

ReadsProfiler

Băbuță Rareș-Mihai II A7

15 Ianuarie, 2016

1 Introducere

1.1 Despre

ReadsProfiler este o aplicație ce oferă acces la o librărie online. Cărțile vor putea fi căutate de către utilizatori după diferite criterii și mai apoi unui utilizator i se va oferi posibilitatea de a descărca sau de a da rating la o carte. De asemenea, fiecare client va primi o serie de recomandări pe baza activității sale, dar și a altor utilizatori, în aplicație.

1.2 Motivație

Interesul pentru cărți și pentru citit a fost dintotdeauna unul primordial. Însă, odată cu evoluția domeniului IT din ce în ce mai multe persoane au renunțat la metoda clasică de a păstra cărțile în biblioteca personală și s-au orientat către **eBooks**, cărțile electronice pe care le citesc pe ecran, pe orice ecran – fie că vorbim de un smartphone, tabletă, eReader sau computer. De aceea, această aplicație are scopul de a oferi la dispoziția utilizatorilor o gamă largă de cărți electronice, ce vor putea fi descărcate și citite în voie pe dispozitivul preferat.

1.3 Utilizare

Pentru a putea profita de beneficiile oferite de aplicație, un client va trebui prima dată să se **înregistreze**, oferind o serie de informații (username, parolă, e-mail, preferințe etc.). Acest pas va fi făcut o singură dată. După aceea, de fiecare dată când un client va dori să folosească aplicația, el va trebui să se **autentifice**, furnizând username-ul și parola oferite la înregistrare. Odată autentificat, fiecare utilizator va putea **căuta** cărți după criterii diverse (gen, autor, titlu, anul apariției, ISBN, rating). Aplicația va oferi ca rezultat al căutării o serie de cărți ce vor putea fi accesate, **descărcate** și **evaluate** de fiecare utilizator în parte. Pe baza acestor informații (căutări, rezultate accesate, descărcări) fiecărui utilizator i se va realiza un **“profil”** ce va fi folosit de aplicație pentru a realiza **recomandări** de cărți. Acest profil va fi actualizat în mod continuu, cu cât activitatea clientului va crește. Astfel, sistemul de recomandare se va **îmbunătăți**, cărțile oferite ca și sugestii unui utilizator fiind din ce în ce mai aproape de gusturile sale.

2 Arhitectură

Modelul **client-server** partajează procesarea între furnizorii de servicii numiți servere și elementele care solicită servicii, numite clienți.

În cazul aplicației ReadsProfiler, clienții vor fi utilizatorii ce vor solicita diferite servicii (înregistrare, autentificare, căutare, descărcare etc.) server-ului. Serverul primește aceste cereri, accesează o bază de date și întoarce rezultatele corespunzătoare fiecărui tip de serviciu cerut de un anumit utilizator.

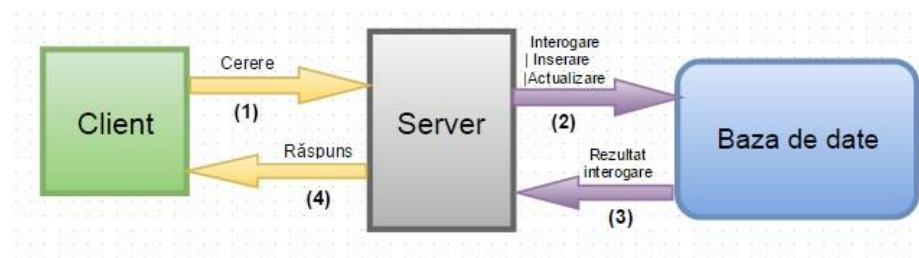


Fig. 1. Diagrama generală a aplicației

Pentru a putea realiza o cerere de tip căutare, accesare, descărcare de carte, cât și pentru a evalua cărțile, fiecare client al aplicației trebuie să fie autentificat. Altfel, singurele cereri pe care fiecare utilizator le poate trimite serverului sunt cele de înregistrare și autentificare.

Cererea de înregistrare constă în completarea datelor necesare de către utilizator. Acestea sunt trimise din client la server care interoghează baza de date pentru a verifica dacă nu exista deja un utilizator cu același nume și în caz negativ, creează noul profil.

