Documentație proiect ReadsProfiler

Olariu Andreea, B4

Universitatea "Alexandru Ioan Cuza", Iași

Rezumat Aplicația ReadsProfiler permite unui utilizator să interacționeze cu o librărie online.

Keywords: TCP · Baza de date · Socket · Fir de execuție · GTK

1 Introducere

Proiectul **ReadsProfiler** constă în realizarea unei aplicații de tip client-server care să permită unui utilizator, reprezentat de client, să acceseze o librărie online, reprezentat de librărie.
În cadrul aplicației, utilizatorul poate să își creeze un cont, să se autentifice într-un cont deja înregistrat în baza de date, să caute și/sau să descarce cărți în funcție de anumite filtre: titlu, autor, isbn, gen, anul publicării, rating. Utilizatorul poate alege un singur filtru sau o combinație dintre acestea.
De asemenea, utilizatorul poate să primească diverse recomandări de cărți pe baza comportamentului său în rețea sau a comportamentului unor utilizatori cu gusturi similare cu el.

2 Tehnologii utilizate

2.1 TCP

Protocolul **TCP** asigură o conexiune full-duplex între client și server, comunicarea realizându-se fără pierderea sau alterarea informației.
Chiar dacă există protocoale care permit o comunicare mai rapidă (de exemplu UDP), TCP asigură că serverul primește cererile de la client în ordine, evitând situațiile de tipul: mai întâi serverul să primească cererea de descărcare după cererea de căutare, acest aspect fiind crucial pentru arhitectura aplicației.
Un al doilea avantaj al protocolului TCP în cadrul acestei aplicații este transmiterea cu acuratețe și fără pierdere a cererilor și răspunsurilor, evitându-se situațiile în care o carte este primită de client parțial sau clientul primește răspuns la o altă cerere decât cea făcută.

2.2 Baza de date relațională

Aplicația utilizează o baza de date relațională, deoarece aceasta este alegerea perfectă pentru a stoca datele pe care librăria trebuie să le manipuleze.
Aceasta este alcătuită din mai multe tabele între care se stabilesc diferite relații (one-to-one, one-to-many) pe baza unor chei străine. Schema bazei este reprezentată în Figura 1 .

2.3 Limbajul SQL

Limbajul SQL este folosit de server în cadrul funcțiilor din biblioteca SQLite pentru a crea baza de date relațională și pentru a manipula datele din tabelele bazei de date conform cu cererile făcute de client.

2.4 SQLite

Biblioteca SQLite oferă un API ușor de utilizat în cadrul aplicațiilor scrise în limbajul C și necesită zero configurare. Cu ajutorul funcțiilor din SQLite, serverul se poate conecta ușor la baza de date, poate să transmită interogări SQL către aceasta și să primească răspunsul la aceste interogări în cadrul funcțiilor de manipulare a răspunsurilor.

2.5 Socket

Comunicarea dintre client și server se realizează prin socket-uri care facilitează comunicare dintre procese aflate pe mașini diferite sau pe aceeași mașină. Din punctul de vedere al programatorului, un socket funcționează ca un descriptor de fișiere și asupra sa pot fi aplicate funcții de scriere și citire specifice librăriei standard C sau API-ului POSIX.
De asemenea, prin API-ul de programare în rețea care utilizează socket-uri se poate realiza ușor protocolul de comunicare TCP între client și server prin funcții specifice precum $bind(), connect(), accept()$.

2.6 Thread

Serverul TCP **concurent** va fi unul **multithreaded**, implementat cu ajutorul mai multor thread-uri.
Folosirea firelor de execuție este motivată de alegerea unui server TCP concurent. Astfel încât, pentru fiecare client care inițiază o conexiune, serverul crează un fir de execuție și toate cererile de la client și răspunsurile de la server vor fi gestionate în cadrul firului de execuție aferent clientului.
Firele de execuție vor fi create și gestionate folosind funcțiile din biblioteca *pthread.h* din API-ul POSIX.

2.7 GTK

Biblioteca GTK este o bibliotecă cu ajutorul căreia se poate realiza o interfață grafică pentru aplicațiile scrise în limbajul C. Am ales această bibliotecă pentru a face interacțiunea utilizatorului cu librăria online mai prietenoasă prin prezența unui meniu.

