Weather Monitor
Grad Dificultate C

Frunza Teodor-Octavian

Universitatea "Alexandru Ioan Cuza" Facultatea de Informatica
frunza.teodor@info.uaic.ro

Abstract. Acest document reprezinta un raport amanuntit asupra proiectului Weather Monitor fiind alcatuit atat din elemente arhitecturale cat si din elemente reprezentate de interactiune cu utilizatorul.

1 Introducere

1.1 Enuntul Problemei

Sa se scrie o aplicatie client/server pentru managementul (e.g., listare, modificare, stergere) informatiilor meteo pentru o anumita zona. La un port separat se va oferi posibilitatea actualizarii informatiilor meteo privitoare la o localitate sau multime de localitati ale zonei considerate.

1.2 Analiza Problemei

Din enuntul problemei deducem ca ni se cere crearea unei aplicatii alcatuita dintr-un server concurent si un client ce se poate conecta la acesta. Astfel, pentru a rezolva aceasta problema vom cauta spre a utiliza o metoda optima de transmitere a datelor de la useri multipli la server si invers fara a exista riscul de a pierde biti de date sau de a corupe mesajele trimise. Mai mult, vom incerca optimizarea acesteia pentru a inlatura posibilitatile de supraincarcare a retelei sau timpii excesivi de lungi pentru asteptarea raspunsurilor.

2 Tehnologii Uitilzate

2.1 TCP/IP

Vom utiliza TCP/IP ca protocol de comunicare în rețeaua aplicației Weather-Monitor datorită aplicațiilor sale vaste în problemele de tip conexiune biunivocă client / server. De asemenea, vom creea un server ce utilizează protocolul de tip TCP/IP sub formă concurentă pentru a permite utilizarea simultana a serverului de mai multi clienti fara a corupe sau pierde datele individuale. In ceea ce urmeaza voi prezenta cateva caracteristici ale protocolului de transmisie TCP/IP datorita carora a fost ales pentru acest proiect si motivul excluderii protocolului UDP.

2.2 Caracteristici TCP/IP

1. Ofera servicii orientate-conexiune, full duplex;
2. Conexiunile sunt sigure pentru transportul fluxurilor de octeti;
3. Vizeaza oferirea calitatii maxime a serviciilor;
4. Controleaza fluxul de date (stream-oriented).

2.3 De ce nu folosim UDP?

1. Ofera servicii minimale de transport, avand riscul de a pierde informatii;
2. Nu ofera controlul fluxului de date;
3. Nu este orientat conexiune.

Astfel se observa ca utilizarea protocolului UDP pentru trimiterea informatiei intre client si server ar fi deficitar in cazul nostru avand un risc destul de mare de a pierde informatii pe parcurs.

3 Arhitectura Aplicatiei

3.1 Metoda de Functionare

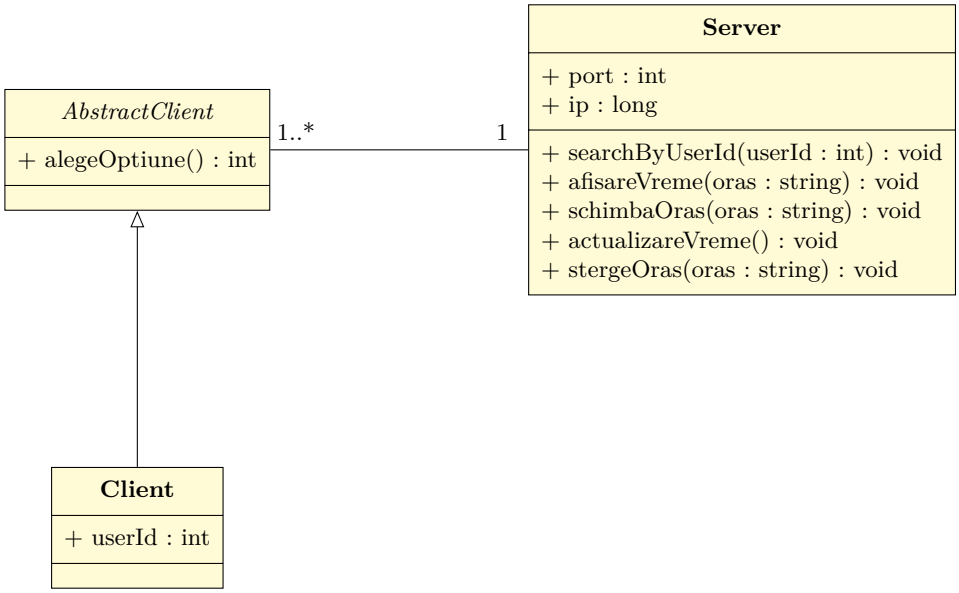
Aplicatia Weather Monitor va fi alcatuita din urmatoarele entitati/participanti:

1. Baza de date / Fisier(e) de stocare a informatiilor meteo;
2. Server TCP/IP concurent pe baza de fork;
3. Client.

Clientul va avea patru optiuni din care va putea alege:

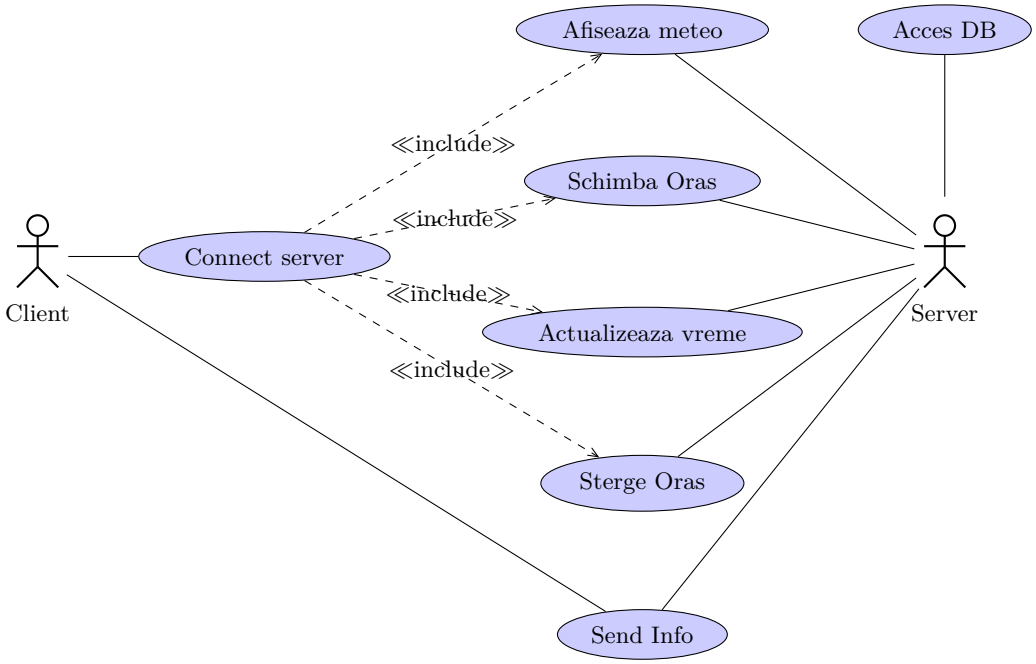
1. Afisarea datelor meteo actuale (afisareVreme());
2. Alegerea altui oras (schimbaOras());
3. Actualizeaza datele meteo (actualizareVreme());
4. Stergerea unui oras (stergeOras()).

