Reconeixedor Cançons

6 - juliol - 2012

Membres del grup

- Bernardo Miquel Riera
- Juan Gabriel Florit Gomila

ÍNDEX DE CONTINGUT

1 Introducció

La realització de la pràctica tenia com a objectiu principal el de fer un assaig de com podem afrontar un problema que a priori no sabem resoldre. És a dir, cercar-nos la vida, de manera que donat un enunciat, ens informem sobre com poder solucionar-lo. Això inclou reutilitzar el codi i llibreries que trobem per Internet, “No hem de descobrir el Mediterrani”, però sempre sabent que estam cercant.

Per realitzar el programa abans hem fet una mica de recerca bibliogràfica per tal d'utilitzar el trobat a la pràctica. A més de la informació proporcionada pel professor hem trobat alguna altra informació útil:

- Bioinformàtica. Adjunt la entrega trobaràs un directori amb el nom “bioinformàtica” en el qual hi ha dos PDFs de les assignatures “Genètica Molecular” (professora Barbarà Terrassa) i “Biologia Molecular de Sistemes” (professor Jordi Oliver), ambdues del Grau en Bioquímica de la UIB.

  En aquests PDFs s'explica com alinear seqüències gèniques (anotacions de les seqüències dels nucleòtids que formen el DNA) per comparar-les entre si i obtenir una puntuació que indica el grau de similitud. Algunes aplicacions són les de trobar seqüències consens, seqüències reguladores, identificar gens, construir arbres filogenètics, etc. No obstant, degut a la naturalesa de com es resolen aquests problemes, el coneixement es pot aplicar també a altres camps ja que en definitiva el que es fa és comparar dues seqüències, ja siguin de nucleòtids o de datapoint's d'una cançó codificada.

  Als PDFs es poden veure exemples de com es resolen aquests problemes i fins i tot hi ha una mica de pseudocodi. Al final no hem implementat el vist a aquests documents però ho hem trobat interessant i ho adjuntam com a tal.

- Llibreries d'Internet.
  - Per a la transformada ràpida de Frourier i treballar amb nombres complexes.
  - Per a llegir metadata de fitxers MP3 (dues famílies de llibreries en realitat).
  - Per a convertir arxius MP3 a WAV (JLayer: jl1.0.jar).
  - Per a usar el reproductor (BasicPlayer3.0), que alhora usa molts altres .jar com tritonus i vorbis.
  - Classe DataPoint treta de la web que ens vas facilitar.

2 Manual d'usuari

S'ha optat per fer-lo en forma de videotutorial. Es pot consultar descomprimint el .rar que hi ha dins el projecte Netbeans entregat. I el present document es pot veure des de el propi programa fent clic a 'Documentació', del menú opcions.

3 Limitacions de la pràctica

- **Reproductor.** No pot reproduir alguns fitxers d'àudio encara que sol poder fer-ho amb la gran majoria. Sobretot si es tracta de fitxers MP3 però també amb WAV i OGG (aquest darrer no s'ha comprovat). Aquesta limitació escapa al nostre control ja que es causada per les llibreries que utilitzam.

- **Metadata.** Al igual que en l'anterior cas, degut a les llibreries que s'utilitzen no sempre és possible llegir la metadata d'un fitxer MP3 (els altres no estan suportats). El motiu és que les llibreries estan fetes per llegir tan sols algunes especificacions ID3 dels tags dels MP3. Per més informació dirigir-se a http://www.id3.org/.

