MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

SISTEM DE RADIO ONLINE BAZAT PE O REȚEA P2P CU SISTEM DE RECOMANDARE INTEGRAT

LUCRARE DE LICENȚĂ

Absolvent: **Denisa-Daniela CERNEA**

Coordonator ştiinţific: **Șef lucr. ing. Cosmina IVAN**

2014

MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

DECAN,
Prof. dr. ing. Liviu MICLEA

DIRECTOR DEPARTAMENT,
Prof. dr. ing. Rodica POTOLEA

Absolvent: **Denisa-Daniela CERNEA**

SISTEM DE RADIO ONLINE BAZAT PE O REȚEA P2P CU SISTEM DE RECOMANDARE INTEGRAT

1. **Enunţul temei:** Proiectul își propune realizarea unui sistem de radio online, bazat pe o rețea peer-to-peer, cu un sistem de recomandare integrat. Principalele funcționalități sunt conectarea la rețea, redarea fișierelor audio, distribuirea, căutarea și descărcarea acestora din rețea. Sistemul generează automat un playlist pentru utilizator, pe baza recomandărilor primite de la sistemul de recomandare.

2. **Conţinutul lucrării:** Cuprins, Introducere, Obiectivele proiectului, Studiu bibliografic, Analiză și fundamentare teoretică, Proiectare în detaliu și implementare, Testare și validare, Manual de instalare și utilizare, Bibliografie, Anexe.
3. **Locul documentării**: Universitatea Tehnică din Cluj-Napoca, Departamentul Calculatoare

4. **Consultanţi**: Șef lucr. ing. Cosmina Ivan

5. **Data emiterii temei:** 1 noiembrie 2013

6. **Data predării:** 3 Iulie 2014

Absolvent: _____

Coordonator ştiinţific: _____

MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

Declaraţie pe proprie răspundere privind autenticitatea lucrării de licenţă

Subsemnatul(a)_____
_____,
legitimat(ă) cu _____ seria _____ nr. _____
CNP _____, autorul lucrării
_____
_____
_____elaborată în vederea susţinerii examenului de finalizare a studiilor de licență la Facultatea de Automatică și Calculatoare, Specializarea _____ din cadrul Universităţii Tehnice din Cluj-Napoca, sesiunea _____ a anului universitar _____, declar pe proprie răspundere, că această lucrare este rezultatul propriei activităţi intelectuale, pe baza cercetărilor mele şi pe baza informaţiilor obţinute din surse care au fost citate, în textul lucrării, şi în bibliografie.

Declar, că această lucrare nu conţine porţiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislaţiei române şi a convenţiilor internaţionale privind drepturile de autor.

Declar, de asemenea, că această lucrare nu a mai fost prezentată în faţa unei alte comisii de examen de licenţă.

In cazul constatării ulterioare a unor declaraţii false, voi suporta sancţiunile administrative, respectiv, *anularea examenului de licenţă*.

Data

_____

Nume, Prenume

_____

Semnătura

Cuprins

Capitolul 1. Introducere

Scopul acestui proiect este de a proiecta și implementa un sistem de radio online bazat pe o rețea peer-to-peer, cu sistem de recomandare integrat. Acest sistem de radio online vine în întâmpinarea dorinței utilizatorilor de a folosi sisteme și aplicații ce le oferă acestora modalități de ascultare, distribuire sau descoperire de noi melodii sau artiști.

Utilizatorul acestui sistem este consumatorul de muzica, persoana care va folosi aplicația pentru a asculta fișierele audio generate de radio. Sistemul se bazează pe o rețea peer-to-peer, unde fiecare membru al rețelei își distribuie propria muzică în rețea, pentru a fi disponibilă și celorlalți utilizatori. Utilizatorii pot căuta, descărca și asculta orice melodie aflată in rețea. Sistemul permite crearea pentru fiecare utilizator în parte a unui playlist de melodii personalizat, realizat în funcție de preferințele muzicale ale acestuia.

Fată de aplicațiile deja existente pe piață, sistemul propus prezintă avantajele folosirii unei rețele peer-to-peer împreună cu o soluție de realizare a unui de sistem de recomandare distribuit.

1.1. Contextul proiectului

Muzica atrage o piață mare de consumatori, ceea ce face din acest domeniu unul atractiv, profitabil și care prezintă interes nu doar celor implicați direct în creare de muzica. Persoanele tind să asculte muzica mai des ca orice altă activitate (ex: televizor, cărți, filme etc.), astfel că muzica devine un instrument esențial de comunicare și exprimare de sine.

Datorită extinderii rapide a rețelelor de calculatoare din ultimele decenii, internetul a devenit principala sursă de utilizare a informațiilor multimedia. In zilele noastre, aplicațiile care utilizează internetul sunt din ce in ce mai căutate pentru descoperirea, descărcarea sau folosirea de media, cum ar fi cărțile, muzica sau filmele.

Odată cu expansiunea muzicală in format digital, s-a acordat o importanță tot mai mare organizării și gestiunii milioanelor de melodii produse de societate. Era tehnologiei a produs astfel schimbări majore în industria muzicală. Dacă în urmă cu două decenii utilizatorii erau nevoiți să cumpere discuri sau să asculte un radio pentru a descoperi muzica, acum, datorită noilor tehnologii, utilizatorul are la dispoziție diverse dispozitive, sisteme sau aplicații pe care le poate folosi pentru a asculta in orice moment orice melodie dorește.

Principala provocare devine găsirea celei mai bune soluții din punct de vedere al perfomanței și al costurilor pentru distribuirea de muzică într-o rețea. Cum procentul sistemelor total distribuite este într-o continuă creștere, rețelele peer-to-peer devin astfel o sursă importantă de dezvoltare, datorită avantajelor pe care le prezintă în fața aplicațiilor clasice, de tip client-server. Această rețea propune dezvoltarea unui sistem scalabil, tolerant la defecte și fiabil. Membrii rețelei vor avea toți aceleasi drepturi, își vor partaja fișierele audio, fără implicarea unui sistem administrativ central.

Un alt aspect important al realizării acestui proiect este găsirea celei mai potrivite metode de construire a unui sistem de recomandare descentralizat, bazat pe o rețea peer-to-peer. Sistemele de recomandare sunt din ce in ce mai utilizate in mediul online si sunt folosite ca instrumente care oferă facilități de predicție a preferințelor utilizatorilor într-

un spațiu de articole dat, pentru a-i ajuta în procesul de selecție a articolelor. Sunt aplicate in diverse domenii, printre care si muzica, unde vin in ajutorul utilizatorilor pentru a filtra și descoperi muzică după preferințele lor.

1.2. Cerințele proiectului

Tema proiectului este crearea unui sistem de radio online bazat pe o rețea peer-to-peer cu sistem de recomandare integrat. Sistemul este dezvoltat pentru utilizatorii simpli, care vor să asculte și să descopere muzică nouă, după preferințele lor.

Sistemul va crea rețele de utilizatori care își vor putea distribui propria muzică în rețea. Din mulțimea melodiilor publicate în rețea, sistemul creaza un playlist personalizat de melodii pentru fiecare utilizator în parte. Playlist-ul este construit pe baza recomandărilor primite de la sistemul de recomandare, în urma realizării profilului muzical al utilizatorului.

Pentru a putea beneficia de funcționalitățile sistemului, un utilizator trebuie să fie conectat la o rețea peer-to-peer. Acesta poate crea o nouă rețea, la care să se conecteze alți utilizatori, sau se poate conecta la o rețea deja existentă, în cazul în care cunoaște adresa unui membru al rețelei respective.

Aplicația oferă utilizatorilor funcționalitatea de adăugare de fișiere într-o rețea P2P, astfel ei își vor putea distribui propriile melodii, pentru a fi disponibile și celorlalți utilizatori.

Sistemul va crea automat un playlist pe baza recomandărilor în urma căutării unei melodii de către utilizator. Căutarea se va face prin intermediul unor cuvinte cheie date de utilizator, ce vor ține de atributele melodiei: gen muzical, artist sau titlu. În funcție de melodia căutată și de istoricul melodiilor ascultate și preferate de către utilizator, sistemul continuă să caute în rețea melodii asemănătoare, în funcție de sugestiile sistemului de recomandare. Melodiile date de sistem vor fi descărcate din rețea și stocate local temporar.

Sistemul de radio online se comportă ca un MP3 Player: utilizatorul poate asculta o melodie (play), poate să o oprească (stop) sau să treacă la următoarea melodie din playlist (skip). După ce o melodie se termină, începe automat difuzarea urmatoarei melodii adăugate în playlist.

Utilizatorul va avea posibilitatea să-și exprime preferințele pentru anumite melodii ascultate, sau, în caz contrar, antipatia fată de acestea. Aceste voturi, pozitive sau negative, sunt folosite la antrenarea sistemului de recomandare și primirea unui playlist de melodii după gusturile și interesele sale.

1.3. Conținutul lucrării

Lucrarea conține 8 capitole în care sunt prezentate, pe rând, obiectivele proiectului, tehnologiile utilizate, analiza, modul de implementare și testare a sistemului, precum și un manual de utilizare.

În primul capitol este descris, într-un mod general, contextul aplicației și cerințele proiectului.

În capitolul 2 vor fi descrise obiectivele principale generale și specifice ale sistemului.

În capitolul 3 se va realiza un studiu bibliografic al componentelor de bază ale aplicației. Vor fi prezentate caracteristicile importante ale unei rețele peer-to-peer,

împreună cu o clasificare și descriere a sistemelor de recomandare. Se va pune accent pe elementele utile sistemului propus: rețeaua peer-to-peer de tip Kademlia și sistemul de recomandare descentralizat. De asemenea, vor fi analizate sisteme similare, pe baza cărora va fi realizată o comparație între sistemul propus și cele deja existente pe piață.

În capitolul 4 vor fi analizate cerințele și cazurile de utilizare ale aplicației. Se va realiza o analiză generală și o arhitectură conceptuală a sistemului, precum și o descriere la nivel teoretic a componentelor aplicației. Vor fi prezentate tehnologiile utilizate în implementarea acestui proiect, justificând fiecare alegere în locul altor tehnologii disponibile pe piață.

În capitolul 5 va fi prezentat modul de dezvoltare a sistemului și detaliile de implementare. Capitolul conține arhitectura software a aplicației, împreună cu o descriere a fiecărui modul în parte.

În capitolul 6 vor fi descrise detaliile despre modul de testare si validare a sistemului.

Capitolul 7 cuprinde manualul de utilizare al sistemului, în care sunt descriși pașii pe care un utilizator trebuie să îi facă pentru a putea rula aplicația și a beneficia de toate funcționalitățile sistemului.

În capitolul 8 vor fi prezentate concluziile referitoare la sistemul dezvoltat, precum si posibile dezvoltări ulterioare .

Capitolul 2. Obiectivele Proiectului

Acest capitol are ca scop definirea obiectivelor pentru sistemul propus, stabilite prin specificarea cerințelor de funcționare ale sistemului, dar și a atributelor de calitate pe care sistemul ar trebui să le îndeplinească.

2.1. Obiective generale

Proiectul își propune realizarea unui sistem de radio online bazat pe funcționalitățile unei rețele peer-to-peer. Sistemul trebuie să fie similar unei aplicații de partajare de fișiere, unde utilizatorii pot distribui propria muzică, respectiv pot descărca melodii aflate în rețea. Sistemul va crea un playlist personalizat pentru fiecare utilizator. Alegerea melodiilor din playlist va fi realizată de către un sistem de recomandare, care va determina gusturile muzicale ale utilizatorului în urma preferințelor acestuia.

Prin folosirea sistemului de radio online se dorește crearea unui mediu în care diverși utilizatori pot face schimb de melodii, cu scopul de a asculta și descoperi muzică nouă aparținând celorlalți membrii ai rețelei, și care să corespundă gusturilor lor muzicale.

Folosirea unei rețele peer-to-peer ca infrastructură prezintă avantajul că toți membrii pot accesa rețeaua cu aceleași drepturi. Fiecare nod acționează atât ca producător cât și consumator de resurse. Membrii din rețea își distribuie resurse fără a mai fi nevoie de o autoritate centrală care să coordoneze activitățile din sistem. De asemenea, topologia rețelei P2P este adaptivă și tolerantă la defecte, nodurile se auto-organizează în vederea menținerii conectivității și performanței rețelei. În momentul în care un nod părăstește sistemul, o astfel de rețea are abilitatea de a trata instabilitatea și variațiile conectivității rețelei, se adaptează automat la erorile survenite sau la dinamicitatea nodurilor.

Prin realizarea unui sistem de recomandare hibrid și distribuit, care să acopere starea globală a melodiilor și starea individuală a fiecărui utilizator, se urmărește obținerea rezultatelor cele mai bune în urma activităților utilizatorilor din rețea, dar și minimizarea cat mai mult a dezavantajelor inerente care vin cu un asemenea sistem, cum ar fi scalabilitatea sau timpul de creare a profilului unui utilizator sau de notare a melodiilor din rețea.

2.2. Obiective specifice

În acest subcapitol vor fi prezentate cerinţele funcţionale şi cele nefuncţionale într-un mod general, ele fiind detaliate ulterior în capitolul 4.

Obiectivele specifice ale acestui proiect care se refera la indicatorii de calitate ai sistemului sunt prezentate in punctele următoare:

- *Scalabilitate*. Rețeaua trebuie să suporte un număr ridicat de utilizatori conectați și un volum mare de date partajate.
- *Eficiență*. Sistemul trebuie să utilizeze cât mai eficient traficul de date între utilizatori, într-un timp cât mai scurt, în momentul descărcării de melodii de la un membru al rețelei la altul.

- *Performanță*. Timpul de răspuns al mesajelor trimise in rețea între oricare doi utilizatori trebuie sa fie cât mai scurt.
- *Fiabilitate*. Sistemul de radio online trebuie să funcționeze fără erori pentru o perioadă cât mai indelungată de timp.
- *Robustețe*. Sistemul trebuie să fie tolerant la defecte și erori.
- *Corectitudine*. Programul software trebuie să se comporte conform specificațiilor funcționale.
- *Mentenabilitate*. Sistemul trebuie să fie ușor de întreținut, efectuarea modificărilor făcute sistemului software nu trebuie să constituie un proces complex.
- *Reutilizabilitate*. Componentele software ale sistemului trebuie să fie reutilizabile.
- *Securitate*. Sistemul trebuie să respecte anumite reguli de criptare a datelor publicate în rețea.
- *Standarde*. Sistemul software trebuie să folosească biblioteci publicate in GPL sau open source.
- *Testabilitate*. Sistemul trebuie să fie ușor de testat, pentru a se observa cât mai ușor si clar modul de funcționare al acestuia.

Obiectivele funcționale care trebuiesc indeplinite în vederea realizării sistemului de radio online sunt descrise în următoarele puncte:

- ***Conectarea la rețea.*** Utilizatorii trebuie să aibă posibilitatea să se conecteze la o rețea de tip peer-to-peer pentru a putea beneficia de funcționalitățile sistemului de radio online. Trebuie să existe două modalități de conectare: prin conectarea directă a utilizatorului la un nod cunoscut al unei rețele existente, sau prin crearea unei noi rețele la care să se poată conecta alți utilizatori.

- ***Distribuirea melodiilor.*** Utilizatorii trebuie să aibă posibilitatea de adăugare a fișierelor audio într-o rețea peer-to-peer. Pe baza melodiilor publicate în rețea de către toți membrii acesteia, vor fi create playlist-uri pe care utilizatorii le vor putea descărca și asculta.

- ***Căutarea unei melodii.*** Utilizatorul trebuie să poată căuta o melodie distribuită în rețea. Căutarea se va face prin intermediul unor cuvinte cheie date de utilizator, ce vor ține de atributele melodiei: gen muzical, artist sau titlu. Sistemul va trimite utilizatorului melodiile găsite, din care utilizatorul își va putea alege melodia pe care dorește să o asculte mai departe.

- ***Crearea unui playlist pe baza recomandărilor.*** Sistemul trebuie să fie capabil să creeze playlist-uri personalizate de melodii pentru fiecare utilizator al aplicației, în funcție de preferințele acestuia. Pe baza istoricului melodiilor ascultate și preferate de un utilizator se va crea un profil muzical al acestuia, care va contribui la realizarea sistemului de recomandare și determinarea melodiilor din playlist. În urma căutării unei melodii în rețea, sistemul va

începe automat construirea playlist-ului prin căutarea și descărcarea altor melodii bazate pe sugestiile venite de la sistemul de recomandare.

- ***Funcționalitate de MP3 Player.*** Sistemul trebuie să se comporte ca un MP3 Player. Utilizatorul trebuie să aibă posibilitatea redării unui fișier audio, opririi acestuia sau trecerii voite la redarea următoarei melodii din playlist-ul radio-ului. Utilizatorul va putea asculta atât fișierele audio proprii, stocate local, cât și melodiile venite din rețea de la ceilalți utilizatori.

- ***Notarea melodiilor.*** Utilizatorul trebuie să aibă posibilitatea să-și exprime aprecierea pentru o melodie, sau, în caz contrar, lipsa de apreciere, în vederea creării playlist-ului după preferințele acestuia. Voturile acordate melodiilor ascultate vor contribui la construirea profilului muzical al utilizatorului.

Capitolul 3. Studiu Bibliografic

Este evident faptul ca muzica atrage o piață mare de consumatori, ceea ce face ca acest domeniu să fie unul atractiv, profitabil și să prezinte interes nu doar celor implicați direct in creare de muzica.

Odată cu expansiunea muzicală în format digital, s-a acordat o importanță tot mai mare organizării și gestiunii milioanelor de melodii produse de societate. Conform articolului [13], cercetări anterioare au arătat că persoanele tind să asculte muzică mai des ca orice altă activitate (ex: televizor, cărți, filme etc.), astfel că muzica devine un instrument esențial de comunicare și exprimare de sine, pentru care e nevoie în continuare de cercetare.

O provocare devine găsirea celei mai bune soluții din punct de vedere al perfomanței și al costurilor pentru distribuirea de muzică într-o rețea.

Cum procentul sistemelor total distribuite este într-o continuă creștere, potrivit articolului [12], rețelele peer-to-peer devin o sursă importantă de cercetare și dezvoltare, datorită avantajelor pe care le prezintă în fața aplicațiilor de tip client-server (exemplu: scalabilitate, toleranță la defecte, fiabilitate etc.).

3.1. Rețele Peer-to-peer

O rețea Peer-to-peer (P2P) este un tip de rețea în care fiecare client are drepturi și responsabilități egale. Clienții partajează o parte din resursele lor în mod direct către alți utilizatori aflati în rețea fără a fi nevoie de un coordonator central cum ar fi un server sau o gazdă stabilă. Fiecare nod poate fi atât furnizor cât și consumator de resurse, în contrast cu tradiționalul sistem client-server, unde serverul este furnizor, iar clientul este consumator. [12]

Rețelele P2P sunt astfel, prin natura lor, sisteme distribuite fără organizare ierarhică sau control centralizat, cum este evidențiat și în [2]. Nodurile formează o rețea auto-organizată peste protocolul internet (IP). Caracteristicile peer-to-peer includ arhitecturi robuste de rutare pe arie largă (wide-area), căutare eficientă a datelor, selecția clienților aflați în vecinătatea imediată, stocare redundanță, permanență, scalabilitate și tolerantă la erori sau eșecuri.

3.1.1. Caracterizare

Rețelele P2P vin cu o serie de caracteristici specifice. Este permisă partajarea resurselor computaționale prin interschimb direct între noduri, și mai puțin prin intermediari oferiți de o autoritate centralizată (ex: server). [8] Serverele centralizate pot fi folosite însă pentru a realiza activități specifice, cum ar fi inițializarea rețelei sau adăugarea de noi noduri în rețea. Ideal, nodurile participă activ și unilateral la realizarea de operații ca localizarea, caching-ul nodurilor și al conținutului, dirijarea informațiilor sau managementul resurselor transferate.

Rețeaua P2P este una suprapusă (overlay) peste cea fizică. Se situează la nivel de aplicație, unde muchiile virtuale sunt conexiuni TCP sau pointeri la adrese IP. Proximitatea fizică a nodurilor nu e importantă, iar menținerea rețelei se face prin verificarea periodică a conectivității ori a existenței celorlalte noduri. Când un nod cade, sistemul P2P ar putea stabili noi muchii.

Topologia rețelei P2P e adaptivă și tolerantă la defecte, nodurile se auto-organizează în vederea menținerii conectivității și performanței rețelei. O astfel de rețea are abilitatea de a trata instabilitatea și variațiile conectivității rețelei, se adaptează automat la erorile survenite sau la dinamicitatea nodurilor.

Astfel de rețele prezintă și alte avantaje față de alte abordări: dependență redusă față de dispozitive individuale și de sub-rețele, scalabilitate mai mare sau reziliență îmbunătățită, unde o resursă este disponibilă în copii multiple.

Pe de altă parte, apar și dezavantaje inerente cu asemenea rețele. Arhitecturile P2P sunt probabilistice, localizarea resurselor este impredictibilă, resursele sunt volatile. Inexistența unui control centralizat duce la apariția problemelor privind impunerea unei autorități asupra aplicațiilor, conținutului și utilizatorilor sau dificultăți în detectarea și identificarea utilizatorilor. Pot apărea probleme de securitate și vulnerabilitate la atacuri. De asemenea, o problemă majoră care poate apărea în cazul aplicațiilor P2P se referă la drepturile de autor asupra conținutului digital, prin folosirea acestora în scop abuziv și ilegal.

3.1.2. Clasificare

În ceea ce privește infrastructura unei rețele peer-to-peer, aceasta poate fi clasificată în funcție de topologia, structura sau gradul de centralizare (figura 3.1).

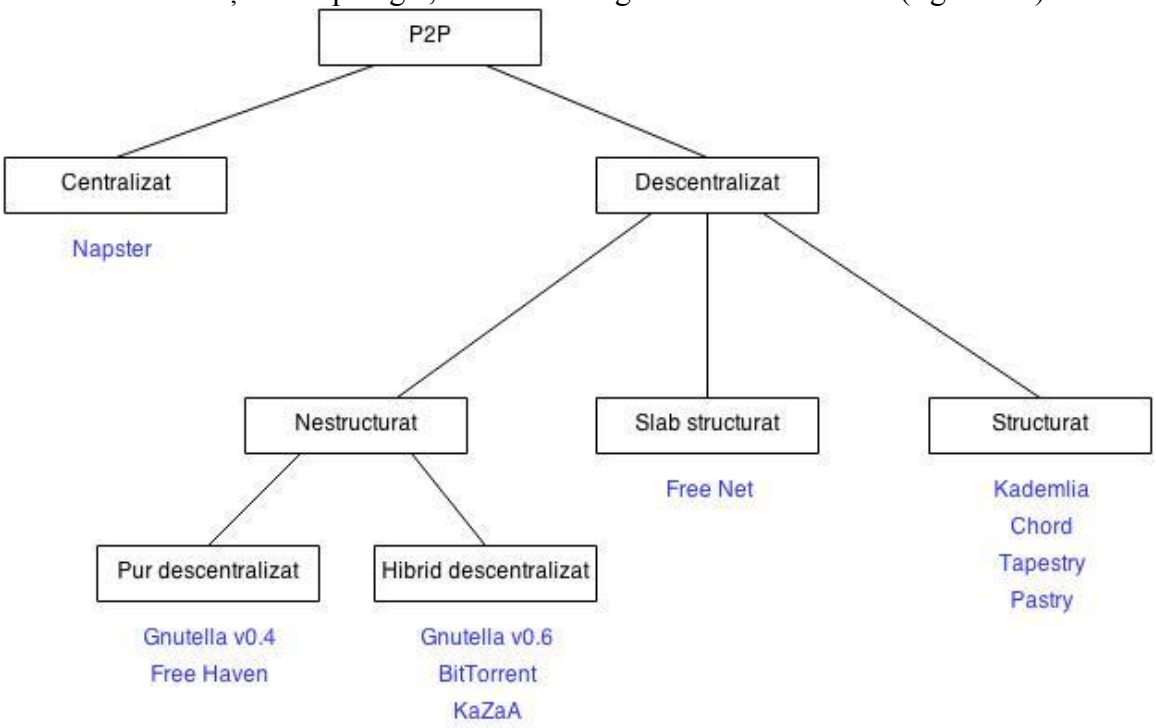


Figura 3.1 Clasificarea rețelelor P2P

În ceea ce privește gradul de descentralizare, rețelele pot fi centralizate, pur sau hibrid descentralizate. O rețea centralizată presupune că toate nodurile sunt legate la o entitate centrală, dar sunt conectate unul la altul pentru transferul fișierelor. Nodul central este strict necesar pentru funcționare și are funcții de indexare și de look-up table. (Figura 3.2 a). Într-o rețea pur descentralizată, toate nodurile au aceleași drepuri și realizează

aceleași activități, având simultan roluri de client și server. Nu beneficiază de o coordonare centrală, stabilirea conexiunilor între peeri este aleatoare, iar eliminarea oricărui nod din rețea nu duce la pierderea funcționalității (Figura 3.2 b). Pe de altă partea, rețeaua hibridă (sau parțial centralizată), comparativ cu rețelele P2P „pure", presupune introducerea unui nivel ierarhic suplimentar: unele noduri au un rol mai important, devenind supernoduri, conform politicilor fiecărui sistem P2P folosit. Rolul de supernod este stabilit în mod dinamic. (Figura 3.2).

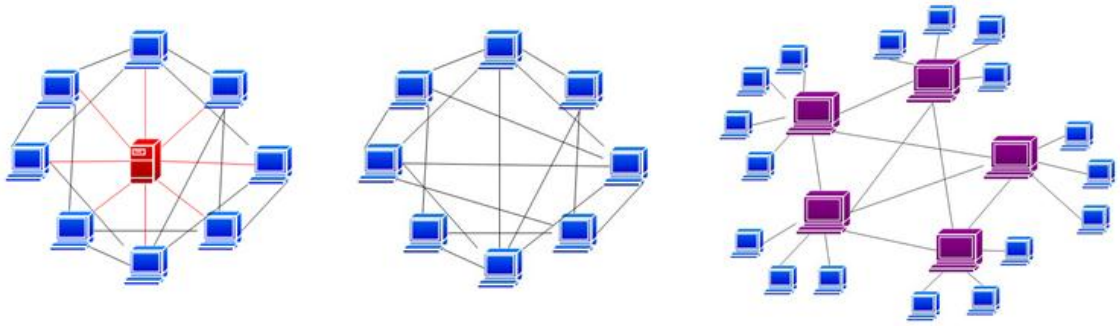


Figura 3.2 a) Rețea P2P centralizată, b) Rețea P2P pur descentralizată, c) Rețea P2P hibrid descentralizată (parțial centralizată)

Din punct de vedere al structurii, rețelele pot fi împărțite în trei categorii: nestructurate, structurate sau slab structurate. Într-o rețea nestructurată, nu există control asupra topologiei și a plasamentului fișierelor, plasarea conținutul este complet independentă de topologia rețelei. Conținutul trebuie localizat, astfel că sunt necesare diferite strategii de căutare: BFS/DFS, drumuri aleatorii, probabilistice etc. Într-o rețea structurată, topologia și plasamentul fișierelor sunt bine controlate. Într-o rețea slab structurată, nodurile pot estima ce noduri stochează resursele căutate. Se folosește o propagare în lanț, în care fiecare nod ia decizii locale privind următorul nod interogat.