3 Arhitectura aplicației

3.1 Protocolul aplicației

Conectarea clientului la server se face folosind schema de stabilire a conexiunii specifică protocolului TCP. Serverul fiind într-o stare de *listen*(), în momentul în care un client conectat la IP și PORT corespunzător transmite o cerere de *connect*(), serverul apelează *accept*() pentru acel client și crează un fir de execuție în cadrul căruia primește cererile și răspunde acestora.
De asemenea, în momentul începerii execuției serverului, acesta se conectează la baza de date folosind funcții din API-ul SQLite pentru C din biblioteca *sqlite3.h*.
În momentul conectării la server, clientul este întâmpinat de un meniu. În acestă etapă, clientul poate alege să:

(1) să se înregistreze în aplicație
(2) să se conecteze în aplicație
(3) să înceapă căutarea cărților dorite
(4) să vizualizeze recomandări
(5) să se deconecteze din aplicație
(6) să părăsească aplicația
(7) să ofere un rating pentru o carte
(8) să-și steargă contul din aplicație

Clientul alege opțiunea pe care și-o dorește folosind o interfață grafică. Opțiunea aleasă va fi transmisă serverului prin socket și serverul va răspunde, după caz, clientului tot prin intermediul meniului grafic.

Alegerea (1):
Clientul este invitat să introducă un username și o parolă. După ce clientul le-a introdus, vor fi trimise serverului prin socket sub forma unei structuri populate cu aceste valori (de tipul username: valoare, parolă: valoare) împreună cu o cerere de înregistrare și serverul va verifica dacă aceste informații se află sau nu în baza de date, mai precis în tabela *users*.
În cazul în care acestea nu sunt prezente în tabel, informațiile introduse de client vor fi adăugate de către server în baza de date și serverul va transmite clientului un mesaj *"Înregistrare cu succes"*. În caz contrar, serverul transmite un mesaj *"Utilizator deja existent"*.

Alegerea (2):
Clientul trimite cererea de conectare împreună cu o structură populată cu username-ul și parola introduse. Informațiile vor fi transmise serverului și serverul verifică dacă utilizatorul se găsește în tabela *users*.
În caz afirmativ, serverul transmite clientului un mesaj *"Conectare cu succes"* și structura globală care memorează informațiile despre client este actualizată. De asemenea, aceste informații sunt memorate și de către server, într-o structură cu informațiile despre thread-ul corespuzător clientului. Altfel, serverul transmite clientului mesajul *"Acest username nu există"*.

Alegerea (3):
Criteriile de căutare sunt: titlu, numele autorului, prenumele autorului, isbn, gen, anul publicării, rating și combinații ale acestora. Clientul transmite serverului o structură populată cu preferințele sale (de tipul cheie: valoare), unde criteriile neutilizate sunt populate cu valoare $NULL$, împreună cu cererea de căutare.
Folosind interogări în limbajul SQL și executându-le utilizând API C al bibliotecii SQLite, serverul primește răspunsul la interogare și trimite clientului rezultatul.
Serverul transmite o cerere clientului prin care întreabă dacă clientul dorește să descarce vreo carte. În caz afirmativ, apare meniul prin care clientul poate să introducă titlul unei cărți pe care dorește să o descarce. Dacă utilizatorul este conectat, clientul transmite serverului o cerere de descărcare a cărții. Altfel, apare mesajul *Descărcare eșuată. Vă rugăm să vă conectați mai întâi."*
De asemenea, când clientul caută sau descarcă o carte, serverul actualizează în tabelul din baza de date care ține evidența cărților cele mai populare informațiile, *top_books_by_genre_popularity*, dar și atributele *downloaded_by* și *searched_by* din tabela *books*.
Cărțile propriu-zise (fișierele care conțin octeții ce alcătuiesc conținutul cărților) sunt stocate pe mașina de pe care rulează serverul într-un director, iar programul serverului are acces la acest director și are permisiuni de citire asupra conținutului acestuia.

Alegerea (4):
Clientul transmite serverului cererea de recomandări. Dacă clientul este conectat în aplicație, serverul face o cerere la baza de date baza unui algoritm de recomandare descris mai jos și transmite recomandările clientului.
Utilizatorul poate, apoi, să aleagă o carte recomandată și să o descarce dacă este conectat, asemănător cu ce se întâmplă atunci când clientul alege să caute o carte.

Alegerea (5):
Clientul transmite serverului cererea de deconectare. Dacă utilizatorul este conectat, atunci serverul transmite clientului un mesaj *"Deconectare cu succes"*. De asemenea, și structura globală care stochează date despre client este reinițializată. În caz contrar, serverul transmite clientului un mesaj *"Nu sunteți conectat la aplicație"*.

