ReadsProfiler

Popescu Georgian Ștefan, A7
georgianspopescu@gmail.com

Facultatea de Informatică Iași

Rezumat ReadesProfiler reprezinta o aplicatie minimalista ce ofera acces la cautarea si descarcarea simpla de carti gratuite.

1 Introducere

ReadsProfiler este o aplicatie client-server care ofera acces la o librarie online. Clientul va avea un cont la care se autentifica prin nume de utilizator si parola. Utilizatorii aplcatiei vor fi de doua tipuri: obisnuiti si administratori. Libraria va contine carti din diverse genuri si subgenuri.

Clientii vor putea cauta o carte dupa atributele acesteia: titlu, autori, gen, an, ISBN, rating. De asemenea, vor putea vizualiza subgenurile unui gen, genurile abordate de un anumit autor, oferi un rating unei carti(cuprins intre 1-foarte slab si 5-excelent), descarca cartea si vizualizarea istoricului cautarilor si descarcarilor.

Serverul ii va oferi recomandari de carti clientului pe baza gusturilor acestuia si a preferintelor altor clienti cu aceleasi gusturi.

2 Tehnologii utilizate

Aplicatia va utiliza tehnologia TCP din urmatoarele motive:

1. stabilirea unei conexiuni intre client si server: autentificarea utilizatorului;
2. conexiune full-duplex: comunicarea dintre client(utilizator) si server(librarie) se bazeaza pe un schimb de mesaje de tipul cerere-raspuns
3. transmiterea sigura a pachetelor intre clienti si server: siguranta ca utilizatorul va primi in intregime raspunsul de la server, descarcarea cartilor.

Servirea concurenta si rapida a clientilor va fi efectuata de un server care foloseste thread-uri.

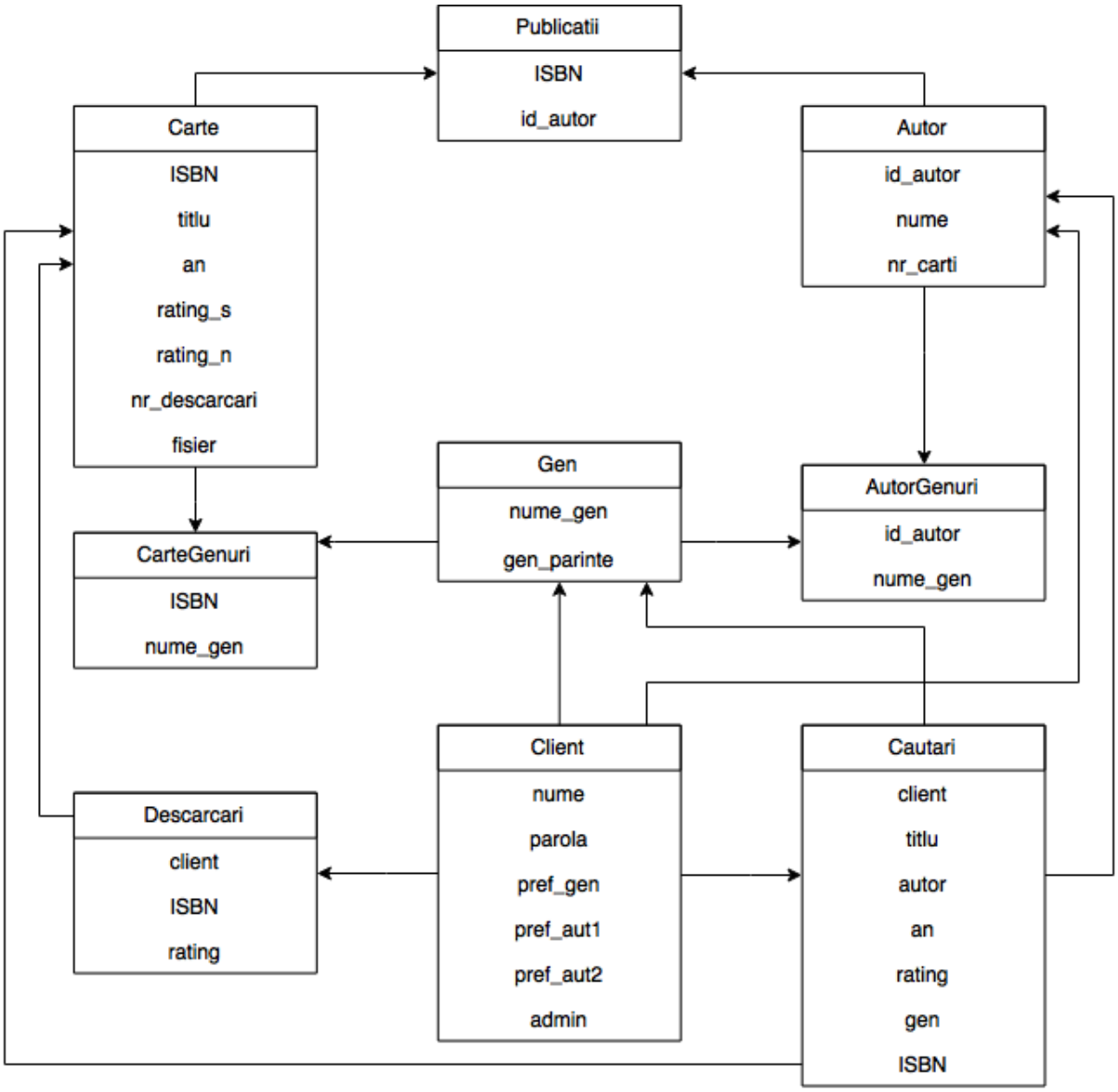
Toate informatiile pe care le utilizeaza serverul vor fi stocate intr-o baza de date relationala. Aceasta are rolul de a simplifica cautarile.