3.1.3. DHT (Distributed Hash Table)

Noțiunea de arhitectură P2P structurată, conform [9], indică faptul că topologia rețelei este strict controlată, iar datele sunt plasate în locații precise, cunoscute, și nu aleator, ceea ce duce la eficientizarea operațiilor de căutare. Asemenea sisteme structurate utilizează tabele distribuite de indici unici (DHT – Distributed Hash Table) ca substrat, în care informațiile de localizare a conținutului (valori) sunt plasate deterministic în nodurile cu identificatori corespunzând indicilor (cheilor) unici ale valorilor. Sistemele bazate pe DHT au proprietatea de a asigura id-uri consistente și uniform distribuite și suportă stocarea și încărcarea scalabilă a perechilor cheie-valoare în rețea. (Figura 3.3) Astfel, prin intermediul funcțiilor de put(cheie, valoare) și remove(cheie) sunt stocate sau aduse valorile corespunzătoare unei chei.

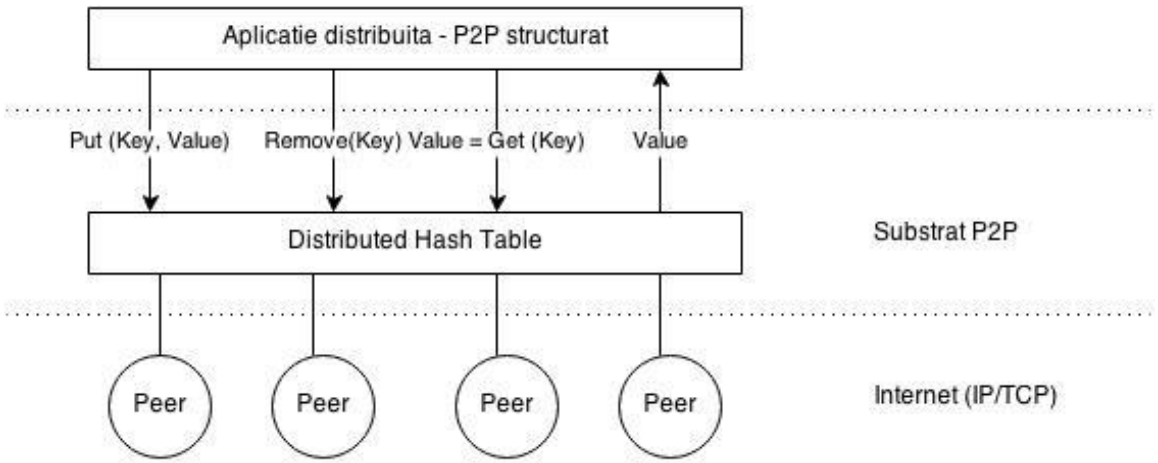


Figura 3.3 Organizarea stratificată a rețelei P2P structurate

Tabela distribuită de indicatori unici (DHT) reprezintă o listă distribuită tuturor nodurilor conectate între ele. În DHT, fiecare nod are un id care aparține unui set de date S (ex: de lungime de 160 biți). Fiecare peer administrează o tabelă de rutare alcătuită din id-urile și adresele IP ale nodurilor vecine. De asemenea, cheile obiectelor de date reținute în DHT, în urma aplicării unei funcții de hash-ing, vor aparține aceluiași set S. O funcție de hash-ing are proprietatea că, aplicată oricăror două valori de intrare, ieșirile vor aparține aceluiași set de date S, dar vor avea mereu valori diferite.

Cererile de căutare sau rutările de mesaje sunt retransmise, într-o manieră progresivă, pe căile acoperite de noduri cu id-uri care sunt mai apropiate de cheia din spațiul de identificatori S, ajungându-se astfel până la nodul căutat.

Teoretic, într-un sistem bazat pe DHT, orice valoare poate fi localizată în medie în O(logN) noduri, unde N reprezintă numărul total de noduri ai rețelei. O proprietate importantă a DHT este că acestea pot administra eficient un număr mare de obiecte de date. De asemenea, numărul de noduri conectate poate fi foarte mare, variind de la câteva noduri la mai multe mii (teoretic, milioane). Din cauza capacității limitate de stocare a fiecărui nod, dar și al costului de adăugare și actualizare a valorilor, nu este fezabilă stocarea tuturor obiectelor în fiecare dintre noduri. De aceea, fiecare peer este responsabil cu stocarea locală a unor porțiuni de obiecte.

Rețeaua P2P structurată asignează chei entităților de date și își organizează peer-urile într-un graf ce mapează fiecare cheie de date câte unui peer. Graful permite descoperirea eficientă a valorilor prin intermediul cheilor. Pe de altă parte, din cauza formei sale simple, sistemul nu permite căutări complexe și implică stocarea unei copii sau a unui pointer către fiecare obiect de date. Conform [7], rețelele P2P structurate cunoscute sunt: Content Addressable Network (CAN), Tapestry, Chord, Pastry, Kademlia sau Viceroy.

CAN este o infrastructură P2P descentralizată ce ofera funcționalitatea tabelei de chei (hash table) la o scară mare, comparabila cu cea a internetului. CAN a fost concepută să fie scalabilă, rezistentă la eșecuri și auto-organizată.

Pastry folosește sistemul Plaxton de rutare bazat pe prefixe pentru a construi o rețea acoperită, descentralizată și auto-organizată în care fiecare peer rutează cererile clientilor și interacționează cu instantele locale a una sau mai multe aplicații.

Sistemul Tapestry are descentralizare aleatoare pentru a obține distribuirea încărcării și rutarea spre locația peer-ilor. Acesta are proprietăți similare cu Pastry însă diferența dintre ele constă în administrarea locațiilor în rețea și replicarea obiectelor de date.

Chord folosește hashing consistent pentru a asigna chei către peer-urile sale. Hashing-ul consistent este proiectat pentru a permite peer-urilor să intre și să iasă din rețea cu minimum de întreruperi. Această schema descentralizată tinde să balanseze încărcarea sistemului, atât timp cat fiecare peer primește aproximativ același număr de chei și nu se fac prea multe mutări de chei atunci când un peer intră sau iasă din sistem.

Rețeaua P2P descentralizată Viceroy este proiectată pentru a descoperi date și resurse într-o manieră dinamică de tip fluture (dynamic butterfly). Viceroy pune la dispoziție un mecanism de hashing consistent astfel încât conferă un echilibru setului de servere și rezistență în cazul intrărilor și a ieșirilor din sistem a serverelor.

În cazul unei rețele Kademlia, se utilizează un algoritm de rutare bazat pe identificatoarele nodurilor pentru localizarea peer-urilor apropiate de o cheie destinatie. Caracteristicile acetstui algoritm vor fi detaliate in subcapitolul următor.

3.1.4. Kademlia

Rețeaua P2P descentralizată și structurată preia abordarea asignării unor identificatori de dimensiune fixă (160 biți) fiecărui peer, potrivit articolului [7]. Astfel, spațiul de date este reprezentat de S = [00...0, 11...1] de 160 biți. O pereche {cheie, valoare} este stocată în cadrul peer-urilor cu id-uri apropiate de valoarea cheii.

Un avantaj pe care îl prezintă acest tip de rețea este că folosește metrica XOR pentru a calcula distanțe între punctele din spațiul cheilor. XOR este o funcție simetrică și permite nodurilor cereri de lookup de la același set de noduri conținută în tabelele acestora de rutare. Folosește un singur algoritm de rutare pentru localizarea nodurilor apropiate de o cheie destinație.

Fiecare mesaj transmis de către un peer include ID-ul său, permițând astfel destinatarului să ia cunoștință de existența nodului respectiv. Cheile, în urma aplicării funcției de hash-ing, au dimensiunea de 160 biți. Pentru a localiza perechi {cheie, valoare}, se calculează distanța dintre doi identificatori, folosind metrica XOR la nivel de bit:

d(a, b) = d(b, a) = $a \oplus b$.

Ecuație 3.1 Disjuncția exclusivă

Astfel, d(a,a) = 0 și d(a, b) > 0, $\forall$ a!=b. De asemenea, este respectată proprietatea inegalității triunghiului: d(a, b) $\leq$ d(a, c) + d(b, c).

Metrica XOR este unidirecțională: pentru orice punct dat x și distanța d > 0, există exact un punct y astfel încât d(x, y) = d. Abordarea unidirecțională asigură faptul că toate căutările pentru aceeași cheie converg către aceeași cale, fără să se țină cont de nodul original.

Peer-ul dintr-o astfel de rețea stochează o listă de triplete *{adresă IP, port UDP, NodeID}* pentru noduri aflate la o distanță cuprinsă între 2i și 2i+1 de el însuși. Aceste liste sunt numite k-buckets. Fiecare k-bucket este păstrată sortată descrescător după timpul de la ultimul acces. Kademlia implementează o politică de LRE (least recently

seen), prin eliminarea nodurilor de care nu s-a mă auzit de cea mai lungă perioadă de timp.

Protocolul de rutare al sistemului Kademlia se realizează prin transmiterea următoarelor mesaje [9]:

- *Ping* reprezintă trimiterea de semnale de verificare a stării de activitate a unui peer.
- *Store* instruiește un peer să stocheze o pereche {cheie, valoare} pentru accesări ulterioare.
- *Find_node* ia ca parametru un ID de 160 de biți și returnează triplete *{adresă IP, port UDP, NodeID}* pentru k noduri despre care știe că sunt cele mai apropiate de ID-ul țintă.
- *Find_value* este similar cu FIND_NODE: returnează triplete *{adresă IP, port UDP, NodeID}*, cu excepția cazului în care un peer primește o comandă STORE pentru acea cheie, caz în care returnează valoarea.

Nodurile din sistemul Kademlia trebuie să localizeze pe k cele mai apropiate noduri de un id dat. Căutarea începe prin alegerea a X noduri din cel mai apropiat nevid k-bucket și apoi trimiterea de cereri asincrone paralele de tip FIND_NODE către cele X peer-uri pe care le-a ales.

Dacă FIND_NODE eșuează să întoarcă un nod care este mai aproape decât cel mai apropiat nod descoperit, se retrimite comanda FIND_NODE către toate cele k cele mai apropiate noduri pe care nu le-a interogat încă.

Pentru a găsi o pereche {cheie, valoare}, un nod începe să trimită lookup-uri de tip FIND_VALUE pentru a descoperi pe cele k peer-uri cu ID apropiat de cheie.

O rețea peer-to-peer presupune că toți membrii pot accesa rețeaua cu aceleași drepturi. Fiecare nod acționează atât ca producător cât și consumator de resurse. Membrii din rețea își distribuie resurse fără a mai fi nevoie de o autoritate centrală care să le coordoneze activitățile din sistem.

Avantajele acestui tip de rețea (toleranța la defecte, scalabilitatea etc.), împreună cu caracteristicile de bază ale acestuia, face ca rețeaua peer-to-peer să reprezinte cea mai potrivită infrastructură pentru sistemul de radio online cu sistem de recomandare integrat.

3.2. Sisteme de recomandare

Sistemele de recomandare reprezintă direcții de cercetare care continuă să primească atenție la nivel academic. Acestea sunt din ce în ce mai utilizate în mediul online și sunt folosite ca instrumente care oferă facilități de predicție a preferințelor utilizatorilor într-un spațiu de articole dat, pentru a-i ajuta în procesul de selecție a articolelor. Sunt aplicate în diverse domenii, printre care și muzica, unde vin în ajutorul utilizatorilor pentru a filtra și descoperi muzica după preferințele lor.

3.2.1. Clasificare

Un sistem de recomandare este utilizat ca o unealtă care încearcă să sugereze utilizatorilor o gama de articole cat mai apropiată de gusturile și interesele lor. Într-un spațiu de date reprezentat de mulțimea utilizatorilor și a articolelor partajate, scopul unui

sistem de recomandare este de a găsi o estimare cat mai bună pentru funcția care măsoară interesul unui utilizator pentru un set de articole.

Potrivit articolelor [1] și [13], există doi algoritmi cunoscuți de realizare a unui sistem de recomandare care produc rezultate bune: *filtrare colaborativă (CF Collaborative filtering)* și *bazat pe conținut (CBM content-based model).*

Algoritmul de *filtrare colaborativă* utilizează entitățile din aplicație (utilizatori, articole, preferințe, acțiuni etc) și construiește un model bazat pe comportamentul anterior al utilizatorului (în cazul de fată, melodiile ascultate sau căutate înainte, notele date melodiilor), dar și pe comportamentul celorlalți utilizatori. [1, 13] Preferințele utilizatorilor pot fi obținute prin exprimarea lor explicită (prin vot pozitiv sau negativ) sau implicită, prin studierea comportamentului. Filtrarea colaborativa poate fi aplicată pe orice fel de elemente: articole, știri, site-uri, filme, muzică, excursii, etc. și nu necesită intervenția umană pentru a adnota conținutul. Metodele de filtrare colaborativă se pot clasifica la rândul lor in două categorii: metode bazate pe utilizator sau metode bazate pe articol. În cazul metodei bazate pe utilizator, se folosesc preferințele utilizatorilor similari pentru un articol dat pentru a descoperi recomandări de noi articole. În cazul metodei bazate pe articol: se găsesc articolele similare unui articol dat prin cuantificarea anumitor aspecte sociale, cum ar în cazul în care mulți utilizatori care au preferat un anumit articol, au votat pozitiv și un alt articol asemănător, care va fi recomandat mai departe.

Un sistem de recomandare cunoscut care folosește acest algoritm aparține site-ului *Last.fm*, care construiește un profil al utilizatorului, prin care observă ce trupe sau melodii ascultă și îl compară cu profilul altor utilizatori. Astfel, Last.fm va sugera melodii care nu apar în lista utilizatorului, dar sunt adesea ascultate de utilizatori cu gusturi similare.

Recomandarea *bazată pe conținut* folosește o serie de caracteristici discrete ale unui articol, și recomandă utilizatorilor articole cu proprietăți asemănătoare. [1, 13] Acești algoritmi se axează pe conținutul efectiv al articolului si oferă ca recomandări articole similare unui articol inițial pe baza unor metrici specifice (de pildă: tag-uri). Punctul slab al acestei abordări este că nu ia în considerare si partea socială, ce constă în modele de similaritate între profilurile de preferințe ale utilizatorilor. *Pandora Internet Radio* utilizează un sistem de recomandare bazat pe conținut, unde proprietățile unei melodii sau ale unui artist sunt folosite mai departe pentru a crea recomandări.

Un alt model de recomandare este cel bazat pe emoții (*EBM Emotion-based model*), care susține că muzica este un mijloc de exprimare a stărilor interioare. S-a demonstrat că emoțiile transmise de muzica au un rol important în alegerea melodiilor, modelul bazat pe emoții devenind astfel un punct de atracție în cercetarea sistemelor de recomandare. [13] *Musicovery* utilizează modelul emoțional fundamental descoperit de psihologi, în care utilizatorii își exprimă emoțiile într-un spațiu 2D: pozitiv-negativ, respectiv excitant-calm. Din muzică se pot extrage diferite trăsături de percepție: energie, ritm, armonie, spectru muzical.

Pentru rezulate mai bune, diferite metode se pot combina într-un sistem hibrid de recomandare, prin cascadare, comutare, combinare a trăsăturilor sau amestecare a rezultatelor. Abordările în cazul modelelor hibride pot fi implementate în mai multe feluri: prin calcularea predicțiilor bazate pe conținut și colaborativ separat, iar apoi combinând rezultatele, prin adăugarea a diferitor capacități ale modelui colaborativ la

modelul bazat pe conținut (sau invers) sau prin unificarea ambelor modele într-unul singur.

Potrivit articolului [13], s-a demonstrat că prin combinarea a cel puțin două modele de sisteme de recomandare, performanțele pot fi mai mari decât în cazul folosirii unei singure metode, din moment ce noul sistem va beneficia de avantajele fiecărui model în parte.

3.2.2. Caracteristici

Obiectivul principal al unui sistem de recomandare este de a găsi algoritmul care să producă cele mai satisfăcătoare rezultate. Pe de altă parte, există și o serie de factori care definesc un sistem de recomandare.

Diversitatea constituie un aspect de dorit în unele sisteme de recomandare, atunci când recomandarea de elemente aproape duplicate sau foarte asemănătoare minimizeaza impactul sau utilizabilitatea sistemului. Utilizatorii tind să fie satisfăcuți cu recomandări mai diversificate, decât cu articole dintr-un spațiu de date apropiat. (de exemplu: recomandarea unei melodii al unui alt artist vs. recomandarea unei melodii al aceluiași artist).

Un alt aspect este legat de *persistența* recomandărilor. În unele situații poate fi mai eficient ca un utilizator să primească recomandări pe care le-a mai primit o dată, elemente deja cunoscute, decât să îi fie mereu sugerate articole noi. Utilizatorii pot alege să ignore elementele pe care le văd pentru prima dată, din cauza lipsei de încredere sau a necunoașterii.

Încrederea în sistemul de recomandare este un criteriu important în stabilirea perfomanței. Un sistem de recomandare poate fi insignifiant dacă utilizatorii nu au încredere în el. Încrederea poate fi construită în timp, prin satisfacerea pretențiilor utilizatorilor și prin explicarea funcționalității sistemului: pe baza căror atribute sugerează un anumit articol.

Transparența recomandării reprezintă o calitate prin care sistemul oferă o motivație pentru recomandarea unui anumit articol, iar astfel poate crește încrederea unui utilizator în calitatea algoritmului de recomandare utilizat.

Scalabilitatea sistemului este importantă în contextul gestiunii de volume mari de date și în continuă creștere. Algoritmii de recomandare sunt în general iterativi, bazați pe procesarea extensivă a datelor. Sunt de dorit timpi de răspuns mici pentru utilizatorul final, iar în acest sens sunt folosite procesări offline, și în mică parte procesări online, la momentul cererii formulate de utilizator.

3.2.3. Probleme ale sistemelor de recomandare

Sistemele de recomandare se pot confrunta inevitabil cu o serie de probleme clasice, cum ar fi „cold start” (e nevoie de timp și date multe pentru a se ajunge la recomandări corecte), înclinație spre melodiile foarte cunoscute și ascultate, efort uman. [13]

Problema „cold start” apare în cazul utilizatorilor noi ai sistemului, atunci când nu există informație explicită despre preferințele acestora pentru a li se putea produce recomandări. În cazul modelului bazat pe filtrare colaborativa, este cunoscută nevoia de date despre utilizatori pentru a putea realiza recomandări precise. O soluție ar fi completarea unui chestionar legat de preferințe la intrarea în sistem (nerecomandat, un

sistem de recomandare inteligent ar trebui să ceară explicit cat mai puține informații de la utilizatori). Altă soluție mai plauzibilă ar fi folosirea unor agenți de învățare care să descopere profilul utilizatorilor pe baza acțiunilor acestuia.

Problema „first rater" este problema echivalentă în cazul articolelor nou intrate în sistem, pentru care încă nu există nicio preferință exprimată. Este mai puțin probabil să fie recomandat vreun articol adăugat recent, în locul unui articol care are deja o istorie îndelungată (exemplu: număr de voturi). În acest caz, soluția ar fi realizarea unui model hibrid de recomandare, care pe lângă filtrarea colaborativa, să realizeze și predicții bazate pe conținutul articolelor, pentru a putea găsi articole similare (exemplu: recomandarea unor melodii de același gen). Aceeași soluție poate fi aplicată și în cazul în care sistemul de recomandare returnează doar cele mai „populare" articole din sistem, dovedind astfel înclinație spre cele mai cunoscute articole.

Poate apărea problema manipulării sau influențării sistemului de recomandare de către un grup mic de utilizatori, de exemplu prin voturi negative la adresa unui articol oferit de competitori. În acest sens, sistemul trebuie să discearnă și să elimine tentativele de manipulare.

În ceea ce privește efortul uman, este complicat de implementat și testat un sistem de recomandare, întrucât necesită multe date de intrare pentru a se atinge rezultate satisfăcătoare. De asemenea, este nevoie de timp în momentul în care un sistem de recomandare pornește de la zero până să ajungă la nivelul în care acesta returnează cele mai potrivite articole.

Pe de altă parte, este știut faptul că nu se poate demonstra dacă un sistem recomandă mereu cele mai bune articole pentru un utilizator. Articolele fac parte din domenii cu grad ridicat de subiectivitate, astfel că nu se poate ști niciodată care este cel mai bun articol pentru un anumit individ la un moment dat. (exemplu: care este cea mai plăcută melodie de către un utilizator). Se poate realiza în schimb găsirea unor articole apropiate de gusturile și interesele acestuia. (exemplu: dacă unui utilizator îi place o melodie de un anumit gen, e foarte posibil să îi placă și alte melodii de acel gen).

3.2.4. Sisteme de recomandare descentralizate

Sistemele de recomandare tradiționale se rezumă deobicei la o relație client-server, și se bazează pe o autoritate centrală (serverul) care deține și gestionează toate datele sistemului de recomandare și cunoaște tot spațiul de date al acestuia: utilizatorii, mulțimea articole, respectiv preferințele utilizatorilor. În cazul acesta, se pot aplica diverse modele de recomandare, cel mai frecvent fiind cel hibrid, prin combinarea modelului bazat pe conținut și cel bazat pe filtrare colaborativa. Toate calculele necesare se realizează în același loc, profilele utilizatorilor sunt de asemenea calculate și reținute pe serverul central, nu este nevoie de comunicare propriu-zisă între utilizatori, ci doar între utilizator și server.

Potrivit articolului [11], într-un sistem de recomandare descentralizat, bazat pe o rețea Peer-to-Peer, modelarea gusturilor unui utilizator devine mai dificilă. Sistemul devine unul distribuit, nicio entitate centrală nu adună datele utilizatorilor, astfel că nu deține nimeni, în niciun moment, cunoștințe complete legate de articole, utilizatori și preferințele acestora. Fiecare nod este astfel responsabil să-și găsească cele mai potrivite articole din rețea în urma calculelor făcute local.

Dacă la un sistem de recomandare bazat pe client-server putea fi aplicată orice metodă de recomandare, într-un mediu descentralizat, distribuit, cum ar fi partajarea de fișiere într-o rețea Peer-to-Peer, condițiile de aplicare a acestor metode nu sunt întotdeauna realizabile.

De-a lungul timpului, au fost propuse mai multe variante de creare a unui sistem de recomandare distribuit. [11] Prima încercare a fost prin difuzarea broadcast a voturilor, ceea ce a produs probleme de scalabilitate și ineficiență. O altă abordare care se bazează pe metoda de filtrare colaborativă propune crearea unor tabele de prietenii bazate pe articole („item-based buddy tables"), în care informații despre articole similare sunt stocate local în tabelele acelor articole. Recomandarea pentru diferiți utilizatori se realizează prin tabelele acelor articole pe care un utilizator le-a descărcat înainte.

Sistemul de recomandare descentralizat DeHinter propune ideea de afinitate între utilizatori: între doi utilizatori care împărtășesc un subset de articole comune există probabilitate destul mare ca aceștia să aibă gusturi comune, astfel că un utilizator poate fi interesat de articolele celuilalt utilizator. [11]

Intr-un mod formal, putem defini $U = \{ u_1, u_2 \dots u_n \}$ mulțimea utilizatorilor din sistem, $A = \{ a_1, a_2 \dots a_l \}$ mulțimea de articole partajate de utilizatori, iar $P(A)$ notăm familia părților lui A (familia submulțimilor). Fie funcția f : U → P(A) care mapează fiecare articol la utilizatorul care l-a introdus in sistem. Astfel, f(u_i) va returna mulțimea de articole distribuite de utilizatorul u_i. În mod evident, $\bigcup_{i=1}^{n} f(u_i) = A$. [11]

Pentru a profită de puterea relațiilor sociale și asemănărilor între membrii unei rețele, a fost introdus termenul de afinitate între utilizatori, unde „prietenia" dintre ei poate fi definită de numărul de resurse pe care le au în comun. Astfel, funcția de afinitate devine: $Aff : U^2 \to \aleph^+$, unde $Aff(u_i, u_j) = | f(u_i) \cap f(u_j) |$.

În urma calculării funcției de afinitate între fiecare dintre utilizatori, se crează așa numita „rețea de afinitate" (affinity network), ce reprezintă un graf în care utilizatorii sunt nodurile. Doi utilizatori u_i și u_j vor fi legați printr-o muchie în graf doar în cazul în care au în comun cel puțin *m* articole.

Astfel, pot fi definite o familie de grafuri , $G^m = (U, E^m)$, unde
$e_{ij}^m \in E^m \Leftrightarrow Aff(u_i, u_j) \geq m$. [11]

În realizarea acestui sistem cu rețea de afinitate, se ține cont de câțiva factori importanți, și anume: se renunță la ideea de profil al utilizatorului sau vector de rating-uri (voturi) specific unui sistem de recomandare centralizat, eliminându-se astfel încărcătura propagării datelor în rețea. Se presupune că dacă un utilizator distribuie un fișier, atunci acesta arată interes în fișierul respectiv. Pentru a genera rețeaua de afinitate, este nevoie să se cunoască setul de date al fiecărui utilizator. Pe de altă parte, sistemul este autonom și complet transparent: articolele sugerate sunt returnate utilizatorului care doar folosește sistemul și nu realizează niciun alt fel de acțiune (cum ar fi votarea unui articol).

În concluzie, principala provocare a acestui proiect este de a găsi metoda cea mai potrivită de creare a unui sistem de recomandare care să lucreze împreună cu structura DHT (Distributed Hash Table) a rețelei P2P. De asemenea, se urmărește minimizarea cat mai mult a dezavantajelor inerente care vin cu un asemenea sistem, cum ar fi scalabilitatea sau timpul de creare a profilului unui utilizator sau de notare a melodiilor din rețea.

3.3. Sisteme similare

Odată cu extinderea rapidă a rețelelor de calculatoare din ultimele decenii, internetul a devenit principala sursă de utilizare a informațiilor multimedia, cum ar fi cărțile, muzica sau filmele. În zilele noastre, aplicațiile care utilizează internetul sunt din ce în ce mai căutate pentru descoperirea, descărcarea sau folosirea de media.