- **Matching de la BBDD amb una gravació des del micròfon.** Degut a que a una gravació hi influeix molt el renou ambient resulta complicat que el matching trobi coincidències correctes en una gravació procedent del micròfon tot i que s'hagi gravat una cançó que tenim codificada a la BBDD. No obstant, si s'obrin els fitxers de text de dues gravacions dels mateixos sons difereixen en quant als datapoint's, però s'observen els mateixos pics i patrons, de manera que si s'apliques un marge prou ampli podríem trobar coincidències positives i trobar la cançó correcta. El problema de donar un marge massa ampli és que també començaria a donar falsos positius amb altres cançons codificades. La solució no passa per ampliar el marge sinó a processar el so (aplicar filtres) de les gravacions per eliminar renous, com segurament fa Shazam. No obstant, això escapa a l'objectiu de la pràctica i no està implementat. Així doncs, el matching de la BBDD amb les gravacions fetes per l'usuari no sempre dona el resultat correcte.

- **Problemes per codificar arxius amb un frame rate molt alt.** Escapa al nostre control ja que depèn de la llibreria "JLayer", que provoca una excepció causada per haver consumit massa memòria quan l'arxiu té un frame rate molt elevat. Quan es dona aquest problema s'avorta la codificació de la cançó.

- **Noms d'arxius amb accents o caràcters especials.** Donen problemes novament degut a les llibreries que treballen amb el nom del fitxer, que no els suporten.

IMPORTANT: per tal d'evitar les limitacions explicades anteriorment és aconsellable usar els arxius d'àudio que adjuntam amb la pràctica i que estan dins el directori "musica_dexemple".

4 Algorisme de matching

4.1 Explicació

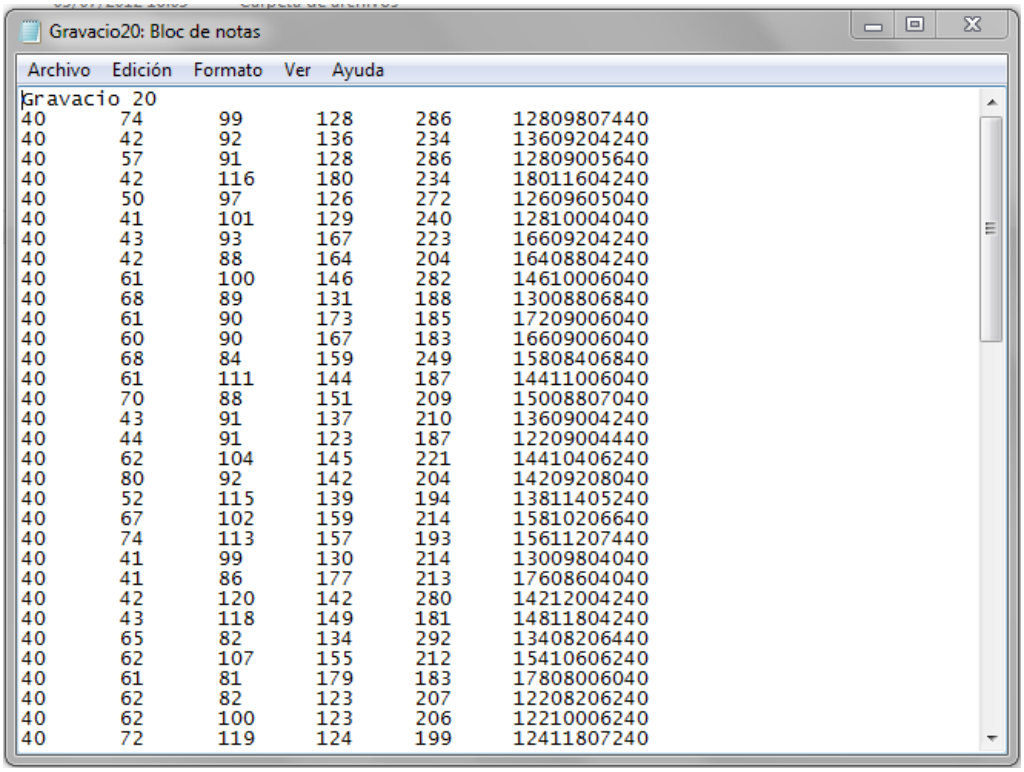
L'algorisme òptim que s'ha estudiat per implementar el procés de matching entre cançons estava basat en programació dinàmica i consistia en trobar, dins un conjunt de cançons o base de dades, la seqüències màxima comuna entre una cançó del conjunt i una cançó a cercar. Si tenim dues seqüències d'entrada $X(X_1,X_2,X_3,..,X_m)$ i $Y(Y_1,Y_2,Y_3,..,Y_n)$ i anomenam L(i,j) a la longitud de la seqüència màxima comú de X_i i Y_j ,és a dir, $X(X_1,X_2,X_3,..,X_i)$ i $Y(Y_1,Y_2,Y_3,..,Y_j)$, s'acompleix el principi d'optimalitat que ens assegura que una subseqüència de longitud màxima està composta per les subseqüències de longitud màxima del mateix problema reduït. Aleshores la funció recursiva que defineix l'algorisme seria:

