RÀDIO PARTICIPATIVA AMB DIFUSIÓ PER INTERNET

Memòria del Projecte Fi de Carrera
d'Enginyeria Informàtica
realitzat per Sergi Pérez Duran
i dirigit per Josep Mª Ganyet
Bellaterra, 2 de Febrer del 2007

Acabes els cinc cursos i tan sols et queda el projecte, això ja està són quince crèdits i ja tens el títol. Com es poca cosa pots combinar-ho amb el treball, comences a treballar i apali vuit hores com a mínim davant de l'ordinador cada dia, arribes a casa i la última cosa que vols fer es posar-te davant l'ordinador, ja l'entregaré al Febrer, al Setembre ... i així vuit anys. La universitat cada cop sembla que estigui més lluny, a més el fet de treballar et fa conèixer una part de la informàtica que no t'agrada i sofreixes una dicotomia que a vegades et fa pensar que no és el teu món, altre vegades que si ... I al final descobreixes una vocació tardia, dedicar-me a la docència, per la que necessites el títol i entre això i les ganes de deixar acabades les coses, sorgeixen les forces per tornar a "reenganxar-se" a la carrera i a fer el projecte, que ara toca!

Voldria agraïr l'ajuda de tots aquells amics, familia que durant aquests vuit anys cada "x" temps em deien:"Sergi i el projecte quan el faràs? Quan tindràs el títol ? Les coses s'han d'acabar!!" Van ser molts els que m'ho deien i la freqüència de repetició era alta, en el moment que m'ho deien: "Uf, que pesats!" Era desagradable per mi però aquesta insistència feu que jo mai me n'oblidés i finalment ho he dut a terme. Gràcies per no haver deixat mai d'insistir !

I personalitzar un agraïment en el Josep Maria Ganyet (encara que se'm titlli de pilota, doncs és el director) per la paciència.

Resum

Aquesta memòria reuneix una explicació del treball que s'ha dut a terme per a desenvolupar el Projecte Final de Carrera. El projecte tracta d'implementar una ràdio musical que s'emet per Internet però amb la característica que té que esser una ràdio que decideixi la programació musical en funció de la seva audiència. Per a realitzar això es distingiran dos tipus d'usuaris els oients i els gestors de la ràdio musical.

El gestors de la ràdio tindràn la responsabilitat d'alimentar de música la ràdio, etiquetar i classificar aquesta música perquè la ràdio estableixi criteris de similitud entre cançons. Els oients seran responsables d'opinar sobre les cançons que emet la ràdio, si els agrada o desagrada, per guiar la ràdio cap al gust musical del propi oient.

La complexitat està en què no es tracta d'una ràdio musical amb un sol oient sinó que hi ha diferents oients i s'haurà de saber gestionar correctament els gustos dispersos dels oients.

En el present document recullim tot els treballs realitzats, l'anàlisis formal del problema, la explicació de la solució proposada i com hem realitzat aquesta solució.

Resumen

Esta memoria reune la explicación del trabajo que se ha llevado a cabo para desarrollar el Proyecto Final de Carrera. El proyecto trata de implementar una radio musical que se emite por Internet pero con la característica que tiene que ser una radio que decida la programación musical en función de la su audiencia. Para realizar esto se distinguen dos tipos de usuarios los oyentes y los gestores de la radio musical.

El gestor de la radio tiene la responsabilidad de alimentar de música la radio, etiquetar y classificar esta música para que la radio establezca criterios de similitud entre canciones. Los oyentes seran responsables de opinar sobre las canciones que emite la radio, si les gusta o desagrada, para guiar la radio hacia el gusto musical del propio oient.

La complejidad está en que no se trata de una radio musical con un solo oyente sino que hay diferentes oyentes y se habrá de saber gestionar correctamente los gustos dispersos de los oyentes.

En el presente documento recojemos todo el trabajo realizados, el análisis formal del problema, la explicación de la solución propuesta y como hemos realitzado dicha solución.

Abstract

This memory has the explanation of the work that has been carried out to develop the Final Project. The project tries to implement one musical radio that is emitted by Internet but with the characteristic that it has to be one radio that decides the musical programming based on the its hearing. In order to make this two types of users are distinguished the listeners and the managers of the musical radio.

The manager of the radio has the responsibility to feed on music the radio, to label and to classificar this music so that the radio establishes criteria of similarity between songs. The listeners you would seran responsible to think on the songs that the radio emits, if they like or dislike, to guide the radio towards the musical taste of own oient.

The complexity is in that is not one musical radio with a single listener but that are different listeners and it will be had to know how to correctly manage the dispersed tastes of the listeners. In made document the present we write all the work, the formal analysis of the problem, the explanation of the propose solution and how we made this solution.

Índex de continguts

1. Introducció

La intenció d'aquest punt és recollir de manera breu que és el que ha de fer el projecte i un descripció dels punts que fna interessant aquest projecte.

1.1. Descripció del projecte

El projecte consisteix en la implementació d'una ràdio musical emesa per Internet amb una peculiaritat: la ràdio ha d'aprendre el gust musical de l'audiència i segons aquest ha de decidir l'ordre de les cançons a escoltar.

Actualment a Internet ja en podem trobar de ràdios:

- ràdios a l'estil de les **tradicionals**, on Internet tan sols afegeix que és un mitjà més pel que emeten.
- ràdios **a la carta**, l'oient tria quin programa ja emès o arxiu sonor (coneguts com a **posdcasts**) vol escoltar. És emissió diferida.
- ràdios **personals**, l'oient defineix uns criteris musicals i sota aquests criteris la ràdio selecciona les cançons a reproduir però en aquest cas la ràdio tan sols té un oient. Les característiques principals d'aquestes ràdios són: permetre a l'oient conèixer noves cançons que encaixen amb els seus gustos, restringir la música emesa als gustos propis i conèixer altres ràdios personals que encaixen amb el gust definit per l'oient.

Per tant el que fa interessant del nostre projecte és la peculiaritat abans descrita doncs és la unió de la ràdio tradicional amb les ràdios personals, ja que en aquest cas la ràdio aprendrà dels criteris musicals de tots els oients: **intentarà adaptar-se a l'audiència**, per tant segons l'audiència que tingui s'anirà especialitzant en un tipus de música o altra.

1.2. Abast

La realització del projecte ha de consistir en l'elaboració dels següents blocs:

1. Donar eines per tal de programar / gestionar la radio a través d'Internet, crear una **administració** de la ràdio que ens permeti seleccionar de quina música disposarà, es facilitarà la pujada de música massiva a la ràdio, organitzar aquesta música, descriure la música i comprovar estadístiques d'ús de la ràdio. Per tant la música s'haurà de **classificar i etiquetar** per tal de poder donar eines a la ràdio perquè pugui trobar similituds entre cançons, perquè la música tingui un significat més enllà d'un fitxer sonor.
2. Dur a terme la **reproducció de la ràdio**, implementar que pugui ser escoltada per Internet. Per tant que hi hagi un espai a Internet on l'oient pugui accedir a la nostra ràdio.
3. Permetre **múltiples instàncies de reproducció** d'una ràdio. Gestionant tot el magatzem musical puguin haver-hi diferents ràdios, emissores, alimentant-se del mateix magatzem i permetre que hi hagi diferents públics.

4. **Aprendre** els gustos musicals de l'audiència, permetre que els oients puguin **pronunciar els seus gustos** així la ràdio és capaç de conèixer l'audiència i també tenir eines per aprendre nous criteris de semblança entre cançons.
5. **Decidir** l'ordre de les cançons a reproduir segons el gust musical de l'audiència. Un cop coneguda l'audiència s'haurà de seleccionar dins el ventall musical disponible **quina és la següent cançó a emetre**.
6. Donar un entorn amigable al oient en el que a més de permetre escoltar la música que emet la ràdio en aquell moment, ens tongui més informació del **context de la ràdio**: mostrar històric de cançons emeses, rebre informació sobre la cançó actual i saber abans quina serà la següent cançó a emetre.
7. Realitzar una **bateria de proves i simulacions per comprovar** el funcionament del algoritme d'aprenentatge i el de decisió. Per aquesta tasca s'utilitzarà diferents programes que ens permetran sotmetre a la ràdio diferents proves d'estress. Per exemple el Jmeter.

2. Anàlisis de requeriments

En aquest punt realitzem un anàlisi dels requeriments del projecte, a continuació descrivim tot el que ha de contenir el nostre projecte, n'hi ha de requeriments funcionals i altres que no.

2.1. Requeriments no funcionals

El anàlisis dels aspectes extrínsecs al projecte a realitzar té en compte aspectes com: llenguatge de programación, requisitos de dissenys, requísits genèrics de l'eina.

1. **Independència del sistema operatiu** i de la **base de dades**, s'ha d'intentar en el disseny de la implementació de l'eina que qualsevol canvi en les peces de software externes no repercuti cap canvi en el software propi del projecte.
2. Aplicació **WEB**, per tant tindrà una part servidora i una part client. La primera serà executada per un servidor Web i l'altre per un client Web (el navegador).
3. Aplicació **multillenguatge**. L'aplicació tingui una interfície que pugui ser escalable a qualsevol idioma de manera senzilla sense repercutir cap canvi en el software propi de l'aplicació.
4. El format musical ha de ser un fitxer de so comprimit, es tria en aquest cas el format **mp3**.

2.2. Requeriments funcionals

En aquest punt llistem les funcionalitats principals que realitzarà la ràdio:

2.2.1. Validació d'usuaris

La ràdio té que oferir una part d'accés privat, s'accedeix aquesta part privada per servei web i haurà de garantir que tan sols els usuaris que tinguin permís puguin accedir-hi, als usuaris que se'ls hi faciliti el codi d'usuari i clau d'accés correcte.

2.2.2. Gestionar una ràdio

Aquesta funcionalitat té un accés restringit i no pot ser accedida sense permís. Des d'aquesta part els **programadors de la ràdio** duran a terme totes les tasques de gestió de la ràdio, aquestes tasques són les següents:

- **Gestionar les cançons** a la ràdio. Eines via web que permetin afegir , eliminar i modificar cançons perquè puguin ser emeses per la ràdio.
- Poder **classificar** les cançons. Per tant ha de facilitar eines per classificar / ordenar aquestes cançons, perquè no sigui un calaix de sastre i també ens permeti tenir un criteri de similitud. Per exemple ubicar aquestes cançons dins de gèneres musicals, estils musicals, associar les cançons a autors i artistes,

agrupar-les per àlbums. Per tant l'eina de gestió ens haurà de permetre gestionar totes aquestes categories on es poden ubicar les cançons.

- **Definir** la música. Ha de tenir eines per definir la música pujada per tal que la ràdio no la vegi tan sols com un arxiu sonor sinó que tingui una sèrie d'atributs que li podrà permetre trobar similituds entre cançons, permet al programador descriure les cançons, aquesta descripció permet a la ràdio conèixer aquestes cançons i treballar-hi amb elles.
- **Eines que facilitin la gestió de la ràdio** dotar al programador de la ràdio unes eines que ens facilitin la feina d'omplir la ràdio de cançons: permetre pujada massiva de cançons (agrupades dins d'un fitxer comprimit), poder extreure la informació (autor, nom de la cançó ...) del fitxer sonor que ens permet classificar la cançó evitant la classificació manual per part de l'usuari.
- **Servei d'estadístiques.** Dins del repertori de tasques que ofereix la part de gestió de la ràdio hi haurà una d'estadístiques que permetrà al programador saber com està funcionant aquesta. Ens permetran conèixer els següents aspectes:
    - **Conèixer dades de les cançons que conté la ràdio**. Per exemple: Saber quantes cançons conté un gènere, quantes han estat definides segons uns certs atributs ...etc. Dades de computació.
    - **Dades d'accés d'oients**. Saber quants oients té la ràdio, per franja horàries ...etc.
    - **Dades de valoració de la musica**. Conèixer quines són les cançons més valorades per l'audiència.