Există pe piață sisteme sau aplicații asemănătoare care permit utilizatorilor să se conecteze sau logheze la sistem, să își distribuie melodiile sau să primească un playlist de melodii sugerate de un sistem de recomandare.

3.3.1. Last.fm

Last.fm[1] este un site web de muzică, fondat în Marea Britanie în anul 2002. Folosind un sistem de recomandare numit „Audioscrobbler", Last.fm construiește un profil detaliat al utilizatorului, în care sunt înregistrate detaliile melodiilor pe care acesta le ascultă. Aceste informații sunt trimise („scrobbled") către baza de date a site-ului prin intermediul unui plugin instalat pe music player-ul utilizatorului (The Scrobbler). Termenul de „scrobbling" este inventat de cei de la Last.fm pentru a descrie noțiunea de înregistrare (logging) a unei melodii în momentul în care un utilizator o ascultă.

Site-ul cere utilizatorilor să se înregistreze si să își creeze propriul cont înainte de a putea utiliza aplicația. Un utilizator își poate crea un profil muzical prin următoarele metode: ascultând colecția personală de muzică pe un player ce are instalat pluggin-ul de „Audioscrobbler" sau ascultând serviciul de radio al site-ului. Fiecare melodie este adăugată unui log (unei înregistrări) care se transmite bazei de date și după care se calculează recomandările muzicale. De asemenea, Last.fm crează automat fiecărui utilizator o pagină cu datele personale (exemplu: nume, poza de profil), data înregistrării și număr total de melodii ascultate. Last.fm pune accent și pe latura socială, astfel că pe pagina de profil un utilizator poate avea o listă de prieteni, grupuri (exemplu: grupuri care au anumite lucruri în comun, cum ar fi un gen, artist etc.), evenimente sau „vecini" muzicali, care au gusturi asemănătoare cu utilizatorul inițial.

Recomandările sunt calculate folosind un algoritm de filtrare colaborativă, unde utilizatorii pot căuta și asculta melodii care nu se află pe lista lor, dar se găsesc pe profilele altor utilizatori cu gusturi similare. De asemenea, utilizatorul poate primi și melodii care i-au fost recomandate direct acestuia sau grupurilor din care face parte. Utilizatorul poate la rândul său să recomande manual anumite melodii, albume sau anumiți artiști altor utilizatori din lista de prieteni sau din grupurile din care face parte. Last.fm dispune de o stație de radio online care generează automat unui utilizator un playlist de melodii filtrate de sistemul de recomandare, bazat pe activitatea acestuia din ultima săptămână.

3.3.2. Pandora Internet Radio

Pandora[2] este un serviciu de difuzare automată și recomandare muzică (music streaming). Pandora își are originea în proiectul Music Genome Project care a fost pentru prima dată realizat în anul 1999 de către Tim Westergren și Will Glazer. Ideea inițială a

[1] *Last.fm*. Preluat de pe Last.fm: http://www.last.fm/

[2] *Pandora Internet Radio*. Preluat de pe Pandora Internet Radio: http://www.pandora.com/

fost crearea unui sistem matematic complex care să clasifice și să grupeze melodiile similare folosind un vector de „gene”. Astăzi, sistemul folosește în jur de 400 de gene diferite pentru a identifica și a grupa melodiile cat mai exact, ceea ce aduce sistemul de recomandare al acestui serviciu în categoria modelelor bazate pe conținut.

Pandora Radio este considerat ca fiind un serviciu de muzică personalizat. În locul unei simple experiențe a utilizatorului de a asculta un post de radio care difuzează muzica fără să țină cont de preferințele acestuia, biblioteca muzicală a serviciului Pandora folosește Music Genome Project pentru a recomanda melodii după gusturile fiecăruia.

O stație de radio este pornită prin specificarea de către utilizator a unui artist, a unei melodii sau combinație a acestora. Ascultătorul poate alege să îi fie difuzate canale deja generate după un anume gen, poate alege canalul altui utilizator sau poate să își creeze propriul playlist în funcție de interesele sale muzicale. Fiecare melodie ascultată poate fi votată printr-un vot pozitiv sau negativ, voturi prin intermediul cărora se pot calcula mai departe următoarele recomandări. De exemplu, două voturi negative consecutive pentru un artist va duce la ignorarea acestuia din playlist-ul format mai departe.

Marele dezavantaj al acestui serviciu este indisponibilitatea sa din cauza licențierii melodiilor, fiind valabil doar Statele Unite ale Americii, Australia și Noua Zeelanda.

3.3.3. Spotify

Spotify[3] este un serviciu de difuzare de muzică (music streaming), care oferă acces la o librărie de peste 8 milioane de melodii, cu drepturi de gestionare restricționat de la case de discuri precum Sony, EMI, Warner Music Group și Universal Music Group. Aplicația oferă posibilitatea de a accesa melodii, albume, artiști, genuri sau case de discuri. Utilizatorul poate construi și distribui playlist-uri de melodii, sau poate primi la rândul sau automat un playlist, ceea ce transformă serviciul într-un radio online.

Conform [6], infrastructura serviciului este realizată prin combinarea arhitecturii client-server cu protocolul Peer-to-peer. Toate datele utilizatorilor sunt reținute în două centre de date (la Londra și Stockolm), la care un peer se conectează în mod aleator. Principalul motiv pentru dezvoltarea unei arhitecturi bazate pe protocolul P2P a fost îmbunătățirea scalabilității serviciului prin micșorarea încărcăturii serverelor și a lățimii de bandă a resurselor. Rețeaua P2P este una nestructurată, permite tuturor nodurilor să participe în mod egal în rețea. Un utilizator se va conecta la un peer nou doar dacă dorește să descarce de la acesta o melodie pe care crede că o are. Utilizatorii stochează local melodiile pe care le-au descărcat și pe care, la rândul lor, le pun la dispoziția celorlalți utilizatori.

Ca funcționalitate, Spotify nu are propriul sistem de recomandare prin care să fie create liste de melodii după preferințele utilizatorilor. În schimb, poate fi folosit împreună cu Last.fm. Integrarea acestui serviciu în Last.fm dă voie utilizatorilor care ascultă muzică de pe Spotify să trimită melodiile ascultate în contul lor de pe Last.fm. „Scrobbling-ul” prin intermediul Last.fm permite astfel ținerea evidenței melodiilor

[3] *Spotify*. Preluat de pe Spotify: http://www.spotify.com/

ascultate, iar cu ajutorul unei aplicații numite Last.fm for Spotify, utilizatorii de pe Spotify primesc melodii bazate pe recomandările de pe Last.fm.

La fel ca în cazul precedent, un dezavantaj al acestui serviciu este faptul că nu e valabil în toate țările (disponibil în aproximativ 20 de țări, printre care România nu se află).

În concluzie, prin prezentarea acestor sisteme deja existente pe piață, similare cu soluția propusă mai departe, s-a încercat punerea în evidență a trăsăturilor fiecărui serviciu de muzică pentru a putea face o comparație între acestea și soluția de fata.

Primele două sisteme prezentate propun o arhitectură client-server, unde fiecare utilizator, pentru a putea utiliza aplicația, este nevoit să se conecteze la un server central, care stochează toate datele din sistem. În ceea ce privește sistemele de recomandare, Last.fm vine cu o metodă de filtrare colaborativa prin care construiește un profil muzical al utilizatorului. Pandora Radio folosește, pe de altă parte, un model bazat pe conținut, unde recomandările se fac în funcție de „genele" melodiilor. Aceste aplicații prezintă caracteristicile unei arhitecturi client-server și a unor sisteme de recomandare centralizate, ceea nu este posibil în cazul sistemului de radio online bazat pe o rețea peer-to-peer cu un sistem de recomandare descentralizat.

Cea de-a treia aplicație, Spotify, deși are la bază o combinație între arhitecturile client-server și peer-to-peer, prezintă marele dezavantaj că nu are un sistem de recomandare integrat. De asemenea, deși căutarea și descărcarea se realizează în rețea, există o entitate centrală care monitorizează toți utilizatorii și la care aceștia se conectează.

Tabelul următor prezintă o comparație între sistemele prezentate în acest subcapitol și sistemul de față:

Tabel 3.1 Comparație între sistemele existente pe piață și sistemul curent

Criteriu	***Radio Online***	***Last.fm***	***Pandora***	***Spotify***
Arhitectură P2P	Da	Nu	Nu	Da
Arhitectură client-server	Nu	Da	Da	Da
Sistem de recomandare integrat	Da	Da	Da	Nu
Sistem de recomandare distribuit	Da	Nu	Nu	Nu
Radio	Da	Nu	Da	Da
Căutare melodii	Da	Da	Nu	Da
Disponibil în România	Da	Da	Nu	Nu

Sistemul propus de noi nu dispune de o autoritate centrală care să rețină sau să gestioneze datele în cazul determinării recomandărilor. Toți utilizatorii conectați la rețea vor avea aceleași drepturi și sarcini, astfel că sistemul de față vine cu o arhitectură peer-to-peer total distribuită, și cu un sistem de recomandare deasemenea distribuit.

Capitolul 4. Analiză şi Fundamentare Teoretică

Cu scopul de a se ajunge la un sistem scalabil, performant şi uşor de folosit de către utilizatori, au fost căutate cele mai potrivite soluții de realizare și implementare a sistemului de radio online. În acest capitol, vor fi prezentate cerințele pe care trebuie să le respecte sistemul și funcționalitățile de bază pe care le oferă. Va fi realizată o analiză amănunțită a sistemului propus, a arhitecturii și algoritmilor folosiți, împreună cu tehnologiile care se mapează cel mai bine pe ideea proiectului.

4.1. Cerințele sistemului

Cerințele reprezintă descrieri ale serviciilor pe care trebuie să le furnizeze sistemul sau a unor constrângeri pe care acesta trebuie să le satisfacă. Implementarea acestor cerințe vor contribui la realizarea sistemului propus, de radio online bazat pe o rețea peer-to-peer cu sistem de recomandare integrat.

4.1.1. Cerințe funcționale

Cerințele funcționale descriu funcțiile pe care trebuie să le realizeze sistemul din punctul de vedere al unui utilizator. Sistemul de față propune realizarea și implementarea următoarelor funcționalități:

Tabel 4.1 Cerințele funcționale

CF 1	**Conectarea la o rețea P2P**
CF 1.1	Conectarea ca Master Peer
CF 1.2	Conectarea la o rețea deja existentă
CF 2	**Distribuirea melodiei**
CF 3	**Creare playlist bazat pe sistemul de recomandare**
CF 3.1	Căutare melodie
CF 4	**Funcționalitate de MP3 Player**
CF 4.1	Redarea melodiei (Play)
CF 4.2	Oprirea melodiei (Stop)
CF 4.3	Redarea următoarei melodii (Skip)
CF 5	**Notarea unei melodii**
CF 5.1	Apreciere pozitivă (Like)
CF 5.2	Apreciere negativă (Dislike)
CF 6	**Deconectare**
CF 6.1	Deconectarea de la rețea
CF 6.2	Închiderea aplicației

4.1.2. Cerințe nefuncționale

Cerințele nefuncționale ale unui sistem specifică proprietățile sau calitățile acestuia, și sunt atașate cerințelor funcționale. De asemenea, definesc constrângerile

impuse, constrângeri legate de cost, calitate, performanță sau construcție. Cerințele nefuncționale ale sistemului propus sunt prezentate în tabelul următor:

Tabel 4.2 Cerințele nefuncționale

CNF 1 ***Construcție-infrastructură***	Soluția propusă se bazează doar pe mecanismele rețelei P2P. Nu există niciun element central (server) care să stocheze si să gestioneze datele din sistem. Nu există niciun administator care să controleze datele din rețea. Toți utilizatorii aplicației vor folosi același produs software și vor avea același rol în sistem. Fiecare utilizator își va crea local propriul profil muzical și va fi responsabil cu determinarea playlist-ului care îi va fi redat, ca urmare a folosirii unui sistem de recomandare distribuit.
CNF 2 ***Scalabilitate***	Rețeaua trebuie să suporte un număr ridicat de utilizatori conectați și un volum mare de date (melodii distribuite în rețea). Sistemul propus beneficiază de avantajele oferite de o infrastructură de tip P2P în ceea ce privește scalabilitatea.
CNF 3 ***Eficiență***	Sistemul trebuie să utilizeze cat mai eficient traficul de date între utilizatori, într-un timp cat mai scurt, în momentul descărcării de melodii de la un peer la altul. Pentru asta, în momentul creării playlist-ului de melodii pentru un utilizator, sistemul descarcă în background un număr minim de melodii în avans, pentru a limita timpul de așteptare al utilizatorilor după o melodie și pentru a evita întârzierile.
CNF 4 ***Robustețe***	Sistemul trebuie să fie tolerant la defecte și erori. Datorită avantajelor rețelei P2P, sistemul nu este afectat în momentul în care utilizator părăsește rețeaua. În momentul apariției unei erori pe partea de interfață grafică (la introducerea unor date invalide), sistemul nu se va bloca și va fi în continuare în stare de funcționare. Interfața grafică îi permite utilizatorului să își aleagă portul de pe care să se conecteze și să scrie adresa ip și portul peer-ului în a cărei rețea vrea să se conecteze. După conectare, interfața grafică îi oferă utilizatorului posibilitatea de a își alege folderul din care vor fi preluate melodiile proprii și descărcate cele noi. Lista melodiilor distribuite în rețea va fi afișată în interfața grafică. De asemenea, va fi afișată și lista melodiilor redate și descărcate de la alți utilizatori, împreună cu adresa peer-ilor de la care au fost descărcate acestea (ip + port). În momentul căutării unei melodii după un cuvânt cheie, interfața grafică va afișa un număr maxim de melodii găsite după acel cuvânt, împreună cu adresa peer-ului la care se găsește acea melodie. Utilizatorul trebuie să aibă posibilitatea să voteze fiecare melodie din rețea prin intermediul butoanelor de like/dislike. Nu este lăsat să dea mai mult de un vot pozitiv sau negativ pentru o melodie.
CNF 5 ***Utilizabilitate***	Sistemul are o interfață ușor de utilizat de către orice utilizator.

CNF 6 ***Mentenabilitate***	Sistemul este ușor de întreținut, datorită utilizării arhitecturii pe nivele. Un asemenea model arhitectural oferă flexibilitate și mentenabilitate, ca urmare a separării dintre interfața grafică, logica sistemului și infrastructura acestuia.

Sistemul poate rula pe oricare din următoarele sisteme de operare Windows: XP, Vista, 7, 8. Nu este nevoie de licențiere pentru acest produs, soluția implementată folosește biblioteci publicate in GPL sau open source.

4.2. Cazuri de utilizare

Cazurile de utilizare oferă o perspectivă de ansamblu a comportamentului și a funcționalității sistemului din punctul de vedere al utilizatorilor. Scopul acestora este de a descrie cerințele funcționale ale sistemului.

Utilizatorul acestui sistem de radio online este consumatorul de muzică, persoana care va folosi aplicația pentru a asculta, pentru a distrbui propria muzica sau pentru a primi melodii de pe stațiile altor utilizatori din rețea. El va reprezenta actorul principal al cazurilor de utilizare pentru soluția propusă, are posibilitatea să aleagă operațiile ce urmează să fie executate.

Modelul use case al sistemului de față, format din totalitatea cazurilor de utilizare împreună cu actorii este reprezentat prin următoarea diagramă use case:

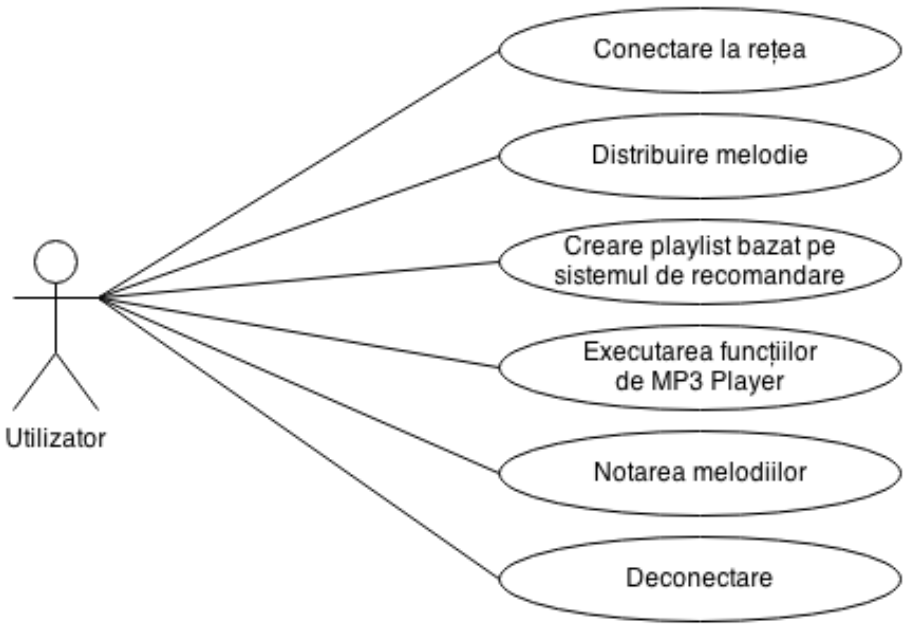


Figura 4.1 Diagrama cazurilor de utilizare

UC 1: Titlu: *Conectarea la rețeaua P2P*

Prima cerință funcțională este descrisă în primul caz de utilizare și este ilustrată în figura 4.2:

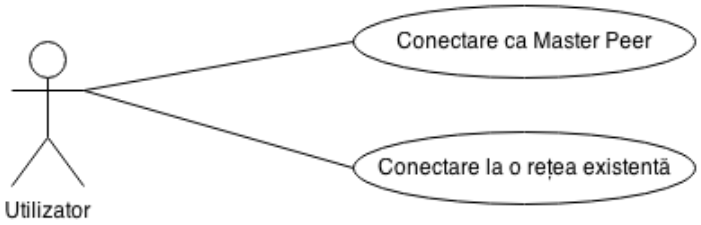


Figura 4.2 Diagrama use case pentru CF 1

UC 1.1 : Titlu: *Conectarea la retea ca Master Peer*
Rezumat: Acest caz de utilizare permite unui utilizator al acestei aplicatii sa se conecteze la o retea P2P pentru a putea beneficia de functionalitatile sistemului.
Actori: Utilizatorul (primar)
Precondiții: Utilizatorul nu trebuie sa fie conectat la nicio alta retea.
Descriere: Aplicatia presupune crearea unei noi retele, unde utilizatorul va reprezenta peer master-ul retelei respective.
Postcondiții: Crearea unui DHT ce contine initial doar id-ul utilizatorului. Urmatorul membru se va conecta direct la adresa lui.
Scenariu de succes:

1. Utilizatorul porneste aplicatia
2. Sistemul ii cere introducerea unei adrese IP sau conectarea ca Master Peer
3. Userul introduce portul de pe care sa se conecteze
4. Userul alege conectarea ca Master Peer
5. Sistemul creaza un nou DHT care contine doar id-ul utilizatorului si informeaza userul de reusita conectarii

Secvenţe alternative:
3A1) Utilizatorul introduce o data invalida in campul de setare a portului propriu: intervine la *pasul 3*

6. Sistemul informeaza utilizatorul ca a introdus o valoare invalida
   Scenariul se intoarce la *pasul 2.*

5A1) Sistemul intampina probleme in crearea tabelului DHT a retelei P2P (exemplu exceptie: timeout): intervine la *pasul 5*

7. Sistemul informeaza utilizatorul de esuarea conectarii
   Scenariul se intoarce la *pasul 2.*

UC 1.2. Titlu: *Conectarea la rețea deja existentă*
Rezumat: Acest caz de utilizare permite unui utilizator al acestei aplicatii sa se conecteze la o retea P2P pentru a putea beneficia de functionalitatile sistemului.
Actori:

- Utilizatorul A (primar)
- Utilizatorul B

Precondiții: Adresa IP si portul nodului la care se contecteaza utilizatorul A este cunoscuta. (ex: adresa utilizatorului B) Utilizatorul nu trebuie sa fie conectat la nicio alta retea.
Descriere: Userul A se conecteaza la o retea existenta: userul cunoaste adresa (IP si port) unui membru din retea si trimite adresa sistemului. Sistemul incearca sa stabileasca o conexiune intre cele doua adrese. Daca incercarea este reusita, structura DHT a retelei este reactualizata cu adresa utilizatorului. Astfel, utilizatorul este de asemenea conectat la retea.
Postcondiții:

- DHT-ul retelei va contine si adresa utilizatorul A.
- Utilizatorul A va avea acces la retea prin intermediul utilizatorului B (va fi conectat de acesta).

Scenariu de succes:

1. Userul porneste aplicatia
2. Sistemul ii cere introducerea unei adrese IP sau conectarea ca Master Peer
3. Userul introduce portul de pe care sa se conecteze
4. Utilizatorul introduce adresa utilizatorul B
5. Sistemul incearca sa conecteze utilizatorul A de utilizatorul B
6. Sistemul actualizeaza DHT-ul retelei cu ip-ul utilizatorului A

Secvenţe alternative:
3A1, 4A1) Utilizatorul introduce date invalida in campurile de setare a porturilor si a adresei IP: intervine la *pasul 3,4*

7. Sistemul informeaza utilizatorul ca introdus o valori invalide
   Scenariul se intoarce la *pasul 2.*

4A2) Userul introduce o adresa care nu respecta formatul unei adrese IP: intervine la *pasul 4*

8. Sistemul informeaza userul ca nu a introdus o adresa IP corecta
   Scenariul se intoarce la *pasul 2.*

4A3) Sistemul intampina probleme in crearea unei conexiuni intre utilizatorul A si adresa introdusa (exemplu exceptie: timeout conexiune): intervine la *pasul 4*

9. Sistemul informeaza userul ca nu se poate conecta la adresa introdusa
   Scenariul se intoarce la *pasul 2.*

5A1) Sistemul nu reuseste sa actualizeze DHT-ul (ex: conexiune slaba, timeout, deconectare internet): intervine la *pasul 5*

10. Sistemul informeaza userul ca nu s-a putut realiza conectarea la retea.
    Scenariul se intoarce la *pasul 2.*

A doua cerință funcțională a sistemului este descrisă de al doilea caz de utilizare:

UC 2. Titlu: *Distribuirea melodiilor in rețea*
Rezumat: Acest caz de utilizare permite unui utilizator sa distribuie melodiile proprii, sa le adauge in retea pentru a fi disponibile si celorlalti membrii.
Actori: Utilizatorul (primar)
Precondiții: Utilizatorul trebuie sa fie conectat la o retea

Descriere: Aplicatia ofera utilizatorilor functionalitatea de adaugare de fisiere intr-o retea P2P, astfel ei pot sa isi distribuie melodiile proprii. Prin operatia de publicare a melodiei in retea, se va crea o noua entitate cheie-valoare in DHT-ul retelei. Metadata melodiei va fi impartita in cuvinte cheie (dupa care va fi posibila cautarea), iar fiecare cuvant va fi la randul sau cheie in DHT. Valoarea va reprezenta un obiect ce va contine metadata melodiei si locatia acestei melodii (hash-ul melodiei).
Postcondiții: DHT va contine hash-ul melodiei (locatia unde se gaseste melodia). Aceasta va fi disponibila pentru download-are tuturor membrilor retelei.
Scenariu de succes:

1. Sistemul cere utilizatorului sa adauge calea catre folderul de unde sa ia melodiile ce vor fi distribuite in retea
2. Utilizatorul alege folderul prin navigare
3. Sistemul verifica duplicitatea fiecarei melodii distribuite de catre user inainte sa o adauge din nou (verificarea metadatelor)
4. Sistemul incepe publicarea pe rand a fiecarei melodii in retea
5. Sistemul anunta utilizatorul de succesul(/insuccesul) adaugarii fiecarei melodii si de finalizarea operatiei
6. Sistemul anunta ceilalti membrii din retea de melodiile distribuite de utilizator, pentru a calcula relatiile de afinitate intre peeri

Secvenţe alternative:
2A1) Userul alege un folder ce nu contine nicio melodie: intervine la *pasul 2*

7. Sistemul informeaza userul ca nu exista nicio melodie in folder
   Scenariul se intoarce la *pasul 1*.

4A1) Sistemul intampina probleme in publicarea melodiei in tabelul DHT a retelei P2P (exemplu exceptie: timeout): intervine la *pasul 4*

8. Sistemul informeaza utilizatorul de esuarea publicarii melodiei respective si trece la melodia urmatoare
   Scenariul continua cu *pasul 4.*

1A1, 2A2) Utilizatorul decide anularea operatiei (apasarea butonului Cancel)

9. Sistemul alege folderul setat implicit de catre sistem (Home) de unde vor fi preluate melodiile.
   Scenariul continua cu *pasul 3.*

6A1) Sistemul intampina probleme la transmiterea de date intre peeri prin intermediul retelei P2P sau in momentul calcularii afinitatii intre acestia(exemplu exceptie: timeout sau in retea exista un singur peer, master peer): intervine la *pasul 6*

10. Sistemul ignora lipsa unui rezultat favorabil al operatiei

A treia cerință funcționala este prezentată in cazul al treilea de utilizare:

UC 3. Titlu: *Crearea unui playlist pe baza sistemului de recomandare in funcție de preferințele utilizatorului, după căutarea unei melodii in sistem*
Rezumat: Acest caz de utilizare permite unui utilizator sa isi creeze playlist-ul cu melodiile sugerate de sistemul de recomandare, in urma unei melodii cautate.
Actori:

- Utilizatorul (primar)

Precondiții: Utilizatorul trebuie sa fie conectat la o retea
Descriere: Sistemul va crea automat un playlist pe baza recomandarilor in urma cautarii unei melodii de catre un utilizator. Căutarea se va face prin adaugarea unor cuvinte cheie de catre utilizator, ce vor tine de atributele melodiei: gen, artist, titlu. Aceste atribute se regasesc in DHT-ul retelei la nivel de chei ale entitatilor. Sistemul trimite utilizatorului melodiile gasite cautate dupa chei in DHT, din care utilizatorul isi alege melodia pe care doreste să o asculte. In functie de melodia cautata si aleasa de acesta, sistemul continua sa caute in retea melodii asemanatoare, in functie de sugestiile sistemului de recomandare si de aprecierile utilizatorului. Melodiile date de sistem vor fi descarcate din retea si stocate local temporar.
Postcondiții: Obtinerea unui playlist pe care utilizatorul sa il poata asculta
Scenariu de succes:

1. Utilizatorul adauga cuvintele cheie in campul de cautare a melodiilor, in functie de atributele melodiei cautate (artist, gen sau titlu)
2. Sistemul cauta in retea melodiile care sa respecte cerintele utilizatorului
3. Sistemul ii ofera lista cu melodiile gasite (lista cu un numar maxim predefinit de melodii)
4. Utilizatorul alege melodia pe care doreste sa o asculte
5. Sistemul descarca melodia din retea pe discul utilizatorului
6. Sistemul continua sa incarce in background un numar predefinit de melodii sugerate de sistemul de recomandare

Secvenţe alternative:
1A1) Userul nu adauga niciun cuvant: intervine la *pasul 1*

7. Sistemul informeaza utilizatorul ca nu a adaugat niciun cuvant dupa care sa se realizeze cautarea

   Scenariul se intoarce la *pasul 1*.