$$L(i,j) = \begin{cases} 0 & si\ i = 0\ o\ j = 0 \\ L(i-1,j-1)+1 & si\ i \neq 0, j \neq 0, x_i = y_j \\ \text{máx}(L(i,j-1), L(i-1,j)) & si\ i \neq 0, j \neq 0, x_i \neq y_j \end{cases}$$

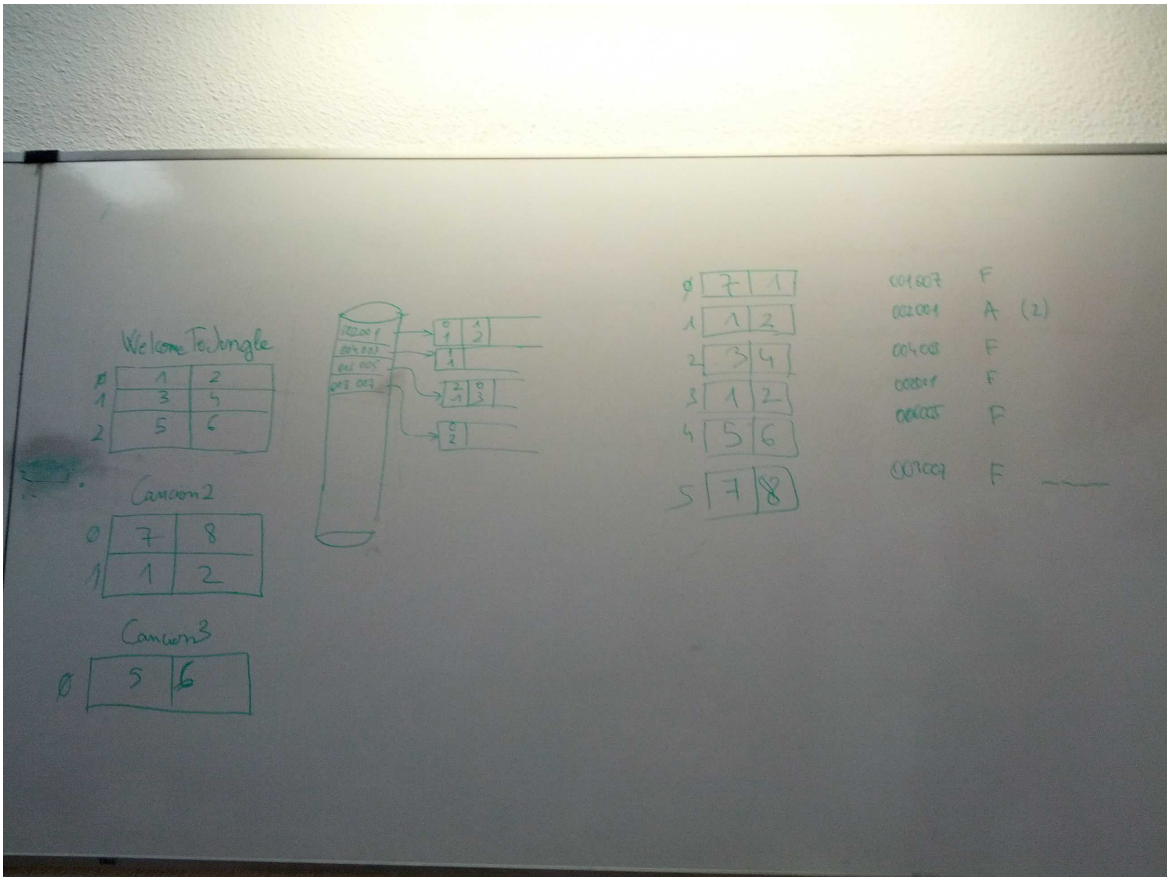
Un cop emplenada l'estructura de càlculs intermigs el que s'hauria de fer és reconstruir el camí que recorre la taula de manera que s'obté el màxim valor de coincidències. Hem fet molta recerca sobre el tema, hem llegit el llibre de Ricardo Peña la bibliografia de l'assignatura que explica el problema de la subseqüència màxima, hem investigat el problema del "Query by Humming", "Query by Humming basat en models ocults de Markov, i problemes de subseqüències màximes en altres disciplines, també hem visitat pàgines web per entendre a fons el funcionament d'aquest algoritme. Una pàgina que recomanam personalment per entendre bé el problema és: http://www.web4bio.com/assets/ProgramacionDinamica-v1.swf .

