ReadsProfiler

Buzenchi Paula-Roberta, grupa 2B4

Universitatea Alexandru-Ioan Cuza, Iaşi, Facultatea de Informatică
buzenchiroberta@gmail.com

1 Introducere

Scopul proiectului **ReadsProfiler** este cel de a realiza o aplicaţie client-server prin care să se ofere acces la o librărie online. Aceasta poate gestiona căutări după criterii multiple şi să ofere rezultate relevante, cât şi posibilitatea de a descărca cărţile. De asemnea, se introduce şi un sistem de recomandări care se bazează pe preferinţele utilizatorilor, cum ar fi genurile literare preferate, genurile abordate de un anumit autor, rating-urile cărţilor, căutările sau descărcările cititorilor. Astfel, obiectivul principal al proiectului este acela de a dezvolta o aplicaţie stabilă, uşor de extins, care să ofere funcţionalităţi clasice utilizatorilor, cât şi mecanisme de recomandări personalizate, aşa încât experienţa acestora pe platformă să fie cât mai plăcută.

2 Tehnologii Aplicate

Implementarea proiectului este realizată în limbajul **C/C++**, folosind funcţii POSIX pentru programarea în reţea. Comunicarea dintre client şi server se realizează prin intermediul protocolului **TCP**, datorită orientării sale asupra conexiunii, proprietăţilor prin care se asigură transmiterea datelor fără pierderi, într-o ordine corectă şi fără duplicări. Aceste caracteristici sunt importante pentru operaţiile de căutare după criterii multiple, accesare de ierarhii de genuri şi subgenuri, descărcare de cărţi şi oferire de recomanări.

Facilităţile POSIX se reflectă prin apeluri precum **socket**, **bind**, **listen**, **accept**, **send**, şi **recv**, şi permit gestionarea conexiunilor simultane şi definirea protocolului la nivel de aplicaţie. Pe partea serverului, datele sunt gestionate prin intermediul unei baze de date **SQLite**, ce permite stocarea informaţiilor despre cărţi, autori, genuri şi istoricul utilizatorilor, uşurând generarea de recomandări.

Clientul integrează o interfaţă grafică atractivă prin care utilizatorul poate trimite comenzi, vizualiza cărţile şi interacţiona cu funcţionalităţile serverului într-un mod cât mai facil.

3 Structura aplicaţiei

Proiectul **ReadsProfiler** este realizat pe modelul **server-client** cu protocol **TCP**. Scopul serverului este cel de a gestiona căutările, baza de date, sistemul

de recomandări, iar al clientului cel de a oferi o interfaţă grafică interactivă pentru utilizatori. Astfel, aplicaţia este organizată pe trei nivele: **nivelul de prezentare**, **nivelul de logică** şi **nivelul de persistenţă**.

3.1 Concepte de Modelare

Modelarea aplicaţiei are la baza mai multe entităţi ce descriu structura librăriei, interacţiunile dintre utilizatori şi funcţionalităţile ce generează recomandări. Fiecare entitate are rolul de a facilita funcţionalităţile de căutare şi descărcare.

- **Carte**
  Este obiectul central al sistemului, pe care toţi utilizatorii o caută, vizualizează şi descarcă
  Atributele esenţiale:
  - titlu
  - autor
  - genuri subgenuri
  - an aparaţie
  - ISBN
  - rating

  Relaţii cu:
  - autorii
  - genurile subgenurile
- **Autor**
  Scriitorii cărţilor.
  Atributele esenţiale:
  - nume
  - genurile literare abordate

  Relaţii cu:
  - cărţile
  - informaţiile despre autor sunt folosite pentru recomandări
- **Gen/Subgen**
  Permite clasificarea cărţilor şi reprezintă un criteriu pentru recomandări.Are structura unui arbore cu niveluri multiple, iar o carte poate aparţine simultan mai multor noduri din arbore.
- **Utilizator**
  Reprezintă entitatea ce interacţionează cu serverul şi generează date pentru algoritmul de recomandare.
  Atributele esenţiale:
  - istoric căutări
  - rezultate accesate
  - descărcări
  - ratinguri
- **Recomandări**
  Această entitate rezultă din analiza mai multor entităţi.
  Sursele de informaţii pentru recomandări:

- genurile preferate de utilizator
- genurile abordate de autorii citiţi de utilizator
- ratingurile acordate cărţilor citite

În continuare vom vorbi despre componentizarea aplicaţiei. Vom delimita responsabilităţile serverului şi ale clientului.

