Documentatie

Rares-Stefan Macovei

December 2022

1 Introducere

ReadsProfiler este o aplicație ce folosește sistemul multithreading pentru a procesa cereri de la un număr de clienți. Ea va crea un profil pentru fiecare utilizator unde acesta poate adăuga în biblioteca virtuală cărți în diferite rafturi ("Reading", "Read", "Dropped", "Planning to Read") și alte informații (de ex. o notă). După mai multe recenzii ale cărților, fiecare utilizator va avea un profil prin care îi vor putea fi recomandate cărți asemănătoare cu ce a citit.

De altfel, pot fi adăugate de către conturile admin alte cărți, alți autori, pot fi făcute căutări dupa autori, genuri, edituri, isbn etc.

2 Tehnologiile utilizate

Pentru acest gen de aplicație, tehnologia TCP ar fi cea mai potrivită deoarece nu permite pierderea de informații între client și server, aceasta modificând profilul fiecărui utilizator în moduri neașteptate și neplăcute.

Pentru datele transmise se vor folosi threaduri pentru fiecare proces, deoarece un thread pentru fiecare utilizator ar limita numărul de utilizatori permiși pe site. Unele threaduri, cum ar fi cel de login și register, vor fi combinate deoarece sunt cereri trimise o singură dată pe sesiune.

Toate datele despre utilizatori, cât și baze de date pentru căutări, vor fi ținute în fișiere de timp xml.

3 Arhitectura sistemului

Aplicația este formată dintr-un fișier pentru client, unul pentru server, un program pentru crearea fișierelor xml și un număr variabil de fișiere xml, în funcție de numărul de utilizatori, de cărți și de autori.

Partea de client va trimite constant cereri către server cu toate funcțiile disponibile în aplicație.

Partea de server va primi cererile clientului și le va rezolva printr-un thread al clientului. El va verifica valabilitatea datelor la login și/sau register, va updata baza de date a utilizatorilor și profilul acestora la fiecare adăugare de

carte în biblioteca virtuală (prin profil înțelegem recomandările pe care le va primi un utilizator în funcție de recenziile sale. Biblioteca virtuală este pagina utilizatorului cu toate cărțile cu status setat). Serverul va crea și fișiere xml noi pentru fiecare adăugare de conținut de către un admin.

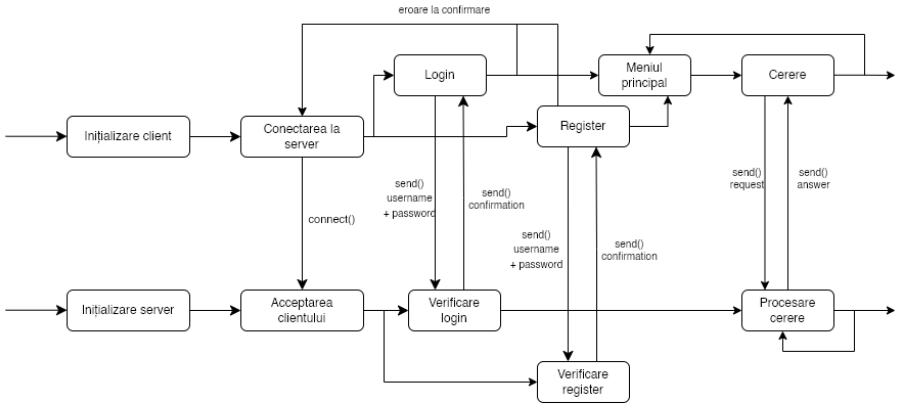

Figura 1: Arhitectura sistemului

4 Detalii de implementare

4.1 Crearea socketului

Prima dată vom crea un socket prin care se va realiza comunicarea client-server.

Crearea socketului

```
int sd;
if ((sd = socket(AF_INET, SOCK_STREAM, 0)) == -1)
{
    perror("Eroare la socket().\n");
    return errno;
}
```

4.2 Conectarea la server

Conectarea la server se va face pe adresa dispozitivului și pe un port specificat in codul serverului. În *sockaddr* vor fi salvate date legate de server.

Conectarea la server

```
int create_connection(char *argv[]){
  int sd;//socket descriptor
  struct sockaddr_in server;//structure used for connecting

  if((sd = socket(AF_INET,SOCK_STREAM,0)) == -1){
    perror(SOCKET_ERROR);
    return errno;
  }

  server.sin_family = AF_INET; //socket family
  server.sin_addr.s_addr = inet_addr(argv[1]); //ip address
  server.sin_port = htons(port); //port

  if(connect(sd,(struct sockaddr *) &server,sizeof(struct
    sockaddr)) == -1){ //connecting to server
    perror(CONNECT_ERROR);
    return errno;
  }
  return sd;
}
```

4.3 Atașarea de socketuri și acceptarea clientului

Vom atașa socketul creat și vom asculta cererile clienților.

Atașarea de socketuri și acceptarea clientului

```
if(bind(sd, (struct sockaddr *)&server, sizeof(struct sockaddr))
    == -1){
    perror("[server]Eroare la bind().\n");
    return errno;
}
if(listen(sd, 5) == -1){
    perror("[server]Eroare la listen().\n");
    return errno;
}
if(accept(sd, (struct sockaddr *)&from, &length) < 0){
    perror("[server]Eroare la accept().\n");
    continue;
}
```

4.4 Crearea de threaduri

Clienții vor fi serviți în mod concurent prin folosirea de threaduri create pentru fiecare cerere.