2A1) Sistemul intampina probleme in cautarea valorilor in tabelul DHT a retelei P2P (exemplu exceptie: timeout): intervine la *pasul 2*

8. Sistemul informeaza utilizatorul de esuarea cautarii

   Scenariul se intoarce la *pasul 1*.

3A1) Sistemul nu gaseste niciun rezultat: intervine la *pasul 3*

9. Sistemul informeaza utilizatorul ca nu a gasit nicio melodie care sa corespunda cuvintelor cheie date de acesta

   Scenariul se intoarce la *pasul 1*.

5A1) Sistemul intampina probleme in descarcarea valorilor din tabelul DHT a retelei P2P (exemplu exceptie: timeout): intervine la *pasul 5*

10. Sistemul informeaza userul de esuarea descarcarii

    Scenariul se intoarce la *pasul 3.*

2A2, 3A1) Userul decide anularea operatiei (apasarea butonului Cancel)

1. Sistemul opreste executarea operatiei.

   Scenariul continua cu *pasul 1*.

Cerința funcțională cu numărul 4 este descrisă în urmatorul caz de utilizare și este ilustrat în figura 4.3:

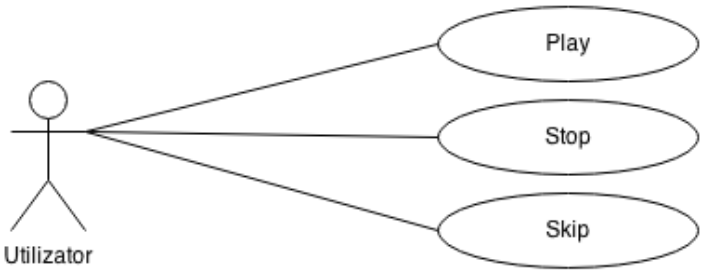


Figura 4.3 Diagrama use case pentru CF 4

UC 4. Titlu: *Executarea funcțiilor de MP3 Player a aplicației*
Rezumat: Acest caz de utilizare permite unui utilizator sa foloseasca aplicatia ca pe un MP3 Player, unde dispune de functiile de play, stop sau skip a melodiilor
Actori: Utilizatorul (primar)
Precondiții: Utilizatorul trebuie sa fie conectat la o retea si sa aiba deja un playlist pe care sa poata sa il asculte (UC 3.3)
Descriere: Sistemul se comporta ca un MP3 Player: utilizatorul poate asculta o melodie (play), poate sa o opreasca (stop) sau sa treaca la urmatoarea melodie din playlist (skip). In momentul trecerii voite la urmatoarea piesa, sistemul de recomandare va nota negativ melodia respectiva (dislike), considerandu-se astfel ca melodia a fost negativ apreciata de catre utilizator. Melodiile ascultate sunt cele incarcate automat in playlist. Dupa ce o melodie se termina, incepe automat difuzarea urmatoarei melodii adaugate in playlist. Daca nu mai exista melodii pe care sistemul sa le descarce, melodiile asculate deja vor fi repetate.
Postcondiții: Redarea melodiilor din playlist
Scenariu de succes:

1.1 Utilizatorul efectueaza operatia de play (redare melodie)

1.2 Utilizatorul efectueaza operatia de stop (oprire melodie)

1.3 Utilizatorul efectueaza operatia de skip (difuzarea urmatoarei melodii)

2.1 Sistemul incepe difuzarea unei noi melodii aleasa din playlist

2.2 Sistemul opreste melodia

2.3 Sistemul trece la difuzarea urmatoarei melodii din algoritm, aleasa aleator din playlist

3.3 Sistemul actualizeaza DHT si structurile de date locale care se ocupa cu gestionarea sistemului de recomandare prin cresterea voturilor negative date melodiei ascultate

Secvențe alternative:
A1) in momentul pornirii executiei unei melodii (butonul Play), utilizatorul nu are la dispozitie niciun playlist din care poate asculta vreo melodie: intervine la *pasul 2.1*

3.1 Sistemul informeaza userul ca nu are generat niciun playlist

4.1 Sistemul trimite utilizatorul la use case-ul UC 3, de creare a playlistului
Scenariul se intoarce la *pasul 5.*

A2) Sistemul intampina probleme in actualizarea valorilor din tabelul DHT a retelei P2P (exemplu exceptie: timeout): intervine la *pasul 3.3*

5. Sistemul ignora lipsa unui rezultat favorabil al operatiei

A cincea cerință funcțională este ilustrată în figura 4.4 și prezentată în următorul caz de utilizare.

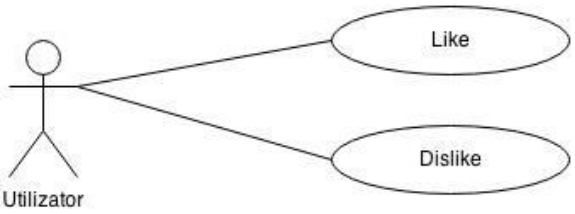


Figura 4.4 Diagrama use case pentru CF 5

UC 5. Titlu: *Notarea melodiei prin apreciere pozitivă sau negativă*
Rezumat: Acest caz de utilizare permite unui utilizator sa aprecieze melodiile pe care le asculta, pozitiv sau negativ, pentru a putea primi melodii de la sistemul de recomandare dupa preferintele sale.
Actori: Utilizatorul (primar)
Precondiții: Utilizatorul trebuie sa fie conectat la o retea si aiba deja un playlist pe care sa il poata asculta (UC 4). Pentru a nota o melodie, aceasta trebuie sa fie in faza de redare (play – ascultare).
Descriere: Utilizatorii pot aprecia pozitiv sau negativ melodiile prin like-uri sau dislike-uri. De asemenea, in momentul in care utilizatorul sare peste o melodie (skip), se considera ca acestuia nu ii place piesa respectiva si va primi un vot negativ (dislike). Aceste notari sunt folosite pentru antrenarea sistemului de recomandare si stabilirea preferintelor utilizatorilor.
Postcondiții: Notarea melodiilor dupa gusturile utilizatorului, pentru antrenarea sistemului de recomandare si primirea unor melodii care sa fie pe gustul sau.
Scenariu de succes:

1.1 Userul efectueaza operatia de like

1.2 Userul efectueaza operatia de dislike

1.3 Userul efectueaza operatia de skip

2.1 Sistemul actualizeaza DHT si structurile de date locale care se ocupa cu gestionarea sistemului de recomandare prin cresterea voturilor pozitive date melodiei ascultate

2.2 Sistemul actualizeaza DHT si structurile de date locale care se ocupa cu gestionarea sistemului de recomandare prin cresterea voturilor negative date melodiei ascultate

2.3 Sistemul trece la difuzarea urmatoarei melodii din playlist

3.3 Sistemul actualizeaza DHT si structurile de date locale care se ocupa cu gestionarea sistemului de recomandare prin cresterea voturilor negative date melodiei ascultate

Secvenţe alternative:
A1) Sistemul intampina probleme in actualizarea valorilor din tabelul DHT a retelei P2P (exemplu exceptie: timeout): intervine la *pasul 2.1, 2.2, 3.3*

4. Sistemul ignora lipsa unui rezultat favorabil al operatiei

Ultima cerință funcțională este descrisă în cazul de utilizare cu numărul 6 și este ilustrată în următoare diagramă:

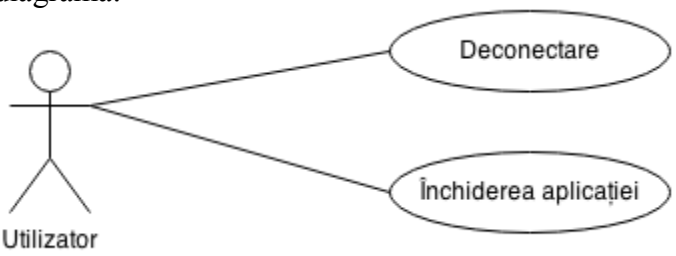


Figura 4.5 Diagrama use case pentru CF 6

UC 6. Titlu: *Deconectarea de la rețea*
Rezumat: Acest caz de utilizare permite unui utilizator al acestei aplicatii sa se deconecteze de la retea, pentru a se putea conecta eventual la o alta retea P2P sau pentru a-si opri aplicatia
Actori: Utilizatorul (primar)
Precondiții: Utilizatorul sa fie logat deja la o retea
Descriere: Daca un utilizator doreste sa se conecteze la o alta retea, inainte acesta e nevoit sa se deconecteze. In momentul deconectarii, memoria de pe disc a aplicatiei este eliberata: melodiile incarcate de pe retea sunt sterse, sistemul de recomandare este reinitializat. Asemenea operatii sunt executate si in momentul in care utilizatorul inchide aplicatia: acesta este deconectat automat de la retea, iar memoria este eliberata.
Postcondiții: Deconectarea de la retea si eliberarea memoriei
Scenariu de succes:

1.1 Utilizatorul apasa butonul de deconectare
1.2 Utilizatorul inchide aplicatia
2. Sistemul deconecteaza userul de la retea
3. Sistemul sterge melodiile descarcate din folderul corespunzator
4.2 Sistemul reinitializeaza sistemul de recomandare – sterge profilul facut utilizatorului

Secvenţe alternative:
2A1) Sistemul intampina probleme in stergerea nodului din tabelul DHT a retelei P2P (exemplu exceptie: timeout): intervine la *pasul 2*

7.1 Sistemul informeaza userul de esuarea deconectarii
Scenariul se intoarce la *pasul 1.1*.
7.2 Sistemul ignora lipsa unui rezultat favorabil al operatiei

3A1) Sistemul nu gaseste nicio melodie descarcata: intervine la *pasul 3*

8. Sistemul nu reactioneaza.
Scenariul se intoarce la *pasul 4.*

3A2) Sistemul nu reuseste sa stearga toate melodiile: intervine la *pasul 3*

9. Sistemul informeaza utilizatorul ca nu a reusit sa stearga toate melodiile din folder si il indeamna pe acesta sa o faca manual
Scenariul se intoarce la *pasul 4.*

4.3. Analiza generală a sistemului

Sistemul de radio online bazat pe o rețea peer-to-peer cu sistem de recomandare integrat este similar unei aplicații de partajare de fișiere, sau mai exact, de muzică (Figura 4.6): utilizatorii au posibilitatea să distribuie sau să descarce muzică din rețea. Sistemul de recomandare oferă posibilitatea creării automate a unui playlist de melodii pentru un utilizator, conform preferințelor sale.

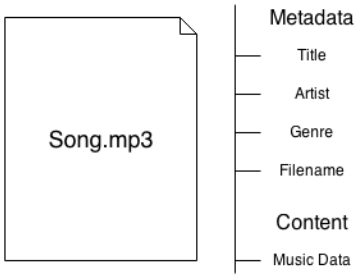


Figura 4.6 Structura unui melodii

O melodie este alcătuită dintr-o structură de metadata, unde sunt ținute detaliile melodiei (titlul, artistul, genul melodiei și numele fișierului) și conținutul propriu-zis al unei melodii.

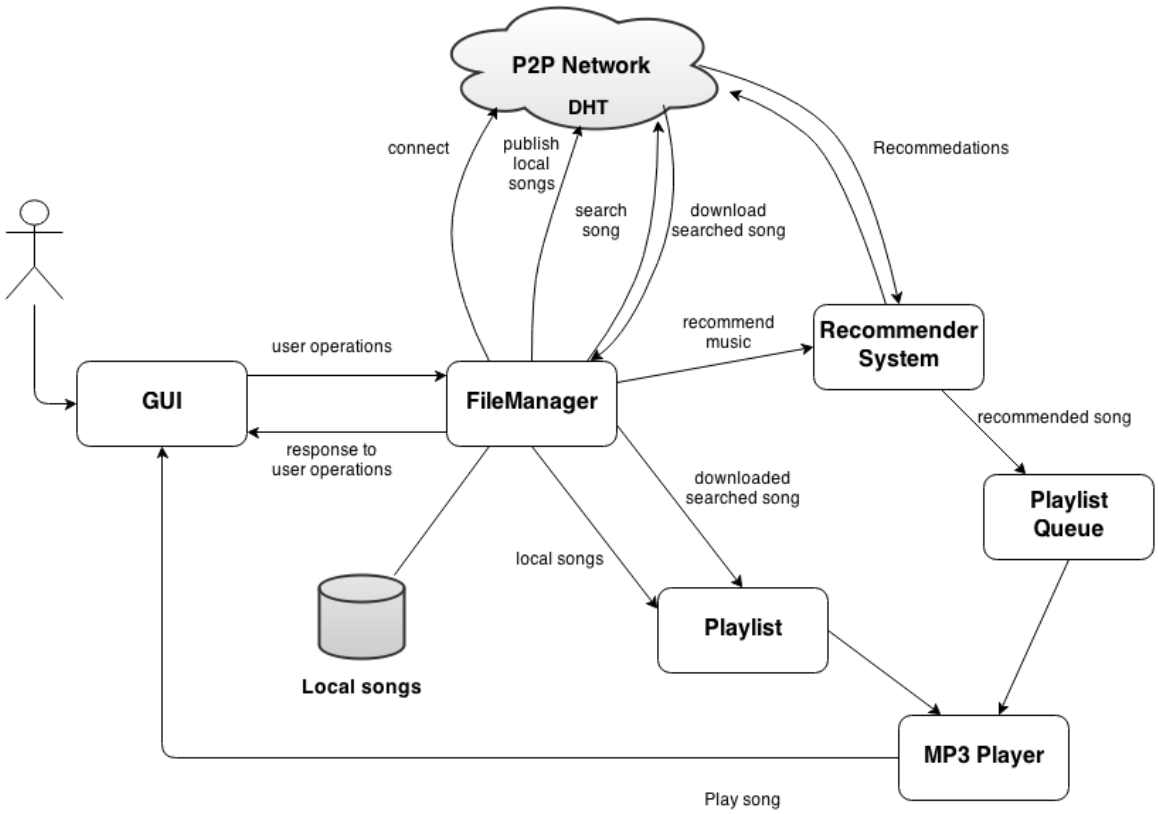


Figura 4.7 Arhitectura conceptuală a sistemului de radio online

Arhitectura conceptuală a sistemului este ilustrată în figura 4.7. Aceasta se bazează pe câteva componente de bază ale sistemului, descrise în acest subcapitol: infrastructura bazată pe rețeaua peer-to-peer și sistemul de recomandare.

Prin intermediul interfeței grafice, utilizatorul are acces la toate funcționalitățile de baza ale aplicației: se conectează la rețea, își publică melodiile de pe stația proprie (melodiile locale), poate căuta și descărca o melodie după cuvinte cheie și poate primi un playlist de melodii descărcate în urma indicațiilor primite de la sistemul de recomandare. Aplicația funcționează ca un MP3 Player, astfel că utilizatorul poate asculta atât melodiile de pe propriul calculator (melodiile locale din Playlist), cat și melodiile descărcate pe parcurs (melodiile recomandate și adăugate în "coada de melodii" Playlist Queue).

4.3.1. Rețeaua P2P

Sistemul de fată este organizat după modelul Peer-to-peer, unde utilizatorii își împart sarcinile pentru buna funcționare a sistemului. Fiecare utilizator este la rândul său producător sau consumator de resurse, este responsabil de distribuirea propriilor melodii în rețea și are dreptul la descărcarea oricăror altor melodii ale utilizatorilor din sistem.

Pentru ca sistemul să funcționeze și utilizatorii să comunice eficient între ei, este nevoie de identificarea și localizarea peer-ilor din rețea. Asemenea tutorialului [10], pornind de la o rețea P2P dezorganizată din punct de vedere logic (Figura 4.8 a), se încearcă găsirea unei metode de a organiza nodurile alocându-le fiecărora un identificator unic global în momentul conectării la o rețea. (Figura 4.8 b) Prin intermediul acestui identificator, un nod poate fi localizat de către oricare alt nod din rețea. Nodurile din rețea creaza un „inel virtual", ordonate după identificatorii lor. Localizarea virtuală diferă de localizarea geografică a peer-ilor, astfel că două noduri, deși situate la distanța mare din punct de vedere geografic, virtual ele pot fi vecine în acest inel. (Figura 4.8 c)

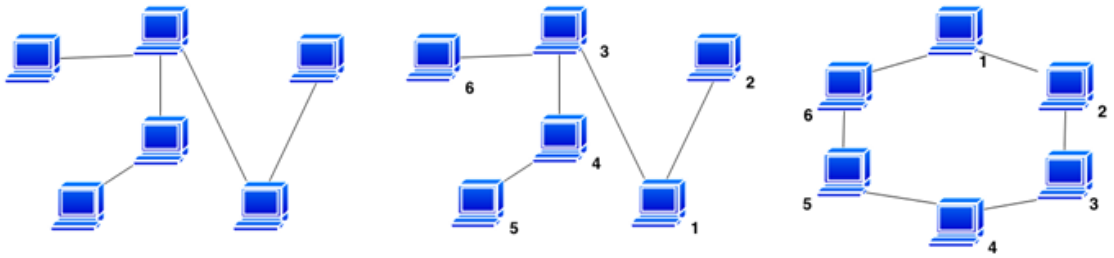


Figura 4.8 a) Rețea P2P dezorganizată, b) Asignarea identificatilor unici (GUID),
c) "Inelul virtual" al rețelei ordonat după GUID

Pentru a putea găsi celelalte noduri din rețea, fiecare peer deține un tabel de rutare (sau DHT), ce conține referințe către un număr de noduri. O referință către un peer constă într-o tupla formată din identificatorul și adresa ip a acestuia: (adresă IP, port UDP, GUID). Într-o rețea P2P cu milioane de noduri, un peer nu poate deține un tabel complet al tuturor nodurilor din rețea, fapt care ar duce la utilizarea ineficientă a resurselor. În schimb, tabela de rutare a unui peer deține referințe către un subset al nodurilor din rețea. Organizarea tabelelor de rutare ține de algoritmul folosit de către rețea, în cazul de fată

fiind folosit Kademlia. Kademlia calculează distanța dintre două noduri după metrica XOR. [9]

Diagramele ilustrate în figurile 4.9 a și b prezintă tabelele de rutare într-o rețea P2P Kademlia din 16 peer-i, unde identificatorii lor unici au dimensiunea de 4 biți. Avantajul utilizării algoritmului Kademlia față de o rețea P2P bazată pe algoritmul Chord constă în ușurința calculării distanței dintre noduri și în simetria rețelei și a valorilor din tabelele de rutare, care lipsește la Chord: dacă nodul 0 are în tabelul de rutare referință către nodul 8, atunci și nodul 8 va avea la rândul său referință către nodul 0. Această simetrie duce la o căutare mai eficientă a nodurilor din rețea.

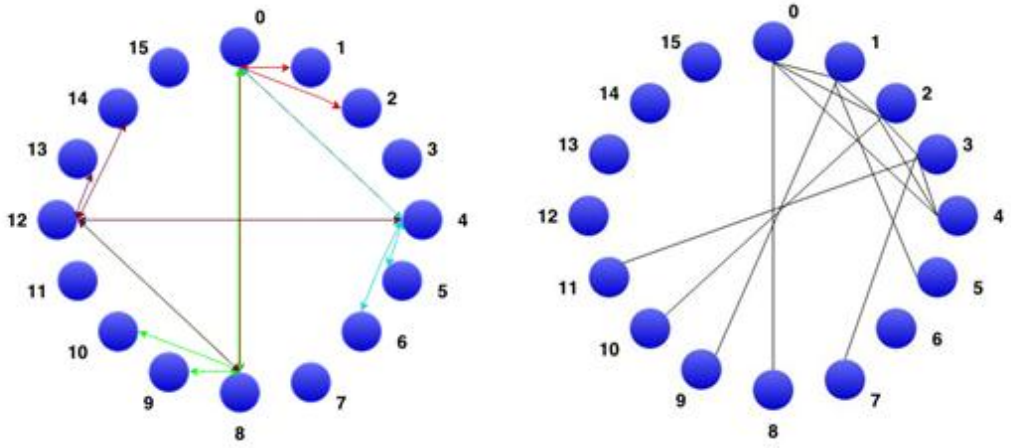


Figura 4.9 a, b) Exemplu de tabel de rutare simetrică dintr-o rețea P2P Kademlia cu 16 noduri

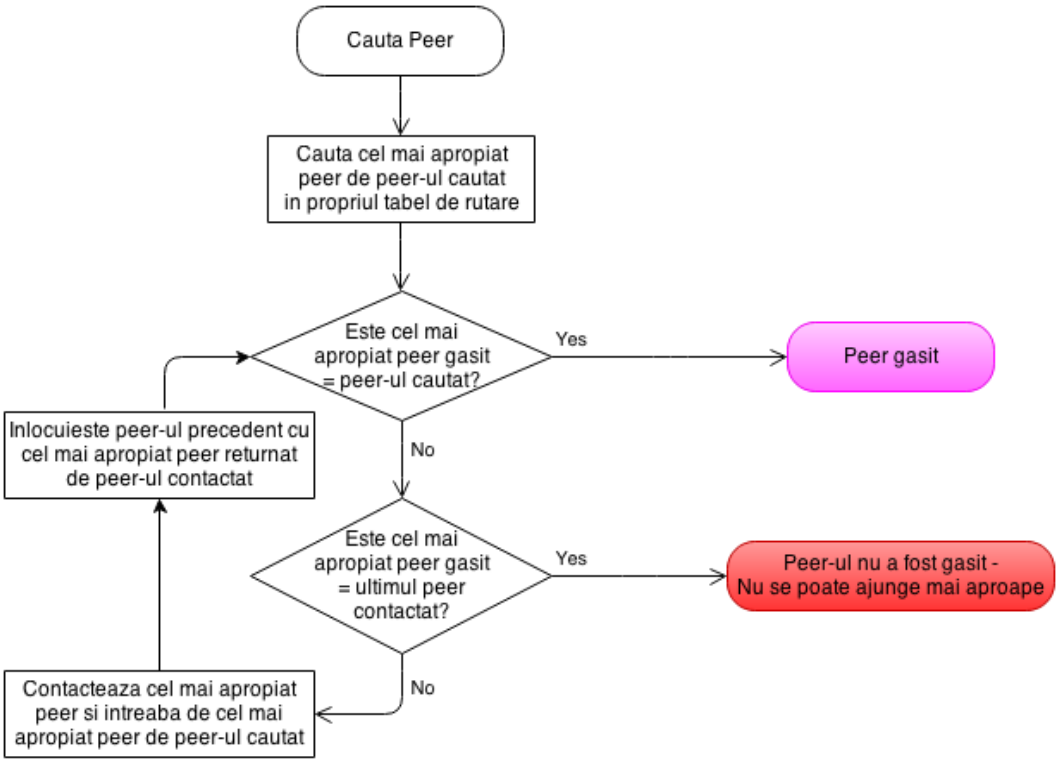


Figura 4.10 Diagrama de flux a precesului de căutare a unui peer

Tabela de rutare poate conține același număr de elemente cat numărul de biți din GUID. În sistemul de față, identificatorii nodurilor au o dimensiune de 160 de biți, ceea ce înseamnă că numărul total de utilizatori conectați poate ajunge până la 2^160.

Găsirea unui nod în rețea se realizează prin intermediul tabelei de rutare (DHT) și respectă algoritmul ilustrat în figura 4.10. Timpul de căutare al unui nod (lookup) într-o rețea Kademlia este de O(log(n)), unde n este numărul total de noduri din rețea.

Operațiile cu structura DHT a rețelei P2P pe care utilizatorul le are la cunoștință vor fi descrise în punctele următoare:

1. *Conectarea la rețea*

Figura 4.11 ilustrează procesul conectării unui utilizator la rețea: acesta trimite unui peer al rețelei, al cărei adresa o cunoaște, un mesaj prin care cere adăugarea lui în rețea (tehnic, adăugarea adresei nodului de la care a venit cererea în tabelul său de rutare). Acesta răspunde și îi trimite valoarea identificatorului unic global (GUID) pe care o va avea de acum încolo în rețea:

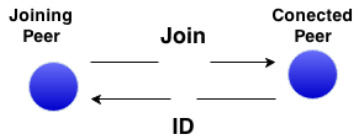


Figura 4.11 Conectarea la un peer din rețea

De asemenea, un utilizator are posibilitatea creării propriei rețele unde el va avea rol de „master peer” și va fi responsabil cu inițializarea rețelei respective. Următorul nod care se va conecta la această rețea trebuie să cunoască adresa IP și portului nodului „master peer”.

2. *Publicarea melodiilor*

Adăugarea unei melodii în rețea reprezintă un aspect important al acestui sistem. Fișierele trebuie să fie disponibile celorlalți utilizatori pentru a le putea accesa și asculta pe stația lor. Astfel, după conectare, utilizatorul alege melodiile locale pe care dorește să le facă publice tuturor utilizatorilor din rețea.

În momentul publicării unei melodii, se construiește un index pentru fișierul respectiv în DHT-ul rețelei (Distributed Hash Table). Indexul este construit cu metadata melodiei, unde metadata este informația conținută de fișierul audio folosită la descrierea anumitor atribute ale conținutului, cum ar fi titlul, genul muzical sau numele artistului. Se creaza astfel 4 domenii, corespunzând următoarelor atribute: titlu, artist, gen și numele fișierului (filename). Metadata unei melodii va fi împărțită în cuvinte cheie ce vor fi publicate în DHT conform domeniului căruia îi corespunde. (put(domain, key, value)). Astfel, în DHT va fi adăugat hash-ul melodiei împreună cu metadata. Ca valoare, hash-ul reprezintă un obiect ce stochează o referință spre locația melodiei din sistemul local, în urma aplicării unei funcții de hash-ing pe 160 biți.

3. *Căutarea unei melodii după un cuvânt cheie*

În sistem se cunoaște pe calculatorul cărui nod se găsește o melodie (referință spre locația acesteia), ce fișiere corespund cuvintelor cheie și cine le deține, datorită structurii DHT a rețelei peer-to-peer.

Utilizatorul va avea posibilitatea căutării oricărei melodii publicate în rețea, în urma introducerii unor cuvinte cheie. Căutarea se va face după atributele melodiei din metadata: titlu, artist, gen muzical sau numele fișierului. Utilizatorul poate primi 0 sau mai multe rezultate, dintre care acesta poate alege melodia pe care dorește să o descarce local si să o asculte.