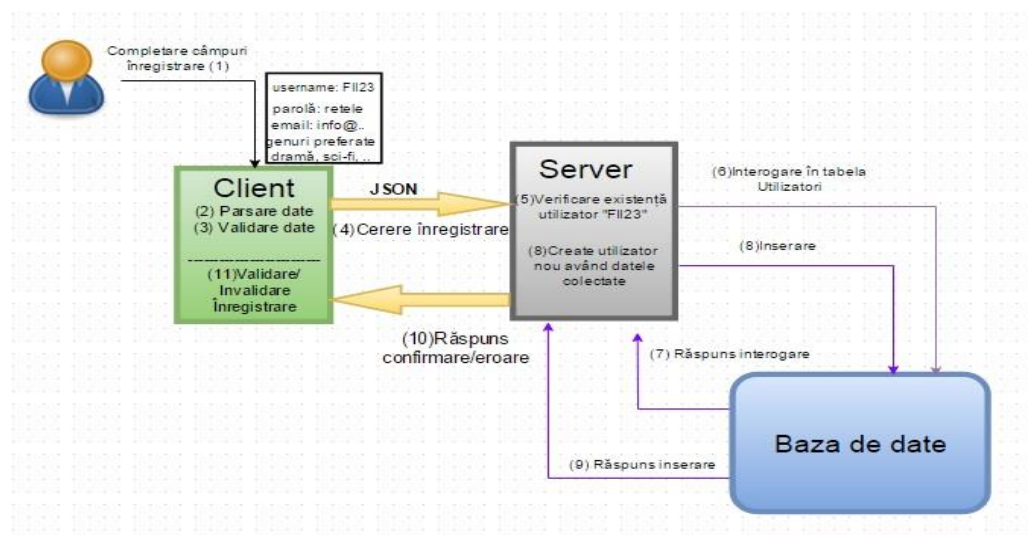


Fig. 2. Serviciul înregistrare

Odată înregistrat, un client se va putea autentifica pe baza numelui de utilizator și a parolei. Acest lucru se realizează printr-o cerere de tip autentificare adresată serverului. Acesta consultă baza de date pentru a valida datele și trimite răspunsul înapoi clientului. În cazul unei erori, acesta înștiințează utilizatorul că datele nu corespund, altfel se realizează procesul de autentificare, se setează flag-ul corespunzător și de acum încolo utilizatorul va putea folosi restul serviciilor oferite de aplicație.

În același timp, pe lângă autentificare, serverul oferă și serviciul recomandare de cărți. Având la dispoziție profilul utilizatorului, se realizează multiple interogări la baza de date. Odată extras un set de cărți, se calculează cele mai bune sugestii ce pot fi oferite utilizatorului respectiv pe baza algoritmului de recomandare ce va fi discutat mai jos, în secțiunea “Detalii de implementare”.

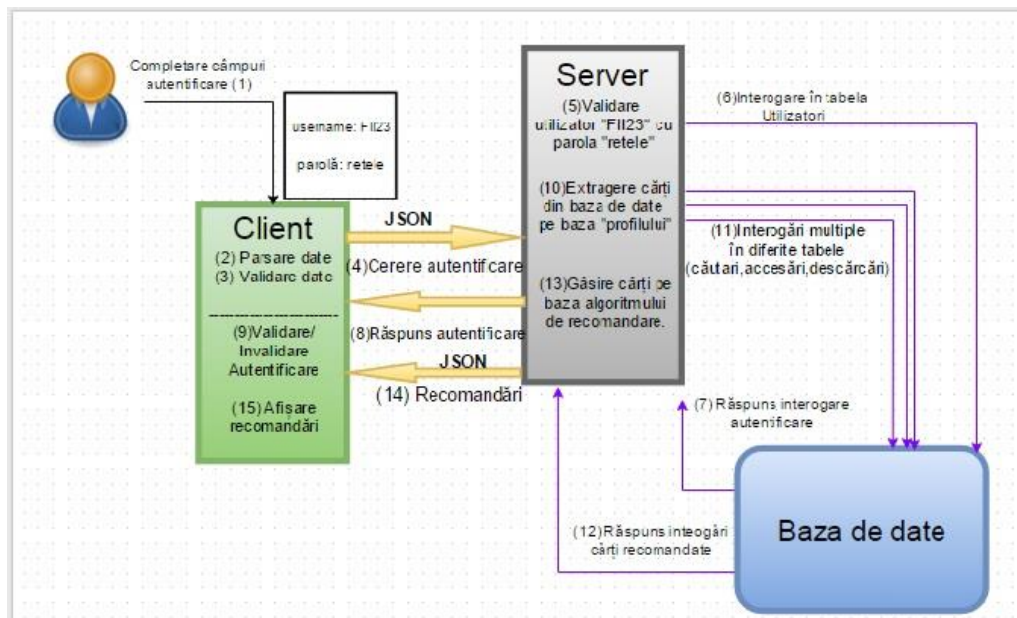


Fig. 3. Serviciul autentificare + Serviciul Recomandări

Serviciile Căutare, Accesare, Descărcare carte etc. sunt asemănătoare serviciului Autentificare, constând în comunicare de tip cerere-interogare-răspuns. De asemenea, dacă în timpul unei sesiuni au fost colectate un număr consistent (stabilit implicit în server) de noi informații legate de gusturile utilizatorului, în timpul oricărei cereri adresate Serverului se va apela (ca și la autentificare) serviciul Recomandare (va fi setat un flag în JSON-ul trimis de la client ce va înștiința serverul să apeleze serviciul).