Resumint es donar una sèrie d'eines al programador per tal que alimenti la ràdio de música, defineixi la música i controli el funcionament de la ràdio.

2.2.3.Emetre la música

Aquesta és la funcionalitat responsable de dur a terme l'emissió de la música / cançó a través d'Internet i així pugui ser escoltada a través de la web. En aquest cas la ràdio no tan sols haurà d'agafar la música i enviar-la per la xarxa de tal manera que pugui ser escoltada via web sinó que haurà de fer un pas anterior, en aquest cas la ràdio ha de ser capaç de decidir quina cançó emetre. Per tant tenint en compte diferents paràmetres la ràdio escollirà una cançó, aquests paràmetres hauran de valorar:

- **Conèixer la seva audiència**. Això ho fa a través de:
    - les valoracions dels oients sobre les cançons emeses.
    - el pes d'aquestes valoracions.
    - quin pes té la gent que valora sobre el reste de gent que està escoltant la ràdio.
    - com evoluciona l'audiència, si es perden oients al posar una cançó.
    - la vida d'aquestes valoracions. Si es tenen en compte tots les valoracions que s'han fet sobre una cançó o només les més recents.

- quantes cançons emeses recentment es tenen en compte per tal de generar la nova cançó.

- **L'historial de cançons emeses**. Cada quan permetem que es repeteixi una cançó, cada quan permetem que es repeteixi una cançó d'un mateix autor.
- **La definició i classificació de les cançons** realitzades pel programador de la ràdio per tal de trobar cançons similars escaients al gust musical de l'audiència.

2.2.4.Escoltar música

La ràdio ha de permetre que l'oient pugui escoltar via web la ràdio, per tant haurà d'incloure una interfície que pugui ser visualitzada a través d'un client web. Aquesta interfície ha de permetre al oient:

- Escoltar la cançó.
- Visualitzar la descripció de la cançó, conèixer la fitxa de la cançó, en quin àlbum ha estat inclosa, quin autor és, quin artista la dur a terme, la història ...etc. En resum una fitxa de la cançó.
- Conèixer el context de les cançons emeses, saber quina ha estat la darrera cançó emesa i quina serà la propera.
- Conèixer les estadístiques de valoracions de les cançons.
- Poder consultar històric de les cançons emeses, així poder veure l'evolució musical de la ràdio i poder donar valoracions sobre cançons ja emeses.

2.2.5.Valorar la música

Des de la funcionalitat abans descrita l'oient podrà també donar a conèixer la seva opinió sobre les cançons emeses, d'aquesta manera la ràdio podrà aprendre el gust musical de l'audiència.

Per tant l'usuari podrà fer valoracions senzilles: la cançó emesa dient si li agrada o no. Però també podrà fer valoracions més complexes proposar un gènere, estil musical o artista o alguna característica musical (algun dels atributs amb què des de la gestió de la ràdio hem definit aquestes cançons). Les valoracions simples tan sols les podrà realitzar una sola vegada per cançó.

3. Especificació

L'etapa d'especificació parteix del anàlisis de requeriments previ per obtenir una descripció funcional completa, no ambigua, de l'aplicació. No s'entra en detalls sobre com es construeix o codifica aquestes funcionalitats (això es deixa per les fases de diseny i implementació).

3.1. Models de cas d'ús

El model de casos d'ús és una primera descripció externa d'una aplicació.

Descriu les operacions que realitza l'aplicació i l'ús que diferentes tipus d'usuaris fan d'aquestes operacions.

Externa perquè té un diagrama d'alt nivell d'abstracció, que no entra en detalles d'implementació.

Descrivim aquest model mitjançant:

- Diagrames UML de cas d'ús.
- Descripció textual dels diferents casos d'ús.

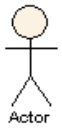
3.1.1.Actors de l'aplicació

La ràdio té dos tipus d'usuaris ben diferenciats, sobretot a l'hora d'interaccionar amb ella, els dos perfils tenen al seu abast un conjunt de funcionalitats bastant diferents; mentre que uns seran els responsables de la ràdio els altres la gaudeixen. Per tant tenim:

- **Administradors / Programadors** Són els que s'encarreguen de gestionar la ràdio, de donar vida a la ràdio, ja que són els que l'alimenten de música i la classifiquen i defineixen per tal de donar-li una informació afegida que ens permet tractar la música més enllà d'un arxiu sonor. Aquests usuaris tenen un accés privat a aquesta gestió de la ràdio, han d'accedir a través d'un codi d'usuari i una clau d'accés.
- **Oients** Són els usuaris de la ràdio, els que usen la ràdio, que escolten la ràdio. L'accés d'aquests serà lliure, però no són uns oients passius sinó que poden interaccionar amb ella valorant la música que la ràdio emet.
- **Sistema** Anomenem d'aquesta manera l'usuari que fa totes aquelles tasques que es duen a terme sense interaccionar amb un usuari real. Per exemple les funcionalitats de validar l'accés d'un usuari a la part restringida, generar la següent cançó, inferir noves similituds entre cançons.

3.1.2.Diagrames de casos d'ús

Per la seva simplicitat y expresivitat, el diagrama de casos d'ús són especialment útil per descriure els requerimients funcionals. Els nostres diagrames de casos d'ús tenen els següents elements:



Actor

És un **perfil** que juga un usuari amb respecte al sistema.

Cas d'ús

És una **operació** que realitza l'aplicació per ordre d'un agent extern o invocada per un altre cas d'ús.

Comunicació

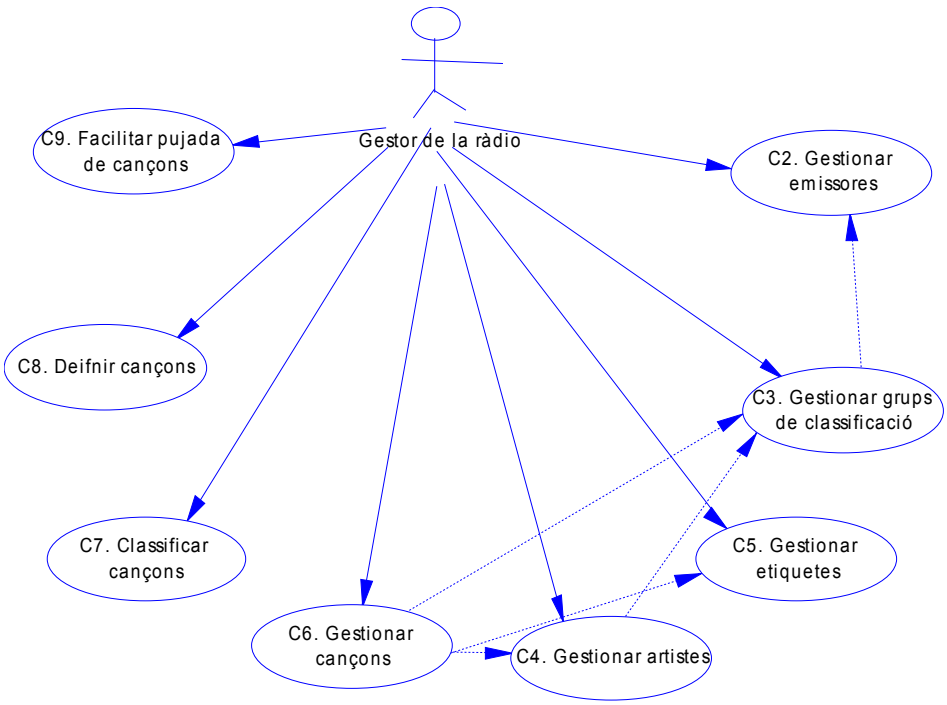
Aquesta relació expressa la **invocació** que fa un actor (o una altre cas d'ús) d'un cas d'ús.

Dependència

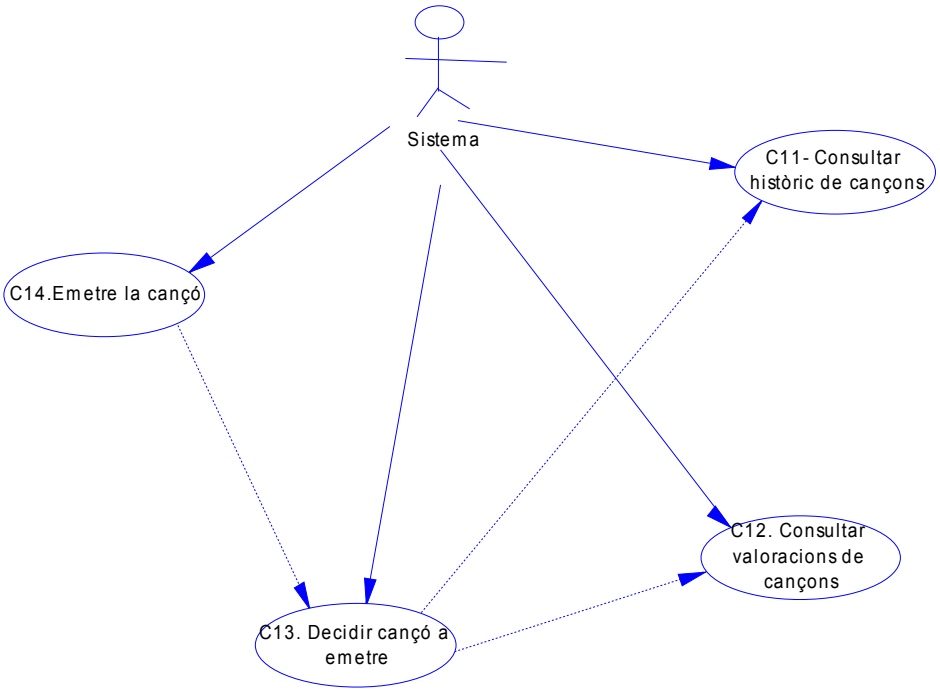
Aquesta relació expressa la **dependència** que té un cas d'ús d'un altre cas.

Per evitar mezclar casos de uso, farem diferents diagrames que agrupen els casos d'ús de funcionalitats relacionades.

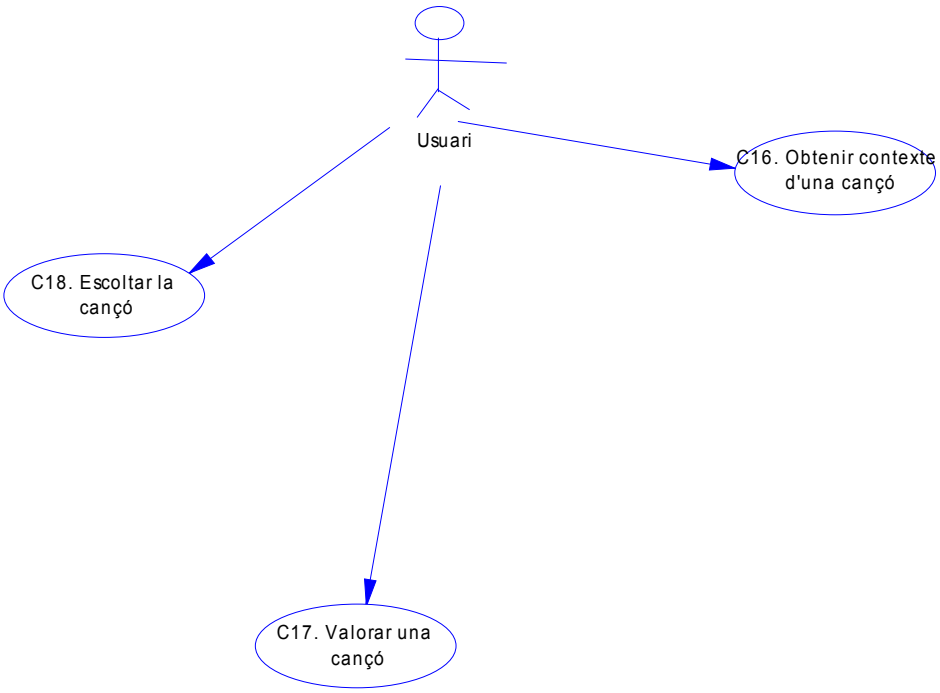
Gestió del magatzem musical de la ràdio



Emetre cançons



Escoltar cançons



3.1.3.Especificació dels casos d'ús

En aquest punt fem una descripció textual dels diferents casos d'ús que contempla l'aplicació, també els agrupem per funcionalitats relacionades. Amb els casos d'ús dividim en petits processos que s'hauran d'implementar per a dur a terme les funcionalitats que requereix la aplicació.