4. *Realizarea recomandărilor*

Infrastructura bazată pe peer-to-peer constituie un element important în realizarea sistemului de recomandare, datorită descentralizării peer-ilor. Acest fapt duce la construirea unui sistem de recomandare distribuit, unde fiecare nod al rețelei este responsabil de gestionarea propriilor recomandări.

În urma căutării unei melodii în rețea de către utilizator, sistemul va construi un playlist bazat pe recomandări pornind de la melodia căutată și va funcționa ca un radio normal, unde melodiile vor fi difuzate automat din playlist-ul creat. Melodiile vor fi descărcate direct de pe calculatoarele celorlalți utilizatori, asemenea unei rețele peer-to-peer convenționale.

Următorul subcapitol va evidenția modul de funcționare al sistemului de recomandare, atât la nivel de structură a aplicației, cat și la nivel de infrastructură și comunicare între peer-i.

4.3.2. Sistemul de recomandare

Sistemul de recomandare vine în ajutorul utilizatorului, oferindu-i acestuia un set de melodii cat mai apropiat de interesele și gusturile sale. Folosirea unui sistem de recomandare distribuit impune anumite limite în comparație cu un sistem centralizat, ce țin de lipsa de cunoaștere a tuturor utilizatorilor și melodiilor. Prin construirea unui sistem de recomandare hibrid, împreună cu structura DHT a infrastucturii, se urmărește obținerea unor recomandări cat mai bune pentru utilizator și minimizarea cat mai mult a dezavantajelor care vin cu un asemenea sistem, cum ar fi timpul de creare al profilului muzical al unui utilizator sau de notare (rating) a melodiilor din rețea.

Sistemul de recomandare propus combină metode de filtrare colaborativa a utilizatorilor sau a melodiilor din rețea cu metode bazate pe conținut, în momentul recomandării după un anumit gen muzical. Va fi acoperită atât starea globală a întregului sistem, cum ar fi ținerea evidenței celor mai populare și apreciate melodii din rețea, dar și urmărirea stărilor individuale a membrilor, prin realizarea de profile muzicale personale și compararea acestora cu profilele celorlalți utilizatori din sistem.

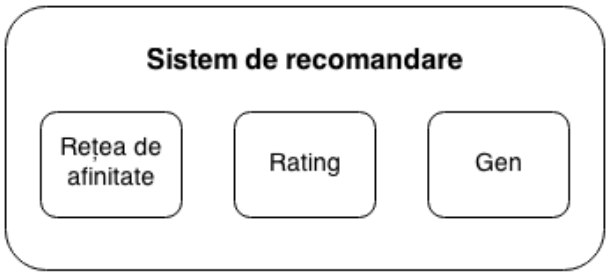


Figura 4.12 Sistemul de recomandare propus

Sistemul este impărțit în 3 componente (Figura 4.11), descrise în punctele următoare:

1. *Rețea de afinitate*

Sistemul de recomandare bazat pe afinitate presupune că între doi utilizatori care distribuie un număr de melodii cu proprietăți asemănătoare există probabilitate destul mare ca aceștia să aibă gusturi apropiate, astfel că un utilizator poate fi interesat de melodiile celuilalt. (Subcapitolul 3.2.4).

O rețea de afinitate se creaza prin calcularea unei funcții de afinitate între fiecare pereche de utilizatori. În sistemul propus, afinitatea între doi utilizatori presupune compararea melodiilor distribuite de fiecare dintre ei, precum în articolul [3]. Similaritatea melodiilor este inspirată de metoda de numărare Borda, folosită în metodele de alegere a liderului (leader election): fiecare candidat este votat în ordinea preferințelor: primul din cei n candidați primește n-1 puncte, al doilea candidat primește n-2, și se continuă până la ultimul candidat care primește n – n = 0 puncte.

Asociată sistemului de fată, metoda Borda returnează următoarele valori în momentul comparării a doua melodii la nivel de metadata:

- sunt identice ⇒ 3 puncte,
- același gen și artist ⇒ 2 puncte,
- același gen ⇒ 1 punct,
- nicio asemanare ⇒ 0 puncte.

Sistemul de recomandare bazat pe afinitățile dintre utilizatori propus urmărește calcularea tuturor afinităților și ordonarea acestor valori pentru a decide care sunt cei mai apropiați membrii din rețea din punct de vedere al preferințelor.Pentru fiecare doi utilizatori din sistem u_i și u_j se va calcula funcția de afinitate astfel: fiecare melodie distribuită de utilizatorul u_i este comparată cu fiecare melodie distribuită de utilizatorul u_j, după metoda Borda definită mai sus. Funcția de afinitate va fi obținută din calcularea sumei valorilor returnate de funcțiile Borda pentru fiecare comparație și apoi normalizată prin împărțirea la numărul total de melodii ale utilizatorului u_j. Normalizarea este necesară pentru a obține valori din același interval de date. Formula acestei funcții este definită în ecuația 4.1

$$Aff(u_i, u_j) = \frac{\sum_{s_i \in S_i} \sum_{s_j \in S_j} m(s_i, s_j)}{|S_j|}$$

Ecuație 4.1 Funcția de afinitate dintre doi utilizatori

unde $m(s_i, s_j)$ este funcția de similaritate dintre două melodii, determinată prin metoda Borda definită mai sus.

Recomandarea bazată pe rețeaua de afinitate constă în găsirea celor mai apropiați utilizatori din punct de vedere al gusturilor și preferințelor și descărcarea melodiilor de pe calculatoarele acestora.

2. *Rating-ul melodiilor*

Utilizatorul are posibilitatea notării unei melodii, printr-un vot de apreciere pozitivă sau negativă (like sau dislike). În momentul unei astfel de operații de votare, sunt notificați și ceilalți utilizatori din rețea cu privire la melodia votată, pentru a-și putea

actualiza tabela melodiilor apreciate. De asemenea, în momentul în care un utilizator dorește să treacă la următoarea melodie din playlist („Skip”), se consideră că acestuia nu i-a plăcut melodia respectivă, motiv pentru care melodia va primi un vot negativ.

În urma numărului de voturi pozitive sau negative se va calcula ratingul fiecărei melodii după următoarea formulă:

$$Rating = \frac{(l-d)}{(l+d)} \in [-1,1]$$

Ecuație 4.2 Rating-ul unei melodii, unde l = numărul de like-uri, iar d = numărul de dislike-uri.

Se crează o stare globală a melodiilor din întregul sistem, unde se încearcă ținerea evidenței a celor mai populare și apreciate melodii ce vor fi recomandate pe urmă utilizatorilor.

3. *Recomandarea după genurile muzicale*

Profilul utilizatorului este creat în funcție de propriile melodii distribuite în rețea și de voturile acestuia acordate melodiilor descărcate. Pe baza acestui profil se realizează o clasificare a genurilor muzicale preferate de utilizator. Acesta va primi ca recomandări un playlist de melodii conform genurilor sale preferate.

Toate aceste componente pot fi combinate sau folosite individual, la alegerea utilizatorului, pentru a obține cele mai bune rezultate.

4.4. Tehnologii utilizate

Sistemul de radio online bazat pe rețea peer-to-peer cu sistem de recomandare integrat este o aplicație ce poate rula pe sistemele de operare Windows. Sistemul este dezvoltat în mediul de programare Eclipse Kepler, iar limbajul de programare folosit este Java, versiunea 7 (Java SE Development Kit 7). Bibliotecile folosite sunt open source sau publicate în GPL.

Pentru realizarea infrastructurii de peer-to-peer a fost aleasă biblioteca TomP2P [13], pentru partea de player a aplicației a fost ales Javazoom Jlayer MP3 Library [5], iar pentru determinarea atributelor unei melodii (metadata) s-a folosit JAudioTagger [4]. Aceste tehnologii vor fi descrise mai departe, argumentând motivele pentru care au fost alese în locul altor biblioteci similare.

4.4.1. TomP2P

Un aspect important al realizării acestui proiect este alegerea bibliotecii potrivite pentru realizarea infrastucturii de rețea peer-to-peer, care să se mapeze pe cerințele sistemului. Biblioteca trebuie să suporte toate funcționalitățile rețelei peer-to-peer și funcțiile tabelelor distribuite de indici unici (DHT).

În urma unui proces de identificare a soluțiilor open source care să ofere funcționalități similare utile cerințelor sistemului, au fost găsite următoarele posibile biblioteci: OpenKad, JDHT, Bamboo-DHT, Pastry, OpenChord sau TomP2P.

JDHT[4] este un tip de date Java ce construiește un DHT și care implementează interfața java.util.Map. Această bibliotecă, ca urmare a serviciilor pe care le oferă, este considerată prea simplă pentru cerințele proiectului actual.

Bibliotecile Bamboo-DHT[5] și Pastry[6] presupun realizarea unei rețele peer-to-peer bazată pe algoritmul Pastry, iar OpenChord[7] propune ideea folosirii unei rețele Chord, ceea ce vine în contradicție cu ideea folosirii algoritmului Kademlia pe care îl propune sistemul de fată.

OpenKad[8] și TomP2P folosesc tabele DHT bazate pe Kademlia, cu mențiunea că OpenKad pune mai mult accent pe rutarea pe bază de cheie și pe găsirea celui mai apropiat peer din rețea ce deține o anumită dată, în timp ce TomP2P este un DHT avansat ce poate stoca mai multe valori pentru o cheie. În urma acestei analize a posibilelor biblioteci, a fost aleasă biblioteca TomP2P pentru realizarea infrastucturii sistemului de radio online.

TomP2P [13] este o bibliotecă software ce are ca scop implementarea unei structuri DHT în Java 6, care la nivel de comunicare folosește Java NIO (non-blocking IO) pentru a trata operațiile concurente. În alte cuvinte, TomP2P reprezintă un DHT extins, care respectă conceptul de tabelă distribuită de indicatori unici: prin intermediul funcțiilor de put(key, value) / get(key) sunt stocate sau aduse valorile din tabela corespunzătoare unei chei. Extinderea constă în adăugarea unor noi operații la nivel de DHT: put(key1,key2,value) / add(key, value). Datorită acestor operații, este permisă stocarea de valori multiple pentru o cheie: put(location_key, content_key, value). TomP2P suportă domeniile (put(key, domain, value) → get(key,domain)) și evită coliziunile pentru aceleași chei.

Prima versiune a bibliotecii TomP2P a fost creată în anul 2004 de către Thomas Bocek. Acesta l-a folosit la proiectul tezei sale de masterat, de DNS distribuit. Prima versiune folosea operații sincrone de intrare/ieșire (blocking IO) și un thread / socket. Versiunile ulterioare au încercat trecerea la operații asincrone de intrare-ieșire (non-blocking IO) prin folosirea a diverse framework-uri: Apache Mina în versiunea 2, respectiv Netty în versiunile următoare. Sistemul de radio online folosește versiunea 4.2 a bibliotecii.

TomP2P respectă cerințele unei structuri DHT în Kademlia (Subcapitolul 3.1.4): fiecare peer are un identificator unic de 160 biți, distanța dintre noduri este calculată prin metrica XOR, cheile sunt plasate pe nodurile ale căror identificatori sunt cel mai aproape de cheile respective. Căutarea unei chei în DHT se realizează iterativ, trecând din peer în peer și verificând mereu cel mai apropiat peer din rețea (calculat prin metrică XOR)) . Timpul de căutare al unei chei poate ajunge până la O(log n), unde n reprezintă numărul utilizatorilor din sistem.

Un concept fundamental al acestei biblioteci este folosirea de către operațiile distribuite a obiectelor de tip Future: se ține evidența evenimentelor viitoare, în timp ce

[4] JDHT: Java Distributed Hash Table. Retrieved from Distributed K-ary System (DKS): http://dks.sics.se/jdht/

[5] The Bamboo DHT. Retrieved from The Bamboo DHT: http://bamboo-dht.org/

[6] Pastry. Retrieved from Pastry: http://www.freepastry.org/

[7] Open Chord. Retrieved from Open Chord: http://open-chord.sourceforge.net/

[8] OpenKad. Retrieved from Google code: https://code.google.com/p/openkad/

programul normal continuă să ruleze fără a fi perturbat de operațiile desfășurate de DHT-ul rețelei.

TomP2P permite conectarea și transferul de date între peer-i după NAT, prin folosirea protocoalelor UPNP sau NAT-PMP și a metodei de port-forwarding. Nodul trimite un request în afară pentru a cere adresa IP văzută de ceilalți membrii ai rețelei. Dacă ceilalți peer-i nu văd aceeași adresă ca a nodului, înseamnă că nodul respectiv se află în spatele NAT.

În ceea ce privește securitatea, asupra identificatorilor unici, cheilor sau mesajelor transmise între noduri se aplică o semnătură digitală folosind funcția de hash-ing SHA1 cu DSA (Digital Signature Algorithm).

4.4.2. Javazoom JLayer

Aplicația nu are nevoie de funcții complexe de redare a melodiilor: va funcționa ca un radio, cu posibilitatea de oprire a acestuia (stop) sau de trecere la următoarea melodie din playlist (skip). Nu va necesită funcționalitatea de pauză și repornire a melodiei, de fast forward sau fast backward. Pentru aceasta este nevoie doar de o bibliotecă simplă care să suporte decodarea fișierelor .mp3 si să asigure difuzarea acelor melodii. Posibile astfel de biblioteci pentru Java sunt Java MediaFramework (JMF)[9] și Javazoom JLayer. Din moment ce JMF este considerat învechit și nu se mai oferă actualizări sau versiuni noi pentru acest framework, biblioteca folosită în sistemul de radio online a fost aleasă Javazoom JLayer, mai precis clasa javazoom.jl.player.advanced.AdvancedPlayer. [5]

Player-ul are o serie de metode specifice, de pornire (play()) sau oprire (stop()) a unei melodii. Clasa AdvancedPlayer vine cu un avantaj fată de Player-ul simplu al aceleiași biblioteci: se poate seta un listener (PlaybackListener) care să trateze evenimentele de oprire sau pornire a unei melodii. Este posibilă astfel și implementarea funcționalității de pauză sau de notificare în momentul terminării unei melodii (utilă în cazul difuzării continue a unui playlist fără intervenția utilizatorului: după se termină o melodie se poate trece automat la redarea următoarei melodii).

4.4.3. JaudioTagger

Metadata unei melodii, care poate fi definită și ca ID3 metadata, este informația conținută de un fișier audio folosită la descrierea anumitor atribute a conținutului audio, cum ar fi titlul, genul melodiei, numele artistului sau albumului etc. Biblioteca aleasă pentru extragerea etichetelor dintr-un fișier audio este JaudioTagger.

JaudioTagger [4] este în continuare o bibliotecă susținută și dezvoltată. Aceasta poate suporta următoarele tipuri de formate: MP3, MP4, Ogg Vorbis, Flac and Wma, și poate avea suport limitat formatele Wav și Real. Pentru formatul MP3, există două tipuri de metadate care sunt folosite în mod regulat: ID3v1 și ID3v2. ID3v stochează informațiile melodiei la finalul fișierului MP3, cu un spațiu de date alocat de 128 octeți. În versiunea a doua, atributele sunt localizate la începutul fișierului și are o capacitate mai mare de stocare, până la 256 octeți. JAudioTagger suportă ambele versiuni și permite conversii între cele două tipuri de metadate.

[9] JMF. Retrieved from Oracle: http://www.oracle.com/technetwork/java/javase/tech/index-jsp-140239.html

În acest capitol s-a realizat o analiză a arhitecturii conceptuale a sistemului prin care au fost determinate cerințele functionale si nefunctionale, împreună cu modul de funcționare ale acestora. De asemenea, au fost alese tehnologiile ce vor sta la baza construirii acestui sistem.

Capitolul 5. Proiectare de Detaliu și Implementare

În acest capitolul este descrisă etapa de implementare a sistemului de radio online bazat pe o rețea peer-to-peer cu sistem de recomandare integrat. Este prezentată arhitectura software a aplicației, împreună cu descrierea fiecărei componente a proiectului.

5.1. Arhitectura sistemului

Având că obiectiv crearea unei aplicații scalabile, flexibile, de complexitate redusă și ușor de întreținut din punct de vedere al dezvoltării software, sistemul de radio online este construit pe o arhitectură pe nivele, ilustrată în figura 5.1. Avantajele folosirii acestui model arhitectural țin de reducerea complexității comunicării dintre diferitele componente ale aplicației, menținerea coeziunii înalte în interiorul unui modul și a cuplajului slab dintre acestea. Arhitectura pe mai multe nivele oferă flexibilitate și mentenabilitate, ca urmare a separării dintre interfața grafică, logica sistemului și infrastructura acestuia.

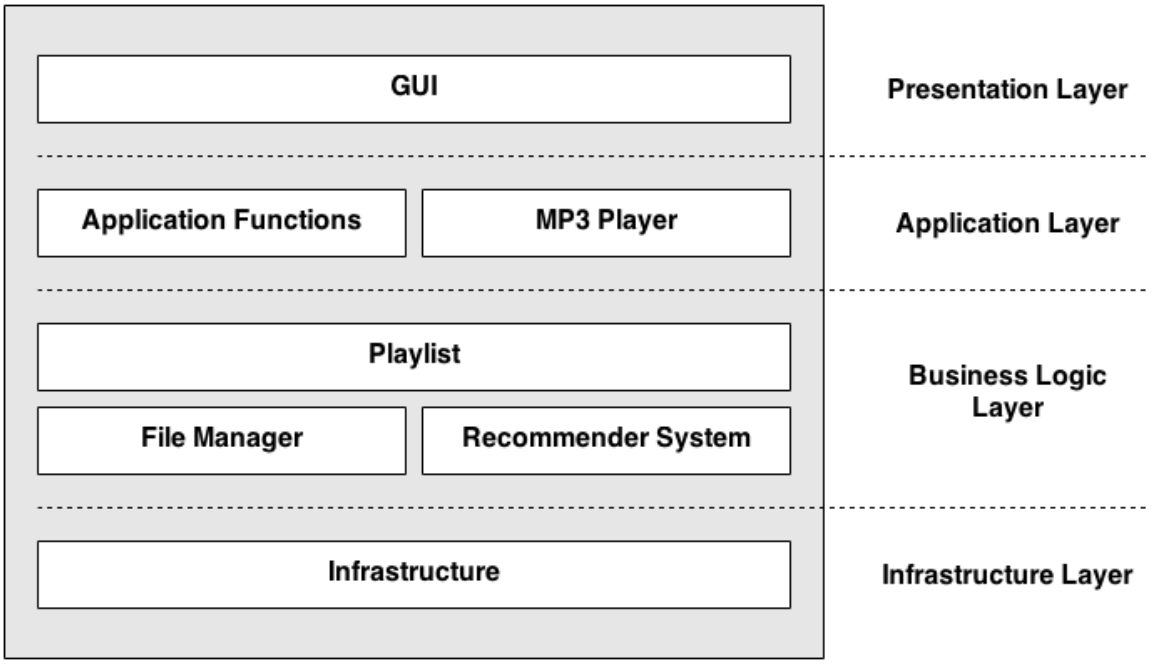


Figura 5.1 Arhitectura software a sistemului de radio online

Arhitectura sistemului este construită pe 4 nivele: nivelul interfeței grafice (Presentation Layer), nivelul aplicației (Application Layer), nivelul logic (Business Logic Layer) și nivelul infrastructurii (Infrastructure Layer).

Prin definirea acestor nivele se realizează o împărțire a responsabilităților pe module, unde fiecare modul va avea propriile sarcini. Legătură dintre componentele aplicației va fi efectuată prin folosirea pattern-ului Facade.

Figura 5.2 ilustrează diagrama de pachete a sistemului, conform împărțirii pe nivele:

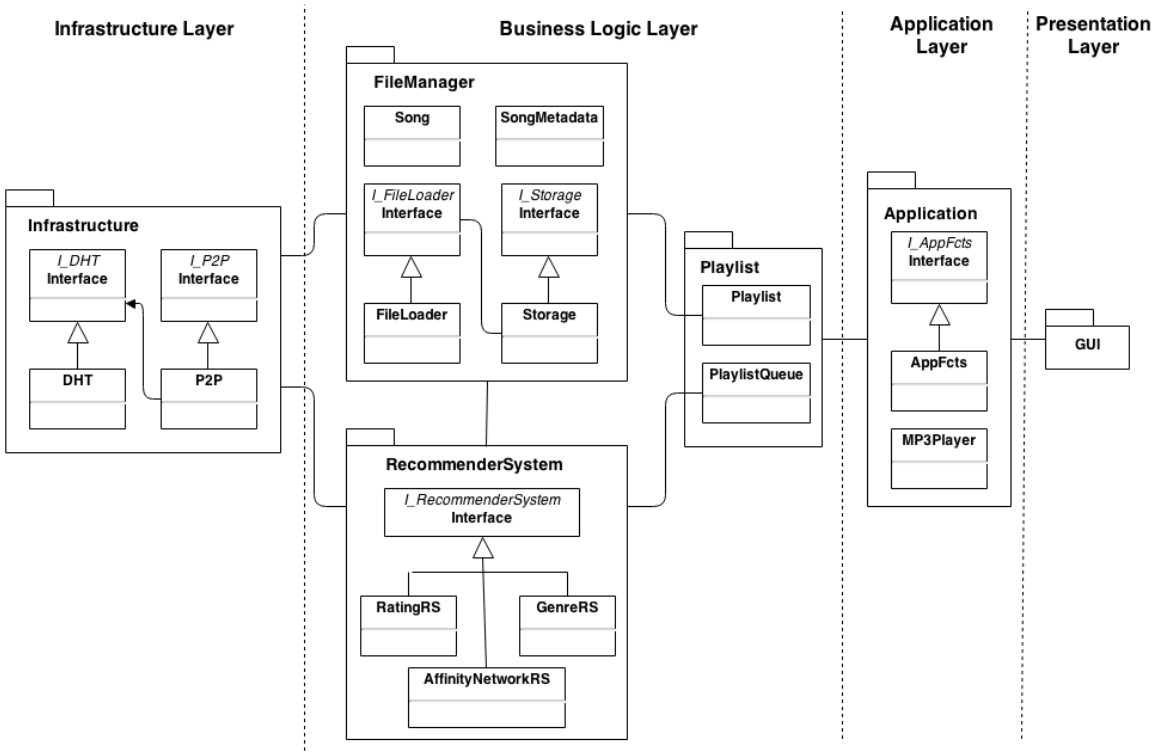


Figura 5.2 Diagrama de pachete

Cele 4 nivele ale arhitecturii, împreună cu pachetele și clasele lor reprezentative, vor fi prezentate pe rând în subcapitolele următoare.

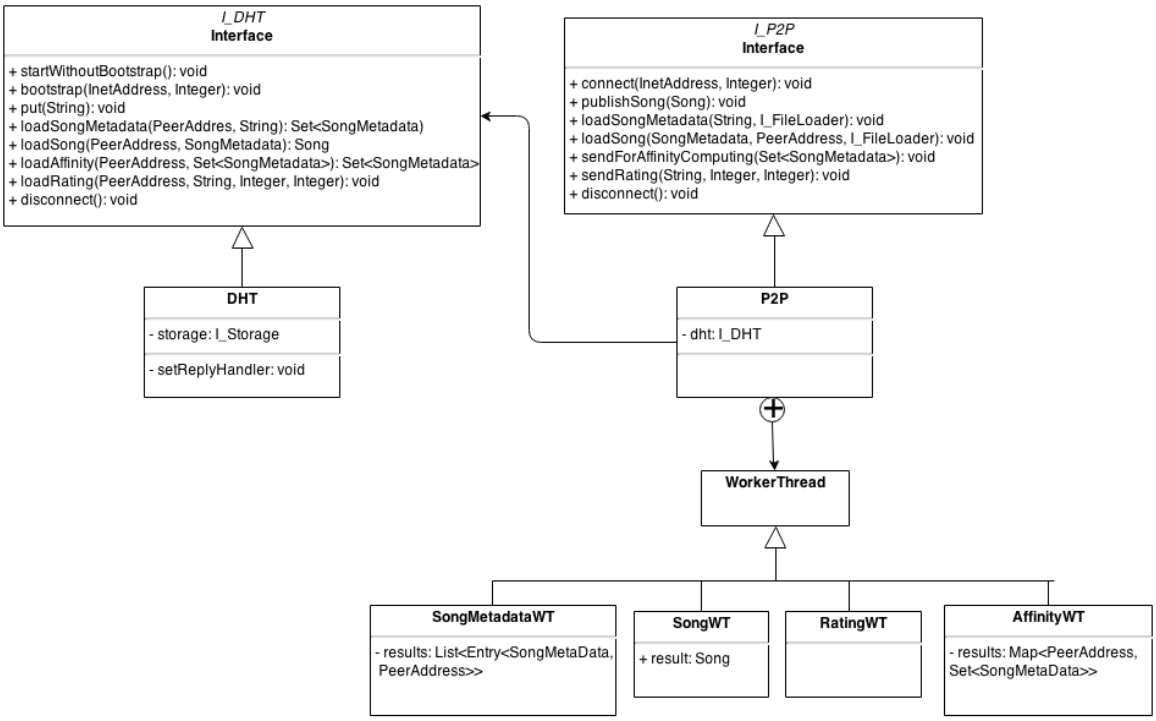
5.2. Nivelul infrastructurii

Această parte a aplicației asigură structura de bază a sistemului, care se ocupă de realizarea tuturor funcțiilor la nivel de rețea, cum ar fi gestiunea operațiilor de intrare/ieșire, conectivitatea la rețea, transferul de fișiere sau streaming-ul de date.

Aceste funcții sunt realizate prin intermediul bibliotecii externe TomP2P, ce vine cu o soluție de creare a unui DHT care să se ocupe de construirea și funcționarea unei rețele de tip peer-to-peer. Această bibliotecă asigură funcțiile necesare implementării acestui sistem de radio online bazat pe o rețea peer-to-peer.

O diagramă a claselor reprezentative ale infrastructurii, împreună cu metodele sugestive ale acestora, este ilustrată în figura 5.3.

Clasa DHT, împreună cu interfața I_DHT, reprezintă o implementare a structurii DHT a unei rețele peer-to-peer. Sunt realizate metode ce stau la baza funcționării întregului sistem, la nivel de rețea: conectarea utilizatorului la o rețea, publicarea, căutarea sau descărcarea unui fișier de la alți utilizatori din rețea. Funcțiile sunt dezvoltate folosind elemente ale bibliotecii TomP2P și sunt explicate în paragrafele următoare.



