquell moment el servidor.
- Cal garantir un disseny modular per tal de facilitar la construcció i manteniment del sistema.
- El sistema ha de ser fàcilment escalable especialment per garantir que en un futur estigui preparat per un gran volum d'usuaris i pugui existir així distribució de càrrega entre servidors.

Per aquests motius, s'ha decidit utilitzar els patrons que a continuació es descriuen i plantejar el disseny del sistema dividint el treball a fer en components independents ,per garantir així part dels punts acabats de citar.

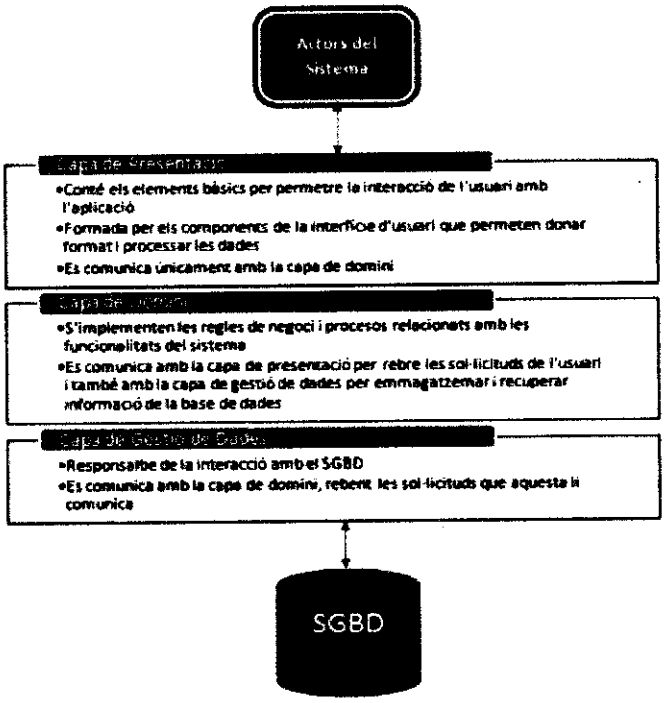


Figura 4.7: Arquitectura en tres capes junt amb les seves responsabilitats

Orientació a objectes

El patró arquitectònic d'orientació a objectes consisteix en veure el sistema com una col·lecció d'objectes que, en resposta a certs estímuls o esdeveniments interns, intercanvien informació, canvien d'estat i eventualment produeixen resultats observables.[AA02, Ten03b]

La orientació a objectes ofereix els següents avantatges:

- Permet que els canvis en el codi no es propaguin a tot el sistema.
- Permet la reutilització i substitució de components per implementacions alternatives.
- Permet l'agrupació de responsabilitats similars que afavoreixen la comprensió i el manteniment.
- Ofereix més facilitats en la portabilitat a altres plataformes.

- Fàcil comprensió: la separació sempre aporta que sigui més fàcil comprendre el funcionament del sistema fins i tot per persones que no l'han desenvolupat.

Com ja s'ha vist en l'apartat de tecnologies utilitzades (apartat 4.2), s'utilitzara Struts 2 i l'arquitectura bàsica d'aquest *framework* implementa aquest patró.

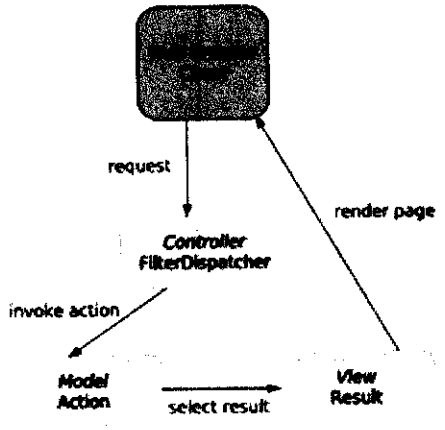


Figura 4.8: Struts 2, que implementa el patró MVC

Patró Dependendy Injection (DI)

Dependency Injection (DI) o Injecció de Dependencies és un forma específica de inversió de control on, en el cas d'aquest patró, el procés que s'inverteix és el d'obtenir les dependències entre objectes o components.

Si es fa us d'aquest patró s'entenen els objectes com a serveis i clients , on un servei es defineix com un objecte que porta a terme una funció coneguda i un client es defineix com un objecte que crida a un servei per executar una funció. En altres paraules, qualsevol consumidor d'un servei és un client.

Basicament, es delega la responsabilitat de crear objectes i les seves dependencies a un altre component, en aquest cas una llibreria externa que és Google Guice.

Alternatives a aquest patró podrien ser:

- La construcció manual, donant a la constructora de cada servei la responsabilitat

Un usuari interacciona amb la capa de domini a traves de les vistes generades amb Freemarker. Tant les vistes, com el controlador o FilterDispatcher i els Actions formen part de l'arquitectura interna de Struts 2. Per tal de comprendre millor com funciona aquest bloc de l'arquitectura, anem a resumir el funcionament de Struts 2. [Blo08, DB08]

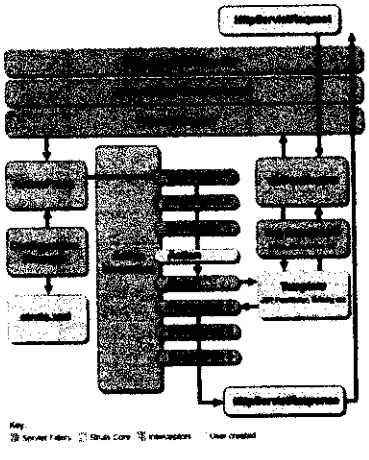


Figura 4.9: Arquitectura interna Struts 2

En el diagrama de l'esquerra, es pot observar que la petició inicial va a parar al contenidor de Servlets i passa per una cadena de filtres. Seguidament, es crida al controlador (FilterDispatcher) que s'encarrega de consultar a l'ActionMapper si la petició ha d'invocar un action. Si l'ActionMapper determina que s'ha d'invocar l'Action, el FilterDispatcher delega al ActionProxy el control. El ActionProxy, consulta els fitxers de configuració de Struts 2 [8] i invoca l'Action que s'ha d'executar, passant abans pels Interceptors. La funció del Interceptor es aglutinar responsabilitats que els Actions comparteixen, per exemple, que abans d'executar-se un action l'usuari hagi iniciat sessió. Abans que l'Action mapejada s'executi, serà interceptada i, si passa els filtres dels interceptors, llavors es podrà executar. Es pot entendre un Action com un cas d'ús del sistema.

[8] Veure fitxers de configuració en apartat de Implementació, punt 4.5.3

incorrectes, portar-lo a una pàgina d'error. Seguidament, el resultat es executat i es mostra de nou en les plantilles, en aquest cas de Freemarker.