Finalment, tot i la recerca, hem consumit molt temps per entendre bé el problema i degut a la complexitat que representava implementar una funció d'aquest tipus hem optat per implementar un algorisme àvid que funciona bé amb una base de dades de fitxers de text. És a dir, per resoldre el problema, hem seguit una heurística que consisteix en triar la millor opció en cada passa local amb l'esperança d'arribar a una solució global òptima.

El punt de partida són els fitxers de DataPoints que genera, per cada àudio implicat, la mateixa aplicació. Un DataPoint consisteix en una tupla de dades formada per l'identificador d'una cançó i un instant de la mateixa cançó. L'estàndard per codificar una cançó consisteix en l'escriptura d'un fitxer amb les freqüències màximes per cada instant de temps d'una cançó. Aquest fitxer també segueix un format estàndard de l'aplicació, primer s'escriu el títol de la cançó, fitxer o gravació i tot seguir, per cada instant de temps, la combinació de freqüències màximes i la clau de hash única que genera la combinació.



Els DataPoints s'organitzen convenientment en una taula de dispersió que ens permet aconseguir a la vegada la seva combinació de freqüències màximes segons el criteri de codificació que esmentat. A continuació es mostra una imatge que representa l'estructura d'informació que empra l'aplicació. A l'esquerra es poden observar 3 cançons molt breus (de 3,2 i 1 instant de temps respectivament), al centre apareix la taula de dispersió dibuixada com a una base de dades de claus generades amb una llista de DataPoints consistents en dos nombres, l'instant de temps (en segons) i l'identificador de la cançó (la posició de la cançó a la base de dades). A la dreta del tot apareix un àudio de 6 instants de temps amb una lletra F o A on F representa un Fail (no es troba a la base de dades) i una A representa un Hit a la base de dades:



Per tant, inicialment es descarrega la base de dades sobre la taula de dispersió de manera que cada clau de la taula representa una combinació de freqüències única, aleshores, per cada combinació única de freqüències el valor a de la taula és una llista de DataPoints que conté la cançó i l'instant de temps en que s'han produït aquestes freqüències. Per tant, el temps de descàrrega és d'ordre lineal O(n) essent n el nombre d'entrades de la base de dades.

Un cop s'ha construït la taula de dispersió, per cada DataPoint de la gravació o fitxer es comprova l'existència de la combinació de freqüències, o el que és equivalent, es comprova si a la taula de dispersió existeix la clau que genera la combinació de freqüències. Si no existeix, evidentment no hi haurà cap tipus de coincidències, i si existeix, s'accedeix a la llista de DataPoint's en un temps O(1) constant i es recorre la seva llista de DataPoints seqüencialment, en un temps O(n), per trobar instants de temps idèntics. Si es troben coincidències en els instants de temps i la combinació de freqüències, s'introdueix a una llista de coincidències l'identificador de la cançó amb un nombre de coincidències de freqüències. Finalment, la cançó amb més coincidències de la llista serà la candidata a ser la cançó cercada.