Alegerea (6):
Clientul transmite serverului o cerere de închidere a aplicației. Se va închide clientul care a inițiat comunicarea, dar serverul rămâne deschis.

Alegerea (7):
În momentul alegerii acestei opțiuni, clientul este invitat să introducă un titlu și un rating de la 0 la 10 pentru cartea cu acel titlu. Aceste informații sunt transmise serverului și câmpurile *rating* și *rated_by* din tabela *books* sunt actualizate, dar și topul cărților pe baza ratingului, stocat în tabela *top_books_by_genre_rating*.

Alegerea (8):
Clientul trimite cererea de ștergere a contului împreună cu o structură populată cu username-ul și parola introduse. Informațiile vor fi transmise serverului și serverul verifică dacă utilizatorul se găsește în tabela *users*.
În caz afirmativ, serverul transmite clientului un mesaj *"Am șters contul!"* și transmite către baza de date o cerere de ștergere a informațiilor despre utilizatorul cu aceste informații din baza de date. Altfel, serverul transmite clientului mesajul *"Acest username nu există"*.

De asemenea, o posibilitate de a se întoarce la meniul inițial rămâne valabilă pe tot parcursul utilizării aplicației.

3.2 Baza de date

Baza de date este organizată conform figurii 1. Legătura dintre server și baza de date este permisă de limbajul SQL și de utilizarea bibliotecii SQLite.

Tabela *books* conține informații despre cărțile care pot fi căutate și/sau descărcate. Fiecare carte poate aparține unui gen, de aceea există relația între tabela *books* și *genres*.

Tabela *authors* conține detalii despre autorii cărților. Fiecare autor poate scrie una sau mai multe cărți, aceste cărți definind genurile abordare de autor.

Tabela *users* reține informații despre toți utilizatorii autentificați în aplicație

Tabelele *searched_books*, respectiv *downloaded_books*, conțin informații despre cărțile căutate, respectiv descărcate de un anumit utilizator identificat de *user_id*.

Tabela *top_books_by_genre_rating* conține informații despre cărțile, identificate prin isbn, care fac parte din topul cărților pentru fiecare gen în funcție de ratingul acordat acestora de utilizatori.

Tabela *top_books_by_genre_popularity* conține informații despre cărțile, identificate prin isbn, care fac parte din topul cărților pentru fiecare gen în funcție de popularitatea acestora în rândul utilizatorilor, adică numărul de descărcări, respectiv căutari ale acestora.

Tabela *book_genre* conține informații despre genurile în care se încadrează o carte.