Qualsevol Business Object i els seus atributs, son coneguts a la capa de presentacio. Així doncs, si a l'Action que s'ha executat tenim una instància de la classe Usuari anomenada u, des de les vistes podrem fer us d'aquesta instancia. Utilitzant `${u.nom}` obtindriem, per exemple, el nom. Per altra banda, per l'entrada de dades, Struts 2 també ofereix tags especials que es poden utilitzar en Freemarker.[str]

Els Services son interficies que contenen operacions que han de realitzar els Actions. En aquest projecte, els Services s'implementen com a classes que s'encarreguen de comunicar-se amb Hibernate per realitzar consultes i modificacions a la base de dades i a l'índex. Tot i així, al ser interfícies, podrien tenir diferents implementacions i aquest fet, també dóna més flexibilitat al sistema.

Com s'ha anat observant, s'ha intentat al maxim construir el sistema de forma modular, per tal de que sigui un sistema escalable, reutilitzable i fàcil de mantenir. Es per aquest motiu que, a part dels Actions, Services i demés elements que formen part del sistema, també existeixen tres components més relacionats amb les tasques de recomanació i la gestió dels serveis d'Imeem.

El component SocialSeeker, dur a terme totes les operacions relacionades amb la recomanacio basada amb la relacio entre els usuaris. Per altra banda, el component ContentSeeker, dur a terme totes les operacions restants de la recomanació basada en la relacio entre elements de la base de dades. Finalment, el component Audio gestiona tota la recuperació de la informació del webservice de Imeem, recuperant bàsicament urls amb les cançons a reproduir o amb informacio a mostrar. D'aquesta manera, s'aconsegueix que si es vol canviar de servei de música o bé es vol canviar de sistema de recomanació, no afecti directament a la part web gestionada per Struts 2.

Pel que fa a domini, sols queda comentar el paper dels Business Objects, les classes de domini que encapsulen totes les dades i els comportaments associats a la entitat que representen. Aquests objectes, seran mapejats a partir de tècniques ORM (Object-Relational Mapping) que implementa en aquest cas Hibernate. Aquest mapeig es realitza a partir del que es coneix com a Hibernate Annotations, que implementen el estándar JPA . En la imatge següent es pot veure una classe mapejada amb Annotations, que són les paraules clau precedides d'un símbol @.[CB07]

1. Els casos d'ús tenen. en general, un diagrama de classes senzill gràcies a l'ús del patró DI. No s'ha d'instanciar cap element, Struts 2 instancia els Actions i Guice els Services i les seves dependències.

2. No es representen totes les crides i instàncies que fa el propi Struts 2 ja que en la majoria dels casos, és transparent pel desenvolupador. Tot i així, es considera interessant veure el cas del procés de validació de les dades a partir del Interceptor de validació (Validation Interceptor). Per aquest motiu, s'adjunta la següent figura que mostra aquest procés, abans que s'executi l'Action.

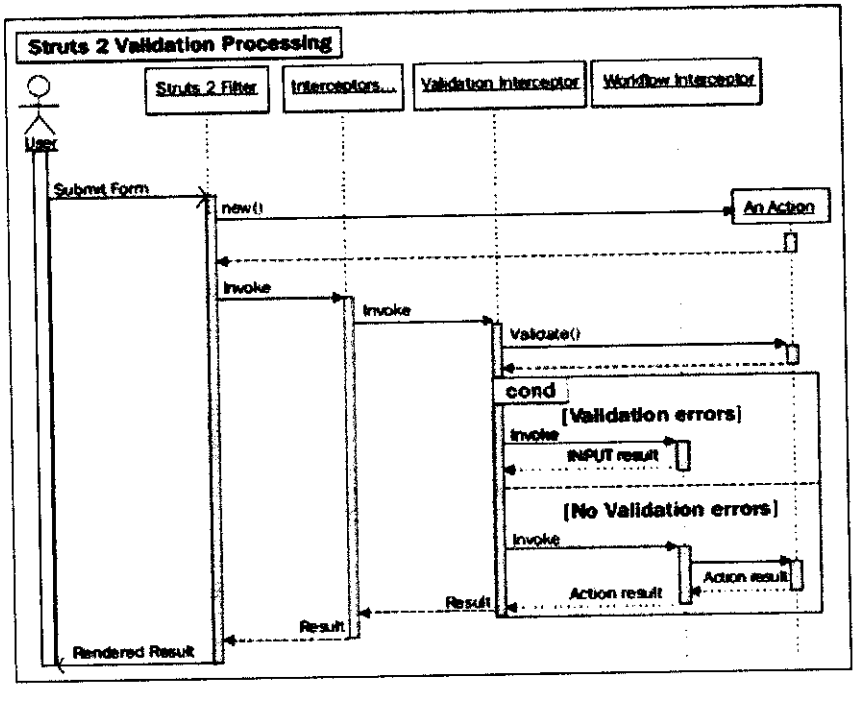


Figura 4.11: Diagrama de seqüència del procés de validació [img]

3. A l'apartat d'especificació s'han detallat totes les operacions del sistema i els seus contractes. En aquest apartat, es detallarà una mostra d'aquests.

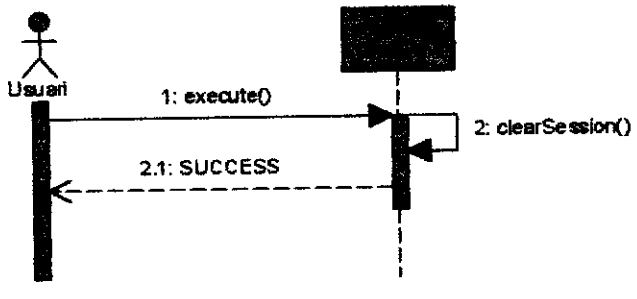
4. Les expressions SUCCESS, ERROR, etc. que es retornen a l'usuari en els dia-

Excepcions: Si el nom d'usuari (user) no existeix la operació find(user) del UserService llança una excepció i l'Action Login retorna ERROR. Igualment, en el cas que trobi l'User u però la contrasenya d'aquest usuari no correspongui a la proporcionada en la operació authenticateUser.