Figură 5.3 Diagrama de clase la nivel de infrastructură

Pentru conectarea la rețea, din punct de vedere al implementării, este nevoie înainte de crearea unui nou peer cu un ID aleator, asupra căruia se aplică funcția de hashing SHA-1, specifică TomP2P (5.1).

```
InetAddress address = Inet4Address.getLocalHost();                    (5.1)
this.localTomP2PPeer = new PeerMaker(Number160.createHash(System.nanoTime())).
  setPorts(this.localPort).setBindings(new Bindings(address)).makeAndListen();
```

Metoda startWithoutBootstrap() conține acest cod și este responsabilă de crearea unui peer de sine stătător. În momentul în care se dorește crearea unei noi rețele, în care utilizatorul să fie primul membru din rețea, este de ajuns să se apeleze doar această metodă. (UC 1.1, subcapitolul 4.2).

În cazul în care utilizatorul dorește să se conecteze la o rețea deja existentă, după o adresă ip și un port cunoscut (UC 1.2, subcapitolul 4.2), este apelată metoda bootstrap(InetAddress address, Integer port). Această metodă apelează la rândul ei metoda anterioară, care creaza peer-ul local.

Următorul pas este să se descopere dacă peer-ul este accesibil rețelei de internet sau este în spatele unui NAT (5.2). Dacă peer-ul nu poate fi văzut și găsit de către ceilalți membrii ai rețelei, nu se poate realiza conectarea la o rețea P2P. Asta presupune că o astfel de rețea peer-to-peer poate fi creată doar în interiorul unei rețele private sau pe internet, de utilizatori cu adrese publice.

```
FutureDiscover fd = this.localTomP2PPeer.discover().                  (5.2)
            setInetAddress(ipTarget).setPorts(portTarget).start();
```

Conectarea propriu-zisă la rețea se realizează prin intermediul funcției de bootstrap a bibliotecii TomP2P (5.3):

```
FutureBootstrap fb = this.localTomP2PPeer.bootstrap()                    (5.3)
       .setInetAddress(ipTarget).setPorts(portTarget).start();
```

În momentul deconectării de la o rețea, prin apelarea metodei disconnect(), va fi eliberată zona de memorie, iar melodiile descărcate pe parcursul rulării aplicației vor fi șterse.

Publicarea unui obiect în rețea este realizată prin metoda put(String key) (5.4):

```
this.localTomP2PPeer.addTracker(n160key).start().awaitUninterruptibly();  (5.4)
```

unde n160key reprezintă valoarea cheii în urma aplicării funcției de hash-ing pe 160 biți. Fiecare cheie va avea alocat un tracker, care va fi responsabil de găsirea nodului de la care a fost adăugată cheia.

Pentru partajarea de fișiere între peer-i este nevoie de transmitere de mesaje de tip Request/Reply între noduri: de exemplu, un nod trimite o cerere către ceilalți membrii ai rețelei pentru căutarea sau descărcarea unei melodii, iar nodul care deține melodia respectivă îi va răspunde acestuia. Metoda privată setReplyHandler() este responsabilă de primirea cererilor de la alți peer-i, determină tipul de mesaj și se ocupă cu trimiterea răspunsurilor, în cazul în care poate îndeplini cererile acestora. (5.5)

```
this.localTomP2PPeer.setObjectDataReply(new ObjectDataReply() {          (5.5)
       public Object reply(PeerAddress sender, Object request){ ... }
       });
```

Trimiterea uneri cereri se realizează prin secvența de cod (5.6):

```
FutureResponse future = this.localTomP2PPeer                             (5.6)
       .sendDirect(peerAddress).setObject(request).start();
```

Rețeaua permite 4 tipuri de cereri, implementate în următoarele metode: (5.6)

- *loadSongMetadata(PeerAddress peerAddress, String keyword)* trimite o cerere către nodul cu adresa peerAddress căruia îi cere o listă cu metadatele melodiilor găsite după cuvântul cheie keyword. În cazul în care acesta deține astfel de melodii, trimite înapoi o listă cu un număr maxim de 10 metadate. Dacă nu deține nicio melodie potrivită, nu răspunde la cerere.
- *loadSong(PeerAddress peerAddress, SongMetadata metadata*) trimite un mesaj de tip Request către peerAddress, de la care cere melodia corespunzătoare metadatei trimise ca parametru.
- *loadAffinity(PeerAddress peerAddress, Set<SongMetadata> mds)* trimite către nodul cu adresa peerAddress lista cu metadatele melodiilor publicate, pentru ca acesta să calculeze afinitatea dintre ei și să-și actualizeze tabelul de afinități cu noua valoare calculată. La rândul său, răspunde cu lista proprie de metadate.
- *loadRating(PeerAddress peerAddress, String song, int nrLikes, int nrDislikes)* trimite un mesaj către peerAddress cu numele melodiei, numărul de voturi

pozitive, respectiv negative. Nodul își va actualiza astfel tabelul cu voturile melodiilor. Nu trimite înapoi niciun răspuns.

Clasa P2P, care implementează interfața I_P2P, vine să completeze atribuțiile clasei DHT, care se ocupă de realizarea funcțiilor sistemului la nivel comunicare între peer-i prin mesaje de tip Request/Reply. Această clasă creaza o legătură între structura pe care se bazează aplicația, adică DHT-ul rețelei, și clasele de la nivelul superior, care se ocupă de logica aplicației.

Metoda *publishSong(Song song)* este responsabilă de publicarea în rețea a melodiei date ca parametru: desparte metadata melodiei în atribute, iar apoi în cuvinte cheie, pe care le publică în rețea apelând metoda *DHT.put(key).* Împărțirea în cuvinte cheie va facilita căutarea în rețea după o melodie anume.

Clasa P2P conține clasa privată WorkThread (clasă ce extinde clasa Thread), împreună extinderile acesteia: SongWT, SongMetadataWT, AffinityWT și RatingWT. Aceste clase sunt responsabile de crearea a câte unui nou thread pe care să se desfășoare operațiile de cerere și trimitere de mesaje, implementate în clasa DHT și descrise mai sus. Aceste clase sunt instanțiate în metodele corespunzătoare din P2P (exemplu: *loadSongMetadata(String keyword, I_FileLoader fileLoader)* rulează un thread de tip SongMetadataWT prin care apelează metoda *DHT.loadSongMetadata(PeerAddress peerAddress, String keyword)* pentru peer-ii din rețea atâta timp cat nu găsește melodia căutată). Fiind pe thread-uri separate, aceste operații nu blochează aplicația, lăsând utilizatorul să aștepte după răspunsurile celorlalți membrii.

5.3. Nivelul logic

Nivelul logic al aplicației (Business Logic Layer) conține toate funcționalitățile creării playlist-ului de melodii al radio-ului online. Este responsabil atât de gestionarea melodiilor locale, cat și a celor externe, descărcate în urma unui proces de căutare realizat de către utilizator sau pe baza recomandărilor date de sistemul de recomandare. Aceste funcționalități sunt realizate prin apelarea funcțiilor infrastructurii.

Acest nivel este împărțit în 3 pachete: FileManager (Gestionarea fișierelor), RecommenderSystem (Sistem de recomandare) și Playlist. Acestea, împreună cu clasele reprezentative, vor fi descrise mai pe larg în subcapitolele următoare.

5.3.1. Gestionarea fișierelor

În figura 5.4 este ilustrată diagrama de clase a pachetului FileManager, împreună cu clasa Playlist aparținând celui de-al treilea pachet din acest nivel (Playlist).

Această parte a aplicației este responsabilă de gestionarea melodiilor locale: clasa Storage, care implementează interfața I_Storage, citește fișierele în format .mp3 din folderul aplicației și le transformă în obiecte corespunzătoare cerințelor aplicației: Song și SongMetadata. În Storage se stochează astfel melodiile locale, care sunt obiecte de tip Song. Acestea pot fi publicate în rețea prin metoda *makeSongPublic(Song song),* care apelează, la rândul sau, metoda din infrastructură *P2P.publishSong(Song song).*

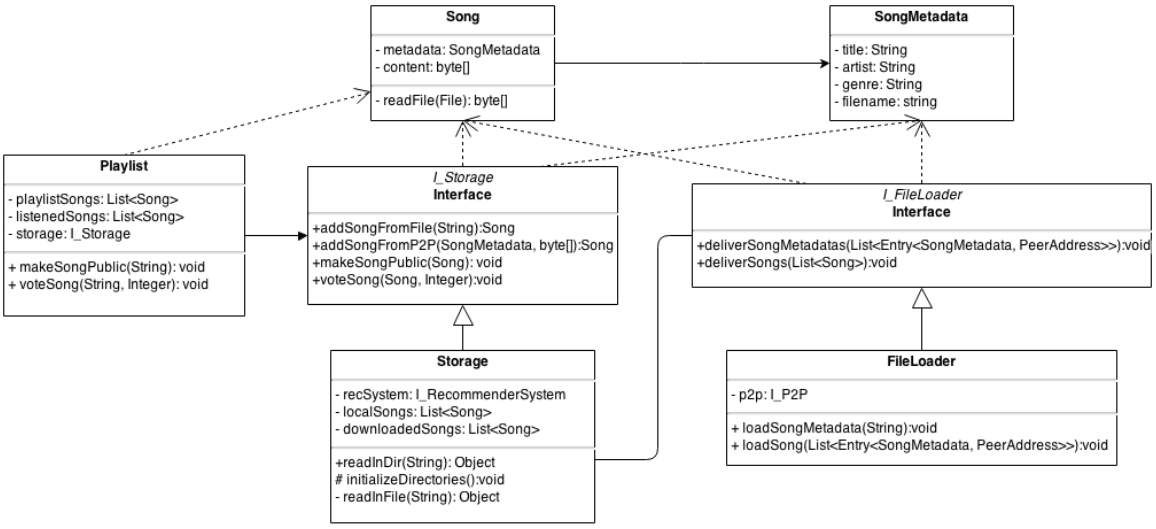


Figura 5.4 Diagrama de clase la nivel logic - pachetul FileManager

Clasele din acest pachet se ocupă de căutarea unei melodii în rețea și descărcarea acesteia. În momentul căutării unei melodii după un cuvânt cheie, se creaza un nou thread de tip FileLoader (care implementează interfața I_FileLoader și extinde clasa Thread) și se apelează *metoda loadSongMetadata(String keyword)*. Execuția thread-ului constă în apelarea metodelor din infrastructură care să caute la celelalte noduri ale rețelei melodii ale căror atribute să coincidă cu cuvântul cheie. În cazul în care ceilalți membrii ai rețelei dețin melodii corespunzătoare, aceștia trimit înapoi doar metadatele melodiei. Astfel, utilizatoarul primește în urma căutării o listă de metadate a melodiilor găsite, împreună cu adresele peer-ilor la care se găsesc acele melodii. Lista de metadate și adrese este trimisă din infrastructură către acest nivel prin apelarea metodei *FileLoader.deliverSongMetadatas (List<Entry<SongMetadata, PeerAddress>>)* de către clasa P2P. Din lista de metadate primită utilizatorul își va putea alege melodia pe care să o descărce si să o asculte.

În cazul descărcării melodiilor ale cărui metadate și adrese ale peer-ilor proprietari sunt cunoscute, este apelată metoda *loadSong(List<Entry<SongMetadata, PeerAddress>>)* din FileLoader, care trimite un apel către clasa P2P. Aceasta răspunde cu lista melodiilor descărcate apelând metoda *deliverSongs(List<Song>).* De asemenea, în momentul descărcării unei melodii de la un alt membru al rețelei, metoda din Storage *addSongFromP2P(SongMetadata metadata, byte[] content)* se ocupă cu transformarea obiectului de tip Song primit de la DHT într-un fișier .mp3 stocat local, pe calculatorul utilizatorului.

Clasa Playlist ține evidența melodiilor stocate local, descărcate sau ascultate.

În momentul votării unei melodii, metoda din Storage voteSong(Song song, Integer like) anunță sistemul de recomandare, pentru ca acesta să-și actualizeze structurile interne în vederea descărcării următoarelor melodii din playlist.

5.3.2. Sistemul de recomandare

Sistemul de recomandare al aplicației se ocupă cu generarea automată a unui playlist de melodii bazat pe preferințele utilizatorului. Preferințele sunt analizate în urma voturilor pozitive și negative acordate și a melodiilor proprii publicate în rețea.

În figura 5.5 este ilustrată diagrama de clase a pachetului RecommenderSystem, împreună cu clasa PlaylistQueue din pachetul Playlist.

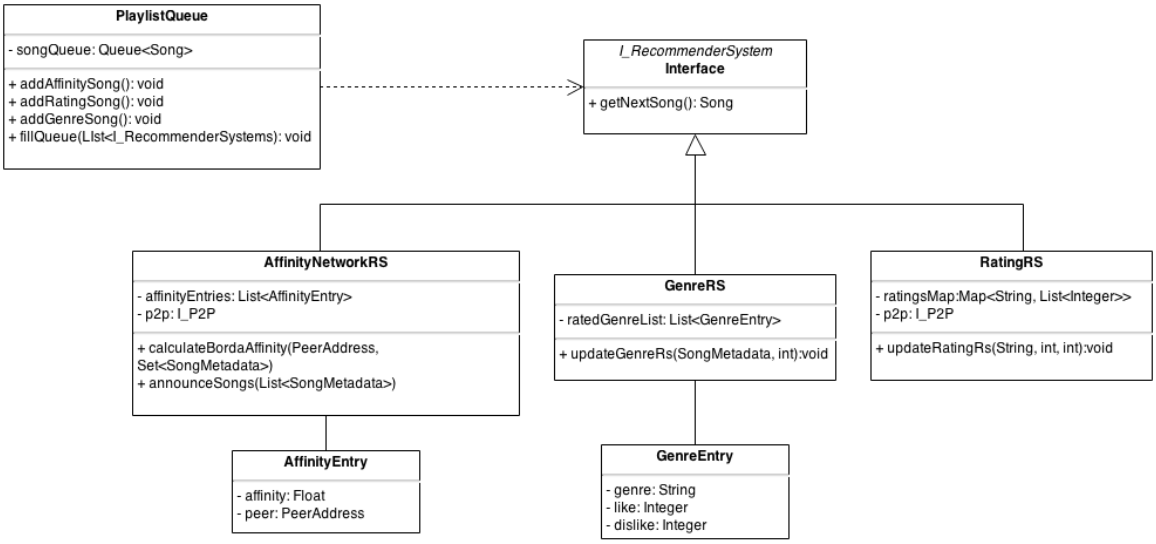


Figura 5.5 Diagrama de clase la nivel logic - pachetul RecommenderSystem

Aplicația propune 3 modalități de creare a recomandărilor, definite în subcapitolul 4.3.2, implementate în clasele AffinityNetworkRS (recomandări bazate pe rețeaua de afinitate), RatingRS (determinarea celor mai populare melodii) și GenreRS (recomandări după genul muzical). Toate aceste clase implementează interfața I_RecommenderSystem și suprascriu metoda de bază a acesteia, prin care se returnează următoarea melodie descărcată și redată utilizatorului în playlist: *getNextSong()*.

Rețeaua de afinitate constă în calcularea afinităților dintre utilizator și ceilalți membrii ai rețelei. În momentul în care acesta își face publice melodiile proprii, sistemul trimite printr-un mesaj de tip Request lista cu metadatele acestor melodii. Ceilalți membrii își calculează afinitatea dintre ei și utilizatorul de la care au primit mesajul și trimit, la rândul lor, metadatele melodiilor proprii distribuite. Utilizatorul, după ce primește răspunsurile, calculează afinitățile și actualizează lista de afinități din clasa AffinityNetworkRS. Lista de afinități este un obiect de tip List<AffinityEntry>, unde clasa AffinityEntry presupune asignarea unei afinități față de un peer anume, determinat prin adresa sa. După ordonarea acestei liste, se va cunoaște care este nodul din rețea cel mai apropiat ca și gusturi de utilizatorul curent. Următoarea melodie descărcată va fi aleasă aleator de pe calculatorul acestuia. Înainte de descărcare, clasa PlaylistQueue verifică dacă nu a mai fost ascultată deja melodia respectivă. În caz afirmativ, este aleasă altă melodie. Dacă au fost descărcate toate melodiile de pe stația aceasta, se trece la următorul peer din lista de afinități.

Partea de rating a sistemului de recomandare contabilizează voturile utilizatorilor. În clasa RatingRS, colecția de tip Map ratingsMap adună toate titlurile melodiei,

împreună cu ratingurile melodiilor respective. Ratingul este calculat în funcție de numărul de like-uri și dislike-uri date de către utilizatori (5.7):

```
rating = (like-dislike) / (like+dislike);
```
(5.7)

Pe lângă votul explicit dat de utilizator (prin apăsarea butoanelor de „Like” sau „Dislike” de pe interfața grafică), este luat în considerare și votul implicit: dacă un utilizator sare peste o melodie („Skip”), se consideră că acestuia nu i-a plăcut melodia respectivă, motiv pentru care va primi un vot negativ.

În momentul în care un utilizator votează pozitiv sau negativ o melodie, sunt notificați toți membrii din rețea printr-un mesaj de tip Request. În mesaj este introdus numele melodiei (de tip String) și numărul de like-uri și dislike-uri acumulate după votul dat de utilizatorul curent (de tip Integer). Fiecare membru, în momentul în care primește un astfel de mesaj, își actualizează colecția de ratinguri cu valorile primite (prin metoda *updateRatingRs(String, int, int)*). Dacă în propria colecție există deja melodia primită prin mesaj, va fi actualizat doar numărul de voturi ale acesteia.

Sistemul bazat pe voturi trimite ca recomandare numele melodiei cu cel mai mare rating din colecție. Înaintea descărcării melodiei cu acel nume, se verifică în Playlist dacă aceasta nu a fost ascultată deja. Dacă melodia a mai fost redată utilizatorului, se continuă cu verificarea următoarei melodii cu ratingul cel mai mare din colecție. Melodia descărcată este adăugată apoi „cozii de melodii” PlaylistQueue.

În cazul recomandării după gen, clasa GenreRS determină preferința utilizatorului pentru anumite genuri. Conține o listă de genuri, împreună cu voturile pozitive și negative corespunzătoare. Lista se actualizează în momentul în care utilizatorul votează o melodie: dacă unui utilizator i-a plăcut o melodie, înseamnă că i-a plăcut și genul muzical al acesteia. Pe de altă parte, dacă utilizatorului i-a displăcut o melodie, și genul acesteia va primi un vot negativ. De asemenea, când un utilizator sare peste o melodie, se consideră că acestuia nu i-a plăcut melodia, și implicit nici genul acesteia, ceea ce duce la acordarea unui vot negativ. După ordonarea listei și găsirea genului muzical preferat, se caută în rețea o melodie de genul respectiv care să fie descărcată și redată în playlistul utilizatorului. La fel ca și în cazul celorlalte două sisteme similare de recomandare, se verifică înainte de descărcare dacă melodia respectivă nu a mai fost difuzată o dată. Altfel, se caută o altă melodie de același gen.

Aceste 3 componente ale sistemului de recomandare pot fi utilizate atât în mod individual, dar și combinate, la alegerea utilizatorului. Pentru a evita întârzierile provocate de căutarea și descărcarea unei melodii, se creaza o coadă de melodii sugerate de sistemul de recomandare și descărcate, care așteaptă să fie difuzate utilizatorului. Numărul de melodii descărcate în avans poate fi între 2 și 5. Astfel, se previne și eșecul apărut în momentul în care un utilizator părăsește rețeaua. Dacă următoarea melodie care trebuia descărcată de către sistemul de recomandare aparține utilizatorului care tocmai a părăsit rețeaua, sistemul nu se blochează, iar după un timp de așteptare în care acesta realizează că melodia sau utilizatorul nu mai pot fi găsite, trece la descărcarea următoarei melodii. Această întârziere nu afectează utilizatorul final, din moment ce melodiile sunt căutate și descărcate în avans.

5.4. Nivelul aplicației

Acest nivel mapează evenimentele produse de componentele interfeței grafice în metode și obiecte înțelese de partea de logică și infrastructură. La acest nivel nu sunt cunoscute detalii de pe nivelele inferioare ale aplicației, dar realizează cerințele provenite de pe interfața grafică prin instanțierea claselor și apelarea metodelor de pe aceste nivele. Figura 5.6 ilustrează diagrama de clase a pachetului Application.

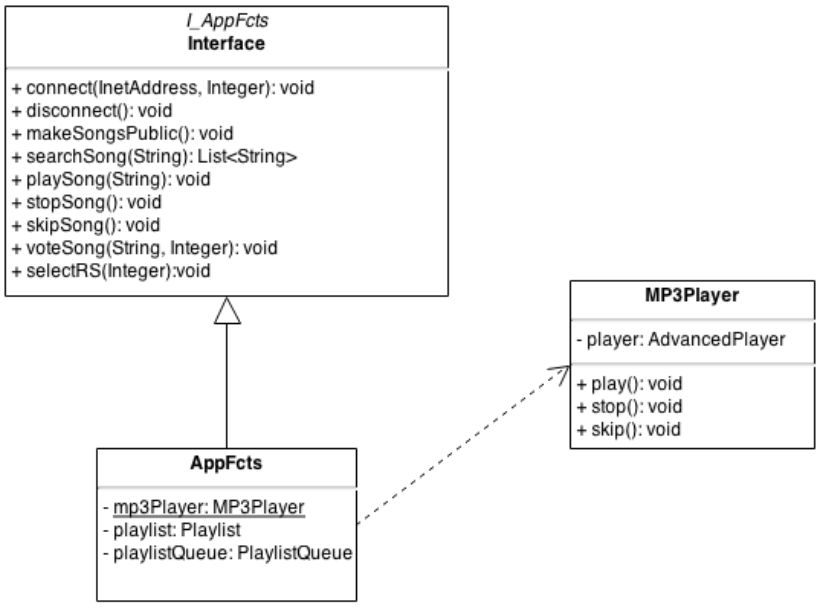


Figura 5.6 Diagrama de clase la nivel de aplicație

Interfața I_AppFcts prezintă funcționalitățile principale ale aplicației, definite și în cazurile de utilizare a sistemului. Clasa AppFcts, care implementează această interfață, este responsabilă de apelarea nivelelor inferioare în momentul realizării unui eveniment în interfața grafică.

Pe lângă rol de mediator între interfața utilizator și logica aplicației, o funcționalitate a acestui nivel este construirea player-ului muzical (clasa MP3Player) și difuzarea fișierelor în format mp3. Clasa oferă metode de pornire a unei melodii (*playSong()*), oprire (*stopSong()*) sau trecere la următoarea melodie din PlaylistQueue (*skipSong()*). În momentul terminării unei melodii, player-ul trece automat la difuzarea următoarei melodii din PlaylistQueue.

Figura 5.7 ilustrează diagrama de secvență a operației de căutare a unei melodii de către un utilizator, pornind de la nivelul aplicației până la DHT-ul rețelei. Funcția *searchSong(String keyword)* al obiectului de tip AppFcts apelează metoda cu același nume din playlist (obiect de tip Playlist, situat la nivelul logic al aplicației). Playlistul cere „depozitului" de melodii (clasa Storage) să găsească melodia potrivită cuvântului cheie dat ca parametru. Prin intermediul obiectului de tip FileLoader, care apelează la rândul sau metodele obiectelor de la nivelul următor, sunt găsite melodiile din rețea care corespund cuvântului căutat, împreună cu adresele peer-ilor de unde sunt preluate acestea.

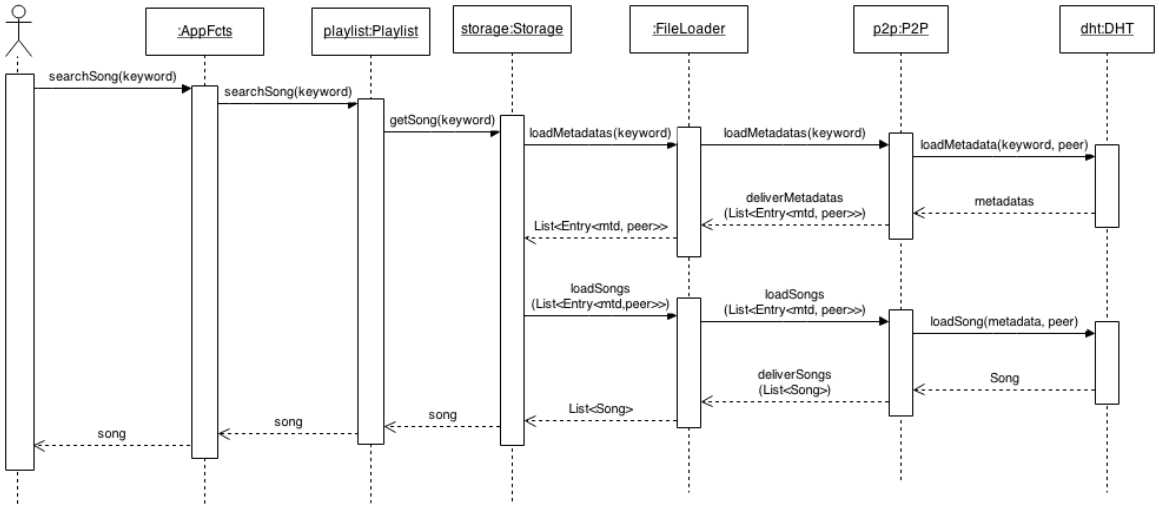


Figura 5.7 Diagrama de secvențe - Căutarea unei melodii de către un utilizator

Figura 5.8 prezintă diagrama de secvență a procesului de recomandare din momentul în care MP3 Player-ul anunță terminarea unei melodii. Obiectul de tip Mp3Player apelează metoda obiectului queue de tip PlaylistQueue prin care cere următoarea melodie pe care să 0 difuzeze (*getNextSong()*). La nivelul logic al aplicației, coada de melodii (PlaylistQueue) solicită sistemului de recomandare o nouă melodie. Clasa care se ocupă de recomandări (și implementează I_RecommenrSystem) se adresează „depozitului" (Storage), căruia îi cere o anumită melodie (după nume sau gen muzical), determinată de sistemul de recomandare respectiv (metoda *getSong(String keyword)*) . În mod asemănător primei diagrame de secvențe, este căutată melodia în rețea, descărcată și difuzată utilizatorului.