- **Gestió del magatzem musical de la ràdio**

  Cas C1. Validació usuari

  Actors: Sistema, Administrador

  Descripció: ens valida que l'usuari que vol accedir a la part restringida de l'eina té els privilegis necessaris per entrar-hi.

  Subcasos de C2. Gestionar emissores

  Cas C2. Gestionar emissores

  Actors: Administrador

Descripció: ens llista les emissores ordenades alfabèticament pel nom per tal que des d'aquest llistat ens permet editar o eliminar l'emissora.

Cas C2.2. Crear emissora

Actors: Administrador

Descripció: es mostra un formulari per tal de poder crear una nova emissora.

Cas C2.3. Modificar emissora

Actors: Administrador

Descripció: es mostra un formulari omplert per les dades de l'emissora que es desitja modificar i ens permet editar i realitzar els canvis desitjats.

Cas C2.4. Eliminar emissora

Actors: Administrador

Descripció: permet eliminar emissores de l'eina. Se'ns mostra una fitxa de l'emissora que es desitja eliminar i se'ns demana la confirmació per tal de validar la seva eliminació o no, si es confirma l'eliminació l'emissora s'eliminaria de l'eina.

Subcasos de C3. Gestionar grups de classificació

Cas C3.1. Llistar gèneres

Actors: Administrador

Descripció: ens llista els gèneres ordenats alfabèticament pel nom per tal que des d'aquest llistat ens permet editar o eliminar els gèneres o estils.

Cas C3.2. Crear gèneres

Actors: Administrador

Descripció: es mostra un formulari per tal de poder crear un nou gènere o estils. En aquest formulari si es tracta d'un estil s'haurà d'omplir el gènere al que està associat, per tant aquest cas d'ús hi ha una dependència del cas d'ús C3.1.

Cas C3.3. Modificar gènere

Actors: Administrador

Descripció: es mostra un formulari omplert per les dades del gènere que es desitja modificar i ens permet editar i realitzar els canvis desitjats.

Cas C3.4. Eliminar gènere

Actors: Administrador

Descripció: permet eliminar un gènere o estil de l'aplicació. Se'ns mostra una fitxa del gènere o estil que es desitja eliminar i se'ns demana la confirmació per tal de validar la seva eliminació o no, si es confirma l'eliminació del gènere s'eliminaria de l'aplicació.

Subcasos de C4. Gestionar artistes

Cas C4.1. Llistar artistes

Actors: Administrador

Descripció: ens llista els artistes ordenats alfabèticament pel nom per tal que des d'aquest llistat ens permet editar o eliminar els artistes.

Cas C4.2. Crear artistes

Actors: Administrador

Descripció: es mostra un formulari per tal de poder crear un nou artista. En aquest artista se li podrà associar un gènere o estil per tant aquest cas depen del cas d'ús C3.1 Llistar gèneres.

Cas C4.3. Modificar artista

Actors: Administrador

Descripció: es mostra un formulari omplert per les dades de l'artista que es desitja modificar i ens permet editar i realitzar els canvis desitjats.

Cas C4.4. Eliminar artista

Actors: Administrador

Descripció: permet eliminar un artista de l'aplicació. Se'ns mostra una fitxa del artista que es desitja eliminar i se'ns demana la confirmació per tal de validar la seva eliminació o no, si es confirma l'eliminació de l'artista s'eliminaria de l'aplicació.

Subcasos de C5. Gestionar etiquetes

Cas C5.1. Llistar etiquetes

Actors: Administrador

Descripció: ens llista les etiquetes ordenades alfabèticament pel nom per tal que des d'aquest llistat ens permet editar o eliminar les etiquetes.

Cas C5.2. Crear etiqueta

Actors: Administrador

Descripció: es mostra un formulari per tal de poder crear una nova etiqueta.

Cas C5.3. Modificar etiqueta

Actors: Administrador

Descripció: es mostra un formulari omplert per les dades de l'etiqueta que es desitja modificar i ens permet editar i realitzar els canvis desitjats.

Cas C5.4. Eliminar etiqueta

Actors: Administrador

Descripció: permet eliminar una etiqueta de l'aplicació. Se'ns mostra una fitxa de l'etiqueta que es desitja eliminar i se'ns demana la confirmació per tal de validar la seva eliminació o no, si es confirma l'eliminació de l'etiqueta s'eliminaria de l'aplicació.

Subcasos de C6. Gestionar cançons

Cas C6.1. Llistar cançons

Actors: Administrador

Descripció: ens llista les cançons ordenades alfabèticament pel nom per tal que des d'aquest llistat ens permet editar o eliminar les cançons.

Cas C6.2. Crear cançó

Actors: Administrador

Descripció: ens mostra un formulari per tal de poder crear una nova cançó.

Cas C6.3. Modificar cançó

Actors: Administrador

Descripció: es mostra un formulari omplert per les dades de la cançó que es desitja modificar i ens permet editar i realitzar els canvis desitjats.

Cas C6.4. Eliminar cançó

Actors: Administrador

Descripció: permet eliminar una cançó de l'aplicació. Se'ns mostra una fitxa de la cançó que es desitja eliminar i se'ns demana la confirmació per tal de validar la seva eliminació o no, si es confirma l'eliminació de la cançó s'eliminaria de l'aplicació.

Cas C7. Classificar cançons

Actors: Administrador

Descripció: permet ubicar una cançó en les diferents possibles classificacions (gènere, estil i artista) que ens permet l'aplicació. Associem a les cançons una classificació. Aquest cas d'ús fara crides als casos d'ús C3.1 Llistar gèneres i C4.1 Llistar artistes.

Cas C8. Definir cançons

Actors: Administrador

Descripció: permet definir una cançó sota atributs definits pel propi programador. Aquest cas d'ús fara crides als casos d'ús C5.1 Llistar etiquetes.

Cas C9. Pujada flexible de cançons

Actors: Administrador

Descripció: el motiu d'aquest cas d'ús es facilitar la gestió al programador de la ràdio, estalviar-li tenir que fer les definicions i pujadas d'arxius manual un a un. Aquest cas d'ús llegirà tota la informació sobre la cançó d'un arxiu mp3, o d'un fitxer empaquetedor (format zip) que contingui un conjunt de fitxers mp3.

Cas C10. Consultar estadístiques de la ràdio

Actors: Administrador

Descripció: ens permet obtenir dades que ens facilitin el coneixement del ús de la ràdio.

- **Emetre cançons**

  Cas C11. Consultar històric de cançons

  Actors: Sistema

  Descripció: Obtenim el llistat de cançons emeses prèviament per una emissora.

  Cas C12. Consultar valoracions de cançons

  Actors: Sistema

  Descripció: Obtenir les valoracions realitzades pels usuaris sobre una cançó concreta.

  Cas C13. Decidir cançó a emetre

  Actors: Sistema

  Descripció: El responsable de decidir quina serà la següent cançó a emetre per la ràdio. Fa ús dels dos casos d'ús anteriors.

  Cas C14. Emetre una cançó

  Actors: Sistema

  Descripció: Transmetre la música extreta de l'arxiu sonor a través de la xarxa fins a l'olient.

- **Apendre similituts**

  Cas C15. Apendre gustos de l'audiència

  Actors: Sistema

  Descripció: És el responsable de fer l'estudi de les valoracions realitzades pels usuaris per tal d'extreure regles de similitud entre cançons.

- **Escoltar cançons**

  Cas C16. Consultar dades de la cançó

  Actors: Oient

Descripció: L'usuari visualitzarà tota la informació associada a una cançó.

Cas C17. Escoltar una cançó

Actors: Sistema

Descripció: El sistema reprodueix l'audio de l'arxiu sonor de tal manera que sigui escoltat per l'oient.

Cas C18. Valorar una cançó

Actors: Oient

Descripció: Permet a l'usuari transmetre la seva opinió sobre una cançó.

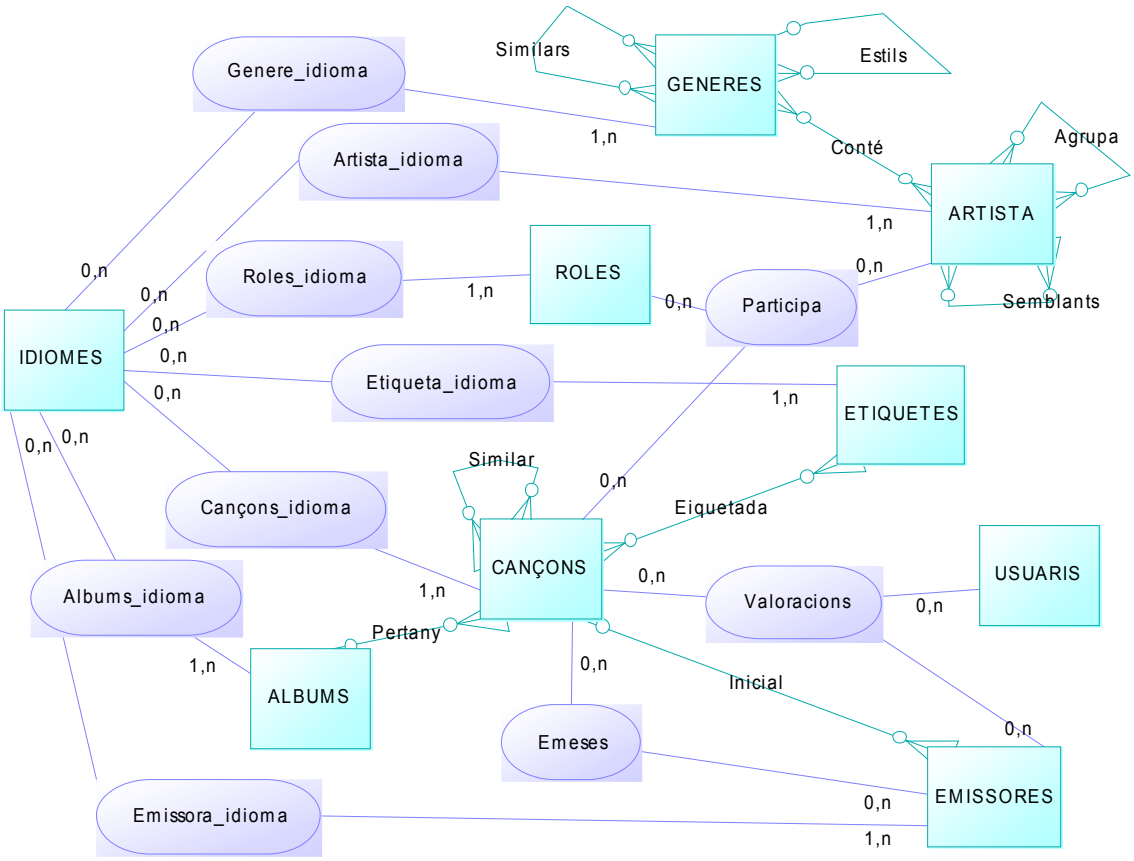
3.2. Model conceptual

En el punts anteriors al definir la majoria de funcionalitats que conté la ràdio, hem fet referència a diferents tipus d'entitats de dades que utilitza la ràdio per emmagatzemar, classificar, definir la música i poder decidir la cançó a emetre; així com per informar a l'oient sobre la cançó que està emetent, el context d'aquesta. En aquest punt expliquem de forma més detallada aquest conjunt de dades lògiques, que implica la implementació de l'eina, i les seves relacions entre elles.