Tancar Sessió

Responsabilitat: Eliminar la sessió actual de l'usuari

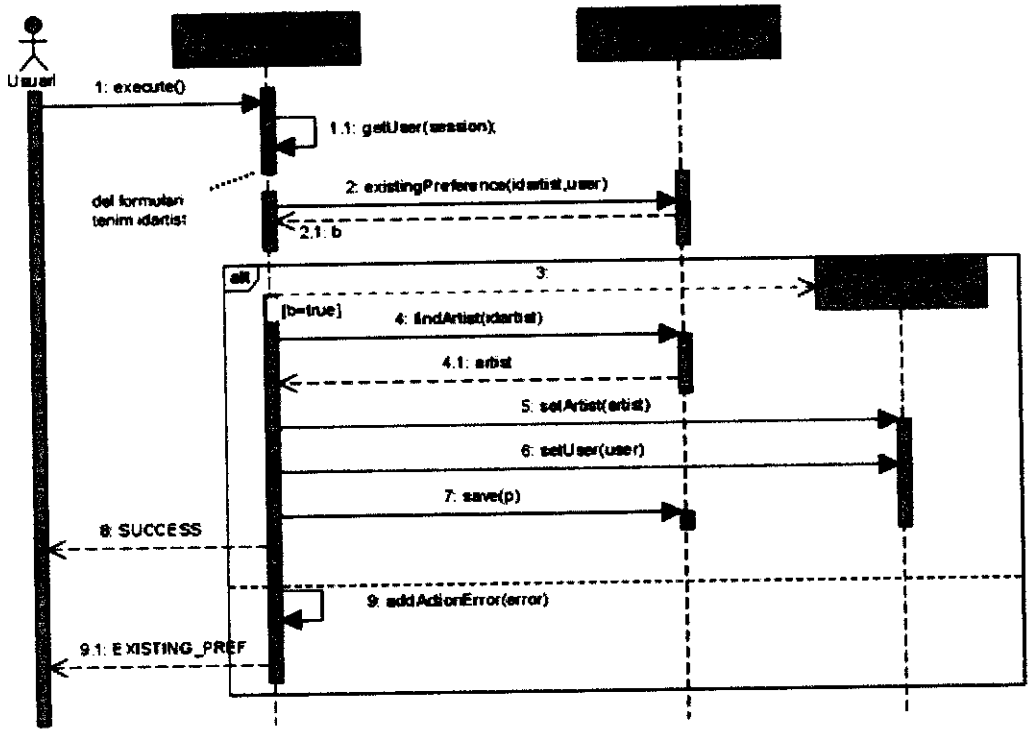
L'usuari demana tancar la sessió en el sistema i llavors s'executa l'Action Logout. Simplement, s'encarrega d'eliminar el que conté l'objecte session.



Alta Usuari

Responsabilitat: Registrar una nova instància d'usuari

L'usuari introdueix en la vista de registre els camps username, pwd i email i l'Action Register s'executa. Aquí es pot notar que no es crea l'instància d'usuari, ja que recordem que és Guice qui la crea. Llavors, simplement es té la instància ja creada i es pot consultar o modificar els atributs. En aquest cas, es té guardat en la instància u el username, pwd i email que ha introduït l'usuari i, a part, l'Action posa el valor per defecte dels atributs huge_content i popular_content. Seguidament, afegeix a la sessió l'usuari i crida al UserService perquè el guardi.

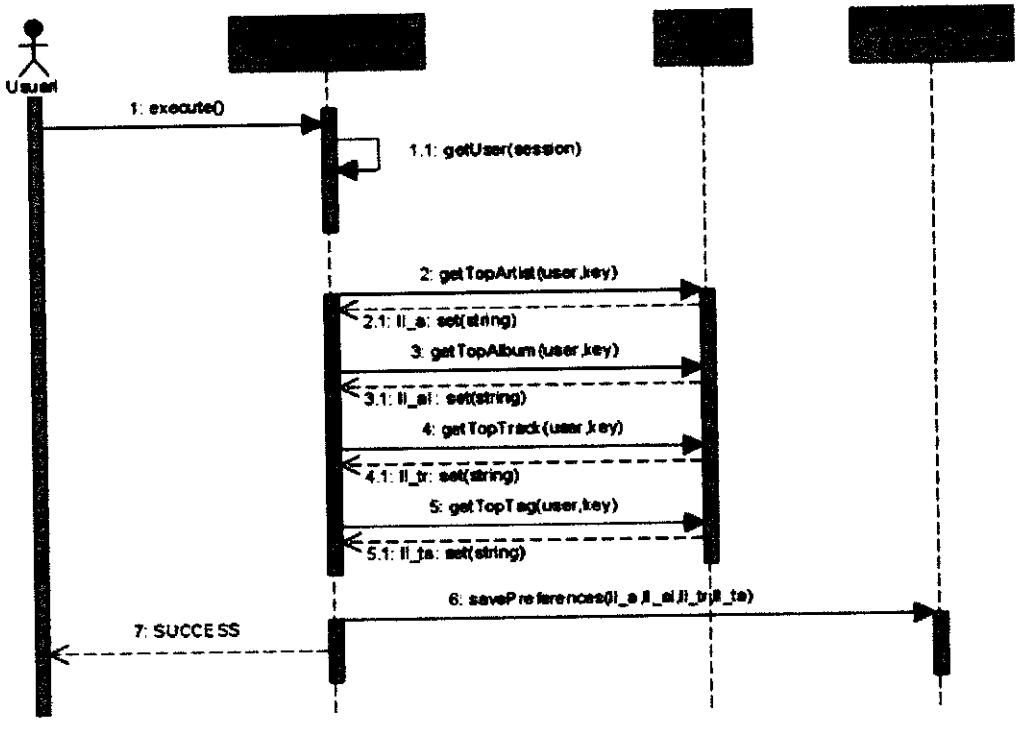


Eliminar Preferència

Responsabilitat:Eliminar la preferència escollida per l'usuari.

Senzillament, del formulari s'obté la id de la preferència a esborrar. El PreferenceService s'encarregarà de buscar la instància de preferència associada a la id i, seguidament, l'Action podrà eliminar la preferència.

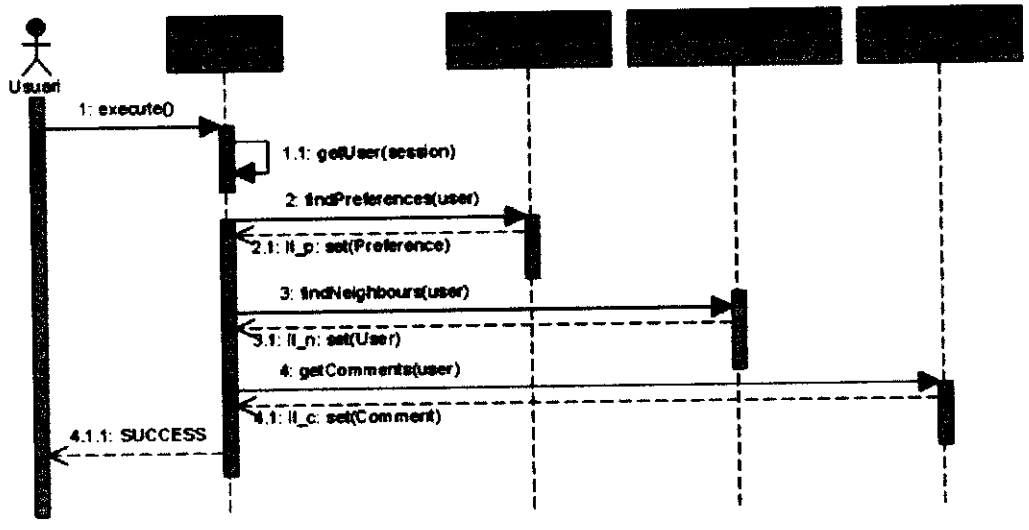
Els casos d'ús d'eliminar instàncies del domini, funcionen de forma similar.