3.2 Diagrama UML



4 Detalii de Implementare

4.1 Scenarii de utilizare



5 Cod Relevant

5.1 Client

```
count = 0;
if (connect (sd, (struct sockaddr *) &server,sizeof (struct sockaddr))
    == -1){
   perror ("[client]Eroare la connect()");
   return errno;
}

if(count == 0){
  if (write (sd, userId, 5) <= 0){
       perror ("[client]Eroare la write() spre server\n");
    return errno;
    }
  if (read (sd, msg, 100) < 0){
         perror ("[client]Eroare la read() de la server.\n");
         return errno;
    }
 printf ("[client]Mesajul primit este:", msg);
}
else{
  count ++;
  bzero (msg, 2);
  printf ("[client]1) Afiseaza meteo \n");
  printf ("[client]2) Schimba oras \n");
  printf ("[client]3)Actualizeaza vreme \n");
  printf ("[client]3)Sterge oras \n");
  printf ("[client]0) Inchide conexiunea \n")
  printf ("[client]Introduceti optiunea dorita: \n");
  fflush (stdout);
  read (0, msg, 2);
  while(msg != 1 || msg !=2 || msg != 3 || msg != 4) {
    printf ("[client]Ati introdus o optiune inexistenta. Va rog
        introduceti optiunea dorita din cele existente(1,2,3,4: ");
    read (0, msg, 2);
  }
}
```

Am ales la client aceasta secventa deoarece reprezinta realizarea conexiunii cu serverul si trimiterea automata a id-ului clientului catre server. In functie de acest Id i se va afisa vremea in functie de ultimele setari. De exemplu daca acesta avea orasul Iasi setat initial, se va afisa vremea din acea locatie. De asemenea se poate observa si un prim prototip de meniu si o verificare pentru optiuni. Daca clientul alege o optiune inexistenta acesta va trebui sa reintroduca pana cand este o optiune valida.

5.2 Server

```
client = accept (sd, (struct sockaddr *) &from, &length);
if(fork()==0){
   close(sd);
}
if (client < 0){
     perror ("[server]Eroare la accept().\n");
     continue;
}
if (read (client, msg, 100) <= 0){
     perror ("[server]Eroare la read() de la client.\n");
     close (client);
     continue;
}
if (verfiId(msg) == true){
   bzero(msgrasp,100);
   importWeather(msgrasp,msg);
   if (write (client, msgrasp, 100) <= 0){
      perror ("[server]Eroare la write() catre client.\n");
      continue;
   }
}
if(verifExit(msg) == true){
   close(client);
}
else{
   bzero (msg, 100);
   if (read (client, msg, 100) <= 0){
      perror ("[server]Eroare la read() de la client.\n");
       close (client);
      continue;
   }
}
//verifId e functie care verifica daca e de forma Id -> daca e aducem
    vremea default din zona atribuita acelui user
//importWeather -> aduce vremea personalizata pentru client
//verifExit -> daca e 0 inchide consexiunea
```

Am ales la server aceasta secventa deoarece reprezinta acceptarea clientului, crearea copilului si inchiderea conexiunii pe tata si de asemenea primirea initiala a idului clientului. Pe baza acestuia se importa din baza de date / fisier ultimele setari si se afiseaza vremea. Dupa aceasta operatiune acesta asteapta o optiune de la client. Daca este 0 serverul inchide conexiunea cu clientul.

6 Concluzii

Pana in momentul de fata avem realizate atat arhitectura aplicatie (UML) cat si scenariile de utilizare (USE CASE). Mai mult avem si un prototip de client si server urmand sa completam cu functiile de verificare si parsare a textului. De imbunatatit ar fi securitatea, viteza si rezolvarea buggurilor din program.

7 Bibliografie

In realizarea acestui document s-au utilizat urmatoarele resurse:

1. https://profs.info.uaic.ro/ computernetworks/cursullaboratorul.php
2. https://profs.info.uaic.ro/ computernetworks/files/NetEx/S5/servTcpIt.c
3. https://profs.info.uaic.ro/ computernetworks/files/NetEx/S5/cliTcpIt.c
4. https://profs.info.uaic.ro/ computernetworks/ProiecteNet2017.php
5. https://stackoverflow.com/questions/3175105/writing-code-in-latex-document
6. https://tex.stackexchange.com/questions/354089/how-to-give-a-name-to-an-association-with-tikz-uml
7. tikz-uml-userguide.pdf
8. llncsdoc.pdf