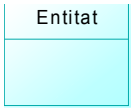
Per especificar aquest model conceptual utilitzem:

- Una **visió gràfica** que ens contempla tot el model de dades implementat per la realització de l'eina, en aquest cas utilitzem un diagrama conceptual basat en la metodologia Merise.
- Una **descripció textual i detallada** de totes les entitats i relacions entre elles que implica el model de dades implementat. Es tracta d'explicar amb més detall el diagrama conceptual.

3.2.1.Diagrama

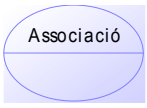


En aquest gràfic s'ha fet servir la notació següent:



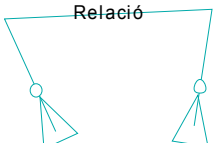
Entitat

És el grup de dades que tenen sentit propi i existència pròpia dins del projecte que hem desenvolupat.



Associació

Ens indica les relacions entre entitats, on aquestes relacions impliquen que tinguin propietats pròpies .



Relació

És la forma de relacionar-se bàsica entre dues entitats

3.2.2.Descripció de les entitats de dades

En aquest punt expliquem de manera més precisa cada entitat implicada en la ràdio, el que significa l'entitat i les diferents propietats de cada entitat. En el cas de la ràdio hem detectat les següents entitats:

- **EMISSORES:** Sota aquesta entitat s'agrupen les dades de les diferents instàncies de la ràdio que podem crear sobre el magatzem comú de música disponible. Cada instància, abans en la introducció ens hem referit a elles com a emissores, té unes característiques pròpies, una audiència pròpia i una emissió única.

Propietat	Tipus	Definició
Oid	**Enter**	Identificador únic de la instància de la ràdio, la emissora. Ens permet distingir de manera única les diferents emissores que sosté el nostre projecte.
CançonsPrevies	**Num.**	Indica el nombre de cançons, emeses, precedents de les que tindrem en compte les seves valoracions a l'hora de decidir la següent cançó. Aquesta propietat ens permetrà definir la velocitat d'adaptació de l'emissora a l'audiència que tingui en aquell moment. Nombre més baix, més s'adapta a la audiència d'aquell moment però més difícil d'adquirir un estil propi.
HistoricValoració	**Num**	Ens indica fins quant de temps enrere tenim en compte les valoracions sobre les cançons. Estarà indicat en unitats de dies, també ens indica la velocitat d'adaptació de la ràdio, quan menor sigui el valor significa que més fidel és a l'audiència actual de l'emissora. Si el valor és negatiu significa que es tenen en compte totes les valoracions realitzades en el temps de vida de la emissora.
RepeticióCançóDia	**Num.**	Ens indica el número de vegades que es pot repetir una cançó en un dia (24 hores).
RepeticióCançó	**Num.**	Ens indica el número de cançons que han de ser emeses abans de repetir una cançó, té relació amb la propietat anterior tan sols el tindrem amb compte aquest valor en el cas que la propietat anterior

		sigui major a 1.
RepeticióArtista	**Num.**	Ens indica el número de cançons que han de ser emeses abans de repetir un artista.
AfegirRegla	**Num.**	Ens indica el nombre d'usuaris que han de coincidir en una regla de comportament per tal d'inferir que dues cançons són similars.
PropagarRegla	**Booleà**	Ens indica si propaguem la regla, apresa de la similitud entre dues cançons, a les categories de classificació, en aquest cas pren el valor "cert".
SIClassificació	**Booleà**	Ens indica si a l'hora de decidir la següent cançó es tenen en compte els criteris de similitud entre categories de classificació de cançons (gèneres, estils i artistes).
SIDefinició	**Booleà**	Ens indica si a l'hora de decidir la següent cançó es tenen en compte els criteris de similitud entre les definicions de les cançons (etiquetes usades).
SIReglaApresa	**Booleà**	Ens indica si a l'hora de decidir la següent cançó es tenen en compte les regles apreses a partir del comportament dels usuaris.
Creació	**Data**	Data en la que s'ha creat la ràdio.

Nota: Aquesta entitat té més propietats pròpies (nom, descripció) però pels requeriments analitzats la ràdio ha de ser multi idioma cosa que fa que aquestes propietats depenguin del idioma, per tant són propietats pròpies de l'associació entre aquesta entitat i l'idioma i ho definim en el següent punt.

- **GÈNERES:** Sota aquesta entitat agrupa els diferents gèneres i subgèneres (estils) musicals en els que podem classificar una cançó, els nivells de classificació. Els subgèneres pertanyen a un gènere concret. Per ex. Gènere -> Rock; Subgènere -> Hard Rock.

Propietat	**Tipus**	**Definició**
Oid	**Enter**	Identificador únic del nivell de classificació.
URL	**Text**	L'adreça, URL, de la pàgina Web que ens permet accedir a més informació del

		gènere o estil.
Imatge	**Text**	Nom de la imatge que ens il·lustra aquest gènere o estil.
Creació	**Data**	Data en la que s'ha creat el gènere.

Nota: Aquesta entitat té més propietats pròpies (nom, breu explicació) però pels requeriments analitzats la ràdio ha de ser multi idioma cosa que fa que aquestes propietats depenguin del idioma, per tant són propietats pròpies de l'associació entre aquesta entitat i l'idioma i ho definim en el següent punt.

- **ARTISTA:** Sota aquesta entitat mantenim la informació sobre totes les persones, pot ser una persona individual o un col·lectiu, que han estat implicades / relacionades en una cançó. Per ex. Bob Dylan, The Cult.

Propietat	Tipus	Definició
Oid	**Enter**	Identificador únic del artista.
URL	**Text**	L'adreça, URL, de la pàgina Web que ens permet accedir a més informació de l'artista.
Imatge	**Text**	Nom de la fotografia de l'artista.
Naixement	**Data**	Data en la que va néixer l'artista o es va formar el col·lectiu.
Nacionalitat	**Text**	País d'on prové l'artista

Nota: Aquesta entitat té més propietats pròpies (breu biografia) però pels requeriments analitzats la ràdio ha de ser multi idioma cosa que fa que aquestes propietats depenguin del idioma, per tant són propietats pròpies de l'associació entre aquesta entitat i l'idioma i ho definim en el següent punt.

- **ROLES:** Agrupem els diferents perfils que pot tenir un artista al col·laborar en una cançó. Per ex. Guitarrista, Autor ... etc. Les propietats d'aquesta entitat són totalment dependents del idioma en que es visualitzi la ràdio, està més explicat en el punt que parlem de les relacions i associacions.
- **ALBUMS:** Ens permet mantenir la informació sobre els àlbums musicals en els que es troben agrupades les cançons.

Propietat	Tipus	Definició
Oid	**Enter**	Identificador únic de l'àlbum.
Any	**Num.**	Any en el que es va produir l'àlbum.
Portada	**Text**	Nom de la fotografia que il·lustra la portada de l'àlbum.

Creació	**Data**	Data en la que s'ha creat l'àlbum.

Nota: Aquesta entitat té més propietats pròpies (explicació adicional) però pels requeriments analitzats la ràdio ha de ser multi idioma cosa que fa que aquestes propietats depenguin del idioma, per tant són propietats pròpies de l'associació entre aquesta entitat i l'idioma i ho definim en el següent punt.

- **CANÇONS:** Agrupa les diferents cançons que tenim disponibles per les diferents entitats.

Propietat	Tipus	Definició
Oid	**Enter**	Identificador únic de les cançons.
Nom	**Text**	L'adreça, URL, de la pàgina Web que ens permet accedir a més informació del gènere o estil.
Any	**Text**	Nom de la imatge que ens il·lustra aquest gènere o estil.
Duració	**Num.**	Temps que dura la cançó expressada en segons.
Fitxer	**Text**	Nom del fitxer sonor associat a la cançó.
Creació	**Data**	Data en la que s'ha creat la ràdio.

Nota: Aquesta entitat té més propietats pròpies (explicació adicional) però pels requeriments analitzats la ràdio ha de ser multi idioma cosa que fa que aquestes propietats depenguin del idioma, per tant són propietats pròpies de l'associació entre aquesta entitat i l'idioma i ho definim en el següent punt.

- **IDIOMES:** Agrupa els diferents idiomes en els que està disponible les interfícies de gestió de la ràdio i la execució de les diferents emissores. Per. Català, Anglès ..etc

Propietat	Tipus	Definició
Locale	**Text**	Codi identificador del idioma, ens base en els codis ISO pel que fa referència als països (http://www.ics.uci.edu/pub/ietf/http/related/iso639.txt)
Descripció	**Text**	Explicació sobre el llenguatge al que fa referència el locale.

- **ETIQUETES:** Ens permet reunir sota aquesta entitat les diferents etiquetes que usarem per definir les cançons. Per ex. Melòdica, Suau, Romàntica, Enèrgica... etc

Propietat	Tipus	Definició
Oid	**Enter**	Identificador únic de l'etiqueta.
Creació	**Data**	Data en la que s'ha creat la ràdio.

Nota: Aquesta entitat té més propietats pròpies (nom, explicació) però pels requeriments analitzats la ràdio ha de ser multi idioma cosa que fa que aquestes propietats depenguin del idioma, per tant són propietats pròpies de l'associació entre aquesta entitat i l'idioma i ho definim en el següent punt.

- **USUARIS:** En aquesta entitat reunim els diferents usuaris que s'han connectat a la ràdio, com l'accés pels oients és obert tan sols tindrem informació d'aquells usuaris que tenen les "cookies" activades cosa que ens permet mantenir un seguiment d'ells i amb això conèixer els seus gustos.

Propietat	Tipus	Definició
Oid	**Enter**	Identificador únic de l'usuari.
Cookie	**Text**	Text identificador únic de l'usuari, valor de la "cookie" que ens permet identifica a l'usuari que ens connecta a la ràdio.
Creació	**Data**	Data en la que s'ha creat la ràdio.

3.2.3.Relacions / Associacions entre les entitats de dades

En aquest punt expliquem les diferents relaciones i associacions que hi ha entre les entitats, a continuació definim textualment els diferents elements que formen part d'aquest conjunt.

Hi ha un subconjunt d'associacions que ens permeten implementar tots aquells aspectes, de les entitats, dependents del llenguatge en el que es visualitzi la ràdio, són:

- **GENERE_IDIOMA:** Ens permet estendre l'entitat GÈNERE per recollir les propietats que depenen de l'idioma escollit per l'oient o el programador. Aquesta associació ens relaciona l'entitat GÈNERE amb l'entitat IDIOMA, amb una cardinalitat de n a n.

  L'associació té les següents propietats pròpies:

Propietat	Tipus	Definició
Nom	**Text**	Nom propi del gènere o estil. Per exemple Rock.
Explicació	**Text**	Breu explicació que ens permet informar de les característiques del gènere o estil.

- **ARTISTA_IDIOMA:** Ens permet estendre l'entitat ARTISTA per recollir les propietats que depenen de l'idioma escollit per l'oient o el programador. Aquesta associació ens relaciona l'entitat ARTISTA amb l'entitat IDIOMA, amb una cardinalitat de n a n.

  L'associació té les següents propietats pròpies:

Propietat	**Tipus**	**Definició**
Explicació	**Text**	Breu biografia de l'artista, permeten ampliar la informació d'aquest.