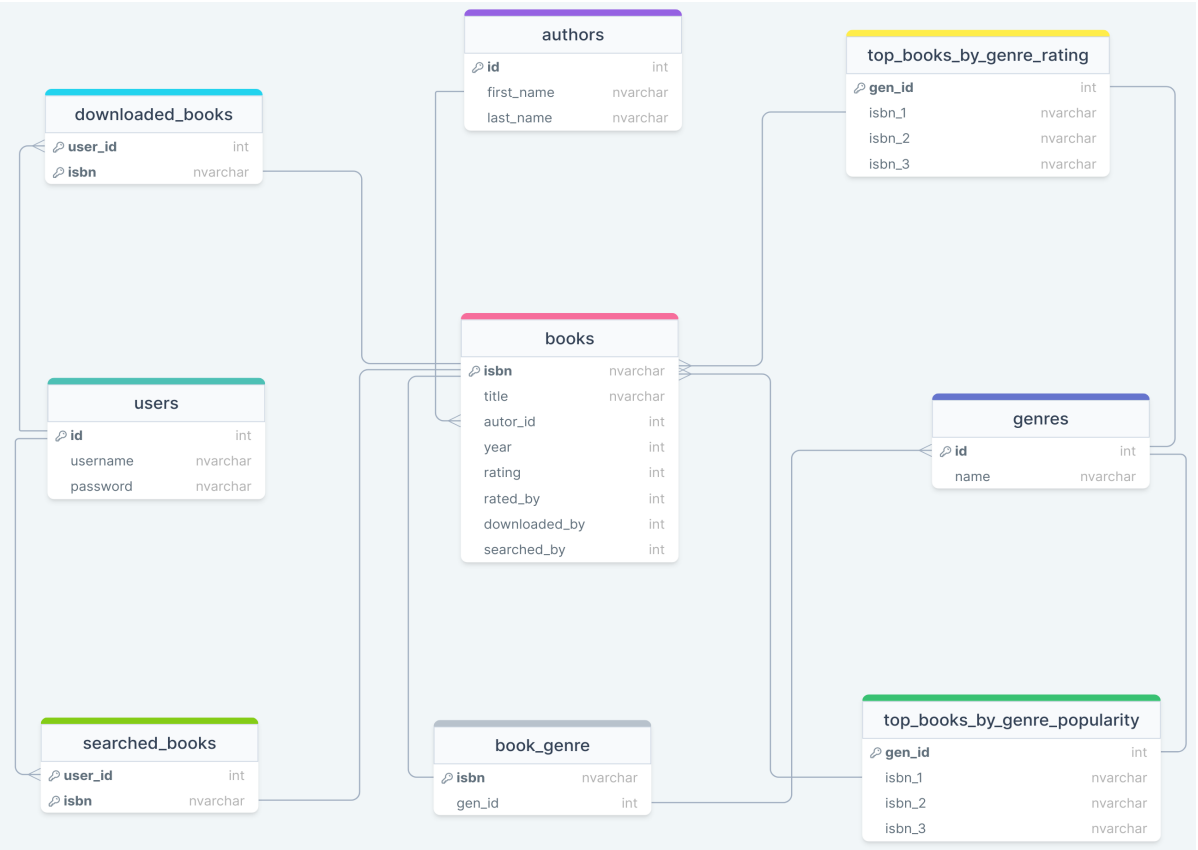

Figura 1. Baza de date

3.3 Diagrama aplicației

În figura următoare se poate observa comportamentul aplicației pentru primele 6 alegeri pe care utilizatorul le poate face în meniul inițial.
Se pot observa atât arhitectura pachetelor trimise, cât și interogările pe care serverul le face bazei de date, dar și posibilele răspunsuri către client.

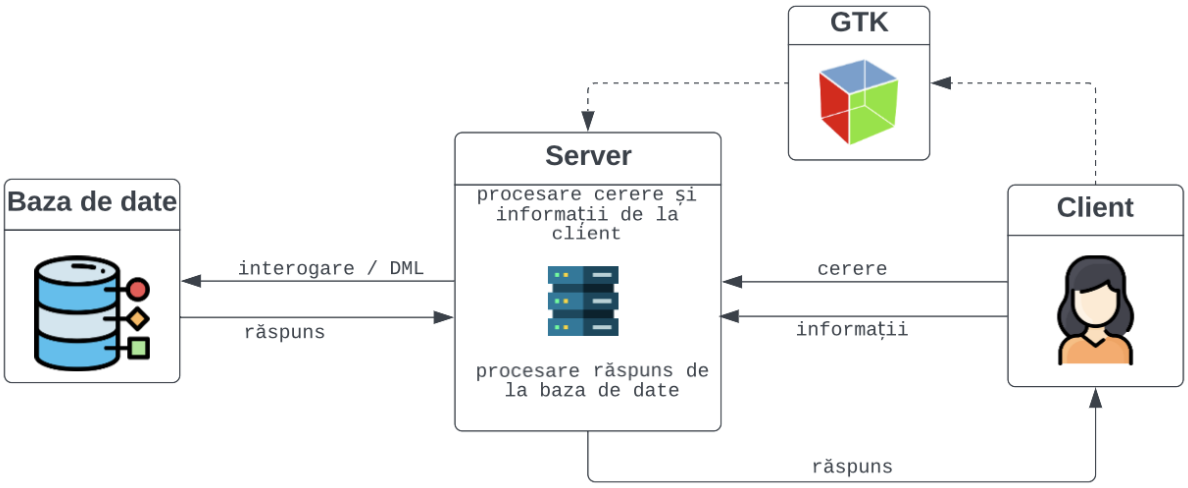

Figura 2. Diagrama aplicației

3.4 Pachete

Pachetele transmise de la client la server, respectiv de la server la client, vor fi prefixate de numărul de octeți pentru ca serverul, respectiv clientul să știe câți octeți să citească din socket.
De asemenea, pachetele transmise de la client la server vor fi prefixate și de un identificator de opțiune pentru ca serverul să cunoască structura pachetului (pachetul din cererea de conectare are o structură diferită față de pachetul din cererea de deconectare) și să citească datele în mod corespunzător.