Modificar Informació Personal

Responsabilitat: Modificar la instància de l'usuari amb la nova informació proporcionada per aquest.

Aquest, és el cas típic de modificació bàsica. Tot cas d'ús que modifiqui una instància del domini del sistema funcionarà de forma anàloga. L'usuari executa el cas d'ús de modificar el perfil, introduint les dades que vol modificar, el Action modifica la instància de l'usuari i, un cop modificada, la salva al sistema.



Mostrar Biografia

Responsabilitat: Obtenir la biografia d'un artista

En aquest cas, es vol mostrar com s'utilitza la informació obtinguda del WebService de Imeem. L'usuari indica que vol veure la biografia d'un artista. S'executa l'Action GetImeemAudio i, aquest Action, crida el mètode corresponent de la component Audio, en aquest cas getImeemBandInfo. A la component Audio, s'obre connexió a la URL del WebService que conté la informació a cercar via querystring. S'obté el Stream, que en el nostre cas és un XML, i es tracta per tal d'obtenir la informació que busquem, la biografia de l'artista.

De forma similar, funcionaria el cas d'escoltar música, on en comptes de recuperar la biografia, al tractar-se el XML s'obtindrien les URLs de les cançons a reproduir.

```
    recuperar(N);
    reutilitzar(Pref,N);
    trobarContingutRelacionat(Pref,N);
faccio
```

Listing 4.2: Pseudocodi operació generaRecomanacions

2. Component Social

El component social s'encarrega de realitzar les operacions de recuperar i reutilitzar tal i com s'ha vist.

```
accio  recuperar(N)
    //recuperar de BD artistes preferits de l'usuari
    llar := obtenirArtistesPreferits(sessio->usuari);
    computarMillorsCandidats(N, llar);
faccio
```

Listing 4.3: Pseudocodi operació recuperar

La operació `recuperar` fa ús d'una altra operació anomenada `computarMillorsCandidats` encarregada de trobar els veïns per un usuari.

A continuació es pot veure el pseudocodi per aquestes operacions.

```
accio computarMillorsCandidats(N, llar)

    //per cada preferència de l'usuari,
    //trobar les altres preferències del
    //sistema que també són del mateix artista

    per cada a de llar fer
        llpref := buscarPreferenciesUsuarisAmb(a);
        afegir(totesPref,llpref);
    fper

    //per cada preferència trobada, contar
    //el nombre d'aparicions que fan els usuaris.
    //A més aparicions serà un usuari més similar.

    per cada p de totesPref fer
        si p->usuari <> sessio->usuari llavors
```

```
        afegir(Pref,p);
    fper

    //contar les aparicions de cada contingut de
    //la preferència (artista, àlbum,etc.) com més
    //repeticions, més popular.

    per cada preferencia de Pref fer
    si Pref conte preferencia->contingut llavors
            count := buscar(preferencia->contingut)
            count := count + 1;
            inserir(Pref,preferencia->contingut,count);
        sino
            inserir(Pref,preferencia->contingut,1);
        fsi
    fper

faccio
```

Listing 4.5: Pseudocodi operació reutilitzar

3. Component Contingut

El component basat en el contingut s'encarrega de trobar contingut relacionat al que l'usuari li interessa.[10]

```
accio trobarContingutRelacionat(Pref, N)

    tags := obtenirTagsMesPopulars(Pref);

    //a partir dels tags més populars,obtenir
    //tot el contingut relacionat amb aquests

    per cada t de tags fer
        c:= obtenirContingutRelacionat(tag);
        afegir(Contingut,c);
    fper

    //s'amplia la llista obtinguda al component social,
```

[10]En diverses operacions s'utilitza buscar(r).Tenint h com una taula de hash, buscar(r) equival al valor de la taula de hash de la clau r

```
            count := buscar(t)
            count := count + 1;
            inserir(tags,t,count);
        sino
            inserir(tags,t,1);
        fsi
    fper

    guardar(tags,10);

   retorna tags;
ffuncio
```

Listing 4.7: Pseudocodi operació obtenirTagsMesPopulars

4. Component Selector: operació mostrarRecomanacions

El component selector, s'encarrega de mostrar les recomanacions segons els paràmetres que l'usuari ha especificat al sistema. Es pot observar el seu funcionament a continuació.

```
funcio mostrarRecomanacions() retorna rec

    rec := seleccionarRecEpoca(u->epoca);

    si u->popular = cert llavors rec := seleccionarRecPopular(rec
       );
    sino rec:= seleccionarRecMesExculsives(rec);

    //n i m enters on n > m
    si u->volum = cert llavors seleccionar(rec,n);
    sino seleccionar(rec,m);

retorna rec;
ffuncio
```

Listing 4.8: Pseudocodi operació mostrarRecomanacions

Tot i que al llarg de l'exposició del pseudocodi del sistema es mostren altres operacions, només s'han descrit les més importants ja que la resta són de fàcil comprensió i acostumen a tenir relació amb consultes a la base de dades.

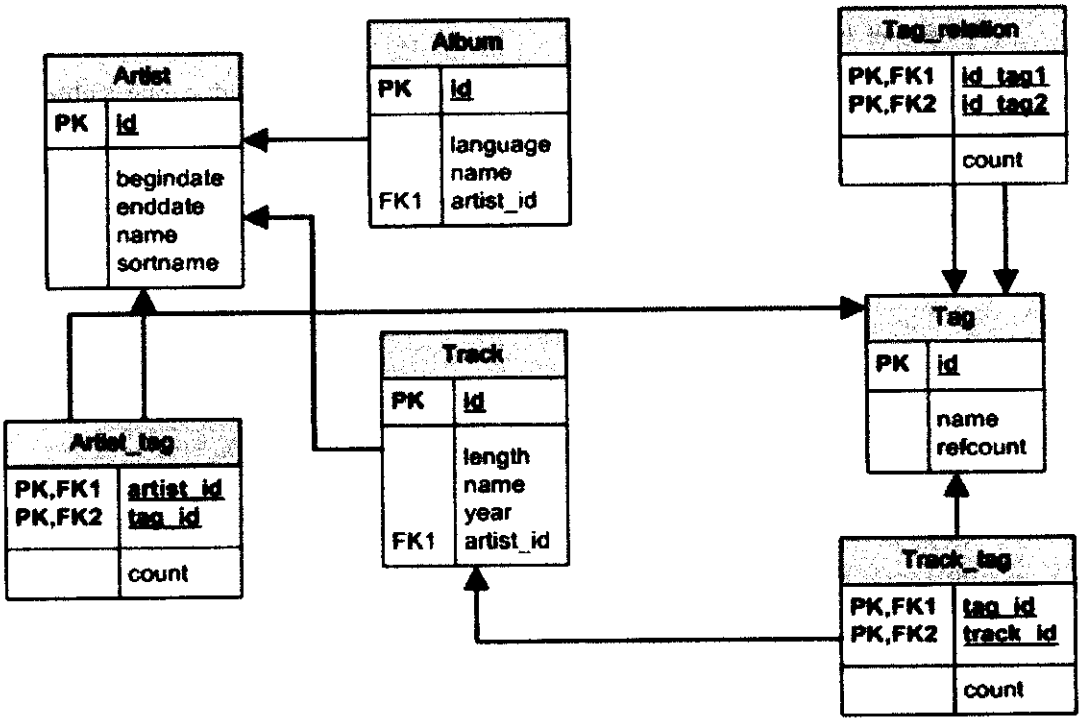


Figura 4.12: Esquema de dades resultat de fer enginyeria inversa a la BD de *MusicBrainz*