- **ROLES_IDIOMA:** Ens permet estendre l'entitat ROLES per recollir les propietats que depenen del idioma escollit per l'oient o el programador. Aquesta associació ens relaciona l'entitat ROLES amb l'entitat IDIOMA, amb una cardinalitat de n a n.

  L'associació té les següents propietats pròpies:

Propietat	**Tipus**	**Definició**
Nom	**Text**	Nom propi del rol. Per exemple Veu, Voice.

- **ETIQUETA_IDIOMA:** Ens permet estendre l'entitat ETIQUETA per recollir les propietats que depenen del idioma escollit per l'oient o el programador. Aquesta associació ens relaciona l'entitat ETIQUETA amb l'entitat IDIOMA, amb una cardinalitat de n a n.

  L'associació té les següents propietats pròpies:

Propietat	**Tipus**	**Definició**
Nom	**Text**	Nom, serveix com a identificador lògic, de la etiqueta. Per exemple Romàntic, Romantico.
Explicació	**Text**	Breu explicació sobre l'ús o motiu d'aquesta etiqueta.

- **CANÇONS_IDIOMA:** Ens permet estendre l'entitat CANÇONS per recollir les propietats que depenen del idioma escollit per l'oient o el programador. Aquesta associació ens relaciona l'entitat CANÇONS amb l'entitat IDIOMA, amb una cardinalitat de n a n.

  L'associació té les següents propietats pròpies:

Propietat	Tipus	Definició
Explicació	**Text**	Breu explicació que ens permet donar informació sobre la cançó a l'oient.

- **ALBUMS_IDIOMA:** Ens permet estendre l'entitat ALBUMS per recollir les propietats que depenen del idioma escollit per l'oient o el programador. Aquesta associació ens relaciona l'entitat ALBUMS amb l'entitat IDIOMA, amb una cardinalitat de n a n.

  L'associació té les següents propietats pròpies:

Propietat	Tipus	Definició
Explicació	**Text**	Breu explicació que ens permet donar informació addicional de l'àlbum perquè pugui ser consultada per l'oient.

- **EMISSORES_IDIOMA:** Ens permet estendre l'entitat EMISSORA per recollir les propietats que depenen del idioma escollit per l'oient o el programador. Aquesta associació ens relaciona l'entitat EMISSORA amb l'entitat IDIOMA, amb una cardinalitat de n a n.

  L'associació té les següents propietats pròpies:

Propietat	Tipus	Definició
Nom	**Text**	Nom propi de l'emissora. Per ex. Ràdio Marina
Explicació	**Text**	Breu explicació que permet detallar les característiques de les emissores. Per ex. Emissora de ràpida adaptació.

El reste de relacions / associacions que ens implementen altres característiques necessàries pel projecte són:

- **Similars:** Ens permet relacionar aquells gèneres o estils que a criteri del programador són similars, que si agrada un gènere pot agradar l'altre. Aquesta associació ens relaciona l'entitat GENERE en si mateixa, amb una cardinalitat de n a n.
- **Estils:** Ens permet informar de la relació jeràrquica de l'entitat GÈNERES doncs ens informa de quins estils té un gènere i a quin gènere pertanyen uns estils. Aquesta associació ens relaciona l'entitat GENERE en si mateixa, amb una cardinalitat de n a n.
- **Conté:** Ens permet ubicar als artistes dins d'uns gèneres o estils concrets. Aquesta associació ens relaciona l'entitat GENERE amb l'entitat ARTISTA,

amb una cardinalitat de n a n; ja que un artista pot estar ubicat en diferents gèneres o estils i un estil o gènere pot contenir diferents artistes.

- **Ubica:** Ens permet ubicar a les cançons dins d'uns gèneres o estils concrets. Aquesta associació ens relaciona l'entitat GENERE amb l'entitat CANÇONS, amb una cardinalitat de n a n; ja que una cançó pot estar ubicat en diferents gèneres o estils i un estil o gènere pot contenir diferents cançons.
- **Agrupa:** Ens permet relacionar diferents artistes dins d'un mateix col·lectiu (per ex. grup musical). Aquesta associació ens relaciona l'entitat ARTISTA en si mateixa, amb una cardinalitat de n a n.
- **Semblants:** Ens permet relacionar aquells artistes que a criteri del programador són similars, que si agrada un artista concret pot agradar un de similar. Aquesta associació ens relaciona l'entitat ARTISTA en si mateixa, amb una cardinalitat de n a n.
- **Participa:** Ens permet definir la implicació d'un artista en una cançó i amb quin perfil s'ha implicat en la cançó. Aquesta associació ens relaciona l'entitat ARTISTA, ROLES i CANÇONS amb una cardinalitat de n a n.
- **Etiquetada:** Ens informa de les diferents etiquetes amb la que ha estat definida una cançó. Aquesta associació ens relaciona l'entitat CANÇONS amb l'entitat ETIQUETA, amb una cardinalitat de n a n; ja que una cançó pot estar etiquetada per diferents etiquetes i sota una etiqueta pot haver diferents etiquetes.
- **Similar:** Ens permet relacionar cançons que siguin similars, que puguin agradar a una audiència de gustos similars. Aquesta associació ens relaciona l'entitat CANÇONS en si mateixa, amb una cardinalitat de n a n.
- **Pertany:** L'usem per indicar a quin àlbum pertanyen unes cançons concretes. Aquesta associació ens relaciona l'entitat CANÇONS amb l'entitat ALBUMS, amb una cardinalitat de n a n.
- **Conté:** Permet definir la primera cançó que emetrà una emissora quan es desconeix totalment l'audiència, és sota criteri del programador definir aquesta propietat o no. Aquesta associació ens relaciona l'entitat CANÇONS amb l'entitat EMISSORA, amb una cardinalitat de 1 a n; ja que una emissora només té una cançó inicial i una cançó pot ser la inicial de diferents emissores.
- **Valoracions:** Permet emmagatzemar les valoracions que facin els usuaris sobre les cançons. Aquesta associació ens relaciona l'entitat CANÇONS amb l'entitat USUARIS, amb una cardinalitat de n a n. Ens interessa guardar informació sobre aquestes valoracions per tant aquesta associació té atributs propis:

Propietat	**Tipus**	**Definició**
Valoració	**Num**	Valor 1 en el cas que agradi o 0 en cas contrari

Data	**Date**	Data en què s'efectua la valoració.
ReglaInferida	**Boolea**	Ens permet indicar si d'aquesta valoració s'ha extret una regla de similitud o no.

- **Emeses:** Guardem l'historial de cançons que ha emès una emissora. Aquesta associació ens relaciona l'entitat EMISSORA amb l'entitat CANÇONS, amb una cardinalitat de n a n.

  L'associació té les següents propietats pròpies:

Propietat	Tipus	Definició
Data	**Date**	Data en que s'ha emès la cançó.
TotalOients	**Num**	Nombre d'usuaris connectats a l'emissora en el moment d'emetre la cançó.

4. Disseny

En aquest punt expliquem com durem a terme la realització de totes les funcionalitats que requereix el projecte.

A l'hora de dissenyar el software que dóna resposta a les necessitats del nostre projecte ha de seguir les següents pautes:

- Modular, per poder reutilitzar el codi i peces de software.
- La solució plantejada és usa la metodologia orientada a objectes.
- Tria un patró de disseny.
- Utilització d'estàndards de codificació.
- Utilitzar software lliure.

Els objectius de la sèrie dels punts que ara entrarem a detallar són:

- Quin model d'implementació utilitzem.
- Com organitzem el software que hem de crear, l'arquitectura del software propi.
- Quines peces de software externes utilitzarem i la justificació del seu ús.

4.1.Plataforma usada

Un dels requeriments no funcionals de la nostra aplicació es que la seva execució ha de ser independent del sistema operatiu per aquest motiu triem utilitzar la plataforma Java, també degut a l'alta experiència i coneixement d'haver treballat amb ella. Veiem les característiques més importants d'aquesta plataforma.

Es tracta d'una plataforma capaç d'executar aplicacions escrites en el llenguatge de programació Java. La plataforma inclou nombroses eines per facilitar el desenvolupament. Les aplicacions s'executen sobre un software anomenat Java Virtual Machine (JVM, maquina virtual Java). Existeixen implementacions de la màquina virtual per a les plataformes de hardware més populars (Windows, Linux, Solaris) de forma que una aplicació Java pot correr en una gran variedad de sistemas diferents amb total **independència del sistema operatiu**, tot això sense necessitat de recompilació.

La plataforma Java 2 de Sun entre altres està formada per:

Java 2 Platform Standard Edition (J2SE), composada per la màquina virtual de Java i un conjunt de llibreries pensades per facilitar el desenvolupament d'**aplicacions de propòsit general**. Algunes d'aquestes llibreries són: java.io (gestió de la E/S), java.sql (acces a base de dades relacionals)...etc.

Java 2 Platform Enterprise Edition (J2EE) és la part de la plataforma per desenvolupar i executar **aplicacions distribuides entre diferents servidors**, està basada en l'execució de components modulars en servidors d'aplicacions. Algunes de les llibreries que pertanyen a aquesta part de la plataforma són: javax.servlet y javax.servlet.jsp (processament d'una petició al servidor web i generació de la resposta), javax.xml (XML)...etc

Algunas conseqüències d'això:

Diferents entorns de treball. La flexibilitat de la plataforma Java ens permite desenvolupar l'aplicació en un ordinador personal amb sistema operatiu Windows i després instalarla amb facilitat a un servidor (per exemple, un SPARC amb sistema operatiu Solaris).

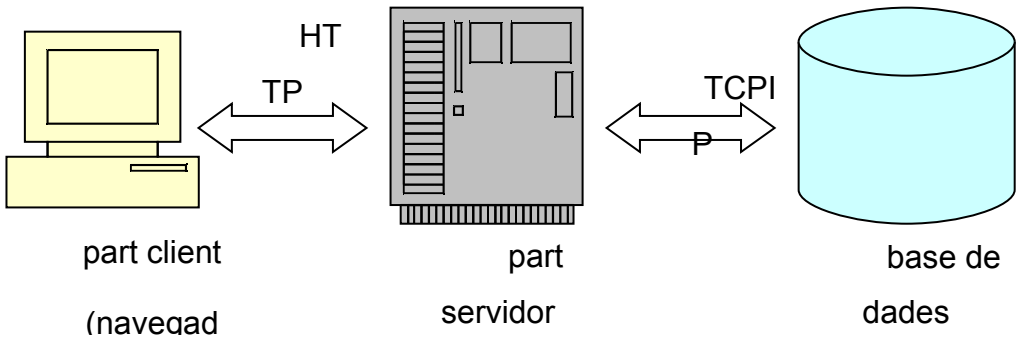
Orientació a objectes. Java és un llenguatge orientat a objectes, per tant encaixa amb el disseny triat per implementar l'aplicació.

4.2.Arquitectura de l'aplicació, patrons de diseny

L'eina ha de ser una aplicació implementada sobre Web. Veiem quins són les directrius, *blueprints* i patrons arquitectònics que han estat desenvolupats per la comunitat d'enginyers de software en Java.

4.2.1.Arquitectura distribuïda

El patró arquitectònic més general a tenir en compte per una aplicació web és el d'una **arquitectura distribuïda** entre les següents parts, que interactuïn en mode client/servidor:



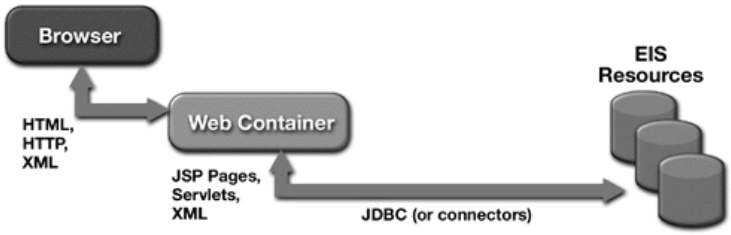
Part client. Consisteix en un **browser o navegador d'Internet** (per exemple, Mozilla o Internet Explorer). A petició de l'usuari, el browser envia peticions HTTP a la part servidor, rep una resposta en els llenguatges XML o HTML i presenta aquesta resposta de manera comprensible a la pantalla de l'usuari.