Serverul reprezintă creierul aplicaţiei. Toate operaţiile de căutare, descărcare şi recomandare sunt executate de către acesta. Vom detalia componentele principale ale serverului.

a. Connection Manager
- se ocupă cu comunicarea din TCP
- deschide socket-ul serverului
- iniţializează socket-ul serverului(**socket, bind, listen**)
- acceptă clienţi noi cu **accept**
- gestionează comunicarea cu fiecare client
- închide conexiunile

b. Request Parser
- rolul său este de a interpreta corect mesajele de la client
- validează structura cererii, identifică operaţia pe care trebuie să o facă (descărcare, căutare, recomandare), şi extrage cuvintele cheie esenţiale

c. Search Engine
- motorul de căutare al aplicaţiei
- pe baza parametrilor primiţi(titlu, autor, gen, an, rating) construieşte interogări **SQL** şi le execută pe baza de date **SQLite**
- rezultatul este o listă de cărţi care respectă preferinţele utilizatorului

d. Recommendation Engine
- are rolul de a genera recomandări personalizate pentru fiecare utilizator pe baza comportamentului din trecut
- extrage istoricul utilizatorilor
- analizează genurile preferate
- identifică autori asemănători celor parcurşi
- generează recomandări după genuri, autori şi utilizatori cu preferinţe similare

e. Download Manager
- se ocupă de descărcarea cărţilor şi transmiterea fişierelor către client
- îi transmite informaţiile prin intermediul **TCP**-ului folosind **send()**
- gestionează erorile

f. SQLite Storage Layer
- reprezintă stratul ce se ocupă cu gestionarea bazei de date
- se conectează baza de date SQLite
- defineşte şi execută interogări SQL
- menţine conexiunile cu tabelele de cărţi, autori, relaţiile dintre entităţi, ierarhia genurilor, utilizatori, istoric, ratinguri şi descărcări
- închiderea conexiunilor şi rezolvarea erorilor

Acum vom analiza componentizarea clientului. Acesta se ocupă cu interaţiunea cu utilizatorul.

a. GUI Layer
- oferă o interfatţă atractivă pentru cititori

b. Command Dispatcher
- trimite cereri către server
- gestionează conexiunea **TCP** către server
- trimite datele prin **send()** şi le recepţionează prin **recv()**