La base de dades de *Musicbrainz* és molt més extensa, tant en nombre de taules, com en nombre d'atributs. No obstant, moltes de les taules han estat irrellevants per aquest projecte i és per aquest motiu que només s'ha dut a terme aquest procés en les taules que han interessat.

Altres consideracions que s'han tingut en compte per construir el model de dades han estat:

- Separar la classe Country a una taula independent en comptes de tenir una columna amb text lliure dins de la taula usuari.
- Eliminar les Especialitzacions/Generalitzacions, convertint les especialitzacions en

4.4.5 Disseny de la capa de presentació

Per tal de fer un prototip de la interfície d'usuari (IU) recuperem alguns dels requeriments relacionats amb aquest apartat que s'han plantejat:

1. L'usuari ha de poder tornar a la home amb un sol click des de qualsevol lloc de la web
2. L'usuari ha de poder tancar sessió des de qualsevol lloc de la web.
3. La manera d'introduir les preferències del usuaris ha de ser molt fàcil, en un màxim de tres passos.
4. El disseny de l'espai on es mostrin les recomanacions ha de ser usable i intuïtiu per l'usuari.
5. L'usuari sempre ha de veure de forma ràpida en quina pàgina es troba
6. El sistema ha de seguir pautes d'usabilitat

Per tal de complir aquests requeriments, s'expliquen a continuació les decisions preses al respecte, dividint per blocs la informació:

Disseny de la interfície d'usuari

Es tindran en compte els següents punts:

- Pel disseny s'utilitzaran recursos comuns de les tendències de disseny web actual, com podria ser cantonades arrodonides, degradats i transparències. Aquests elements acabats de citar s'identifiquen actualment amb el concepte de web 2.0 que és on el nostre sistema es pot classificar.
- La tipografia serà una sans-serif, estàndard per web. La escollida és l'Arial ja que és la més utilitzada actualment, sense tenir en compte el cas de Facebook que utilitza Tahoma o les webs que s'han desenvolupat pensant en usuaris de mac i que utilitzen Helvetica.
- Es desenvoluparà un disseny híbrid entre interfície web i aplicatiu, ja que tot i que el projecte es troba ubicat en un context de web, té una clara vocació d'aplicació.

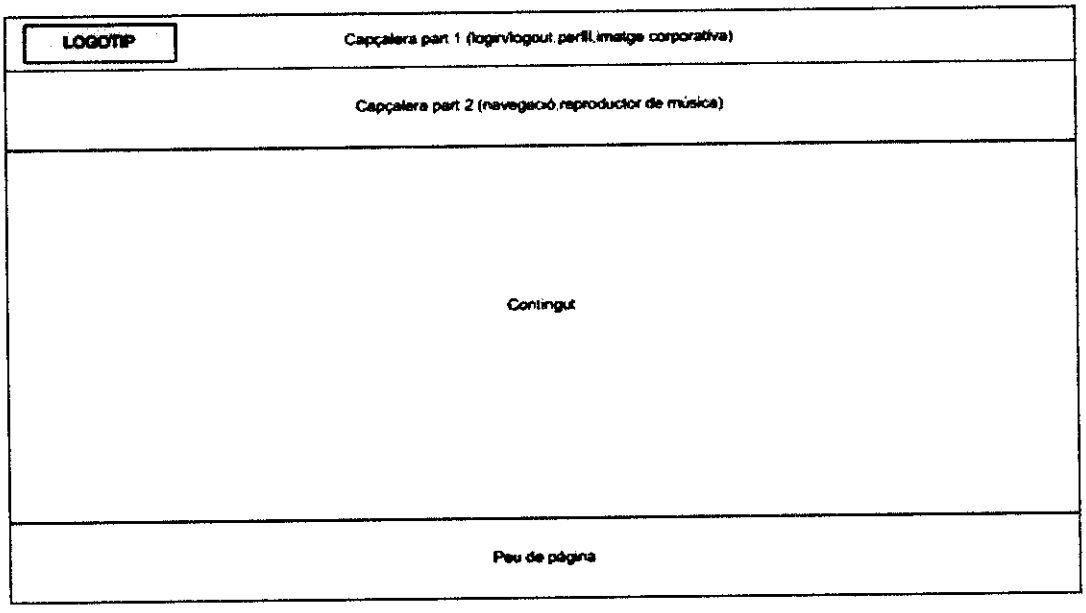


Figura 4.14: Estructura de les pàgines de recmusic

- **Capçalera:**

  La capçalera es dividirà en dues parts. La part superior contindrà la imatge corporativa, és a dir, el logotip que representa el sistema així com els enllaços per iniciar sessió/tancar sessió i registrar-se. La segona part de la capçalera, contindrà la navegació i el reproductor de música. La capçalera estarà sempre present en qualsevol de les pàgines que visiti l'usuari.

- **Peu de pàgina**: Al tractar-se d'un prototip, no conté tota la informació que acostumaria a tenir una pàgina web d'aquest estil com podria ser avisos legals i demés. Tot i així, s'ha inclòs en el disseny perquè és important mantenir l'estructura a la que l'usuari està acostumat. També serà sempre visible.

- **Contingut**: Part dinàmica del disseny de l'aplicació web. En aquest espai s'aniran mostrant els diferents continguts.

Aquesta estructura es desenvoluparà en *Freemarker* de la següent manera:

1. Creació de plantilles per blocs, és a dir, una plantilla que contingui la capçalera part 1, una el peu de pàgina i una altra la capçalera part 2.

```
<@head />
    <body>
        Welcome to recmusic!
    </body>
<@footer />
```

Listing 4.12: Exemple de document freemarker corresponent a una possible vista

Navegació

La figura 4.15 mostra el flux d'informació bàsic que es podrà dur a terme en el sistema. Aquest flux, determina la navegació de la web tot i que pot estar subjecte a alguna decisió final d'implementació. S'ha intentat dirigir a l'usuari a través de la web de forma senzilla i intuïtiva, intentant que qualsevol acció que vulgui dur a terme es pugui fer de manera bastant directe.