4 Implementare

Compilarea serverului și a clientului se face folosind următoarele comenzi. Este necesară instalarea bibliotecilor Sqlite3 și GTK3.

```
gcc `pkg-config --cflags gtk+-3.0` -o client client.c `pkg-config --libs gtk+-3.0`
gcc server.c -o server -lpthread -lsqlite3 -std=c99
```

4.1 Porțiuni de cod

Funcția care permite afișarea meniului de Sign Up:

```
1  static gboolean SignUpMenu(GtkButton *btn, gpointer p) {
2      // 1 = sign up
3      int opt = 1;
4      if (write(sd, &opt, sizeof(int)) != 4) {
5          perror("[client] Eroare la write in sign up!\n");
6          exit(1);
7      }
8
9      remove_children(GTK_CONTAINER(window));
10
11     GtkEntry *username = GTK_ENTRY(gtk_entry_new());
```

```
    GtkEntry *password = GTK_ENTRY(gtk_entry_new());
    gtk_entry_set_placeholder_text(
        GTK_ENTRY(username), "username...");
    gtk_entry_set_placeholder_text(
        GTK_ENTRY(password), "password...");
    gtk_entry_set_visibility(password, FALSE);

    user_data.username = username;
    user_data.password = password;
    GtkButton *sign_up_btn = GTK_BUTTON(gtk_button_new_with_label("Sign up"));
    g_signal_connect(sign_up_btn, "button-press-event", G_CALLBACK(
    user_info_interaction), NULL);

    GtkButton *back = GTK_BUTTON(gtk_button_new_with_label("Back"));
    g_signal_connect(back, "button-press-event", G_CALLBACK(
    BackUserInfoInteraction), NULL);

    GtkBox *wrapper = GTK_BOX(gtk_box_new(GTK_ORIENTATION_VERTICAL, 0));
    gtk_container_add(GTK_CONTAINER(wrapper), GTK_WIDGET(username));
    gtk_container_add(GTK_CONTAINER(wrapper), GTK_WIDGET(password));
    gtk_container_add(GTK_CONTAINER(wrapper), GTK_WIDGET(sign_up_btn));
    gtk_container_add(GTK_CONTAINER(wrapper), GTK_WIDGET(back));

    gtk_container_add(GTK_CONTAINER(window), GTK_WIDGET(wrapper));
    gtk_widget_show_all(GTK_WIDGET(window));
}
```

Functia care transmite serverului filtrele de cautare a cărților:

```
static gboolean search_book(GtkButton *btn, gpointer data) {
    struct book_search book;
    bzero(&book, sizeof(book));

    strcpy(book.title, "");
    strcpy(book.author_first_name, "");
    strcpy(book.author_last_name, "");
    strcpy(book.isbn, "");
    strcpy(book.genre, "");
    strcpy(book.rating, "");
    strcpy(book.publish_year, "");

    // Extragerea filtrelor de cautare
    if (GTK_IS_ENTRY(book_data.title) == TRUE && gtk_entry_get_text_length(
    GTK_ENTRY(book_data.title)) != 0) {
        strcpy(book.title, gtk_entry_get_text(GTK_ENTRY(book_data.title)));
        strtrim(book.title);
    }
    if (GTK_IS_ENTRY(book_data.author_first_name) == TRUE &&
    gtk_entry_get_text_length(GTK_ENTRY(book_data.author_first_name)) != 0) {
        strcpy(book.author_first_name, gtk_entry_get_text(GTK_ENTRY(book_data.
    author_first_name)));
        strtrim(book.author_first_name);
    }
    if (GTK_IS_ENTRY(book_data.author_last_name) == TRUE &&
    gtk_entry_get_text_length(GTK_ENTRY(book_data.author_last_name)) != 0) {
        strcpy(book.author_last_name, gtk_entry_get_text(GTK_ENTRY(book_data.
    author_last_name)));
        strtrim(book.author_last_name);
    }
    if (GTK_IS_ENTRY(book_data.isbn) == TRUE && gtk_entry_get_text_length(
    GTK_ENTRY(book_data.isbn)) != 0) {
        strcpy(book.isbn, gtk_entry_get_text(GTK_ENTRY(book_data.isbn)));
        strtrim(book.isbn);
    }
```

```
30      if (GTK_IS_ENTRY(book_data.genre) == TRUE && gtk_entry_get_text_length(
    GTK_ENTRY(book_data.genre)) != 0) {
31          strcpy(book.genre, gtk_entry_get_text(GTK_ENTRY(book_data.genre)));
32          strtrim(book.genre);
33      }
34      if (GTK_IS_ENTRY(book_data.publish_year) == TRUE && gtk_entry_get_text_length(
    GTK_ENTRY(book_data.publish_year)) != 0) {
35          strcpy(book.publish_year, gtk_entry_get_text(GTK_ENTRY(book_data.
    publish_year)));
36          strtrim(book.publish_year);
37      }
38      if (GTK_IS_ENTRY(book_data.rating) == TRUE && gtk_entry_get_text_length(
    GTK_ENTRY(book_data.rating)) != 0) {
39          strcpy(book.rating, gtk_entry_get_text(GTK_ENTRY(book_data.rating)));
40          strtrim(book.rating);
41      }
42
43      // Scriem spre server filtrele
44      if (-1 == write(sd, &book, sizeof(book))) {
45          perror("[client]Eroare la write() spre server!\n");
46          exit(1);
47      }
48
49      char return_msg[301] = "";
50      read_string_from_fd(sd, return_msg, "[client]Eroare la citirea de la server!")
    ;
51      strtrim(return_msg);
52
53      int download;
54      if (read(sd, &download, 4) != 4) {
55          perror("[client]Eroare la read() de la server!\n");
56          exit(1);
57      }
58
59      if (download == 1) {
60          AskToDownloadBookMenu(return_msg);
61      } else {
62          ShowMessage(return_msg);
63      }
64  }
```