c. Response Handler
- preia răspunsul serverului şi actualizează GUI

3.2 Diagrama detaliată a aplicaţiei

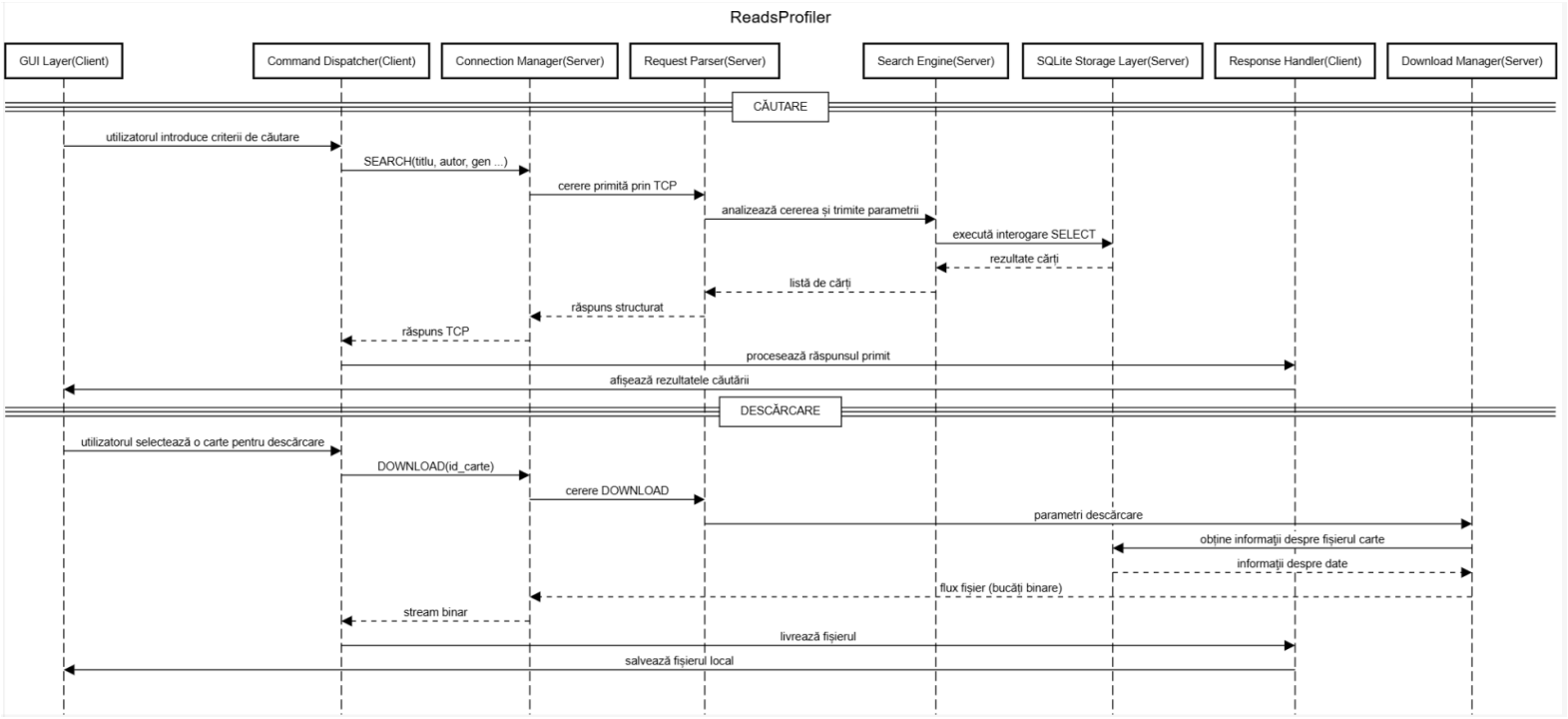

Fig. 1.

4 Protocolul de comunicare

Comunicare dintre cele două părţi componente ale aplicaţiei se realizează prin **TCP**. Se folosesc mesaje text care respectă formatul **COMANDA parametri**.

Protocolul include următoarele tipuri de mesaje trimise de client:

- **search** <criterii>: căutarea cărţilor se bazează pe un criteriu specific(titlu, autor, gen, subgen, an, rating)
- **download** <id_ carte>: descarcă o carte pe baza unui id
- **recommendations** <criterii>: solicită recomandări personalizate pentru fiecare utilizator
- **quit**: închide aplicaţia

Serverul răspunde fiecărui mesaj cu una dintre următoarele comenzi:

- **validate** <comanda>: confirmă primirea comenzii şi procesarea acesteia
- **error** <motiv>: ne spune ce eroare am obţinut şi motivul ei
- **bookList**: returnează informaţiile despre cărţile care îndeplinesc criteriile de căutare
- **dataDownload**: returnează confirmarea descărcării cărţii solicitate
- **dataRecommendations**: returnează recomandările personalizate pentru utilizator
- **notifyRecommendation** <id_carte>: informează utilizatorul despre o recomandare nouă

Comenzile vor fi scrise pe linii separate. În cazul în care o comandă nu este corectă sau validă, serverul va trimite un mesaj de eroare.