4.5 Implementació

L'objectiu d'aquest capítol és presentar els entorns de desenvolupament utilitzats, així com els detalls, avantatges i problemes trobats durant la fase d'implementació del sistema.

4.5.1 Entorns pel desenvolupament del sistema

Eclipse

S'ha emprat la versió 3.4 d'aquest famós IDE fet en Java per a desenvolupar el sistema. Eclipse, apart de donar suport als diferents llenguatges utilitzats pel desenvolupament (Java, HTML, CSS, JavaScript, XML), permet la integració de diversos plug-ins que faciliten la creació i manteniment d'aplicacions.

Eclipse també incorpora per defecte la integració del servidor web o d'aplicacions. D'aquesta manera, dins del propi Eclipse es pot arrancar, parar i configurar, en aquest cas, el servidor Tomcat.

PgAdmin III

Es tracta d'una aplicació gràfica per gestionar el SGBD PostgreSQL, sent la més completa i popular sota llicència Open Source.

Luke

Permet inspeccionar els índexs de Lucene creats per Hiberante Search de manera ràpida i còmoda. És una eina molt simple però bastant indispensable que bàsicament ajuda a:

- Veure documents individuals
- Executar consultes i veure el resultat
- Seleccionar documents de l'índex concrets i eliminar-los
- Saber la freqüència d'una paraula

L'autor de Luke és Andrzej Bialecki, i manté activament Luke perquè s'adapti a les últimes versions de Lucene. Pot ser descarregat via Java WebStart JNLP o mitjançant llibreries. En aquest projecte s'ha utilitzat la versió JNLP d'aquest.

4.5.2 Estructuració

El projecte s'ha estructurat en paquets:

1) **Actions**: conté tota la lògica de negoci del sistema, en altres paraules, tots els Actions que Struts 2 pot executar

2) **Components**: conté els components independents de Struts, com pot ser el component d'audio o els de recomanació.

3) **Domain**: conté els business objects, les classes de domini que són mapejades també a la base de dades.

4) **Indexes**: conté les classes encarregades d'indexar la informació.

5) **Interceptor**: conté els interceptors desenvolupats a mida, com ara l'Interceptor de autenticació.

6) **Service**: conté tots els services i les seves implementacions que es comuniquen amb la BD.

Seguidament trobem els fitxers de configuració que veiem en el següent apartat.

La part de contingut web, és a dir, el conjunt de vistes, es troben en la carpeta WebContent, així com els css, javascripts i imatges utilitzades per la interfície de l'usuari.

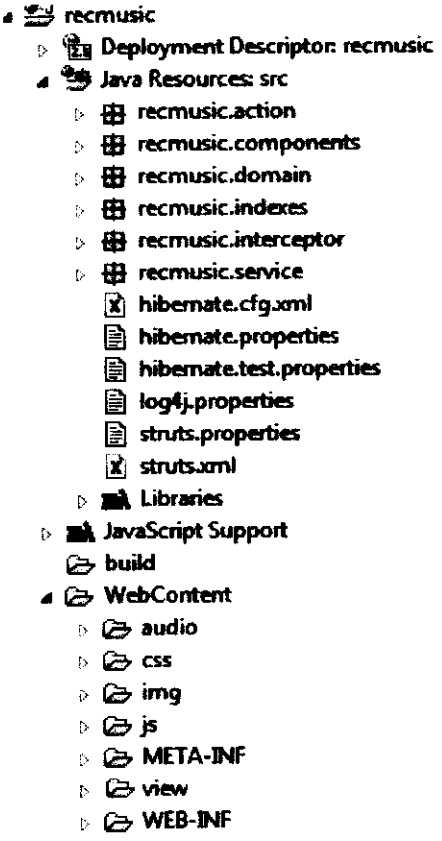


Figura 4.16: Estructura de les carpetes del projecte

```
            <interceptor-ref name="hibernate" />
            <interceptor-ref name="defaultStack" />
        </interceptor-stack>
    </interceptors>
<default-interceptor-ref name="daoStack" />
    <global-results>
        <result name="error">/view/common/error.ftl</result>
        <result name="login">/view/login.ftl</result>
    </global-results>
    <global-exception-mappings>
        <exception-mapping result="error" exception="java.
            lang.Exception" />
    </global-exception-mappings>
    </package>
    <package name="secure" extends="default">
    <interceptors>
        <interceptor name="auth"
                    class="recmusic.interceptor.
            AuthenticationInterceptor" />
        <interceptor-stack name="secureStack">
            <interceptor-ref name="auth" />
            <interceptor-ref name="hibernate" />
            <interceptor-ref name="fileUpload">
<param ame="allowedTypes">
image/png,image/gif,image/jpeg
</param>
                <param name="maximumSize" >
1900000
</param>
            </interceptor-ref>
            <interceptor-ref name="defaultStack" />
        </interceptor-stack>
    </interceptors>
    <default-interceptor-ref name="secureStack" />
     </package>
</struts>
```

Listing 4.13: Fitxer de configuracio Struts 2

Més endavant trobem on es guarden els documents indexats i seguidament totes les entitats que seran mapejades a la BD, corresponents al package domain que em vist en el punt de estructuració (4.5.2).

```
<!DOCTYPE hibernate-configuration SYSTEM "http://hibernate.
   sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
   <session-factory>
    <property name="hibernate.connection.driver_class">
        org.postgresql.Driver
        </property>
    <property name="hibernate.connection.url">
jdbc:postgresql://localhost/recmusic_bd
</property>
    <property name="hibernate.connection.username">anna</property
       >
    <property name="hibernate.connection.password">admin</
       property>
    <property name="hibernate.dialect">
org.hibernate.dialect.PostgreSQLDialect
</property>
    <property name="show_sql">
true
</property>
    <property name="format_sql">
false
</property>
        <property name="hibernate.search.default.indexBase">
/Users/anna/indexed
</property>
        <property name="hibernate.hbm2ddl.auto">update</property>
        <mapping class="recmusic.domain.Album"/>
        <mapping class="recmusic.domain.Artist"/>
        <mapping class="recmusic.domain.Artist_tag"/>
        <mapping class="recmusic.domain.Country"/>
        <mapping class="recmusic.domain.PreferenceArtist"/>
        <mapping class="recmusic.domain.PreferenceTag"/>
        <mapping class="recmusic.domain.PreferenceAlbum"/>
        <mapping class="recmusic.domain.PreferenceTrack"/>
        <mapping class="recmusic.domain.Tag"/>
        <mapping class="recmusic.domain.Tag_relation"/>
```

2. ***Tokenization* dels índexs**

   Per defecte, quan indiques a Hibernate Search que ha d'indexar un atribut d'una classe, aquest separa en tokens tots els termes dels camps. En un principi, les cerques donaven mals resultats perquè no es va tenir en compte que tots els termes estaven separats per tokens. Per exemple, si un usuari buscava '*The Rolling Stones*' i en comptes d'escriure-ho correctament escrivia '*Rolling*' només, el sistema no era capaç de trobar com a primers resultats '*The Rolling Stones*'.