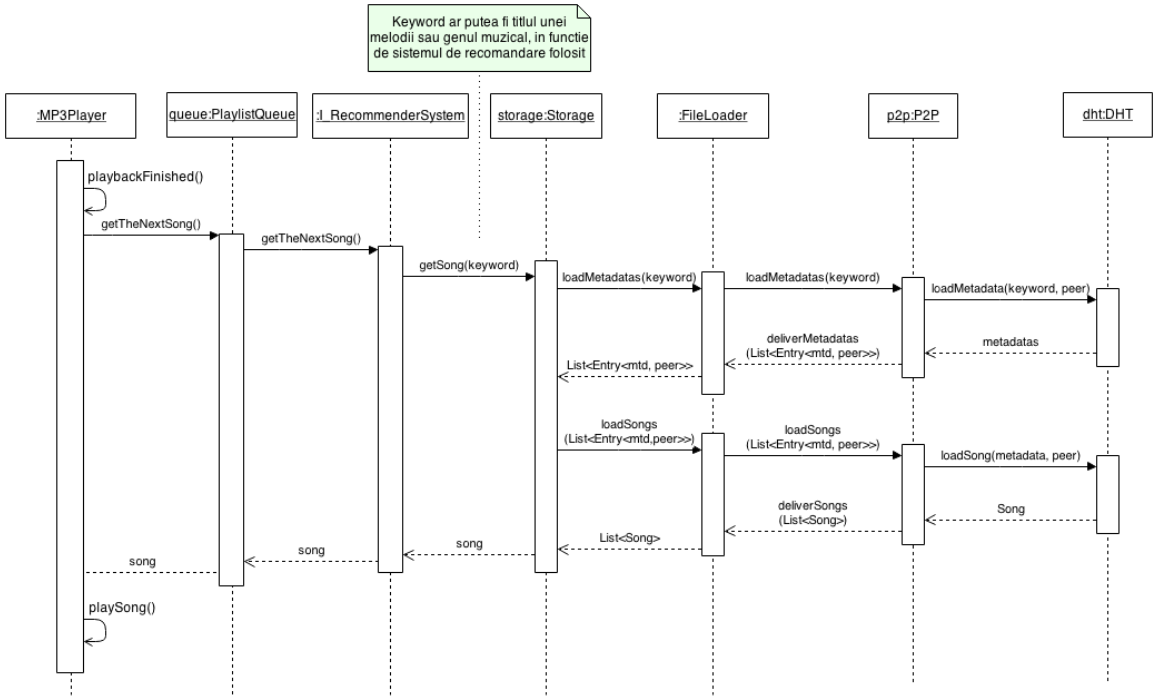


Figura 5.8 Diagrama de secvențe - Recomandarea unei melodii

Prin prezentarea acestor diagrame de secvențe este evidențiat principiul de bază al arhitecturii pe nivele, în care partea de logică este separată de infrastructura aplicației. Se observă trecerea datelor prin aceste nivele, pornind de la nivelul superior până la cel inferior, și înapoi.

5.5. Nivelul prezentării

Nivelul de prezentare conține clasele responsabile de construirea interfeței grafice. Fiecare caz de utilizare al sistemului poate fi realizat pornind de la componentele interfeței utilizator.

Diagrama de clase a pachetului GUI este reprezentată în figura 5.9.

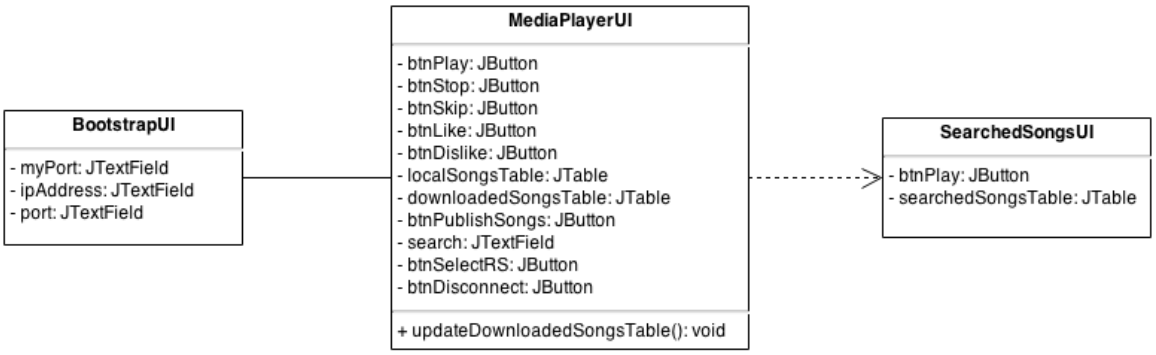


Figura 5.9 Diagrama de clase la nivel de prezentare

În momentul pornirii aplicației, se deschide prima fereastră, corespunzătoare clasei BootstrapUI. Utilizatorul poate completa atât propriul port de pe care să se conecteze la rețea, cat și adresa și portul nodului la care să se conecteze. Acesta poate alege să se conecteze ca Master Peer, creându-și propria rețea, sau să se conecteze la o rețea deja existentă.

Dacă este realizată cu succes conectarea, se deschide o nouă fereastră, în care utilizatorul își va alege folderul aplicației de unde vor fi preluate melodiile locale. În acest folder va fi creat un folder provizoriu (numit „download”) în care vor fi stocate temporar fișierele descărcate din rețea. La deconectare, atât melodiile descărcate cat și acest folder provizoriu vor fi șterse. În cazul în care conectarea la rețea ar fi eșuat, utilizatorul era atenționat și îndemnat să încerce din nou.

După alegerea folderului, se deschide fereastra principală, MediaPlayerUI, unde se regăsesc toate elementele de bază ale aplicației. Există un tabel cu melodiile locale și un tabel cu melodiile descărcate și ascultate de utilizator. În cel de-al doilea tabel, pe lângă numele melodiei, este afișată și adresa nodului de la care a fost preluată melodia.

În câmpul de căutare melodie, utilizatorul introduce cuvântul cheie după care dorește să fie realizată căutarea. În cazul în care nu este găsită nicio melodie, utilizatorul este înștiințat și îndemnat să încerce cu un alt cuvânt. Altfel, se deschide o nouă fereastră, SearchedSongsUI, unde sunt afișate într-un tabel titlurile melodiilor găsite, împreună cu adresele nodurilor de la care provin melodiile. Utilizatorul poate alege dintre aceste titluri ce melodie dorește să asculte. Melodia respectivă va fi descărcată pe calculatorul personal și redată prin MP3Player.

Tot în fereastra principală, prin intermediul butoanelor disponibile, utilizatorul e liber să-și exprime preferințele prin acordarea de voturi pozitive (Like) sau negative (Dislike). De asemenea, poate utiliza aplicația ca pe un MP3 Player: poate porni, opri sau sări peste o melodie (butoanele de Play, Stop, Skip).

Implementarea arhitecturii pe nivele oferă o împărțire a responsabilităților și o separare a infrastructurii de logica sistemului și de interfața grafică. Se dezvoltă astfel o aplicație flexibilă, cu module slab cuplate, care pot fi mai ușor de testat în mod individual.

Capitolul 6. Testare şi Validare

Dezvoltarea unui produs software presupune etape de analiză, proiectare, testare si mentenanță a codului sursă al programelor. Deși sunt considerate etape separate, între ele există o puternică relație de interdependență. În acest capitol vor fi prezentate etapele de testare, verificare și validare a sistemului de radio online.

Testarea s-a realizat după fiecare etapă de proiectare a aplicației, respectiv după implementarea fiecărui nivel arhitectural al programului software. Au fost utilizate mai multe modalități de testare, care vor fi prezentate în subcapitolele următoare.

6.1. Testare funcțională

Testarea funcțională se aplică pentru a verifica dacă sistemul se comportă și funcționează corect, conform specificațiilor din proiect. Modul de realizare a acestei testări a fost prin crearea unor scenarii de test create în conformitate cu cerințele și funcționalitățile sistemului, cu scopul de a găsi posibile erori în aplicație .

În continuare vor fi descrise 2 cazuri de testare care au fost utilizate pentru testarea sistemului, și anume conectarea la o rețea peer-to-peer, respectiv căutarea unei melodii în rețea.

TC 1. Titlu: Conectare la retea
Descriere: Utilizatorul trebuie sa se poata conecta la o retea pe baza adresei sale ip si a portului calculatorului. Conectarea se realizeaza prin introducerea adresei unui nod al retelei respective.
Scenariu 1:

1. Oprește conexiunea la Internet
2. Pornește sistemul
3. Introdu adresa ip si portul nodului la care sa se realizeze conexiunea
4. Apasă „Connect”

Rezultat așteptat: Un mesaj de eroare care să sugereze imposibilitatea realizării conexiunii, pe motiv ca nu găsește in rețea nodul cu adresa respectivă.

Scenariu 2:

1. Pornește conexiunea la Internet
2. Pornește sistemul
3. Introdu un numar de port in câmpul „My port” de pe care s-a mai realizat deja o conectare la o rețea
4. Introdu adresa ip si portul nodului la care sa se realizeze conexiunea
5. Apasă „Connect” (sau „Connect as Master Peer”)

Rezultat așteptat: Un mesaj de eroare care să atenționeze utilizatorul că portul introdus este deja utilizat de către sistem.

Scenariu 3:

1. Pornește sistemul
2. Introdu un numar de port invalid in câmpul „My port”
3. Apasă „Connect as Master Peer”

Rezultat aşteptat: Un mesaj de eroare care să atenționeze utilizatorul că a introdus un număr de port invalid.

Scenariu 4:

1. Pornește sistemul
2. Introdu adresa ip sau port invalid
3. Apasă „Connect”

Rezultat aşteptat: Un mesaj de eroare care să atenționeze utilizatorul că adresa sau portul introdus este invalid.

Scenariu 5:

1. Pornește sistemul
2. Introdu adresa ip a nodului la care sa se realizeze conexiunea si lasa campul portului necompletat
3. Apasă „Connect”

Rezultat aşteptat: Un mesaj de eroare care să atenționeze utilizatorul că nu a completat toate câmpurile.

TC 2. Titlu: Căutarea unei melodii în rețea

Descriere: Utilizatorul trebuie să aibă posibilitatea să caute și să descarce o melodie din rețea. Căutarea se va realiza pe baza unui cuvânt cheie ce ține de atributele din metadata melodiei căutate.

Scenariu 1:

1. Pornește sistemul
2. Conectează-te la o rețea
3. Apasă „Search Song”

Rezultat aşteptat: Mesaj de eroare care îndeamnă utilizatorul să completeze câmpul de căutare a melodiei

Scenariu 2:

1. Pornește sistemul
2. Conectează-te la o rețea
3. Oprește conexiunea la Internet
4. Introdu cuvântul după care să se realizeze căutarea melodiei
5. Apasă „Search Song”

Rezultat aşteptat: După expirarea timpului alocat sistemului de a căuta o melodie în rețea, va apărea un mesaj de eroare prin care utilizatorul e anunțat că nu a reușit să găsească nicio melodie.

Scenariu 3:

1. Pornește sistemul
2. Conectează-te la o rețea
3. Introdu cuvântul după care să se realizeze căutarea melodiei
4. Apasă „Search Song”
5. Oprește conexiunea la Internet
6. Din lista de titluri de melodii găsite și returnate de către sistem, alege melodia pe care dorești să o asculți

Rezultat aşteptat: După expirarea timpului alocat sistemului de a descărca o melodie în rețea, va apărea un mesaj de eroare prin care utilizatorul e anunțat că nu s-a putut realiza descărcarea melodiei căutate.

6.2. Testare unitară

Testarea unitară s-a impus in ultima perioadă in dezvoltarea proiectelor scrise în limbajul Java, pe măsura apariției unor utilitare gratuite de testare a claselor care au contribuit la creșterea vitezei de programare și la micșorarea drastică a numărului de defecte. Cel mai folosit utilitar pentru testarea unitară a claselor Java este Junit, care a fost utilizat și în cadrul acestui proiect.

În cazul acestei aplicații, au fost implementate clase de test pentru testarea infrastructurii sistemului și a conectivitatii rețelei.

Figura 6.1 prezintă diagrama de clase a pachetului Test.

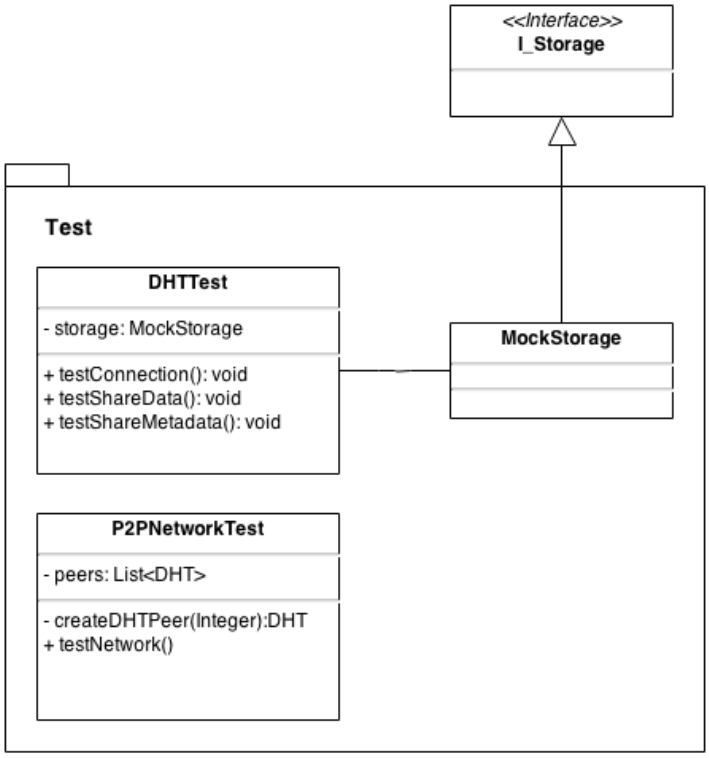


Figura 6.1 Diagrama de clase a pachetului Test

Clasa DHTTest simuleaza funcționalitățile structurii DHT a unei rețele peer-to-peer. Conține 3 metode de test, descrise în punctele următoare:

- testConnection() - În această metodă se testează conectivitatea între 3 peer-i, unde un peer este reprezentat de instantierea unui obiect de tip DHT, din pachetul de Infrastructură. Acești peer-i au aceeași adresă ip (adresa locală a calculatorului de pe care se face testarea), doar numărul portului diferă. Se dorește verificarea funcționării corecte a metodelor *startWithoutBootstrap()*, respectiv *bootstrap(InetAddress address, Integer port)* a clasei DHT.
- testShareData() - Metoda de test realizează conexiunea între 3 peer-i, asemenea metodei anterioare. În plus, este testată și funcționalitatea de adăugare a datelor în

rețea (metoda put(String key) din clasa DHT), și de căutare a nodurilor care conțin o anumită cheie introdusă în rețea (metoda *getPeers(String key)*) din DHT.

- testShareMetadata() - Această metodă testează aceeași funcționalitate ca și metoda precedentă. Datele introduse vor fi obiecte de tip SongMetadata. Pe lângă publicarea în rețea și căutarea nodurilor care dețin o anumită cheie, de data aceasta va fi testată și descărcarea datelor aparținând cheii respective. Se va testa astfel și metoda *loadSongMetadata(PeerAddress peerAddress, String keyword)* a clasei DHT.

Pentru realizarea acestor metode, este creată o clasă ce implementează interfața I_Storage din pachetul FileManager (clasa MockStorage), pe baza căruia va fi construit fiecare DHT al unui peer.

Clasa P2PNetworkTest simulează la un nivel mai complex crearea unei rețele peer-to-peer. Nodurile vor fi reprezentate printr-o listă de obiecte de tip DHT. Este testată conectivitatea rețelei, împreună cu funcționalitatea de publicare, căutare și descărcare a unor obiecte de tip Song din rețea. Fiecare nod va avea propriul folder de pe care își va prelua fișierele audio, și în care va descărca alte fișiere din rețea.

Aceste clase de test au fost implementate cu scopul de a verifica comportamentul infrastructurii sistemului înainte de a trece la implementarea componentelor de la nivelul superior al arhitecturii software. Au fost testate astfel funcționalitățile de bază pe care o rețea peer-to-peer trebuie să le îndeplinească: crearea, conectivitatea, publicarea, căutarea și extragerea datelor din rețea.

6.3. Testare nefuncțională

Testarea nefuncțională consta în testarea calitatilor si cerințelor nefuncționale ale sistemului. Pentru verificarea atributelor de calitate ale sistemului a fost nevoie de crearea unei rețele peer-to-peer cu un număr de 5 utilizatori. Au fost testate două cazuri: (1) utilizatorii erau toți conectați la aceeasi rețea privată de internet sau (2) utilizatorii aveau adrese publice de internet.

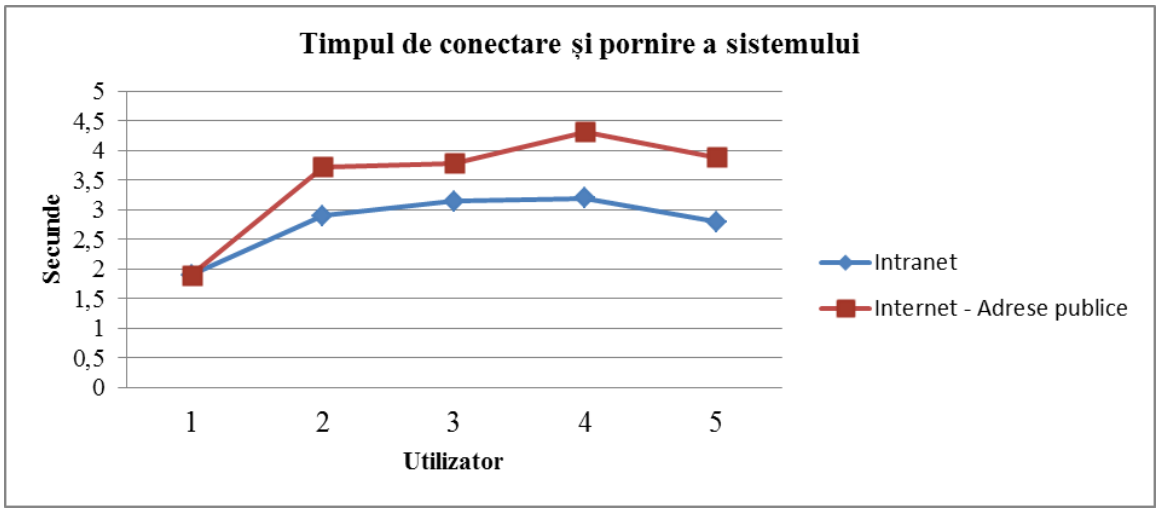


Figura 6.2 Timpul de conectare și pornire a sistemului pentru o rețea de 5 utilizatori

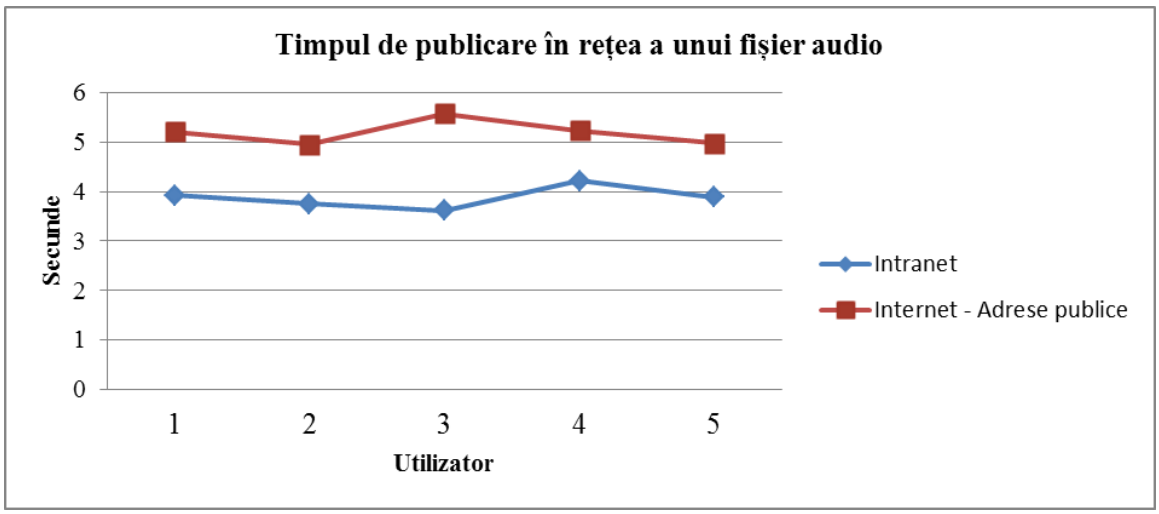


Figura 6.3 Timpul de publicare a unui fișier audio într-o rețea de 5 utilizatori

Figura 6.3 ilustrează timpii de publicare a unui fișier audio într-o rețea peer-to-peer de 5 utilizatori. Timpul mediu pentru un utilizator de adăugare a unui astfel de fișier într-o rețea peer-to-peer de utilizatori din aceeași rețea privată este de aproximativ 3,5 secunde, în timp ce într-o rețea peer-to-peer din internet este aproape cu o secundă mai mult (≈ 4,5 sec).

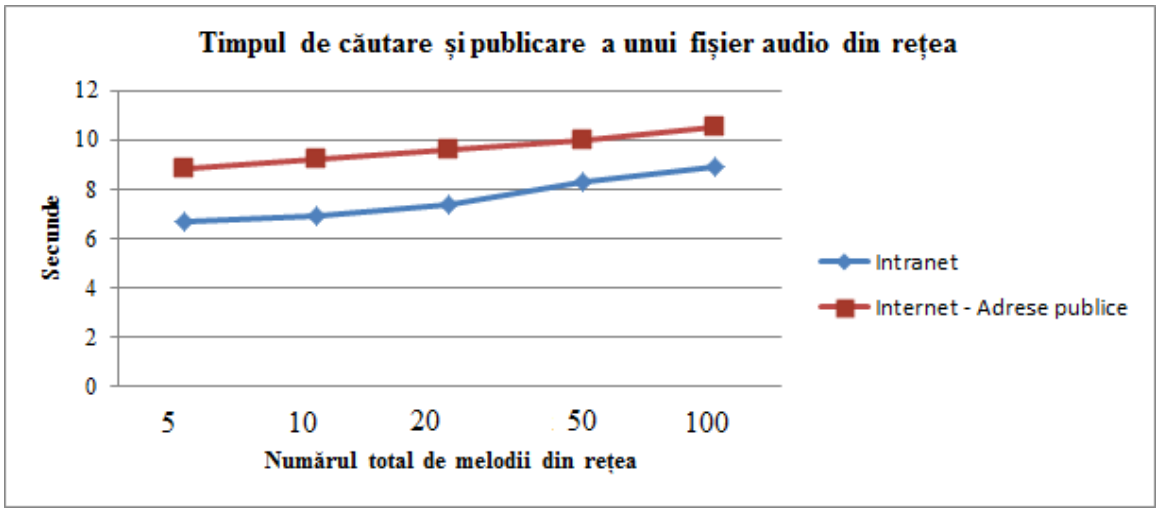


Figura 6.4 Timpul de căutare și descărcare a unei melodii din rețea

Figura 6.4 prezintă timpii de căutare și descărcare a unui fișier audio dintr-o rețea peer-to-peer din intranet, respectiv internet. Căutarea a fost realizată într-o rețea în care au fost publicate în total 5, 10, 20, 50 sau 100 de melodii. Timpul mediu pentru o astfel de operație intr-o rețea privată este de aproape 8 secunde (≈ 7,7 sec), iar pe internet este de aproximativ 10 secunde (≈ 9,6 sec)

Prin crearea acestei rețele de 5 utilizatori a fost de asemenea testată și componenta de recomandare. Fiecare utilizator a publicat în rețea între 10 și 30 de melodii preferate:

- Utilizatorul cu numărul 1 a adăugat o listă de melodii cu predilecție spre genul Rock. Pe lângă acestea, pe lista sa s-au mai găsit și câteva melodii de Folk sau Pop.
- Utilizatorul 2 a distribuit o listă de melodii diversificată, aparținând mai multor genuri muzicale: Pop, Rock, Blues, Folk, Etno sau Clasic.
- Utilizatorul 3 a adăugat o listă de melodii aparținând în general genurilor Dance și House.
- Utilizatorii 4 și 5 au distribuit liste de melodii comerciale, de diverse genuri: Pop, R&B, Dance sau Hip-Hop.

Fiecare utilizator a avut posibilitatea să aleagă tipul de sistem de recomandare care să îi construiască un playlist.

Primul utilizator a optat pentru recomandări în funcție de genul muzical preferat, iar sistemul i-a recomandat alte melodii Rock din rețea.

Al doilea utilizator a ales sistemul de recomandare bazat pe rețeaua de afinitate. Sistemul i-a recomandat melodiile celui mai apropiat utilizator din rețea ca și gusturi, în cazul acesta fiind vorba de primul utilizator.

Cel de-al treilea utilizator a ales să îi fie creat un playlist după genurile sale preferate. Sistemul i-a recomandat alte melodii de genul Dance și House, primite de la utilizatorii 4 și 5.

Utilizatorii 4 și 5 au optat pentru un sistem de recomandare combinat. Pe baza rețelei de afinitate, sistemul le-a recomandat melodii de la utilizatori apropiați din punct de vedere al melodiilor distribuite în rețea. Astfel, utilizatorul 5 a primit melodii de la utilizatorul 4, iar utilizatorul 4 a primit melodii de la utilizatorul 5, aceștia având gusturi asemănătoare. Le-au mai fost recomandate melodii în funcție de genurile lor preferate. De asemenea, playlist-urile lor au conținut și melodiile cele mai apreciate din rețea.

Având în vedere că melodiile aparțin unui domeniu cu grad mare de subiectivitate, nu se poate decide întotdeauna care este cea mai bună melodie pe care utilizatorul o poate aștepta de la sistemul de recomandare. S-a încercat în schimb găsirea unor melodii apropiate de preferințele și gusturile utilizatorilor, care să ducă la satisfacerea cerințelor acestora.

Capitolul 7. Manual de Instalare si Utilizare

Acest capitol prezintă o descriere detaliată a punerii în funcțiune a sistemului de radio online, atât din punct de vedere al instalării, cat și al utilizării aplicației pe calculatorul personal.

7.1. Manual de Instalare

Sistemul de radio online bazat pe o rețea P2P poate rula pe oricare din următoarele sisteme de operare Windows: XP, Vista, 7, 8. Sistemul este dezvoltat în mediul de programare Eclipse iar limbajul de programare folosit este Java (versiunea 7).

Astfel, pentru a putea rula aplicația, utilizatorul trebuie să aibă platforma Java instalată, versiunea 7 (Java SE Development Kit 7).

7.2. Manual de Utilizare

Acest subcapitol descrie pașii pe care trebuie să îi urmeze utilizatorul pentru a putea folosi cu succes aplicația.

7.2.1. Fereastra de conectare la sistem

La deschiderea aplicației utilizatorul trebuie să se conecteze la o rețea P2P, pentru a putea folosi în continuare aplicația. Acesta are două posibilități:

1. **Conectare ca Master Peer**

Utilizatorul își crează propria rețea, unde va avea rol de Master Peer. Următorul membru al rețelei se va conecta la adresa sa.

Utilizatorul își poate introduce propriul port de pe care să se conecteze, în câmpul *„My Port”*. În cazul în care acest câmp rămâne necompletat, sistemul va seta automat portul 6000.

Pentru conectare ca Master Peer și crearea propriei rețele, utilizatorul va apăsa butonul *„Connect as Master Peer”*.