Crearea de threaduri

```
td = (struct thData*)malloc(sizeof(struct thData));
td->idThread = i++;
td->cl = client;
pthread_create(&th[i], NULL, &treat, td);
```

4.5 Verificarea existenței utilizatorului

Se va citi de la client un username cerut in prealabil, după care se va verifica existența fișierului xml cu acest nume în folderul *users*. Dacă el există, atunci vor fi preluate statusul (admin sau utilizator obișnuit) și parola din fișier cu ajutorul unei funcții de parsare a xml-ului, iar cea de a doua va fi comparată cu parola cerută clientului (vezi următorul subcapitol). Bineînțeles, în funcția *register* se va verifica să nu existe un utilizator cu același nume pentru a nu suprascrie un fișier de utilizator. Serverul va trimite un 1 dacă userul există și este unul obișnuit, 2 dacă este admin și 0 dacă nu există. În client va fi salvat statusul userului.

Verificarea existenței utilizatorului

```
char user[1001],fisier[1021],parola[1001];
int i=0;
struct thData tdL;
tdL= *((struct thData*)arg);
if(read (tdL.cl, &user,sizeof(user)) <= 0){
    printf("[Thread %d]\n",tdL.idThread);
    perror ("Eroare la read() de la client.\n");
}
printf("[Thread %d]Mesajul a fost
    receptionat...%s\n",tdL.idThread, msg);
strcat(fisier,".users/");strcat(fisier,user);strcat(fisier,".xml");
if((file = fopen(fisier,"r"))!=NULL){
    printf("[Thread %d]Utilizatorul a fost
    gasit!\n",tdL.idThread);
    char user_status[10];
    cautare_xml(fisier,"user",user_status); //verificare status
    admin
    cautare_xml(fisier,"parola",parola); //functie de parsare a
    xml-ului
    if(user_status=="user" && write (tdL.cl, "1", 2) <= 0){
        printf("[Thread %d] ",tdL.idThread);
        perror("[Thread]Eroare la write() catre client.\n");
    }
    if(user_status=="admin" && write (tdL.cl, "2", 2) <= 0){
        printf("[Thread %d] ",tdL.idThread);
        perror("[Thread]Eroare la write() catre client.\n");
    }
    else{
        printf("[Thread %d]Mesaj trimis.\n",tdL.idThread);
    }
}
else if(write (tdL.cl, "0", 2) <= 0){
    printf("[Thread %d] ",tdL.idThread);
    perror("[Thread]Eroare la write() catre client.\n");
}
else{
    printf("[Thread %d]Mesaj trimis.\n",tdL.idThread);
}
```

4.6 Verificarea parolei

Parola găsită la subcapitolul anterior este comparată cu parola primită de utilizator. Serverul va trimite un 1 dacă parolele se potrivesc sau 0 dacă nu.

Verificarea parolei

```
char parola_input[1001];
int i=0;
struct thData tdL;
tdL= *((struct thData*)arg);
if(read (tdL.cl, &parola_input,sizeof(parola_input)) <= 0){
    printf("[Thread %d]\n",tdL.idThread);
    perror ("Eroare la read() de la client.\n");
}
printf("[Thread %d]Mesajul a fost
    receptionat...%s\n",tdL.idThread, msg);
if(strcmp(parola,parola_input)==0){
    if(write (tdL.cl, "1", 2) <= 0){
        printf("[Thread %d] ",tdL.idThread);
        perror("[Thread]Eroare la write() catre client.\n");
    }
    else{
        printf("[Thread %d]Mesajul a fost transmis cu
    succes.\n",tdL.idThread);
    }
}
else if(write (tdL.cl, "0", 2) <= 0){
    printf("[Thread %d] ",tdL.idThread);
    perror("[Thread]Eroare la write() catre client.\n");
}
else{
    printf("[Thread %d]Mesajul a fost transmis cu
    succes.\n",tdL.idThread);
}
```

5 Concluzie

Unul din lucrurile care ar putea fi adăugate ar fi transformarea site-ului într-o rețea de socializare în care persoanele pot adăuga prieteni, recenzii la cărți sau intra în grupuri. Prin crearea unui sentiment de comunitate, utilizatorii vor folosi mai des site-ul și se vor simți mai bine, dând personalitate aplicației.

Cea mai atractivă parte a site-ului este sistemul de recomandări, iar acesta poate fi îmbunătățit prin folosirea de Machine Learning pentru a afla procentele optime în calculul recomandărilor. Acest procent poate fi mai departe interpolat cu recomandările altor persoane pentru a primi recomandări în cadrul unui grup/cerc literar.

O altă funcționalitate ce ar putea fi imbunătățită este sistemul de căutare astfel încât să existe căutări parțiale ale titlului, updatări în timp real a rezultatelor, mărirea numărului de cărți de pe site.

Baza de date ar putea fi updatată la una de tip SQL pentru a gestiona mai

ușor volumul de informații în cazul în care apar foarte mulți utilizatori, dar și pentru a fi mai ușor de transferat pe un alt site sau pentru a introduce alte câmpuri.

Pot fi organizate concursuri, polluri, quizuri care să facă aplicația mai atractivă și să dea libertate utilizatorilor. Bineînțeles, ar trebui moderate într-un fel aceste cereri, dar cred că vor fi permise doar la nivel de text și va fi o listă de cuvinte interzise pe site, creata si editata de utilizatorii de tip admin. Tot legat de asta, ar putea fi detectat comportamentul abuziv al utilizatorilor (număr exagerat de taguri, cereri neobișnuite către server, recenzii exclusiv de o stea) și eliminați de pe site.

6 Bibliografie

https://profs.info.uaic.ro/~computernetworks/files/NetEx/S12/ServerConcThread/servTcpConcTh2.c

https://profs.info.uaic.ro/~computernetworks/files/NetEx/S12/ServerConcThread/cliTcpNr.c

https://www.w3schools.com/xml/xml_whatis.asp