3 Tehnologii folosite

3.1 TCP/IP

Deoarece sistemul se dorește a fi unul fiabil, cu siguranța că datele ajung la destinație intacte, în ordinea în care au fost trimise, protocolul utilizat în aplicație este TCP/IP. TCP/IP vine de la Protocolul de Control al Transmisiei/Internet Protocol și este principala metodă de comunicare din cadrul rețelelor, fiind alcătuită din mai multe protocoale. Protocolul de control al transmisiei este unul folosit de obicei de aplicații care au nevoie de confirmare de primire a datelor. Efectuează o conectare virtuală full duplex între două puncte terminale, fiecare punct fiind definit de către o adresa IP și de către un port TCP.

3.2 Socketuri

Comunicarea server-client se va realiza folosind socketuri. Un socket este un punct terminal de comunicare între două sisteme din rețea. El este o combinație de adresă IP și Port. Socketurile pot fi folosite în diferite limbaje, însă în acest proiect el a fost folosit în forma sa pură (limbajul C).

3.3 Unix

Programarea în C se va realiza pe un sistem de operare Unix. Sistemul de operare Unix este folosit pe scară largă atât pentru servere cât și pentru stații de lucru. Mediul de dezvoltare Unix și modelul de programare client-server au fost esențiale în dezvoltarea Internetului și trecerea de la sistemele de calculatoare individuale la sistemele de calculatoare în rețea.

3.4 MySQL

Fiind o librărie online, aplicația va trebui să stocheze un număr foarte mare de date, atât pentru cărți, autori, cât și pentru căutările, descărcările și ratingurile realizate de un utilizator. Din acest motiv este necesară utilizarea unei baze de date. MySQL este un sistem de gestiune a bazelor de date relaționale. Se poate folosi împreună cu limbajul C, folosind un API specific.

3.5 Qt

Qt este un framework ce permite dezvoltarea de aplicații software ce folosesc o interfață grafică (GUI). Folosit pentru a realiza partea de Front-End a aplicației adică realizează unei interfețe grafice user-friendly ce va facilita interacțiunea utilizator-client.

4 Detalii de implementare

După cum am precizat mai sus, partea de comunicare în rețea va fi realizată în C folosind socketuri.

```
int sd = socket(AF_INET, SOCK_STREAM, 0);
```

AF_INET - specifică familia de protocoale cu care va comunica socketul și anume IPv4.

SOCK_STREAM – transmisie sigură, orientată conexiune

0 – selectarea protocolului implicit corespunzător (IPPROTO_TCP)

4.1 Client

Conectarea la server se va realiza folosind primitiva connect.

```
connect (sd, (struct sockaddr *) &server,  
        sizeof (struct sockaddr))
```

Odată stabilită conexiunea, se va trimite structura către server, conținând datele colectate de la utilizator, sub forma unor perechi cheie : valoare. Aceste date vor fi trimise împreună cu un caz aferent serviciului.

Trimiterea datelor către server:

```
Cautare::Cautare()  
{  
    caz = 3;  
    bzero(&date, sizeof(detaliiCautare));  
}  
bool Cautare::trimite_datele_la_server(char *nume)  
{  
    if(conectat == 0){  
        conectare_la_server();  
    }  
    if(write(sd, &caz, sizeof(int)) < 0){  
        conectat = 0;  
        return 0;  
    }  
    if(write(sd, &date, sizeof(detaliiCautare)) < 0){  
        conectat = 0;  
        return 0;  
    }  
}
```

Primirea datelor de către server:

```
detaliiCautare cautare;
bzero(&cautare, sizeof(detaliiCautare));
if(read(client_descriptor, &cautare,
        sizeof(detaliiCautare)) < 0)
{
    perror("Eroare la citirea datelor in serviciul
           cautare.\n");
}
char username[50];
if(read(client_descriptor, &username, 50) < 0)
{
    perror("Eroare la citirea numelui de utilizator
           in serviciul cautare cautare.\n");
}
```

Odată primit răspunsul înapoi de către client, de la server, se vor procesa datele și se va înștiința utilizatorul prin afișarea lor în interfața grafică.