4.2 Cost

Per tant, tal com s'ha comentat en el punt anterior, s'ha implementat un algoritme àvid que en cada DataPoint que comprova elegeix la cançó amb més coincidències. El temps de descàrrega de dades des del fitxer es fa en un temps lineal O(n) essent n les entrades o línies de la base de dades, posteriorment s'accedeix a les llistes de DataPoints en un temps O(1) gràcies a l'estructura de taula de hashing, tot seguit es recorre aquesta llista de DataPoints per cercar coincidències en l'instant de temps amb una complexitat O(n) essen n el nombre de DataPoint's de la llista, finalment es recorre un a llista petita de cançons candidates per extreure la informació de les cançons en un temps O(n) essent n el nombre de cançons coincidents. Per tant, seqüencialment s'executen 3 recorreguts d'ordre lineal O(n), resultant una complexitat asimptòtica O(n) tot i que som conscients de que la constant multiplicativa també influeix en l'hora d'accedir a fitxers tant per llegir com per escriure.

Finalment, durant les proves d'execució, el temps per codificar una cançó depèn de la longitud de l'àudio (evidentment), però només la cerca d'una d'aquesta cançó no supera mai el límit d'un segon.

5 Extres de la pràctica

5.1 Reproductor d'àudio

El reproductor d'àudio és capaç de reproduir arxius amb el format MP3, OGG i WAV. S'han implementat les accions de “play”, “pause” i “stop”.

A més, també s'ha inclòs un comptador de temps per saber en quin punt estam de la cançó i una barra de reproducció que també indica gràficament el temps transcorregut. La barra de reproducció s'ha fet mitjançant un Jslider que s'actualitza periòdicament sincronitzada amb el comptador de temps (un Jlabel).

5.2 Lectura metadata de fitxers d'àudio

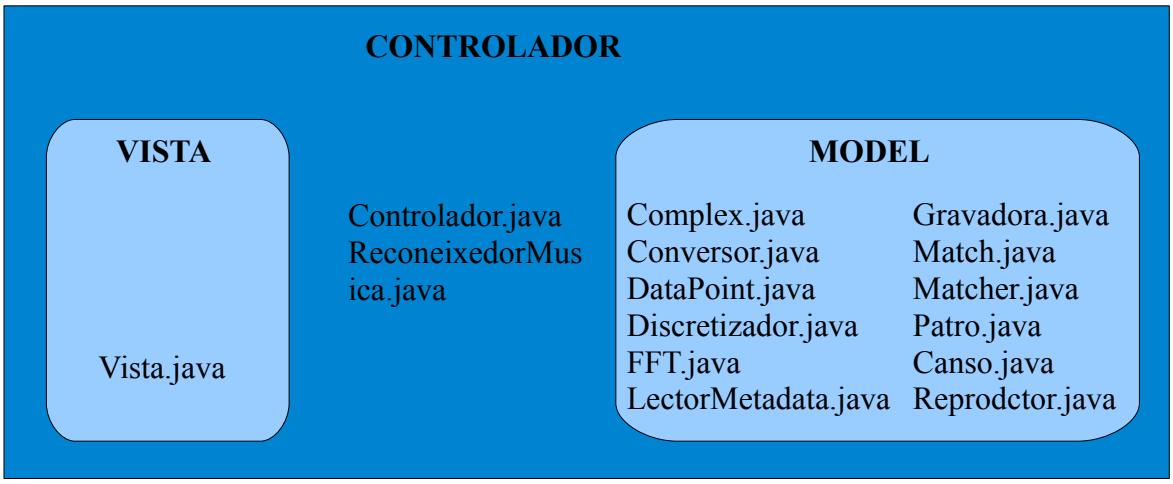
S'han emprat dos conjunts de llibreries per llegir la metadata dels fitxers d'àudio. En principi sols funcionen per arxius MP3.