   Per aquest motiu, finalment es van reindexar els atributs evitant que es separessin els tokens i les cerques obtingudes via *fuzzy queries* han estat correctes.

3. **Components de recomanació**

   L'algorisme de recomanació dissenyat, al haver-se de recórrer un gran volum de files, sovint obtenia temps massa llargs. Actualment, la base de dades conté uns 500.000 artistes i unes 8.000.000 de cançons. A part, com que el sistema s'ha desenvolupat en un PC, sense disposar d'un servidor amb suficient RAM i processador, també ha fet que sovint els temps obtinguts no siguin tant bons com els desitjats.

   Per aquest motiu, s'han introduït els següents filtres:

   - El component Social Seeker només utilitza els 10 millors veïns per tal de trobar els elements més similars a l'usuari pel que s'estan generant les recomanacions
   - El component Content Seeker només fa ús dels tags que ha introduït l'usuari i dels tags dels seus 5 millors veïns per tal de trobar contingut relacionat amb el contingut de l'usuari pel que se li estan generant les recomanacions.

   A part, s'ha tingut molt en compte les estructures de dades utiltizades, principalment TreeMaps i HashSets que han permès un estalvi substancial de temps a l'hora de realitzar l'algorisme. També s'ha intentat realitzar el mínim de consultes a la base de dades durant el procés, intentant mantenir en memòria tota la informació ja carregada.

4. **Problemes amb els navegadors**

   Ja és totalment conegut el fet que una maquetació web es pot veure molt diferent en un navegador que en un altre. Al voler fer una maquetació cross-browser i estàndard, s'han tingut força problemes per aconseguir que l'aplicació es veiés bé tant en Internet Explorer com en Firefox i Chrome, que són els navegadors més utilitzats i els que s'han tingut en compte.

D. Poc coneixement de les preferències de l'usuari

La part de contingut del sistema actualment és estàtica. S'ha instal·lat com s'ha explicat anteriorment, la base de dades de MusicBrainz i s'ha fet un *dump* de la seva estructura i informació a una data determinada.

Per altra banda, per tal de dur a terme les diferents proves en les situacions plantejades, s'han importat un seguit de perfils d'usuari de Last.fm. Aquest perfils consten bàsicament de les preferències d'uns 33.000 usuaris, per tal de poder dur a terme les proves pertinents del component social del sistema.

Així doncs, idealment es treballarà sota aquesta situació, amb un volum d'usuaris mitjà, com el que acabem de comentar (situació A) i s'anirà jugant amb les situacions C i D per veure la repercussió que tenen aquestes situacions en el sistema.

No obstant, també es realitzaran proves partint d'un volum d'usuaris baix per veure l'impacte del component social en el sistema.

4.6.2 Resultats obtinguts

S'ha dut a terme la bateria de proves especificada. Per cada context s'han obtingut certes conclusions que s'expliquen a continuació així com les limitacions trobades i les accions preses.

Observacions

- El component social perd el sentit en contextos amb un nombre d'usuaris alt però amb un nombre de preferències d'artistes baixa. Penalitza en temps i en la majoria dels casos, no aporta valor. En el cas extrem, en que l'usuari només té una preferència, s'han tingut en compte les següents situacions.

  Accions considerades

  a. Eliminar l'ús del component social en aquest cas.
  b. Consultar directament a la base de dades [11] sota les condicions que es coneixen que són:
  - Intersecció: sempre serà 1

[11] En l'estudi inicial, s'ha observat que Last.FM actua així

Limitacions

Després d'haver realitzat les proves s'han detectat certes limitacions:

1. **Escalabilitat d'usuaris**:

   Si l'entorn creixés molt en nombre d'usuaris, es considera que podria ser inviable el càlcul de les recomanacions a temps real, ja que implica un gran consum de memòria del sistema per poder realitzar les recomanacions en un temps acceptable.

   Per garantir els temps però s'hauria de provar l'aplicació en una màquina més potent de la qual no s'ha disposat en aquest projecte.

2. **Paràmetres de configuració del sistema**

   La determinació del valor dels paràmetres que s'utilitzen per que el sistema sigui més eficient i no tingui en compte casos sense valor (H2) s'ha determinat de manera empírica en un valor de 15 % per casos normals i d'un 25% en casos d'usuaris amb un volum de preferències elevat. No obstant, tot i que per la situació actual es consideren òptims, no es pot garantir que en un suposat creixement del sistema no s'hagin d'ajustar. Pot ser que s'hagin de determinar de nou d'una manera més teòrica.

plantejar la viabilitat d'un sistema de recomanació d'aquest tipus a temps real. Aplicant certs filtres, però, s'ha aconseguit que el temps de recomanació sigui acceptable.

Construcció del pilot

Cal destacar que, gràcies a l'especial atenció que s'ha prestat en la decisió d'utilitzar les tecnologies que finalment s'han emprat, s'ha aconseguit desenvolupar un sistema escalable, que fa ús de tecnologies punteres i fàcil de mantenir. L'aprenentatge ha estat elevat ja que les tecnologies utilitzades eren desconegudes per mi.

Pel que fa a l'aplicació web, s'ha intentat crear un entorn el màxim de realista, que pogués ser realment publicat a la xarxa i que tingués utilitat. S'ha dissenyat pensant molt en l'usuari, intentant que fos el màxim d'atractiu per a ell. No obstant, es considera que si es publiqués el sistema, valdria la pena oferir alguna funcionalitat extra per al que també s'ha intentat que el sistema sigui fàcilment ampliable.

5.2 Revisió de la planificació

Finalment, hi ha hagut una lleugera desviació en el temps estimat de finalització del projecte. El projecte s'ha finalitzat aproximadament amb un 10% més de temps total.

Aquesta desviació, ha estat deguda a tres motius:

- El projecte es va començar més tard, i es va reduir en dues setmanes el més estimat de vacances.
- La corba d'aprenentatge de les tecnologies.
- La implementació del sistema va portar dues setmanes més de feina, per tal de fer millores al sistema.