4.2 Server

Serverul este cel ce oferă servicii clienților, la diferite porturi. Clienții vor fi tratați concurent folosind threaduri:

```
if ( (client = accept (sd, (struct sockaddr *) &from,
                      &length)) < 0)
{
    perror ("[server]Eroare la accept().\n");
    continue;
}
td=(struct thData*)malloc(sizeof(struct thData));
td->idThread=i++;
td->cl=client;

pthread_create(&th[i], NULL, &treat_client, td);
```

```

static void *treat_client(void * arg)
{
    struct thData tdL;
    tdL= *((struct thData*)arg);
    printf ("[thread] S-a conectat clientul %d...\n",
            tdL.idThread);
    pthread_detach(pthread_self());
    int caz;
    bool clientulEsteConectat = true;
    while(clientulEsteConectat)
    {
        if(read(tdL.cl, &caz,sizeof(int)) <= 0){
            printf("[Thread %d]\n",tdL.idThread);
            perror ("Eroare la read() de la client pt
                    caz. Se inchide conexiunea...\n");
            close ((long)arg);//inchide conexiunea
            return(NULL);
        }
        else
        {
            switch(caz)
            {
            case 0:
                printf("Am inchis conexiunea pentru
                        clientul %d.\n",tdL.idThread);
                close ((long)arg);//inchide conexiunea
                clientulEsteConectat = false;
                return(NULL);
            case 1:
                printf("Se incearca autentificarea pentru
                        clientul %d.\n",tdL.idThread);
                autentificare(tdL.cl);
                break;
            case 2:
                printf("Se incearca inregistrarea pentru
                        clientul %d.\n",tdL.idThread);
                inregistrare(tdL.cl);
                break;
            case 3:
                printf("Se incearca cautarea de carti
                        pentru clientul %d.\n",tdL.idThread);
                cautare(tdL.cl);
                break;
            }
        }
    }
}

```

```

        case 4:
            printf("Se incearca recomandarea de carti
                pentru clientul %d.\n",tdL.idThread);
            recomandare_carti(tdL.cl);
            break;
        case 5:
            printf("Se incearca Upload de carte
                pentru clientul %d.\n",tdL.idThread);
            upload_carte(tdL.cl);
            break;
        case 6:
            printf("Se incearca descarcarea
                detaliilor unei coperti pentru
                clientul %d.\n",tdL.idThread);
            trimite_detalii_la_client(tdL.cl);
            break;
        case 7:
            printf("Se incearca descarcarea unei
                continut de carte pentru clientul
                %d.\n",tdL.idThread);
            trimite_continut_la_client(tdL.cl);
            break;
        case 8:
            printf("Se incearca votarea unei carti de
                catre clientul %d.\n",tdL.idThread);
            voteaza_carte(tdL.cl);
            break;
        case 20:
            printf("Clientul %d a parasit serviciul
                inregistrare.\n",tdL.idThread);
            break;
    }
}
}
}

```


4.3 MySQL

Tot în funcția `apeleaza_serviciul` se vor realiza operațiunile asupra bazei de date.

Inserare :

```
sprintf(inregistrare, "INSERT INTO utilizatori VALUES ("%s" ,
"%s" , "%s", "%s");", register_request.username, regis-
ter_request.parola, register_request.email, register_re-
quest.profil);
```

```
if(mysql_query(db, inregistrare)) db_error(db);
```

Interogare :

```
sprintf(interogare, "SELECT * FROM utilizatori WHERE
(username,password)= ("%s","%s")", login_re-
quest.username, login_request.parola);
```

```
if(mysql_query(db, interogare)) db_error(db);
```

```
result = mysql_store_result(db);
row = mysql_fetch_row(result);
```

Tabelele:



```
mysql> show tables;
+-----+
| Tables_in_readsprofiler |
+-----+
| accesari_utilizatori    |
| autori                  |
| carti                    |
| cautari_utilizatori     |
| descarcari_utilizatori  |
| genuri                   |
| genuri_autori            |
| genuri_carte             |
| rating                   |
| rating_utilizatori      |
| subgenuri                |
| subgenuri_carte          |
| utilizatori              |
+-----+
13 rows in set (0.00 sec)
```