La primera llibreria serveix per llegir els tags nom, grup/artista, àlbum, gènere i any. La segona llibreria serveix per llegir la portada/caràtula del disc. Com que sol fallar gran part de les vegades, quan es dona el cas, s'intenta cercar la portada d'entre una serie d'imatges que tenim emmagatzemades dins una carpeta del projecte. Si tot i així no es troba la portada, se'n mostra una per defecte (la mateixa que es mostra quan s'inicia el programa).

Per més informació dirigir-se a la web http://www.id3.org/ i també a http://javamusictag.sourceforge.net/api/index.html.

5.3 Patró MVC (patró de sistema)

Hem decidit reutilitzar els coneixements que varem adquirir els membres de la pràctica a Ampliació de Programació Orientada a Objectes per implementar un patró. Ja que el nostre programa utilitza una GUI era quasi obligatori optar per aplicar-hi el Model-Vista-Controlador, que facilita molt la interacció de la interfície amb tots els algoritmes que hi ha al seu darrera.

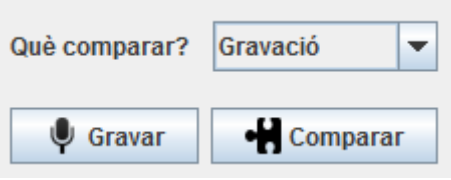


5.3.1 Descripció de les classes

- Controlador.java: Classe amb el rol de controlador al patró MVC. S'encarrega dels esdeveniments i de mantenir la comunicació entre les distintes classes que formen el model així com la finestra gràfica.
- Vista.java: Finestra principal & missatges
- Complex.java: Classe per treballar amb nombres complexes.
- Conversor.java: Classe per convertir diferents fitxer i formats a estàndards de l'aplicació.
- DataPoint.java: Classe que representa un punt clau en la cerca.
- Discretizador.java: Classe per codificar tots els fitxers i gravacions al format adequat per al procés de matching.
- FFT.java: Classe per realitzar la transformació ràpida de Fourier.
- LectorMetadata: Llegeix la metadata d'un arxiu.
- Gravadora.java: Classe per recollir gravacions des del micròfon.
- Match.java: Representa la coincidència d'una cançó i la quantitat de coincidències de DataPoints.
- Matcher.java: Classe que s'encarrega de fer els matchings entre els DataPoints.
- Patro.java: Classe d'informació genèrica per totes les classes. Es tracta de l'estàndard de l'aplicació a l'hora de codificar àudio.
- Canso.java: Representa el concepte de cançó dins el programa: una cançó està formada per diferents tags com són el seu nom, el grup, l'àlbum... i també el nom de l'arxiu del qual n'hem extret la informació.
- ReconeixedorMusica.java: Classe des d'on s'inicia el programa
- Reproductor.java: Aquesta classe s'encarrega d'instanciar i controlar un player baixat

5.4 Gravació d'àudio des del micròfon

S'ha implementat que un cop l'usuari pitja el botó “Gravar”, el programa faci una gravació de 5 segons en la que pren mostres del micròfon de l'ordinador. Aquestes mostres es guarden dins un fitxer únic (el seu nom és el temps del sistema .txt) de datapoint's que després es pot comparar amb la BBDD si hem triat aquesta opció al JcomboBox. L'altra opció del JcomboBox és la de comparar una cançó (la que està oberta actualment) amb les cançons que tenim a la BBDD.