Part servidor. Consisteix en un **servidor d'aplicacions web** capaç de rebre a través de la xarxa peticions en el protocol HTTP, procesarles, construïr una

resposta, codificada típicament en els llenguatges XML o HTML, i enviar-la a la part client, que se la presentarà a l'usuari.

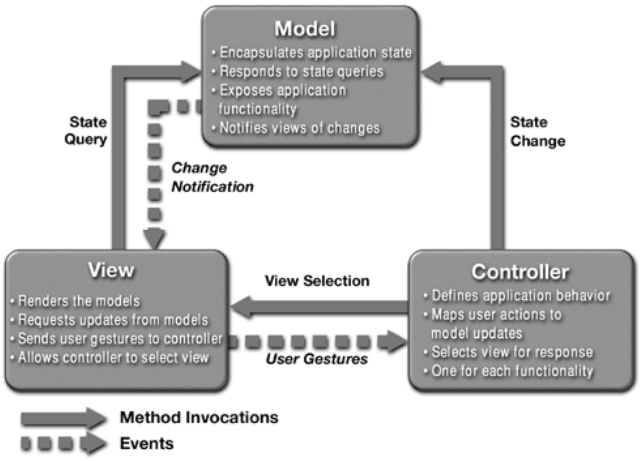
Base de dades. Gestiona de manera eficient i segura la persistència de les dades empleades per l'aplicació. Pot ser un SGBD relacional, una base de dades orientada a objectes, etc. Té que existir un protocol de comunicació entre la part servidor i la base de dades. Per exemple: JDBC per l'accés des d'aplicacions Java a bases de dades relacionals.

El següent diagrama [Singh, Stearns, Johnson 2002] es una traslació d'aquest model a l'arquitectura Java:



4.2.2.Patró Model / Vista / Controlador

Un dels patrons arquitectònics més utilitzats per aplicacions interactives és el denominat **Model-Vista-Controlador**. Aquesta arquitectura divideix les aplicacions en tres capes de responsabilitats clarament diferenciades. Cada capa realitza determinades tasques y ofereix serveis a les altres capes. Una bona descripció d'aquest patró es troba en [Singh, Stearns, Johnson 2002];



La **capa model** representa la informació i la lògica de negoci, és a dir, les operacions que permeten accedir i modificar les dades de l'aplicació. Quan hi ha un canvi en el model aquesta capa s'encarrega de notificar-ho a les vistes. Proporciona

a la capa vista els mètodes necessaris per consultar l'estat del model. Proporciona a la capa controlador la funcionalidad encapsulada pel model.

La **capa vista** presenta el contingut del model. Accedeix a les dades i defineix tècniques de presentació. Recull la interacció de l'usuari i com a resultat d'aquesta interacció envia peticions a la capa controlador. Quan hi ha canvis en el model, és informada per la capa model i notifica a l'usuari d'aquests canvis.

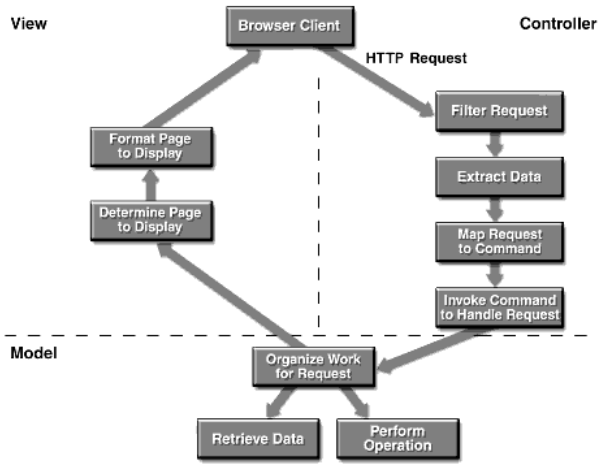
La **capa controlador** rep les peticions dels usuaris, invoca a les accions de la capa model que hi ha que cridar per resoldre cada petició, i finalment selecciona la vista que s'utilitzarà per presentar el resultat a l'usuari.

Les aventatges d'aquesta separació són:

- Es redueix la duplicació de codi, al estar clarament delimitades les responsabilitats de cada capa.
- Facilitat de manteniment, el codi estructurat es més fàcil de mantenir, a més pot haver perfils especialistes en cada capa.
- Facilitat per afegir nous tipus de client, doncs no implica modificar la lògica de negoci. Per exemple, afegir a una aplicació web soport per a clients tipo PDA, tan sols hauriem d'afegir la vista adequada aquest dispositiu.

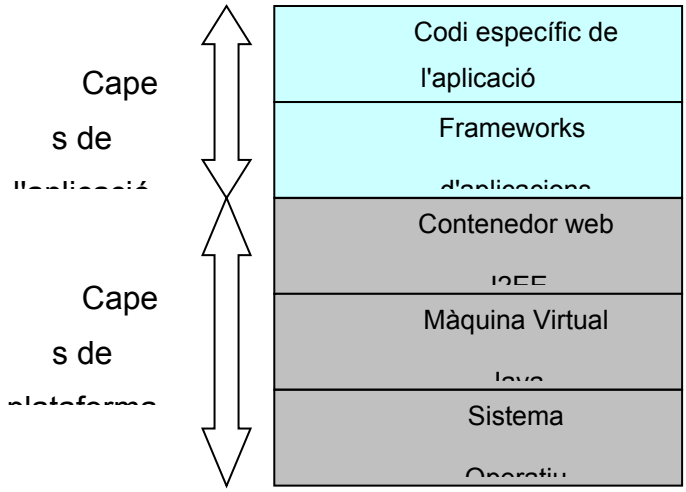
En l'escenari d'aplicació web, l'usuari interacciona des de la part client mitjançant un navegador. Les seves peticions HTTP són enviades al servidor d' aplicacions web. El servidor d'aplicacions web utilitza el controlador per atendre la petició; el controlador fa operacions sobre la capa model i tria una vista per presentar el resultat a l'usuari. La vista genera la resposta en llenguatge HTML o XML i la envia de tornada al navegador client.

La següent figura (extreta de [Singh, Stearns, Johnson 2002]) ilustra com encaixen les diverses peces de la arquitectura web Java amb les tres capas del patró:



4.2.3.Capes de plataforma vs capes d'aplicació

Una aplicació basada en la plataforma Java disposa de les següents capes :



Las capes que formen part de la plataforma són: **sistema operatiu**, sobre la que s'executa la **Java Virtual Machine**, sobre la que s'executa el **servidor d'aplicacions web J2EE**.

La nostra aplicació s'executa sobre el servidor d'aplicacions web J2EE. A la seva vegada, aquesst diagrama mostra la nostra aplicació dividida en dues capes: **framework d'aplicació** i **codi específic d'aplicació**. El framework d'aplicació ofereix un conjunt de funcionalitats necessàries per construir les nostres aplicacions i que no són provistes pel servidor d'aplicacions web J2EE. Aquest framework d'aplicació ofereix aquestes funcionalitats, pot ser compartit entre diferents aplicacions diferents.

En una aplicació web, el framework d'aplicación proporciona funcionalitats comuns como: atendre peticions, invocar mètodes del model, seleccionar i ensamblar les vistes.

Les ventatges d'utilitzar un framework d'aplicació existent, en lloc de construir un propi, són:

- Els frameworks existents estan basats en patrons arquitectònics de provada utilitat, els implementen de manera efectiva.
- Proporcionen l'estructura sobre la que construir aplicacions complexes de la forma més ordenada i sistemàtica possible.
- Al tenir les aplicacions més estructurades es facilita el testeig unitari i el manteniment.

- Proporcionen gran part de las funcionalitats necessaris pel 99% de les aplicacions: templates (plantilles), internacionalització (I18N – multiidioma), sistema de log (rastreig per comprovar possibles errors i fer un seguiment de l'eina).
- Promouen la utilització d'estàndards.

- Permeten separar els diferents rols dels desenvolupadors: desarrolladores especializados en capa presentación y escribir JSPs, especialistas en capa modelo y persistencia de datos, etc.

4.2.4.Alguns patrons de diseny apropiats

Extraiem de [Singh, Stearns, Johnson 2002] diferents patrons de diseny útils per la nostra aplicació:

Patró Filtre Interceptor

Aquest patró intercepta les peticions que arriben al servidor i aplica un pre-procés i un post-procés. Per exemple: comprensió de la resposta, validació de la IP de la que procedeix la petición, incloure cookies en la resposta, etc. Una implementació d'aquest patró és la interfície javax.servlet.Filter.

Patró Ajudant de Vista

L'ajudant de vista encapsula porcions de vista que desitgem ocultar o reaprofitar, amb l'objectiu de fer més sencilles les vistes. Un exemple d'aquest patró són els tags de JSP de Java.

Patró Vista Composta

Consisteix en crear plantilles pels diferents models de vistes o de porcions de vistes que es repeteixen freqüentment. Per ex. Existeix una implementació d'aquest patró anomenada Tiles integrada dins del framework Struts (http://struts.apache.org).

Patró Controlador Frontal

Consisteix en tenir un controlador central que rep totes les peticions de l'aplicació. El controlador frontal decideix a quins components distribueix la petició per donar resposta a la mateixa. **Utilizem aquest patró per disenyar la capa controlador de la nostra arquitectura Model / Vista / Controlador**.

Patró DAO (Data Access Object)

L'objectiu d'aquest patró es encapsular totalment les classes que accedeixen a la base de dades per que els detalls de la mateixa siguin totalment transparents al reste de capes. **Utilizarems aquest patró per gestionar la persisteècia de les entitats de la capa Model**.

4.2.5.Sistema d'informació o base de dades

Una altra part vital de la plataforma de la nostra aplicació és el sistema d'informació corporativa o base de dades, ja que per la nostra aplicació es necessari persistir les dades pel seu ús posterior. També hem de tenir en compte un dels

requeriments no funcionals que el software que desenvolupem sigui independent de les peces de software amb les que relaciona, per tant la nostra aplicació ha de ser independent del gestor de base de dades.

Veiem como independitzar la nostra aplicació:

→ del **tipus** de sistema d'informació (relacional, orientada a objectes, bd XML, etc)

i

→ d'un **fabricant** determinat (Oracle, mySQL, SQL Server, etc).

Les diferents entitats de l'aplicació (emissores, cançons, artistes, etc) es **modelitzen com objectes**. Aquests objectes persistirán en el sistema d'informació escollit mitjançant el patró Data Access Objects (DAO). L'aventaja d'aquest patró es que independitza les aplicacions del destí final de les dades que manipula. Com resultat, si hi ha que fer alguna manipulació sobre com s'emmagatzema les dades, n'hi ha prou amb reimplementar els objectes DAO mantenint la API; d'aquesta manera no serà necessari fer cap canvi al reste de components de l'aplicació.

Els diferents objectes DAO ofereixen mètodes per accedir i modificar els diferents tipus d'objectes ocultant els detalls del sistema d'informació concret. Aquest podria ser una BD relacional, ficheros XML planos, etc.

El patró DAO es considera una bona pràctica de programació en aplicacions J2EE. Amb aquest patró independitzem totalment la nostra aplicació dels detalls concrets sobre el sistema d'informació usat.