5 Scenarii de utilizare

5.1 Scenarii de succes

∗ **Căutarea după criterii multiple**
   a. Utilizatorul accesează aplicaţia şi introduce criterii pentru căutare(titlu, rating, autor, gen, rating)
   b. Clientul trimite cererea de căutare către server, serverul interoghează baza de date SQLite, iar rezultatele căutării sunt retrimise clientului şi afişate pe interfaţa grafică
   c. Cititorul vede o listă de cărţi ale căror caracteristici corespund criteriilor introduse de utilizator

∗ **Descărcarea unei cărţi**
   a. Utilizatorul selectează una sau mai multe cărţi pentru descărcare
   b. Clientul trimite cererea către server, iar acesta trimite fiecare fişier ce se doreşte a fi descărcat
   c. În final, toate cărţile sunt descărcate corect şi salvate local

∗ **Generare de recomandări**
   a. Cititorul cere recomandări pe baza istoricului căutărilor şi descărcărilor
   b. Serverul procesează istoricurile, identifică genurile şi autorii cei mai citiţi de utilizator şi generează recomandări
   c. Utilizatorul primeşte o listă de recomandări

5.2 Scenarii de eşec

* **Căutare cu parametri incorecţi**
  a. Cititorul introduce criteriile de căutare într-o ordine greşită, incorecţi sau incompleţi
  b. Clientul trimite cerere către server cu parametri invalizi
  c. Serverul returnează un mesaj de eroare (error <motiv>) care explică ce s-a greşit, astfel încât ca utilizatorul să poată corecta
* **Descărcare eşuată din cauza unui fişier inexistent**
  a. Utilizatorul încearcă să caute o carte care nu mai există sau introduce un ID greşit
  b. Clientul trimite cererea de descărcare către server, însă acesta eşuează în a găsi cartea
  c. Serverul trimite înapoi un mesaj de eroare care explică de ce nu a putut găsi cartea
* **Recomandări eşuate din cauza lipsei istoricului**
  a. Cititorul cere recomandare, însă nu a căutat sau descărcat nicio carte
  b. Clientul trimite cererea către server, dar acesta nu poate procesa datele
  c. Serverul returnează un mesaj de eroare care spune faptul că nu există informaţiile necesare pentru a face recomandări

6 Concluzii

Aplicaţia **ReadsProfiler** se bazează pe sistemul server-client ce gestionează o librărie online, cu algoritmi de căutare, descărcare şi de generare de recomandări. Sistemul de recomandări ar putea fi îmbunătăţit prin utilizarea unor tehnici de învăţare automată, care să ofere recomandări mai precise pe baza istoricelor de căutări şi descărcări ale utilizatorului. În mod similar, interfaţa grafică ar putea fi extinsă cu opţiuni de filtrare şi sortare mai avansate, iar notificările pentru recomandări ar îmbunătăţi interacţiunea dintre utilizatori şi aplicaţie. Aceste optimizări ar contribui la creşterea performanţei şi scalabilităţii proiectului.

7 Referinţe bibliografice

1. *Computer Network – Course Materials*. Se găsesc la adresa: https://edu.info.uaic.ro/computer-networks/
2. *The Linux Programming Interface*. Disponibil la : The Linux Programming Interface - TCP/IP
3. *TCP Server – Client implementation in C*. Disponibil la: https://www.geeksforgeeks.org/c/tcp-server-client-implementation-in-c/
4. Springer. *Lecture Notes in Computer Science – Author Guidlines*. Disponibil la: https://www.springer.com/gp/computer-science/lncs/conference-proceedings-guidelines
5. SequenceDiagram.org. *Online Sequence Diagram Editor*. Disponibil la: https://sequencediagram.org
6. Cartea: *Computer Networking – A Top – Down Approach*, Kurose Ross