În figura 7.1, utilizatorul alege să se conecteze în rețea ca Master Peer, de pe portul 5000.

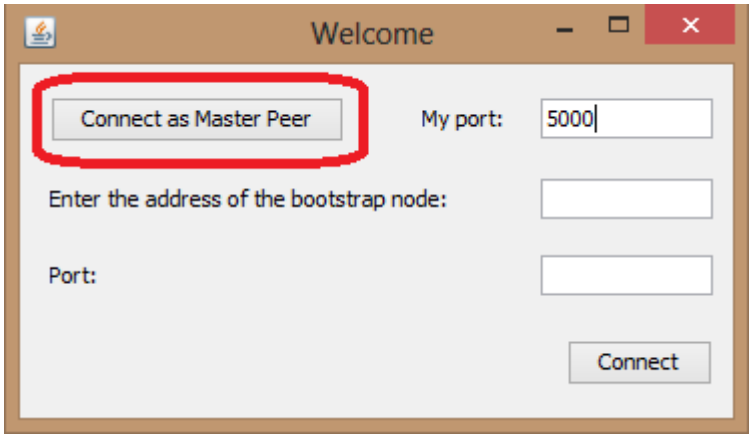


Figura 7.1 Conectare ca Master Peer

2. Conectare la o rețea existentă

Utilizatorul se poate conecta la o rețea deja existentă, cu condiția ca acesta să cunoască adresa unui peer al rețelei respective. Poate completa portul propriu, de pe care să se conecteze. Dacă acest câmp *(„My Port”)* rămâne necompletat, sistemul setează implicit portul 6000.

În câmpurile de adresa IP și port, utilizatorul completează adresa nodului la care dorește să se conecteze, iar pe urmă apasă butonul *„Connect”*.

În figura 7.2, utilizatorul alege să se conecteze la nodul cu adresa 192.168.56.1, port 5000, de pe portul 4000 al calculatorului propriu.

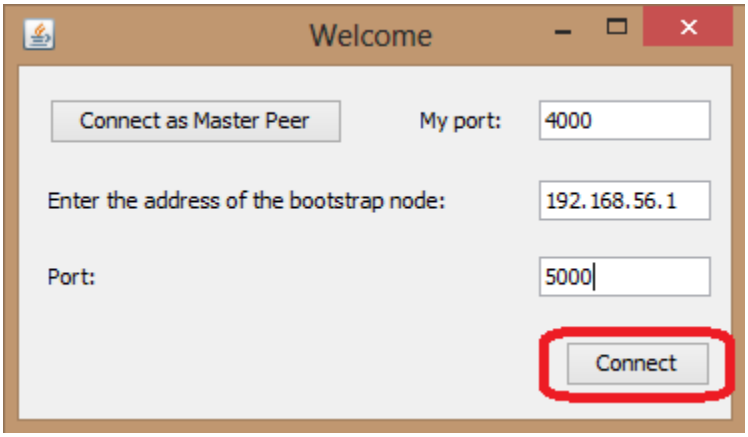


Figura 7.2 Conectare la o rețea existentă

În cazul eșuării conectării, utilizatorul este avertizat și îndemnat să încerce din nou. Sistemul expune cauza eșecului (ex: câmpuri completate greșit, adresă inexistentă sau indisponibilă etc.).

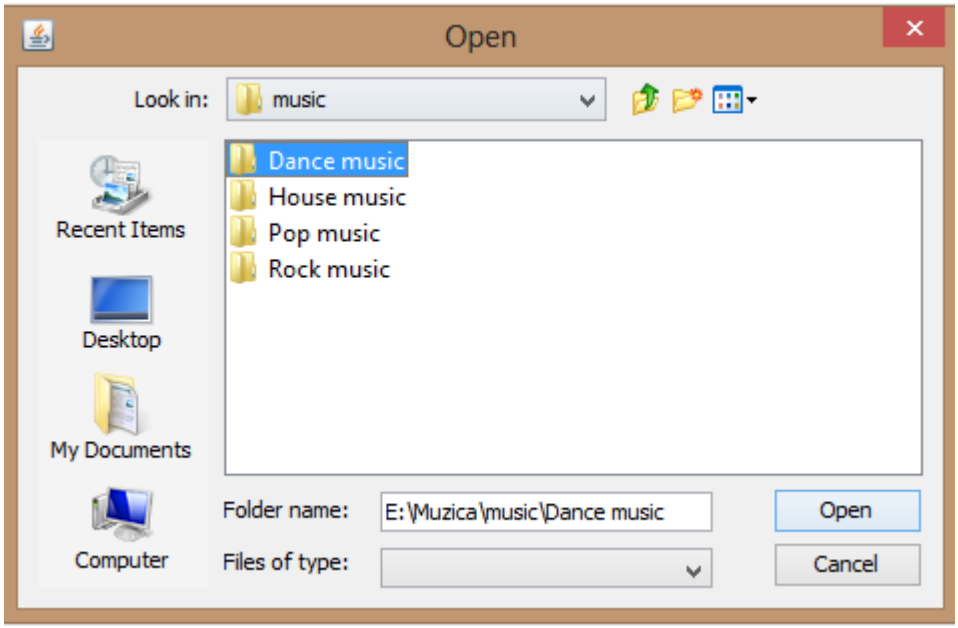


Figura 7.3 Alegerea folderului aplicației

După ce utilizatorul este conectat la rețea, acesta își va alege folderul aplicației de unde vor fi preluate melodiile locale. (Figura 7.3) În acest folder va fi creat un folder provizoriu (numit „download”) în care vor fi stocate temporar fișierele descărcate din rețea. La deconectare, atât melodiile descărcate cat și acest folder provizoriu vor fi șterse. În cazul în care utilizatorul nu alege niciun folder (apasă butonul *„Cancel”* sau *„Close”*), sistemul setează implicit folderul de Home.

7.2.2. Fereastra principală de ascultare melodii

După conectarea cu succes la o rețea, utilizatorului i se deschide fereastra principală. De aici, acesta poate utiliza funcționalitățile de baza ale aplicației: redarea fișierelor audio, căutare melodiilor în rețea sau notarea unei melodii prin vot pozitiv sau negativ. Sistemul permite doar utilizarea fișierelor in format .mp3. Pentru o cât mai bună utilizare a sistemului de radio online, este recomandat ca fișierele audio să aibă etichetele de metadata corecte, cu atributele de titlu, artist sau gen muzical completate corespunzător.

În figura 7.4 este ilustrată fereastra principală, imediat după conectarea utilizatorului la rețea.

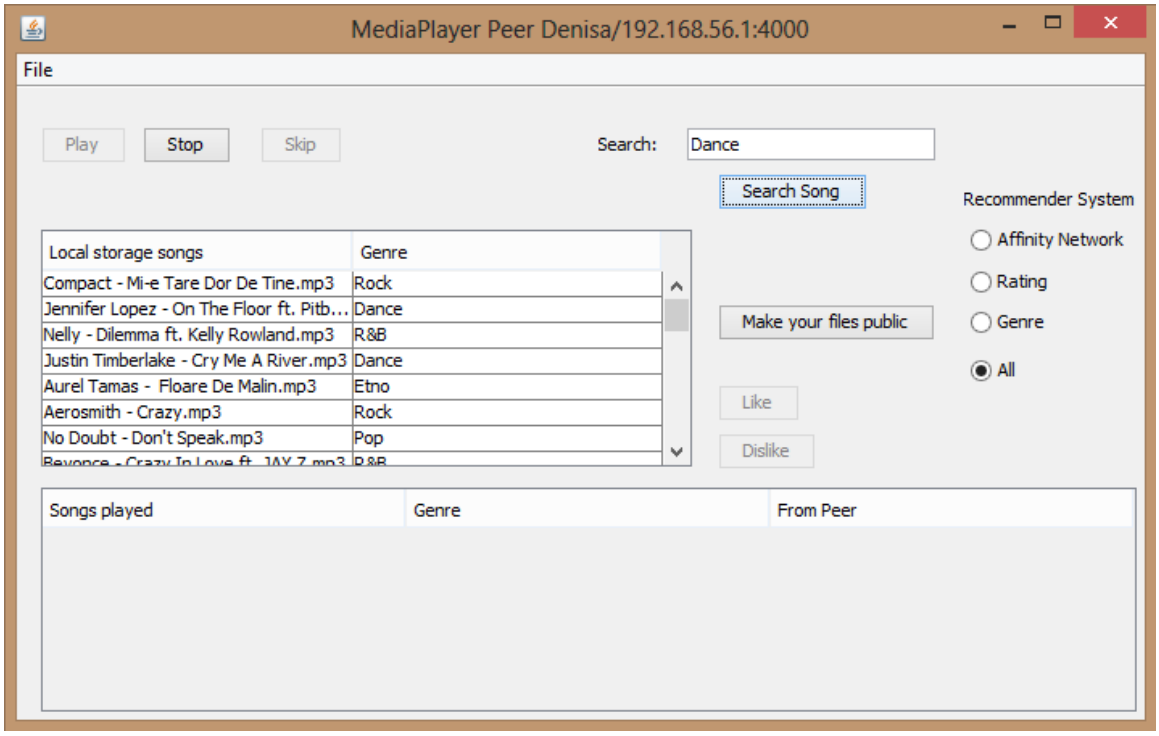


Figura 7.4 Ferestra principală a aplicației dupa conectarea la rețea

Aceasta fereastra conține două tabele:

- tabela cu melodiile locale (*„Local storage songs”),* ce vor fi distribuite în rețea prin apăsarea butonului *„Make your files public”*
- tabela cu melodiile ascultate de către utilizator, care va fi reactualizată după fiecare melodie ascultată.

Butonele de *Play*, *Stop* și *Skip* asigură funcționalitatea de MP3 Player a sistemului. Prin apăsarea butonului *Play* se poate porni o melodie din playlist-ul local (*„Local storage songs”*). *Stop* realizează oprirea difuzării unui fișier audio. Butonul de *Skip* realizează trecerea la următoarea melodie din playlist-ul generat automat de către sistem. Acest buton este activat doar în momentul difuzării playlist-ului de radio online.

Utilizatorul poate selecta tipul de sistem de recomandare care să îi determine următoarele melodii redate: după afinitățile dintre el și ceilalți membrii rețelei (*Affinity Network*), după popularitatea unei melodii (*Rating*), după genul muzical preferat (*Genre*) sau după o combinație a acestora.

Pentru a primi automat un playlist de melodii de la sistemul de radio online, utilizatorul trebuie să caute o melodie în rețea, după care vor fi generate și celelate fișiere audio pe baza recomandărilor sistemului de recomandare. Căutarea se face după un cuvânt cheie al melodiei (ex: din titlu, artist sau gen muzical). Acest cuvânt trebuie introdus în câmpul de căutare.

După apăsarea butonului *„Search song”*, se deschide o nouă fereastră cu melodiile găsite pe baza cuvântului cheie introdus. În figura 7.5 este ilustrată o astfel de fereastră, în urma căutării melodiilor cu genul muzical „Dance”. Dintre acestea, utilizatorul poate alege ce fișier audio să asculte mai departe (prin apăsarea butonului *Play*). În cazul în care sistemul nu găsește nicio melodie corespunzătoare cuvântului dat, atenționează utilizatorul și îl îndeamnă să încerce altă căutare.

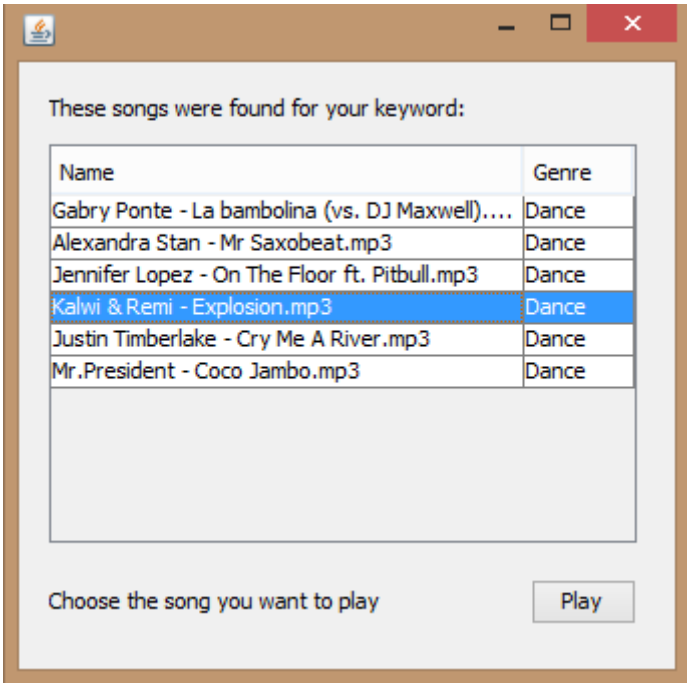


Figura 7.5 Fereastra melodiilor găsite după un cuvânt cheie

După ce a ales melodia pe care dorește să o asculte, fereastra cu melodiile găsite este închisă, iar utilizatorul este întors la fereastra principală a aplicației. Tabelul melodiilor ascultate este actualizat cu melodia aleasă, împreună cu adresa nodului de la care a fost descărcată aceasta.

Pornind din acest moment, utilizatorul va primi automat un playlist de melodii determinat de sistemul de recomandare. Tabelul melodiilor difuzate este reactualizat după fiecare melodie ascultată. (Figura 7.6)

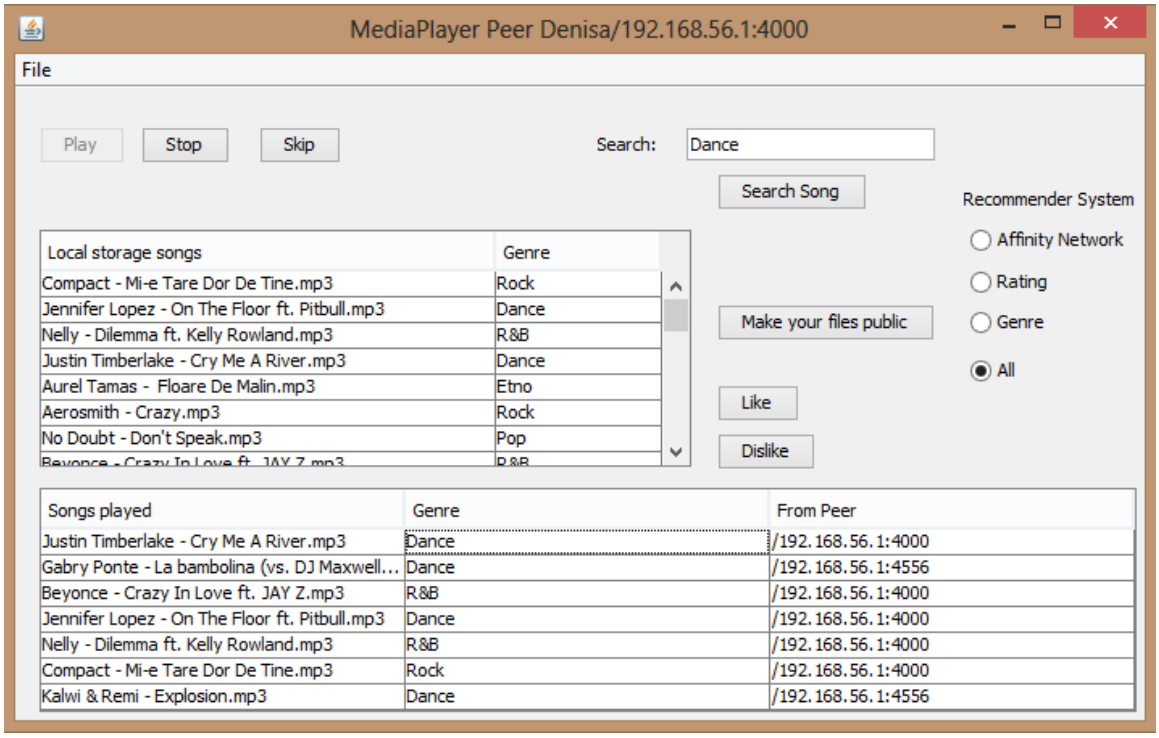


Figura 7.6 Fereasta principală a aplicației după difuzarea unui playlist generat de sistemul de recomandare

Utilizatorul poate aprecia o melodie, pozitiv sau negativ, prin apăsarea butoanelor de Like sau Dislike. Aceste voturi sunt folosite la antrenarea sistemului de recomandare, care va recomanda melodiile următoare din playlist-ul de radio online pe baza preferințelor utilizatorului.

În orice moment, utilizatorul poate începe o nouă căutare a unei melodii. De asemenea, se poate deconecta de la această rețea, prin selectarea File > Disconnect din bara de meniuri. După deconectare, utilizatorul este adus din nou la fereastra de conectare, unde se pote conecta la o alta retea peer-to-peer. În momentul deconectării sau închiderii aplicației, este eliberată zona de memorie de pe calculatorul utilizatorului, fișierele audio descărcate pentru playlist sunt șterse automat.

Capitolul 8. Concluzii

Acest capitol prezintă obiectivele atinse în urma etapelor de realizare a acestei aplicatii software, precum și posibile îmbunătățiri aduse programului sau dezvoltări ulterioare.

8.1. Obiective realizate

Prin proiectarea și implementarea acestui sistem de radio online bazat pe o rețea peer-to-peer, cu sistem de recomandare integrat au fost puse în practică noțiunile învățate in decursul perioadei de studii universitare. Dezvoltarea proiectului a presupus un proces continuu de analiză, proiectare și testare. Astfel, au fost parcurse următoarele etape:

- stabilirea obiectivelor generale și specifice ale acestui proiect
- documentarea domeniului în care se situează tema proiectului, mai exact studierea carecteristicilor unei rețele peer-to-peer sau a unui sistem de recomandare
- studierea sistemelor similare deja existente pe piață
- analizarea sistemului, identificarea tuturor cerințelor proiectului și găsirea tehnologiilor care să poată îndeplini aceste cerințe
- determinarea modelului arhitectural software, de tip multi-layer
- dezvoltarea produsului software într-o manieră incrementală și iterativă: în cadrul unei iterații a fost realizată analiza cerințelor ce urmau să fie implementate, implementarea și testarea acestora, după niște scenarii de test.
- verificarea și validarea sistemului de radio online.

Au fost realizate toate funcționalitățile definite în capitolul 2:

- Conectarea la o rețea peer-to-peer
- Distribuirea melodiilor în rețea
- Căutarea și descărcarea unei melodii din rețea
- Crearea unui playlist pe baza recomandărilor
- Funcționalitatea de MP3 Player: redarea fișierelor audio de format .mp3
- Notarea melodiilor de către utilizator

Pentru a putea utiliza radio-ul, utilizatorul se poate conecta la o rețea peer-to-peer. Acesta își poate alege o listă de melodii pe care să o distribuie în rețea, pentru a fi accesibile și celorlalți membri ai rețelei. Sistemul permite utilizatorilor să caute și să asculte melodii din rețea. Căutarea se face pe baza atributelor din metadatele fișierelor audio, de format .mp3. Sistemul de recomandare crează un profil muzical al utilizatorilor, în funcție de melodiile proprii publicate în rețea, melodiile căutate și de voturile de apreciere sau lipsă de apreciere acordate melodiilor ascultate. După căutarea și descărcarea unei melodii, radio-ul construiește pentru utilizator un playlist cu melodii determinate de sistemul de recomandare.

Utilizatorul poate alege între 3 variante de construire a playlist-ului: prin găsirea utilizatorilor cu gusturi asemănătoare și descărcarea melodiilor acestora, prin ascultarea celor mai populare melodii din rețea sau prin descărcarea melodiilor cu genul preferat. Prin posibilitatea de selecție a tipului de recomandări, utilizatorul poate evita anumite deficiențe ce apar în urma folosirii unui sistem de recomandare, cum ar fi nevoia de

antrenare a sistemului pentru obținerea unor rezultate bune (problemele „cold start” sau „first rater” descrise în subcapitolul 3.2.3). Recomandările în funcție de rating sau gen solicită implicarea continuă a utilizatorilor în procesul de votare a melodiilor, e nevoie de timp de creare a istoricului unui utilizator sau a unei melodii. Sistemul de recomandare bazat pe afinitățile dintre utilizatori vine cu o soluție de diminuare a acestor deficiențe, din moment ce identificarea utilizatorilor cu gusturi asemănătoare se face imediat după conectare, prin intermediul melodiilor proprii publicate în rețea.

Ca în orice altă aplicație software, mereu vor putea fi aduse îmbunătățiri pentru a face aplicația mai performantă sau să răspundă mai bine nevoilor utilizatorilor. Posibile astfel de îmbunătățiri sau dezvoltări ale programului vor fi prezentate în subcapitolul următor.

8.2. Dezvoltări și îmbunătățiri ulterioare

Prin proiectarea și implementarea acestui sistem de radio online s-a dorit realizarea unei aplicații performante și ușor de folosit de către utilizatori. Un aspect important al acestei aplicații o reprezintă scalabilitatea: programul trebuie să suporte un volum mare de date și un număr ridicat de utilizatori conectați la o rețea peer-to-peer. O problemă a acestui sistem este reprezentat de creșterea timpului de căutare și descărcare odată cu mărirea numărului de utilizatori sau de date partajate în rețea. Căutarea unei melodii într-un volum mare de astfel de fișiere poate duce la blocarea temporară a aplicației: interfața grafică afișează melodiile găsite doar în momentul în care au fost găsite deja 10 melodii, s-a căutat prin toată rețeaua sau s-a ajuns deja la un timp maxim de așteptare. Pentru a evita astfel de momente în care utilizatorul este nevoit să aștepte după sistem, fără să primească nicio notificare din partea acestuia, se pot afișa rezultatele în interfața grafică încă din momentul găsirii, în locul afișării a tuturor rezultatelor deodată.

O altă problemă este legată de streaming-ul de date. Pentru a putea fi redată de către MP3 Player, fișierul audio trebuie să fie descărcat în totalitate pe calculatorul utilizatorului. Fișierul este extras de la un singur peer din rețea, ceea ce limitează perfomanta aplicației, din cauza cantității maxime de date pe care un peer poate să trimită odată. O soluție pentru rezolvarea echilibrării încărcării datelor poate fi replicarea fișierelor audio și descărcarea simultană a unui fișier de la mai mulți utilizatori.

O deficiență a acestui sistem ține și de persistența datelor. În momentul părăsirii rețelei, fișierele audio distribuite de utilizator devin indisponibile celorlalți membri ai rețelei, iar utilizatorul deconectat pierde și datele reținute de sistemul de recomandare, privind voturile acestuia date melodiilor ascultate. Pentru evitarea acestei probleme, ar trebui găsite soluții prin care entitățile (cheie, valoare) introduse în DHT-ul rețelei peer-to-peer să fie menținute mereu active. De asemenea, trebuie căutate soluții de reținere a datelor sistemului de recomandare și după deconectare, pentru a avea deja un profil muzical în momentul conectării la o altă rețea.

Alte îmbunătățiri pot fi aduse și în ceea ce privește securitatea sistemului și a rețelei. În sistemul actual, nu sunt rezolvate problemele de securitate, cum ar fi introducerea intenționată de chei și metadate greșite în DHT, sau de manipulare a sistemului de recomandare prin voturi acordate greșit.

Modificări cu scopul de a construi playlist-uri cu recomandări mai bune pentru utilizatori pot fi aduse și sistemului de recomandare. O astfel de modificare ar putea fi

introducerea termnului de similaritate între atributele melodiei (de exemplu, genul Rap este *apropiat* de Hip-Hop, artistul Shakira este foarte *diferit* de Bon Jovi).

Pe termen lung, o posibilă dezvoltare a acestui sistem poate fi realizarea unei aplicații similare pentru un dispozitiv mobil.

Bibliografie

[1] Aiolli, F. „A Preliminary Study on a Recommender System for the Million Songs Dataset Challenge.” *Preference Learning: Problems and Applications in AI (PL-12).* Montpellier, France, August 28, 2012.

[2] Dădârlat, Vasile Teodor, și Emil Cebuc. „Rețele Locale de Calculatoare - de la cablare la interconectare.” Cluj-Napoca: Editura Albastră (MicroInformatica), 2005.

[3] Hecht, și alții. „Radiommender: P2P On-line Radio with a Distributed Recommender System.” *Peer-to-Peer Computing (P2P), 2012 IEEE 12th International Conference.* Tarragona, 3-5 Sept. 2012.

[4] *JAudioTagger*. http://www.jthink.net/jaudiotagger/.

[5] *JLayer - MP3 library*. http://www.javazoom.net/javalayer/javalayer.html.

[6] Kreitz, Gunnar, și Fredrik Niemelä. „Spotify – Large Scale, Low Latency, P2P Music-on-Demand Streaming.” *Proceedings of IEEE P2P*. 2010.

[7] Lua, Eng Keong, Jon Crowcroft, Marcelo Pias, Ravi Sharma, și Steven Lim. „A Survey and Comparison of Peer-to-Peer Overlay Network Schemes.” *IEEE Communications Surveys & Tutorials.* Martie 2004.

[8] Mahanta, Kashyap, și Guruprasad Khataniar. „Peer-to-peer (P2P) file sharing system: a tool for Distance Education.” *International Journal of Computer Science and Information Technology & Security (IJCSITS), Vol. 3, No.2*, April 2013.

[9] *Peer-to-peer*. http://en.wikipedia.org/wiki/Peer-to-peer.

[10] *Peer-to-Peer (P2P) Networks Tutorial.* http://tutorials.jenkov.com/p2p/index.html

[11] Ruffo, Giancarlo, și Rossano Schifanella. „Evaluating peer-to-peer recommender systems that exploit spontaneous affinities.” *Proceedings of the 2007 ACM symposium on Applied computing.* Seoul, Korea, March 11-15, 2007.

[12] Schollmeier, Rüdiger. „A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications.” *Proceedings of the First International Conference on Peer-to-Peer Computing, IEEE.* 2002.

[13] Song, Yading, Simon Dixon, și Marcus Pearce. „A Survey of Music Recommendation Systems and Future Perspectives.” *9th International Symposium on Computer Music Modellin and Retrieval (CMMR 2012).* 19-22 June 2012 .

[14] *TomP2, a P2P-Based Key-Value Pair Storage Library*. http://tomp2p.net/.

Anexe

Glosar de termeni

P2P – Peer-to-Peer
DHT – Distributed Hash Table
MP3 - MPEG-1 or MPEG-2 Audio Layer 3 (Moving Picture Experts Group)
IP – Internet Protocol
NAT – Network address translation
NAT-PMP - NAT Port Mapping Protocol
UPNP - Universal Plug and Play
SHA1 - Secure Hash Algorithm 1
GUI – Graphical User Interface

Listă figuri, tabele și ecuații