4.4 Algoritm recomandare

Algoritmul de recomandare de cărți pentru fiecare utilizator în parte are următoarea structură :

```
sugestiiCarti = extrage_carti_din_bd(tabela,profil);
pentru fiecare carte din sugetiiCarti executa
{
    gradRecomandare = nrOrdineTabela2 x importanta(carte);
    recomandari.push_back(<carte,gradRecomandare>);
}
sorteaza_dupa_grad_recomandare(recomandari);
response = selecteaza_top_recomandari(recomandari);
return response;
```

Gradul de recomandare al unei cărți se calculează după cum se vede în pseudocod după următoarea formulă:

```
gradRecomandare = nrOrdineTabela2 x importanta(carte)
unde:
```

$nrOrdineTabela$ = Importanța tabeli din care au fost extrase cărțile. De exemplu, o carte extrasă din tabela “Cautari” va avea o importanță mai mică în calcularea gradului de recomandare decât o carte extrasă din tabela “Descarcari”. Motivul este evident: o posibilă sugestie rezultată în urma faptului că utilizatorul a descărcat pentru citire o carte din genul dramă este, în majoritatea cazurilor, mult mai aproape de gusturile utilizatorului decât o carte din genul sci-fi rezultată în urma faptului că utilizatorul doar a căutat cărți din genul sci-fi.

$importanta(carte)$ = suma importanțelor fiecărei categorii din care provine cartea (gen, subgen, autor, titlu etc). Asemănător numărului de ordine al tabeli, și în acest caz o categorie din care face parte cartea poate juca un rol mai decisiv în algoritmul de recomandare decât altă categorie din care face parte aceeași carte.

Exemplu: utilizatorul a căutat o carte din genul sci-fi, apărută în anul 2005. Este mult mai probabil ca o posibilă carte recomandată ce provine tot din genul sci-fi să fie mult mai aproape de gusturile utilizatorului decât o carte recomandată doar pentru că a apărut tot în anul 2005. Ambele criterii sunt importante însă importanța genului este mai mare decât importanța anului apariției.

După ce se calculează gradul de recomandare pentru fiecare carte se aleg primele n cărți cu gradul de recomandare cel mai mare, unde n va fi setat implicit.

4.5 Aplicația finală

Reads Profiler

Bine ai venit, raresbabuta

Reads Profiler

Nume autor

Verne

Gen

roman

ISBN

Prenume autor

Jules

Subgen

Rating

5

Titlu

An

Cautare






Rezultate

Upload

Nr.	Titlu	Autor	Genuri	Subgenuri	An	Rating
1	Capitan la cincisprezece ani	Jules Verne	roman cala...	aventura	1878	★★★★★
2	O calatorie spre centrul p...	Jules Verne	roman cala...	aventura ro...	1863	★★★★★

Acceseaza

S-ar putea sa iti placa:



Reads Profiler

Bine ai venit, raresbabuta

Reads Profiler

Nume autor

Verne

Gen

roman

ISBN

Prenume autor

Jules

Subgen

Rating

5

Titlu

An

Cautare

Rezultate

Upload

Detalii

O calatorie spre centrul pamantului

Autor:

Jules Verne

Genuri:

roman calatorie

Subgenuri:

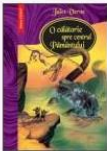
aventura romantic subteran sci-fi

An aparitie:

1863

ISBN:

565-5-21-574322-8



★★★★★






Descriere

Dupa descoperirea unui manuscris vechi cu rune, un savant, nepotul sau si ghidul lor intreprind o calatorie spre

Voteaza

Descarcare continut

Utilizatorii au apreciat de asemenea:



5 Concluzii

Proiectul ReadsProfiler a fost ales întrucât reprezintă o aplicație complexă ce combină :

1. tehnici de programare în rețea
2. lucru cu baze de date
3. programare orientată obiect
4. algoritmică și structuri de date
5. interfață grafică

5.1 Posibile îmbunătățiri

1. Oferirea posibilității utilizatorilor de a-și seta singuri anumite opțiuni și criterii ce vor fi utilizate în recomandări.
2. Oferirea posibilității utilizatorilor de a încărca propriile cărți în librărie.
3. Opțiunea de a urmări activitatea altor utilizatori.
4. Abonarea la un anumit autor, gen, subgen etc. iar de fiecare dată când va fi încărcată o carte ce corespunde listei bifate de fiecare abonat acesta să fie înștiințat prin e-mail.

6 Bibliografie

http://thor.info.uaic.ro/~adria/teach/courses/net/files/7rc_ProgramareaInRe-teaIII.pdf

<http://profs.info.uaic.ro/~eonica/rc/index.html>

<http://zetcode.com/db/mysqlc/>

<http://doc.qt.io/qt-5/qtexamplesandtutorials.html>