Breu Descripció de les estructures de dades i algorismes utilitzats per a implementar les funcionalitats principals.

Per tal de satisfer totes les nostres necessitats per implementar el projecte, em generat diverses estructures de dades. A continuació us exposem de forma breu la descripció de cada una d'elles ordenades per clases.

Llistat de les classes que contenen les estructures de dades utilitzades:

- Edge
- Node
- Graph
- Pair
- Ponderation
- Reproduction
- User

CLASS EDGE:

Aquesta classe és molt simple degut a que és la representació d'una relació entre nodes. Utilitzarem aquesta estructura de dades per tal de poder separar els nodes en comunitats i així saber quins nodes tenen relació amb quins altres depenen del "pes" que li posem.

CLASS NODE:

Aquesta classe és molt simple degut a que és la representació d'un node dins d'un graf. Utilitzarem aquesta estructura de dades per tal de poder respresentar o bé una canço o bé una reproducció.

Aquesta estructura conté només un identificador per diferenciar nodes i una funció per tal de saber si un node és igual a un altre node.

CLASS GRAF:

Aquesta classe és més complicada degut a que és la representació o bé de totes les cançons o bé de totes les reproduccions, on els nodes són les cançons o les reproduccions i les arestes (Edge) la relació que hi ha entre elles. La utilitzarem per tal de poder establir les comunitats desitjades amb les ponderacions establertes per l'usuari. No és si més no, una estructura de dades en forma de graf (no binari) per tal d'utilitzar l'algoritme d'ordenació de comunitats que volguem.

Al ser una de les estructures de dades principals, conté moltes funcions, entre elles de control per tal de gestionar-la bé.

CLASS PAIR:

Aquesta classe és necessària degut a que en java no existeix l'estructura pair. La utilitzem sobretot per saber quins dos nodes connecta una aresta. És una classe molt simple que retorna o bé la parella de valors que conté o bé el valor del primer paràmetre o el valor del segon.

CLASS PONDERATION:

Aquesta estructura de dades conté tot el que considerem com rellevant per establir les relacions entre nodes, és a dir, l'autor, l'estil, l'any… etc. A més cada ún dels paràmetres té un factor d'importància que defineix l'usuari, per exemple, l'usuari pot decidir que l'estil és molt més important que no pas l'any (per defecte tots els paràmetres tenen el mateix factor d'importància).

CLASS REPRODUCTION:

Aquesta estructura de dades representa una reproducció. La utilitzem per tal de saber en quin momento un usuari a escoltat una cançó. La reproducció conté data i hora, autor i títul. També se sap si ha escoltat en un mateix dia la mateixa cançó ja que una reproducció s'identifica per la data i hora.

CLASS USER:

Aquesta classe és una estructura que representa un usuari. L'usuari conté un nom, una edat i una llista de reproduccions. Com el graf és una estructura molt important no només perqué ens identifica els usuaris sinó perqué conté per cada usuari una llista de reproduccions. Aquestes ens serviran per definir relacions entre reproduccions depenent el momento en que s'han reproduït. També el fet de l'usuari en si, és a dir, el seu identificador, és un paràmetre per definir relacions entre cançons i entre reproduccions.

Algorismes utilitzats per a implementar les funcionalitats principals.

Funcionalitats Principals:

- Girvan Newman
- Solution
- FileManager
- FileParser
- History

Girvan Newman

Aquest algorisme és el que ens separarà el graf en diferents comunitats, passant-li nosaltres el graf. Funciona de la següent forma:

1. Es calcula primer el pes de totes les arestes existents al graf.
2. S'elimina les arestes dels pesos més alts.
3. Es torna a calcular el pes de les arestes borrades.
4. Es repeteixen els passos 2 i 3 fins que no hi ha arestes.
5. Finalment es retorna els componentts que encara queden connnectats.

Solution

Aquesta classe conté els algoritmes necessaris per gestionar una solución, és a dir, controla les entrades i les sortides d'una solución.

File Manager

Al tenir tants arxius requerim d'una classe que controli tota la gestió d'aquests. Això és el que fa File Manager. Te diverses funcionalitats però les principals serien, SaveData, LoadData, EraseData i les seves derivades com SaveSong , RemoveSong.

SaveData

Bàsicament li passem un String amb la direcció on es guardarà el fitxer i un ArrayString amb el contingut de fitxer. Comprovem que existeixi la carpeta on es guardarà i si no la creem i en un arxiu txt escrivim tot el que hi ha a l'ArrayString.

LoadData

En aquest cas simplement li passem la dirección (path) d'on volem carregar l'arxiu. Comprovem que sigui correcte i finalment el carreguem.

EraseData

Aquest cas és el més simple de tots. Fem exactament el mateix que LoadData però en contes de carregar l'arxiu, indiquem que l'eliminem amb una fucionalitat de java.io.File que és diu .delete();

File Parser

Les funcionlitat d'aquesta classe és molt rellevant degut a que fa processos intermitjos. Com per exemple, agafa un usuari concret de l'arxiu o una cançó concreta.

Funcions rellevants, GetUser, GetReproduction, GetSong i GetGraph.

GetUser

Aquesta funció li passem un string, que representa la línia on es trova l'usuari desitjat. I el retorna.

Les demés funcions tenen els mateixos paràmetres d'entrentada i la sortida és una una reproducció en el cas de GetReproduction i una cançó en el cas de GetCançó.

GetGraph

Aquesta funció és més complicada. D'entrada li passem una string que és la direcció d'on es trova el graf. Creem un nou graf i llavors comencem a llegar el ja disponible i a introduir-ho en aquest nou graf. Finalment retornem aquest graf.

History

En aquesta classe, fem tota la funcionalitat que seria referent a l'historial. Fem servir les funcions GetSolutions, SaveSolution i RemoveSolution.

GetSolution

Aquesta funcionalitat agafa totes les solucions existents i les retorna en una array de solucions. Bàsicament se'n va al directori que l'indiques a través d'un paràmetre que li envíes mitjançant un String, i recorre totes les carpetes agafant les solucions que va trobant.

SaveSolution

Com el seu nom indicia, guarda una solución només passant-li com a paràmetre la Solució i l'identificador de la solución.

RemoveSolution

Una vegada indicat el nom de la solució aquesta funció te l'elimina.