5. Implementació

En aquest punt definim com tècnicament duem a terme el que hem disenyat, els punts següents els dividirem en dos grans grups:

- per una banda expliquem com implementem el patró de disseny triat (Model /Vista / Controlador)
- per l'altra banda entrem més al detall en els dos algoritmes més complexos : "Decisió de la següent cançó", "Apendre similituds de cançons".

5.1. Organitzacions de les classes: model / vista / controlador

Durant la fase de diseny establim que implementem l'aplicació seguint el patró arquitectònic model / vista / controlador. La implementació de les classes de les diferents capes s'han organitzat per packages (organitzacions. Veiem cómo els diferentes packages donen soport a aquestes capes

Capa controlador	**pfc.radio.action** Implementa la lògica dels casos d'ús.
Capa model	**pfc.radio.dao** Gestió de la persistència de les entitats. **pfc.radio.model** Entitats persistens de l'aplicació (per ex. Emissora, canço ...etc)
Capa vista	**pfc.radio.form** Beans per la presentació i validació dels formularis. **pfc.radio.cancons** Beans per la presentació del context de la cançó que està escoltant l'oient en el moment actual. **Jsp** Els fitxer encarregats de implementar les interfícies d'usuari de l'aplicació.

Transversal a totes les capes	**pfc.radio.util** Ens implementa funcionalitats útils comuns ques

5.2.Capa model

La capa model està encapsulada en diferents packages que s'utiltizen pel següent:

Package pfc.radio.model

- Entitats pròpies de l'aplicació (àlbums, gèneres, cançons ...etc), modeladas como Java Beans persistentes.
- Estructures de dades necessàries per donar soport a processos complexos (històric de cançons, registre de votacions, extreure regles de similitud entre cançons recorrent històric de votacions dels usuaris).
- Java Beans utilitzats per intercanviar informació amb les capes Controladora i Vista.

Package pfc.radio.dao

- Objectes DAO que gestionen la persistència de les classes persistents, seguint el patró DAO (Data Access Object) del que hem parlat a la fase de diseny.

Veiem tots aquests components detalladament:

5.2.1.Beans persistents

Existeix un package que encapsula totes les **entitats persistents** de l'aplicació herramienta de gestión de horas: Package pfc.`radio.model`

Conté classes Java que representen en memòria les entitats de l'aplicació: emissores, cançons, gèneres, etc. També conté classes que formen **estructures de dades** necessàries per processar algunes de las operacions més complexes de l'aplicació (apendre similituds entre cançons, decidir següent cançó a emetre) com serien les entitats votacions i cançons emesses. Finalment, el package model inclou també beans que s'utilitzen per passar informació des de la capa model cap a les capes controlador i vista.

Per evitar **un ús inadequat de la memòria**, materialitzem les entitats només quan les anem a fer servir, doncs per exemple seria del tot insostenible mantenir tota la informació associada a les cançons, que tenim disponibles per a les emissores, en memòria ja que pot ser un nombre molt alt, i les persistirem a la base de dades per assegurar que les podem materialitzar quan ens interessei. La gestión de la materialització i la persistència d'entitats a una base de dades es realitzarà mitjançant la capa DAO.

5.2.2.Capa DAO

Els objectes d'aquesta capa estan encapsulats dins del següent package Java: pfc.`radio.dao`. Recordem que aquesta capa és la responsable d'interactuar amb el sistema físic que utilitzem per persistir els objectes per tant la que ens independitza totalment l'aplicació d'aquest sistema físic triat.

Per a cada entitat persistent de l'aplicació, aquest package conté una classe DAO. Els nombres de les classes DAO s'han escollit per facilitar l'identificació de quina classe gestionen. Per exemple:

Entitat: Emissora; Java Bean (Classe lògica) : pfc.radio.model.Emissora; Classe DAO:pfc.radio.DAO.EmissoraDAO.

5.2.3.Elecció d'un framework de persistència OO-relacional

Existeix una API de Java anomenada JDBC que és la que permet realitzar operacions amb la base de dades des d'una aplicació Java.

JDBC és un API a molt baix nivell. L'usuari de JDBC ha d'escriure moltes línies de codi i manegar objectes com Conexions, ResultSets, Statements. A més a pesar de l'adopció del estàndard SQL-92 és difícil escriure sentències SQL que siguin 100% compatibles amb la majoria de SGBDs del mercat.

Dins la comunitat Java existeixen diferents **frameworks de persistència** open source. En el nostre cas triem un framework de persistència OO-relacional, és a dir que ens converteix els objectes en taules d'una base de dades relacional.

Les **aventatges** d'aquest tipus de frameworks són les següents:

- Ofereixen una API de programació senzilla 100% independent del SGBD concret.
- Inclouen funcionalitats avançades per millorar la eficiència: caché, connection pooling, gestió de transaccions distribuïdes.

Les **desvantatges**:

- Si bé ofereixen APIs molt més senzilles i potents que JDBC, la **corba d'aprenentatge** de la seva utilització és llarga.
- Determinades operacions són més **eficients** si es realitzen directament en JDBC.

Decidim utilizar un framework de persistència perque facilita enormement la construcción d'una capa de persistència. Ara ha de saber-se quin triar.

L'escollit és: **Hibernate** (http://www.hibernate.org). Hibernate és probablement el framework de persistència OO-relacional més utilitzat. Els motius del seu èxit són: potència, senzillesa, versatilitat. Veiem algunes de les seves característiques més rellevants:

- Permet persistir qualsevol classe Java sobre una taula de base de dades definint un fitxer XML de mapejos.
- Ofereix una API senzilla i potent per manipular la base de dades.

- Permet realitzar transaccions combinant operacions de bd realitzades amb la API de Hibernate i operaciones realitzades en JDBC.
- Implementa polítiques de concurrència optimistas y pesimistas. Per a les nostres aplicacions utilitzarem una política optimista, basada en l'ús de **versions**.
- Implementa un potent sistema de caché que millora enormement el rendimient de les operacions a base de dades.

5.2.4.Implementació dels objectes DAO: JDBC/Hibernate

Hem comentat abans que el framework de persistència que utilitzarem permet combinar operacions en la API pròpia del framework amb operacions JDBC. D'aquesta manera podrem desenvolupar la capa DAO amb rapidesa i optimitzarem algunes operacions concretes, particularment costoses, directament en JDBC.

Totes les operacions JDBC i Hibernate estan encapsulades dins dels objectes DAO, de forma que resulten totalment transparents al reste de les capes de l'aplicació.

Dins de la capa DAO tenim la classe DAOConnection que gestiona la connexió a la base de dades i l'estat de la transacció. La classe GenericDAO ens ofereix mètodes genérics per executar operacions sobre JDBC e Hibernate.

5.2.5.Concurrència

Totes les entitats persistents pròpies d'aquesta aplicació té un atribut TIMESTAMP. Aquest atribut també està present en les taules de la base de dades. S'utilitza per controlar les versiones de les diferentes entitats amb l'objectiu d'evitar errors de concurrència.

En un gestor de concurrència optimista quan una transacció modifica un objecte, registre o dada comprova la versió que està guardada: si la versió guardada és superior a la que ha modificat, significa que una altra transacció ha modificat la mateixa dada. Es diu que és una gestió **optimista** perquè es fa la hipotèsis que els accesos concurrents es produeixen rarament. Si es produeixen, l'última transacció es cancel·la i es torna un missatge d'error a l'usuari.

¿Podem considerar que accessos concurrents a una mateixa dada de la nostra aplicació es produiràn estranyament? Es dir, ¿podem acceptar la utilització d'una política optimista de concurrència? La resposta es SÍ, degut a dos motius:

- Les transaccions a la base de dades tan sols les du a terme un únic gestor.
- Les úniques transaccions que poden ser concurrents són les votacions però tan sols son insercions a la base de dades, en cap moment l'usuari pot modificar una votació, per tant no hi ha accés exclusiu a aquestes dades.
- Les operacions d'acces a dades són molt breus (no hi ha bloquejos).

5.3.Capa controlador

5.3.1.Elecció d'un framework M/V/C per l'aplicació web

Com ja hem comentat durant el diseny i seguint les recomanacions de Sun pel desenvolupament d'aplicacions web en Java, utilitzarem un framework existente pel desenvolupament d'aplicacions basades en M/V/C.

El framework escollit és Jakarta Struts (http://struts.apache.org). Els motius d'aquesta elecció són:

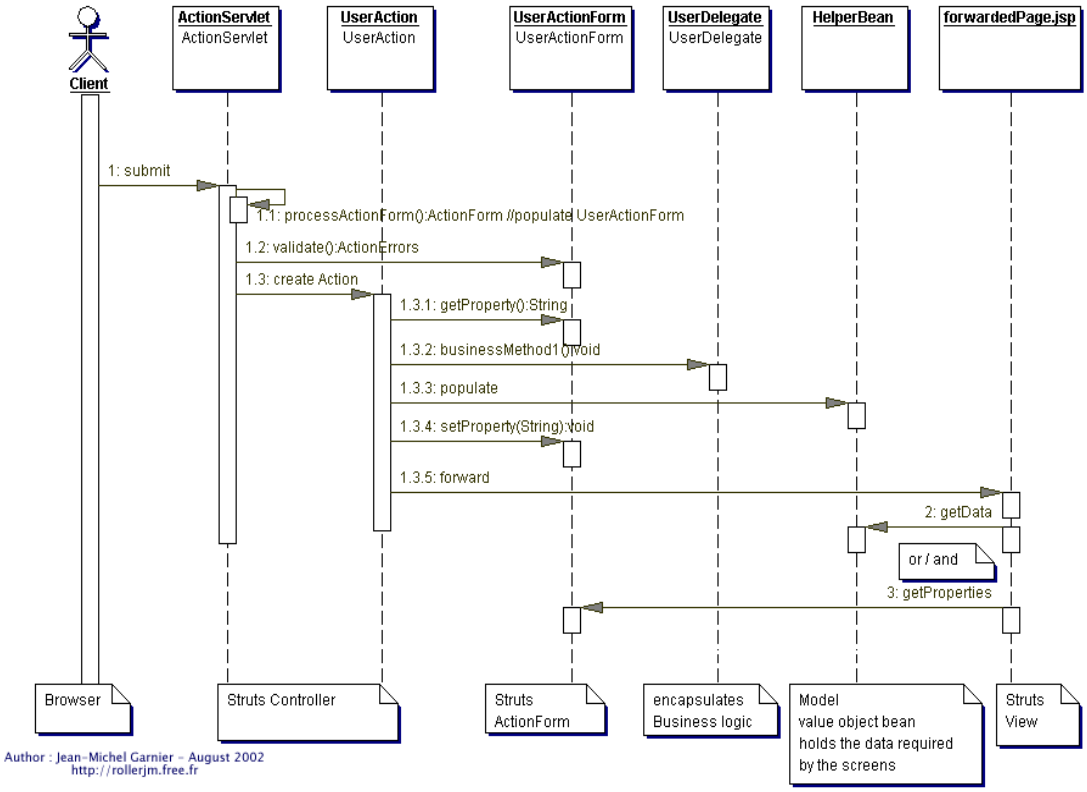
- Proporciona directrius molt precises per realitzar una aplicació estructurada i per tant més fàcil de mantenir o modificar.
- Multitut d'aplicacions implantades amb Struts garanteixen el seu èxit.
- Existeix un bon suport per desenvolupar aplicacions sota aquest framework.
- Inclou components que faciliten la construcció de la capa Vista.

5.3.2.Descripció de Struts

El component central de Struts és:

- El controlador, és a dir, un Servlet que rep les peticions HTTP. Aquest servlet consulta un fitxer de configuració XML que indica, per a cada tipus de crida: una classe que executarà la petició (la anomenada "action"), i una vista (generalment JSP) per mostrar els resultats.