Conectarea la baza de date:

```
1       int rc = sqlite3_open("../database/reads_profiler.db", &db);
2       if (rc != SQLITE_OK) {
3           fprintf(stderr, "Cannot open database: %s\n", sqlite3_errmsg(db));
4           sqlite3_close(db);
5           return 1;
6       }
```

Funcția callback a creării thread-ului:

```
1   static void *thFunc(void *arg) {
2       struct thData tdL;
3       tdL = *((struct thData *)arg);
4       printf("[Thread %d] - Asteptam mesajul...\n", tdL.idThread);
5       fflush(stdout);
6       pthread_detach(pthread_self());
7       get_option_from_client((struct thData *)arg);
8       close((intptr_t)arg);
9       return (NULL);
10  };
```

Funcțiile de citire/scriere de la server, respectiv client:

```
1   int read_string_from_fd(int fd, char* storage, const char err_msg[]) {
```

```
    int cod_term, index = 0, nr_bytes;
    char ch;
    read(fd, &nr_bytes, sizeof(int));
    while (nr_bytes--)
    {
        cod_term = read(fd, &ch, 1);
        if(cod_term <= 0) {
            break;
        }
        storage[index++] = ch;
    }
}
void write_string_to_fd(int fd, char to_write[], char err_msg[]) {
    int len = strlen(to_write);
    if(write(fd, &len, sizeof(int)) < 0) {
        perror(err_msg);
        exit(0);
    }
    if(len != write(fd, to_write, len)) {
        perror(err_msg);
        exit(0);
    }
}
```

Adăugarea unei cărți în tabela *downloaded_books*

```
void add_book_in_downloaded(struct user current_user, char *book) {
    char query[501] = "";
    strtrim(current_user.username);
    strtrim(book);
    sprintf(query,
            "INSERT INTO downloaded_books VALUES((SELECT id FROM users WHERE
    username='%s'),"
            "(SELECT isbn FROM books WHERE LOWER(title)=LOWER('%s')));",
            current_user.username, book);

    int rc = sqlite3_prepare_v2(db, query, -1, &res, 0);
    if (rc) {
        fprintf(stderr, "Eroare!: %s\n", sqlite3_errmsg(db));
        sqlite3_close(db);
        exit(0);
    }
    int step = sqlite3_step(res);
    sqlite3_finalize(res);
    sprintf(query,
            "UPDATE books SET downloaded_by = downloaded_by + 1 WHERE "
            "isbn=(SELECT isbn FROM books WHERE LOWER(title)=LOWER('%s'));",
            book);
    modify_table_query(query);
}
```