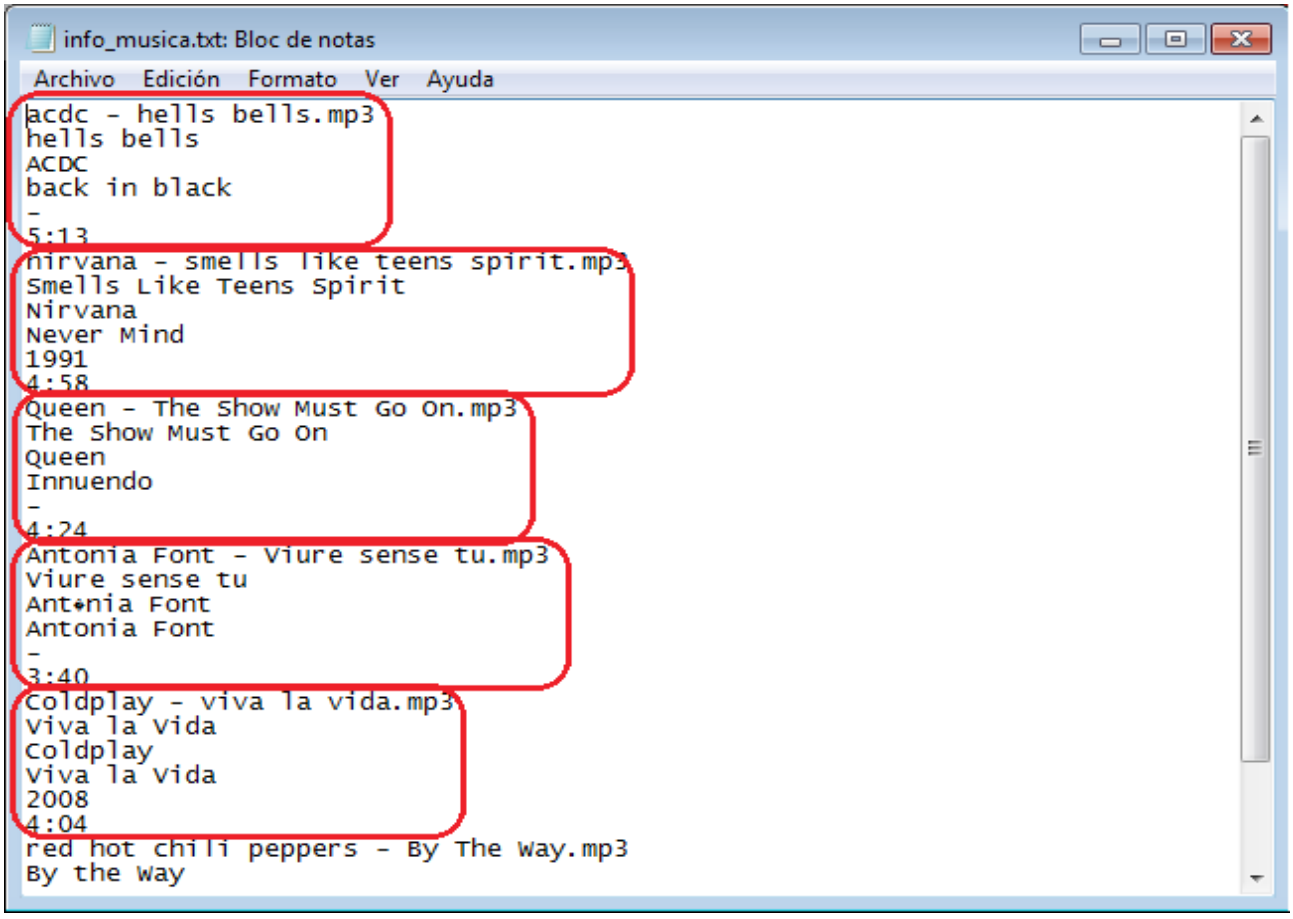
5.5 Gestió de la informació de les cançons

Al marge del fitxer “BBDD.txt” que emmagatzema tots els datapoint's també tenim el fitxer “info_musisca.txt”. Als dos fitxers s'hi emmagatzema informació diferent, tot i que la clau per accedir a cada conjunt d'informació és el nom de l'arxiu.