El següent diagrama de seqüència resumeix com el controlador de Struts aten cada petició HTTP:



Struts, a més, inclou software per internacionalitzar les aplicaciones (suport multiidioma) i per construïr les vistes (llibreries de tags per a JSP).

5.3.3.Implementació de capa controladora basada en Struts

Com hem vist, el controlador de Struts distribueix les peticions a unas classes denominades "Actions" que són les que implementen els diferents casos d'ús. Aquestes classes hereden d'una "ActionBase" que forma part del framework. Les "actions" específiques de la nostra aplicació están recullides en el següent package:

Package pfc.radio.`action`

5.4.Capa vista

5.4.1.Capa vista basada en Struts

Com ja hem comentat en la secció dedicada a la capa controlador, el framework Struts inclou components per facilitar la construcció de les vistes. Struts defineix en el seu fitxe de configuració struts-config.xml quines JSPs s'utilitzen per resoldre cada cas d'ús. Es defineix en el fitxer a través d'aquest mapeig:

```
<action path="/admradio/entrar"
        type="radio.action.AdminAction"
        name="AdminForm"
        scope="request"
        validate="false">
        <forward name="success" path="/admin/jsp/inici.jsp" />
        <forward name="failure" path="/admin/jsp/error.jsp" />
</action>
```

Com pot observar-se en el fragment del fitxer de configuració, per a cada URL peticionada de l'aplicació es defineix: la classe Action que implementa la lògica del cas d'ús (atribut "type") i la JSP que construeix la vista (element "forward").

Las JSPs contenen el codi HTML que dona estructura a les pantalles de l'aplicació. A aquesta estructura es formateja segons les regles d'un fitxer d' **estils en cascada CSS**. El fitxer CSS determina tipografia a utilitzar, tamany, colors, la posició dels diferents elements de cada pantalla, l'estil de les caixes de text (vores, marges), etc.

5.4.2.JSP i taglibs

Les vistes de l'aplicació estan construïdes mitjançant JSPs. Una JSP és un fitxer que combina fragments de codi HTML (estàtic) amb petits scripts programats en Java que serveixen per les parts dinàmiques. Com el llenguatge Java no es barreja massa bé amb el HTML hi ha una altra tècnica per crear contingut dinàmic en les pàgines: la utilització de llibreríes de tags. Els tags encapsulen el codi Java en tags similars als del propi llenguatge HTML.

Existeixen multitut de llibreries de tags disponibles per usar en les JPSs. Struts inclou les següents llibreries, que utilitzem per programar les vistes de la nostra aplicació:

- Llibreria de tags LOGIC: permet codificar condicionals i bucles.
- Llibreria de tags HTML: facilita la construcció de formularis amb validació de camps en el costat servidor.
- Llibreria de tags BEAN: per presentar les diferents propietats de Beans Java.

També és possible desenvolupar llibreries de tags a mida per cada aplicació, en aquest cas no ho utilitzem.

5.5. Apendre similituds entre cançons

Pretenem fixar-nos amb les diferents valoracions que va fent l'audiència per si es segueix un patró del que puguem inferir similituds entre cançons no definides pel programador de la ràdio, d'aquesta manera fem que la emissora aprengui dinàmicament nous criteris de similitud que ens permeten apropar més el comportament de l'emissora a l'audiència real.

El sistema cada rastrejarà l'històric de valoracions per tal que coincideixi que dues cançons han estat valorades de la mateixa manera, en positiu o negatiu, un nombre igual o superior a la propietat AfegirRegla de l'emissora i aleshores inferir una regla de similitud, o positiva o negativa, entre aquelles dues cançons.

Això ho implementarem amb una consulta sobre la taula Valoracions i afegirem un registre a la relació Similar.

5.6. Decisió de la següent cançó

L'algoritme per seleccionar la cançó a emetre es basa en els següents criteris:

1. Històric de cançons emeses
2. Valoracions fetes pels usuaris
3. Criteris de similituds creats pel programador
4. Similituds apreses del seguiment de les valoracions dels usuaris
5. Tenir en compte els criteris de repetició
6. Tenir compte el nombre d'oients que van tenint cada cançó ens permet relativitzar les valoracions i si alguna cançó produeix fugida d'audiència.
7. Afegir un factor aleatori per poder introduir cançons que puguin sorpendre a l'audiència

L'algoritme implementat segueix els següents passos:

1. **Seleccionar candidates**.

   Treballem sobre el conjunt de cançons emesses anteriorment a l'actual, la quantitat de cançons a que ens remuntem és el nombre que hem definit al crear la emissora.

   Per tant per cada cançó del conjunt a estudiar farem:

   1. Si la cançó ha estat valorada almenys per un percentatge llindar de l'audiència d'aquell moment, per ex. Un 5% és un criteri configurable, i el 60% de les valoracions positives aleshores afegim al conjunt les següents cançons:

1. Les cançons similars pel criteri de similitud entre cançons, les puntuarem amb 5 punts.
2. Les cançons similars pel criteri de similitud per etiquetatge i no hagin estat introduïdes en el pas anterior.
3. Les cançons similars pel criteri de similitud de l'artista implicat en la cançó i no hagin estat introduïdes en els pasos anteriors.
4. Les cançons similars pel criteri de similitud de gènere o estil i no hagin estat introduïdes en els pasos anteriors.

2. **Seleccionar excloses**.

   És el mateix que el que el primer pas però en aquest cas tan sols treballem amb aquelles cançons que han estat valorades per sobre el llindar fixat i d'aquestes valoracions més d'un 60% negatives, aleshores cerquem les cançons similars de la mateixa manera que abans però les guardem a un conjunt de cançons que etiquetem com a excloses doncs són aquelles que no es poden emetre.

   A part diaquestes cançons afegim al conjunt d'excloses les següents cançons que no es poden afegir per criteris de repetició:

   1. Les cançons que no es poden repetir degut a què s'han emès ja el nombre màxim de vegades en aquell dia.
   2. Les cançons que no es poden repetir degut a què ja s'han emès cançons del mateix artista en les últimes cançons emesses.

3. **Indiferència.** Si la cançó sobre la que treballem no ha tingut prou pes les valoracions sobre l'audiència total o si la diferència entre les valoracions positives o negatives no tenen prou majoria per tant no tenim valoració correcta perquè ens provoqui cap acció sobre aquella cançó. En aquest cas no realitzem cap acció amb aquesta cançó.

4. **Netejar les cançons candidates**.

   En aquest pas eliminem del conjunt de candidates totes aquelles cançons que estan incloses en el conjunt d'excloses.

5. **Afegim cançons aleatòries.** Per afegir-li un factor sorpresa afegim un cert nombre de cançons aleatòries al conjunt de candidates. En aquest cas això ho realitzem tenint en compte que aquestes cançons no poden estar incloses ni en el conjunt de candidates ni en el conjunt de cançons excloses.

6. **Puntuem les cançons candidates.** Donem un valor a les cançons del conjunt candidates segons el motiu de similitud pel qual han estat escollides:

   1. 5 punts en el cas que la cançó hagi estat seleccionada a través del criteri de similitud entre cançons.
   2. 4 punts en el cas que la cançó hagi estat afegida a través de criteris de similitud d'etiquetatge.

3. 3 punts en el cas que la cançó hagi estat afegida per criteris de similitud d'artista.
4. 2 punts en el cas que la cançó hagi estat afegida per criteris de similitud per estil o gènere.
5. 1 punt en el cas que la cançó hagi estat afegida per criteri aleatori.

7. Assignar probabilitats a cada cançó candidata.

Repetim cada cançó dins del conjunt de candidates tantes vegades com la puntuació que se li ha assignat en el pas anterior. Un cop fet això desordenem el conjunt.

8. **Generar un número aleatori que ens permeti triar una cançó**.

Si el conjunt de candidates té membres generem una posició aleatòria del conjunt, i la posició aleatòria resultant ens fixarà la cançó seleccionada.

Si el conjunt de candidates arribes buit, triariem una cançó aleatòria entre tot el ventall disponible en el magatzem i que no estigués dins del conjunt de cançons excloses.

9. Registrem l'emissió

Un cop ja hem obtingut la cançó adient per l'audiència guardem un registre de la seva emissió a l'historial de canóns emeses per la emissora en concret amb la que treballem.

10. Recullim informació de la cançó i l'emetem per la xarxa.

En el cas que sigui la primera cançó a emetre d'una emissora emetrem la cançó que ha seleccionat el programador al crear la emissora o si està buida aquesta propietat ens dirigirem directament al pas 7 amb el conjunt de candidates buit.

6. Pla de proves

En aquest punt explicarem les diferents proves que s'han realitzat sobre l'eina per tal de garantir un bon funcionament d'aquesta.

6.1. Funcionament de l'eina

Sotmetem l'aplicació a un conjunt de proves per tal de comprovar el seu funcionament de l'eina si realment aprèn o no la ràdio, per això provem diferents casos i comprovem que les respostes s'atenen a la resposta esperada. Al realitzar aquestes proves també ens permet trobar els valors més adients per a l'algoritme de decisió, les proves ens permet ajustar aquests valors, en aquest cas valors que fan referència a:

- percentatge de valoracions sobre audiència total que s'ha d'obtenir per tal de prendre en consideració les valoracions dels usuaris.
- percentatge necessari del sentit d'una valoració sobre les valoracions total per tal de considerar que una valoració té un sentit o l'altre, sinó hi ha prou majoria s'estableix que agrada i desagrada de manera igual per tant no es fa cap acció associada doncs es produeix associada.
- percentatge d'aleatorietat.

Un parell d'exemples provats:

1. Dues emissores creades amb les mateixes característiques, al llarg de les seves primeres cinc cançons se li fan a una sempre 100 valoracions positives de cada cançó i comprovem que les cançons que comencen a generar són d'estils totalment oposats.
2. Dues emissores amb el mateix històric de valoracions però amb el nombre de cançons emeses que es consulten una molt menor que l'altre (2 contra 15), aleshores realitzem una valoració negativa massiva de la cançó que s'està emeten en aquell moment i comprovem que la resposta de la ràdio una s'adapta totalment a aquest canvi d'audiència i comprovem que canvia d'estil en canvi l'altra no s'adapta tan ràpidament.

6.2. Rendiment de l'eina

Per tal de garantir un correcte rendiment de l'eina mitjançant una eina de software lliure com el JMeter es faran simulacions d'estres sobre l'eina. Amb aquesta eina ens permet fer un conjunt de proves a l'eina simulant l'accés a l'eina de manera simultània de varis usuaris. Amb això volem provar el correcte funcionament de l'eina en situacions extremes i que els temps de resposta siguin raonables.

S'han realitzat proves amb diferent nombre d'oients conjunts de cinc, deu i cinquanta oients concurrents. I el temps de resposta domat és raonable.

7. Bibliografia

S'han consultat els següents llibres:

- *Enginyeria del Software: Especificació. Especificació de sistemes orientats a objectes amb la notació UML*

  Costal, Dolors / Sancho, M. Ribera / Teniente, Ernest

  Edicions UPC, 2000

- *Designing Enterprise Applications with the J2EE Platform, Second Edition*

  Singh, Inderjeet / Stearns, Beth / Johnson, Mark

  Addison-Wesley, 2002

I hem consultat les següents URL:

1. Web de classificació musical

   http://www.allmusic.com

2. Ràdios personals:
   1. http://www.last.fm
   2. http://www.pandora.com
   3. http://www.irateradio.com
   4. http://www.jamendo.com
   5. http://www.musicstrands.com
3. Music Gnome Project, projecte per definir tot tipus de música, defineix més de 400 atributs per una cançó: http://en.wikipedia.org/wiki/Music_Genome_Project.
4. http://struts.apache.org
5. http://www.hibernate.org