4.2 Strategie algoritm de recomandare

În Figura 1 se poate vedea structura pe tabele a bazei de date. În tabela *top_books_by_genre_popularity* se memorează primele 3 cărți cele mai populare pentru un anumit gen specificat prin atributul *gen_id*, iar în tabela *top_books_by_genre_rating* se memorează primele 3 cărți cu ratingul cel mai mare pentru un atributul *gen_id*.
Pentru a oferi recomandări clientului pe baza gusturilor sale, pentru acel client identificat prin *id* din tabela *users* vom parcurge tabelele *downloaded_books* și *searched_books* și vom considera doar cărțile descărcate, respectiv căutate, de utilizatorii care au $user\_id = id$. Pentru acele cărți, vom contoriza de câte ori se repeta un anumit gen și, în funcție de acest contor vom face un top al genurilor preferate de utilizator. Pentru fiecare gen din top, vom recomanda un număr de cărți din tabelele *top_books_by_genre_popularity* și *top_books_by_genre_rating* proporțional cu locul în care genul se situează în top.

De asemenea, în cazul în care utilizatorul nu a frecventat aplicația atât de des, la topul cărților pe baza preferințelor se adaugă un top default care conține cărțile cu ratingul cel mai mare pentru toate genurile. Acest algoritm a fost implementat folosind operații precum join-uri, subinterogări sau putem folosi alte tabele suplimentare.

4.3 Meniul realizat folosing GTK

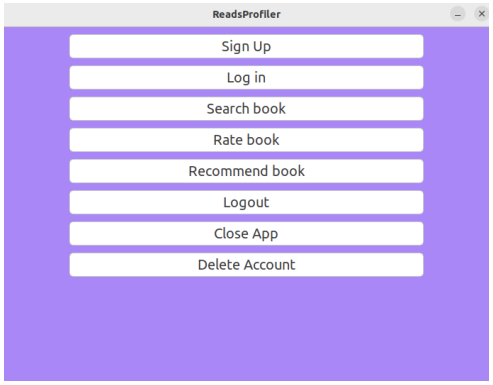

Figura 3. Meniul principal

Figura 4. Meniul de Sign Up

Figura 5. Meniul de căutare cărților

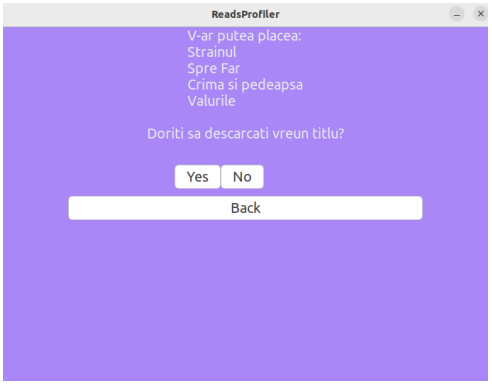

Figura 6. Meniu ce oferă recomadări pe baza algoritmului de recomadare

4.4 Scenarii de utilizare

Un scenariu de utilizare este redat în Figura 7 în care un utilizator conectat, având username-ul ”andreea” transmite o cerere de căutare a unei cărți cu titlul ”Străinul” și autorul ”Albert Camus” către server. Serverul transmite o interogare bazei de date, iar baza de date transmite serverului un string ce conține titlurile cărților care respectă filtrele aplicate de client, în cazul acesta stringul care conține titlul ”Străinul”, deci cartea căutată se află în baza de date. Serverul transmite stringul mai departe clientului, iar utilizatorul face o cerere de descărcare a cărții.
Serverul transmite bazei de date o interogare prin care verifică dacă utilizatorul este conectat și, cum utilizatorul cu username-ul ”andreea”, care a trimis cererea de descărcare, este conectat, baza de date transmite răspunsul utilizatorului serverului și serverul transmite cartea utilizatorului (ca un șir de bytes). De asemenea, serverul transmite bazei de date să actualizeze datele din $downloaded\_books$ și $searched_books$.