3 Arhitectura aplicatiei

3.1 Stocarea datelor

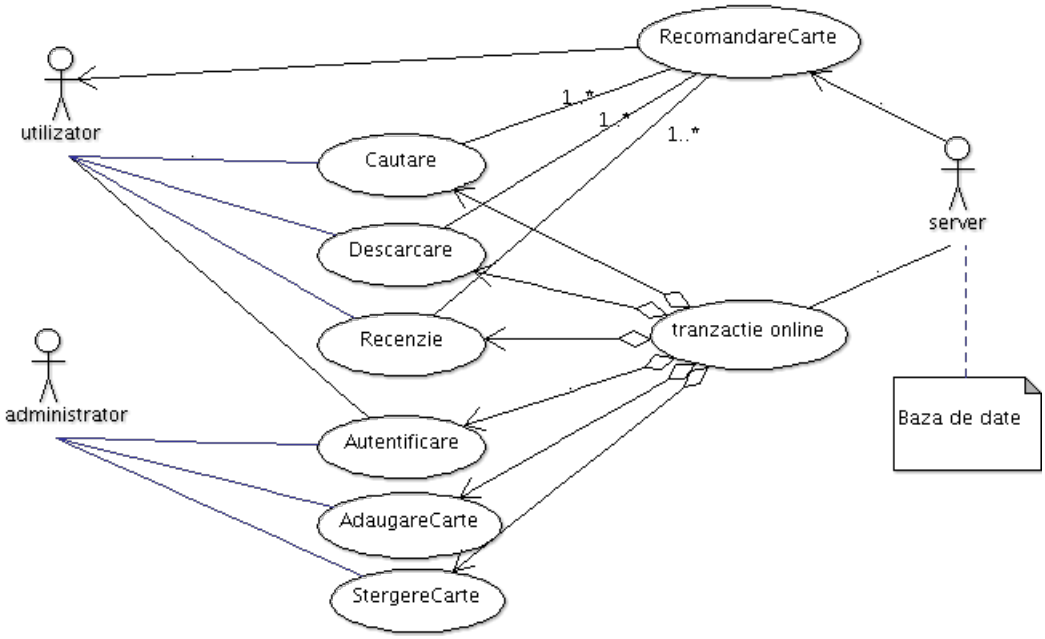
Baza de date contine tabelele *Carte*, *Autor* si *Genuri*, care vor avea ca tebele intermediare de legatura *Publicatii*, *CarteGenuri* si, respectiv, *AutorGenuri*.

In doua tabele se vor retine informatii despre cautarile si descarcarile clientilor, iar in tabela *Client* vom stoca datele de autentificare, genul preferat si 2 autori preferati pentru fiecare utilizator, informatii ce ne sunt utile pentru gestiunea sistemului de recomandari.