Nom cançó	Grup	Àlbum	Any	Duració
hells bells	ACDC	back in black	-	5:13
Smells Like Teens ...	Nirvana	Never Mind	1991	4:58
The Show Must Go ...	Queen	Innuendo	-	4:24
Viure sense tu	Ant□nia Font	Antonia Font	-	3:40
Viva la Vida	Coldplay	Viva la Vida	2008	4:04
By the Way	red hot chili peppers	By The Way	2002	4:18

Centrant-nos en l'arxiu “info_musisca.txt”, -que és el que guarda la informació de les cançons-, aquest consta de l'anomenada capçalera (nom de l'arxiu), que serveix de clau per identificar la informació única i també actua com a foreign key de l'altre arxiu també anomenat.

La informació que s'emmagatzema de cada cançó prové de la metadata del fitxer i és la següent: nom de la cançó, grup/artista compositor, àlbum/disc, any d'estrena i la duració.



La interfície consta d'un boto “Afegir” per codificar la cançó oberta en aquell moment. Durant la codificació es van guardant els datapoint's al fitxer “BBDD.txt” i després es guarda la informació de la cançó al fitxer “info_musica.txt”. Finalment s'actualitza el Jtable per donar feedback a l'usuari.

També existeix un botó “Eliminar” que marca als dos fitxers la cançó oberta actualment com a brossa i actualitza la taula.

5.6 Javadoc & convenis nomenclatura

S'ha emplenat tot el javadoc del codi de la pràctica (classes, constructors i mètodes). Es pot obrir des de la pròpia interfície.

**També s'han usat comentaris aclaratius d'una sola línia que es poden veure directament al codi. Aquests comentaris són per facilitar la comprensió del funcionament de l'aplicació.*

S'han seguit els convenis de nomenclatura que venen donats per Java:

Convenis de nomenclatura		
Identificador	**Convenció**	**Exemple del programa**
Paquets	El prefix d'un nom de paquet (únic) s'escriu sempre en minúscules de l'ASCII. Els components posteriors del nom del paquet varien en funció dels convenis de nomenclatura interns.	reconeixedormusica
Classes	Per als noms de classes posar la primera lletra en majúscules i les altres en minúscules. Cada paraula nova s'inicia amb majúscula.	LectorMetadata
Mètodes	Els mètodes han de començar amb un verb i aquest en minúscula. Si té més d'una paraula, s'escriuen juntes i sempre amb la primera lletra en majúscula.	transformarGravacio()
Variables	El nom d'una variable ha de ser breu però significatiu (significat mnemotècnic). No hauria de començar ni amb signe de subratllat ni $, tot i que estan permesos. S'alternen majúscules i minúscules començant per una minúscula.	llistaCansons
Constants	Estan declarades com “static final”. Totes les seves lletres van en majúscula i la separació entre paraules es fa amb “_”.	CHUNK_SIZE

6 Conclusió

Hem trobat interessant la realització de la pràctica per diversos motius. Un d'ells és que el seu enunciat és un problema que ens podem trobar al llarg de la nostra vida laboral. És a dir, si se'ns presenten amb un problema per resoldre, molts de cops el treball a realitzar no consistirà en posar-se a picar codi directament, fins i tot prèvia fase d'anàlisis per veure com estructurar el problema. En ocasions haurem d'anar més enllà i moure'ns per cercar aquella informació que ens sigui útil per resoldre el problema, ja que sense aquesta possiblement el resoldríem de manera incorrecta. O potser no, però segurament si ho féssim ens estalviaríem feina i com a -futurs- enginyers hem d'anar a lo pràctic. Així doncs, en aquest sentit ens ha paregut positiva la pràctica ja que fuig dels típics enunciats acadèmics que acaben fent-se avorrits de programar. Un punt negatiu que podríem citar era la data inicial d'entrega, ja que creim que hagués estat complicat per la majoria de grups entregar una bona pràctica en aquella data.