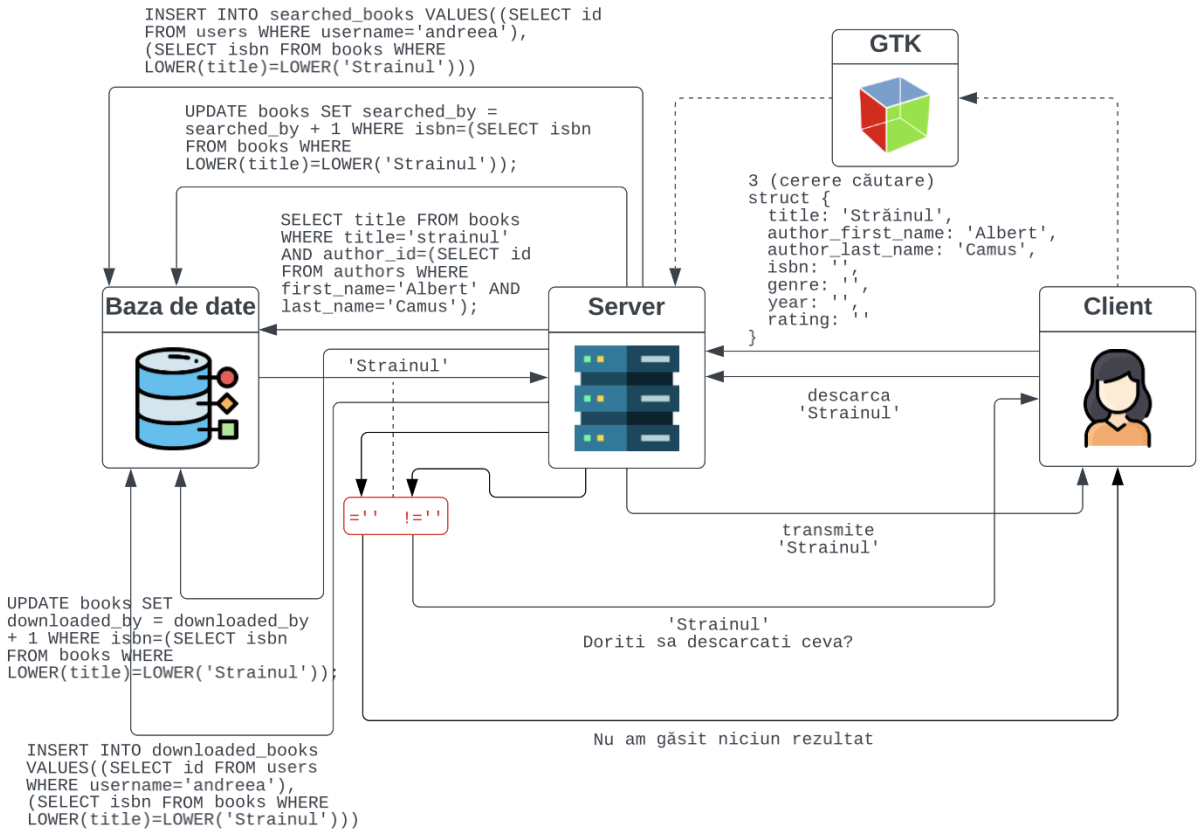

Figura 7. Scenariu utilizare

5 Concluzii

Aplicația ReadsProfiler folosește elemente de grafică care permit utilizatorului să transmită diferite cereri descrise serverului. Pachetele transmise diferă în structură de la opțiune la opțiune și, pentru fiecare client, serverul face un nou fir de execuție pentru a răspunde la cererile clientului respectiv.
Informațiile despre cărți, utilizatori, autori și genuri sunt stocate într-o bază de date relațională manipulată cu ajutorul limbajului SQL și a bibliotecii SQLite, iar cărțile propiu-zise sunt stocate într-un director aflat pe mașina pe care rulează serverul, acesta având drepturi de citire asupra fișierelor din acel director.

Îmbunătățire
O îmbunătățire considerabilă ar aduce-o un algoritm bazat pe învățare automată care să prezică cu o mai mare acuratețe recomandărilor pe care serverul le poate face utilizatorilor.
O altă îmbunătățire a aplicației ar fi utilizarea unui protocol specializat pentru schimbul de fișiere precum FTP, SFTP, precum și encriptarea parolelor.

Bibliografie

1. https://docs.gtk.org/gtk3/index.html
2. https://en.wikibooks.org/wiki/Regular_Expressions/POSIX_Basic_Regular_Expressions
3. https://www.tutorialspoint.com/sqlite/
4. https://zetcode.com/db/sqlitec/
5. https://www.sqlite.org/c3ref/intro.html
6. https://profs.info.uaic.ro/ bd/wiki/index.php/Pagina_principală
7. https://profs.info.uaic.ro/ computernetworks/cursullaboratorul.php
8. https://ro.wikipedia.org/wiki/SQLite
9. https://www.gtk.org
10. https://www.geeksforgeeks.org/regular-expressions-in-c/