3.2 Scenariu de utilizare

Comunicarea dintre client si server se va desfasura dupa urmatorul scenariu:



3.3 Formatul liniei de comanda

1. nume_comanda // login |cauta |descarca |review |istoric |exit
2. argument_comanda

Exemple corecte:

```
[input] login presedinte parolaMea
[output] conectare reusita

[input] cauta
[output] Introdu titlu sau ALL pentru a selecta toate titlurile
[input] ALL
[output] Introdu autorii despartiti prin virgula
[input] Liviu Rebreanu
[output]  Introdu genurile despartite prin virgula
[input] universala,istorie
[output] Introdu subgenurile despartite prin virgula
[input] beletrictica
[output] Introdu ISBN
[input] ALL
[output] Ion - Liviu Rebreanu,universala,beletristica,2010,9786064301277
         Ciuleandra - Liviu Rebreanu,universala,beletristica,2012,4786064301279
```

3.4 Formatul pachetului

```
unsigned int       length
char               type
char[length - 1]   data payload
```

Unde type este:

```
l = logare
c = cautare
d = descarcare
r = review
a = adauga carte
s = sterge carte
h = istoric cautari
n = raspuns normal de la server
e = eroare
x = delogare
```

Continutul payload-ului difera in functie de tipul de serviciu:

```
type            payload
l               nume_utilizator parola
c               titlu|ALL$nume_autor|ALL$gen1[,gen2,...]$subgen[,subge2,...]$an$ISBN
d               ISBN
r               ISBN
a               titlu$nume_autor$gen1[,gen2,...]$subgen[,subge2,...]$an$ISBN
s               ISBN
h               -
n               raspuns
e               descriere_eroare
x               -
```

3.5 Proiectarea si implementarea modelului client/server TCP

Serverul va fi implementat folosind tehnologia TCP – prethreaded, cu blocare pentru protectia *accept*(). Serverul creeaza un numar de thread-uri cand este pornit si aceastea vor servi clientii. Pentru ca doar un singur thread sa apeleze *accept* la un moment dat, se va folosi *mutex lock* pe apelul primitivei *accept*. Serverul va tine evidenta numarului de thread-uri active(care deservesc un client) si va creea noi thread-uri cand numarul clientilor se apropie de numarul total al thread-uilor create. In momentul cand numarul thread-urilor nefolosite atinge o cota, acestea isi vor termina executia, notificand in acelasi timp procesul principal.

Clientul va fi implementat folosind tehnologia TCP. Aplicatia-client ii cere utilizatorului sa se autentifice si apoi, in cazul unei acceptari din partea serverului, clientul va putea introduce comenzie la care are acces. Utilizatorul va

putea introduce alta comanda dupa primirea raspunsului de la server. Incheierea sesiunii se va face prin comanda *exit*.

3.6 Proiectarea sistemului de recomandari

```
bool Recomandare::adauga(carte)
{
    if(carte != NULL && adaugata_deja(carte) == NULL)
    {
        recomandari++;
        carti_recomandate[recomandari] = carte;
        return true;
    }
    return false;
}

 Recomandare Recomandare::recomanda(Client client)
{
    Gen gen = client.gen_preferat();
    Autor aut1 = client.autor_preferat1();
    Autor aut2 = client.autor_preferat2();
    Carte carte = new Carte;
    int nr = 0;
    Recomandare carti_rec = new Recomandare;

    // recomanda carti scrise de autorii preferati din genul preferat
    while(nr < 3 && (carte = cauta(aut1)) != NULL))
    {
        if(carti_rec.adauga(carte))
            nr++;
    }
    while(nr < 6  &&  (carte = cauta(aut2)) != NULL))
    {
        if(carti_rec.adauga(carte))
            nr++;
    }

    // recomanda carti de alt gen, scrise de aceeasi autori
    while(nr < 9 && (carte = cauta(aut1,aut2)) !=NULL)
    {
        if(carti_rec.adauga(carte))
            nr++;
    }

    // recomanda 2 carti descarcate de alti clienti cu aceleasi gusturi
```

```
        while(nr < 11)
        {
            Client alt_client = new Client;
            alt_client = client.cautaclient(gen,aut1,aut2);
            carte = cauta(alt_client);
            if(carti_rec.adauga(carte))
                nr++;
            else
                alt_client = client.cautaclient(gen,aut1,aut2);
        }

        // recomanda o carte de la un autor cautat recent
        Autor aut = client.autor_ultim();
        carte = cauta(aut);
        carti_rec.adauga(aut);
    }
```

4 Concluzii

Aplicatia ofera functionalitatile de baza ale unei librarii online. Modul de implementare a serverului ofera servirea rapida, concurenta si dinamica a clientilor, iar folosirea tehnologiei TCP asigura siguranta transferului de date.

Odata cu cresterea numarului de utilizatori, numarul de cautari poate creste exponential. Folosirea unor *functii de hash* pentru gestionarea cautarilor ar reprezenta o metoda mai eficienta de implementare in acest caz.

Bibliografie

[1.] https://profs.info.uaic.ro/ computernetworks/files/7rc_ProgramareaInReteaIII_Ro.pdf
[2.] https://www.ietf.org/rfc/rfc4253.txt
[3.] https://www.tutorialspoint.com/sqlite/sqlite_c_cpp.htm