Així doncs el total de dies destinats han estat els següents:

Taula 5.2: Costos del projecte

Tasca	Recurs	Preu/hora	Hores	Total
Direcció del projecte	Director	50 €	20	1.000 €
Investigació	Analista	30 €	100	3.000 €
	Programador	15 €	20	300 €
Conceptualització	Analista + Arquitecte	30 €	100	3.000 €
	Programador	15 €	20	300 €
Especificació	Analista	30 €	28	840 €
Disseny	Arquitecte	30 €	40	1.200 €
Implementació	Programador + Dissenyador web	15 €	300[1]	4.500 €
Proves	Tester	10 €	28	280 €
Documentació	Programador	15 €	12	180 €
TOTAL	-	-	**668**	**14.600 €**

5.4 Futur del sistema de recomanació

El sistema, com a prototip, no és un sistema definitiu. Ha nascut amb la idea de plasmar les teories establertes en els primers capítols i posar a prova la recomanació a temps real. D'aquest fet, ja s'han detectat certes limitacions que es podrien intentar pal·liar en futures millores.

Pel que fa a altres tipus d'ampliacions, i com ja s'ha comentat al llarg de la memòria, si es volgués llençar aquest producte a la xarxa caldria considerar la necessitat d'oferir més funcionalitats a l'usuari. També seria interessant la creació de *widgets* o d'eines d'integració per a altres plataformes, ja que seria una manera de que recmusic fos un servei més visible.

Per altra banda, recmusic s'ha introduït en un entorn d'una xarxa social però allunyant-se una mica de crear un espai 'on trobar gent propera a tu'. S'ha intentat que per sobre de tot fos una utilitat. Cada cop més, els usuaris d'Internet troben més atractives les eines *on-line* que els hi són realment útils, que els ajuden a solventar una necessitat. Potser la frase sembla òbvia però el que s'està intentant explicar és que, les xarxes socials en certa manera són una moda que no desapareixerà del tot però que ja comença a causar estralls entre els usuaris, que se senten molts cops sobrecarregats d'informació irrellevant. Per aquest motiu, si s'hagués d'ampliar el producte, s'intentaria enfocar

sistemes com aquest, consumeixen molta memòria.

Com a conclusions personals respecte als sistemes de recomanació, he vist que realment tenen moltes limitacions, que es veuen condicionats per molts paràmetres que els fan ser difícils de configurar. No obstant, segueixo pensant que el futur de la xarxa està en una web confeccionada a mida per a l'usuari, ja sigui amb un sistema de recomanació, amb sistemes d'agregació d'informació personalitzats o qualsevol altre sistema que proporcioni a l'usuari allò que necessita pràcticament amb un sol click.

Glossari

Adobe Flex	es tracta d'un kit de Adobe Systems creat amb la finalitat de poder desenvolupar Rich Internet Applications (aplicacions RIA), és a dir, aplicacions web que tenen gran part de les característiques d'una aplicació d'escriptori, 82
Annotations	és una forma d'afegir metadades al codi font Java els quals estan disponibles a l'aplicación en temps d'execució. Moltes vegades s'utilitzen com a alternativa a la tecnología XML., 76
Apache Software Foundation	organització no lucrativa creada per donar suport als projectes de software sota el nom Apache. Es tracta d'una comunitat descentralitzada de desenvolupadors que treballen projectes de codi lliure., 74
Back-office	part d'un sistema software que serveix per gestionar-lo i que no està visible a l'usuari, 87
Cloud Computing	tendència actual en la xarxa, utilitzant-la com si es tractés d'un sistema operatiu, incorporant així software online que substitueix les aplicacions instal·lades en els ordinadors de cada usuari, 149

JDBC	acrònim de Java Database Connectivity, es una API que permet l'execució d'operacions sobre la base de dades des del llenguatge de programació Java, independentment del sistema operatiu on s'executi o de la base de dades en la qual s'accedeixi., 74
Llibreria	col·lecció de material útil (mètodes, interfícies, implementacions,etc.) per el ús general empaquetat per ser utilitzat sota una plataforma concreta., 78
Look & Feel	terme utilitzat per descriure les principals característiques de l'aparença d'un producte., 79
PHP	llenguatge de programació interpretat, dissenyat originalment per a la creació de pàgines web dinàmiques., 73
REST	és una arquitectura de programari pensada per sistemes distribuïts basats en hypermedia, com ara la web. Tot i que en un principi REST es referia tan sols a un conjunt de principis d'arquitectura de xarxa i la definició i adreçament dels recursos, actualment aquest concepte s'utilitza per referir-se a una interfície web que utilitza XML i HTTP sense cap conjunt de capçaleres, 82
Ruby	llenguatge de programació interpretat, reflexiu i orientat a objectes., 73
Servei web	és una col·lecció de protocols i estàndards que serveix per intercanviar dades entre aplicacions, 82

W3C
: acrònim de World Wide Web Consortium, defineix el consorci internacional que regula i proposa bones pràctiques en tot el que està relacionat amb la WWW., 79

WebWork
: framework basat en Java desenvolupat per la comunitat OpenSymphony que ha acabat convertint-se en l'actual Struts 2. Va ser desenvolupat amb la intenció de millorar la productivitat del programador i aportar simplicitat al codi., 75

Widget
: element d'una interfície gràfica d'usuari que té com a objectiu oferir una funcionalitat concreta de totes les disponibles en una aplicació. En altres paraules, els widgets són blocs de l'aplicació que en un conjunt poden dur a terme totes les funcionalitats de l'aplicació., 82

XML
: llenguatge de marcatge creat per transportar dades que es caracteritza per tenir una sintaxis descriptiva., 76

[EB09] John Griffin Emmanuel Bernard. *Hibernate Search.* Manning Publications Co., 2009.

[GG:a] Google collections framework. `http://code.google.com/p/google-collections/`.

[GG:b] Google guice. `http://code.google.com/p/google-guice/`.

[img]

[j2e] J2ee design patterns. Web: `http://java.sun.com/blueprints/corej2eepatterns/Patterns`, `http://java.sun.com/blueprints/patterns`.

[JFKa] Enterprise javabeans technology. Web Oficial: `http://java.sun.com/products/ejb/`.

[JFKb] Java spring framework. Web Oficial: `http://www.springsource.org/`.

[JFKc] Javaserver faces technology. Web Oficial: `http://java.sun.com/javaee/javaserverfaces/`.

[Las] Last.fm: Music to listeners' ears. WIRED: `http://www.wired.com/culture/lifestyle/news/2003/07/59522`.

[Mon03] Miquel Montaner. *Collaborative Recommender Agents Based on Case-Based Reasoning and Trust.* PhD thesis, Universitat de Girona, September 2003.

[Pag] Oficial Web Page. The apache velocity project. `http://velocity.apache.org/`.

[PvM] Mysql vs postgresql. `http://tweakers.net/reviews/649/7`.

[Rei06] José L. Ruiz Reina. Introducción a las redes bayesianas, 2006. `http://www.cs.us.es/cursos/ia2-2005/temas/tema-08.pdf`.

[RvM] Maarten van Someren Robin van Meteren. Using content-based filtering for recommendation1. Whitepaper: `http://www.ics.forth.gr/~potamias/mlnia/paper_6.pdf`.

[Ser] Apache tomcat. Web Oficial: `http://tomcat.apache.org/download-60.cgi`.