Weather Monitor
Grad Dificultate C

Frunză Teodor-Octavian

[1] Universitatea "Alexandru Ioan Cuza"
[2] Facultatea de Informatică
frunza.teodor@info.uaic.ro

Rezumat Acest document reprezintă un raport amănunțit asupra proiectului "Weather Monitor", fiind alcătuit atât din elemente arhitecturale cât și din elemente reprezentate de interacțiunea utilizatorului cu aplicația.

1 Introducere

1.1 Enunțul Problemei

Să se scrie o aplicație client/server pentru managementul (e.g., listare, modificare, ștergere) informațiilor meteo pentru o anumită zonă. La un port separat se va oferi posibilitatea actualizării informațiilor meteo privitoare la o localitate sau mulțime de localități ale zonei considerate.

1.2 Analiza Problemei

Din enunțul problemei deducem că ni se cere crearea unei aplicații alcătuite dintr-un server concurent și un client ce se poate conecta la acesta. Astfel, pentru a rezolva această problemă vom căuta spre a utiliza o metodă optimă de transmitere a datelor de la useri multipli la server și invers fără a exista riscul de a pierde biți de date sau de a corupe mesajele trimise. Mai mult, vom încerca optimizarea acesteia pentru a înlătura posibilitățile de supraîncărcare a rețelei sau timpii excesivi de lungi de așteptare a răspunsurilor.

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 simultană a serverului de către mai mulți clienți fără a corupe sau pierde datele individuale. În ceea ce urmează voi prezenta câteva caracteristici ale protocolului de transmisie TCP/IP datorită cărora a fost ales pentru acest proiect și motivul excluderii protocolului UDP.

2.2 Caracteristici TCP/IP

1. Oferă servicii orientate-conexiune, full duplex;
2. Conexiunile sunt sigure pentru transportul fluxurilor de octeți;
3. Vizeaza oferirea calității maxime a serviciilor;
4. Controlează fluxul de date (stream-oriented).

2.3 De ce nu folosim UDP?

1. Oferă servicii minimale de transport, având riscul de a pierde informații;
2. Nu oferă controlul fluxului de date;
3. Nu este orientat conexiune.

Astfel se observă că utilizarea protocolului UDP pentru trimiterea informației între client și server ar fi deficitar în cazul nostru, având un risc destul de mare de a pierde informații pe parcurs.

3 Arhitectura Aplicației

3.1 Metoda de Funcționare

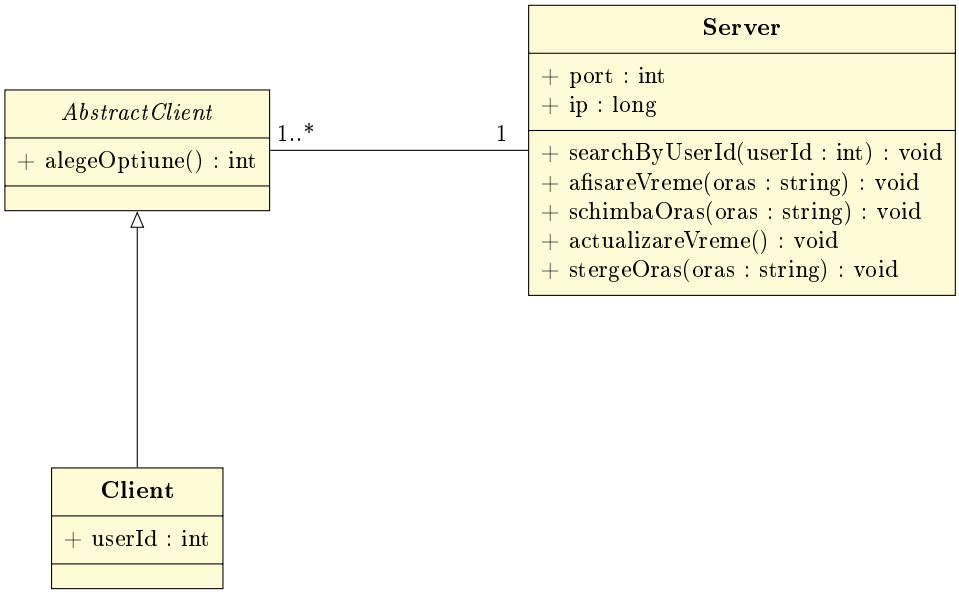
Aplicația "Weather Monitor" va fi alcătuită din următoarele entități/participanți:

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

Clientul va avea patru opțiuni din care va putea alege:

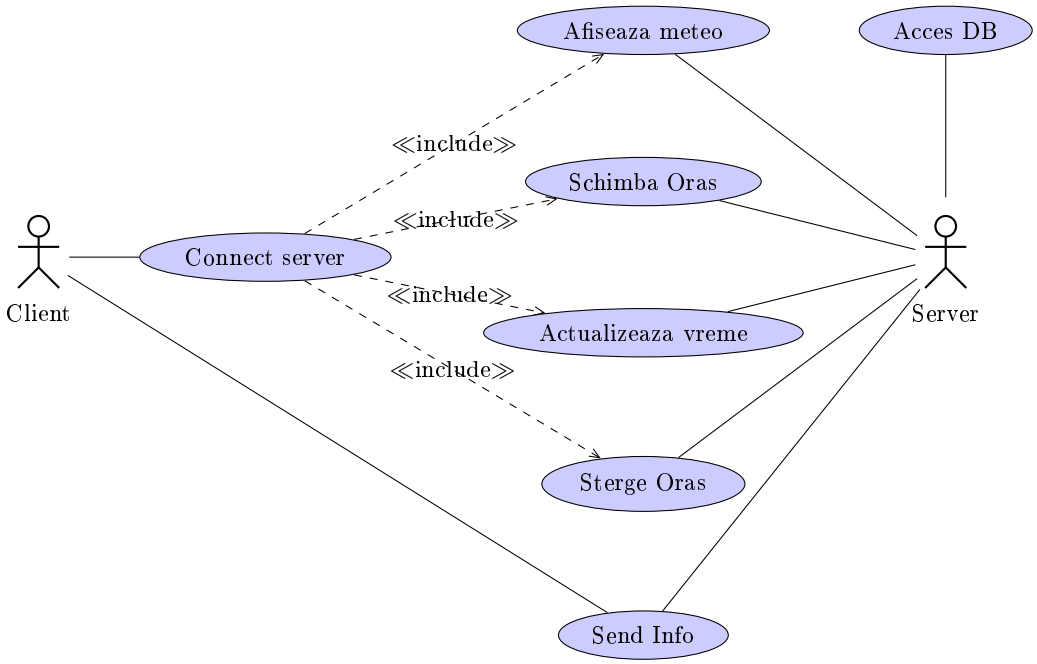
1. Afișarea datelor meteo actuale (afisareVreme());
2. Alegerea altui oraș (schimbaOras());
3. Actualizarea datele meteo (actualizareVreme());
4. Ștergerea unui oraș (stergeOras()).

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);
  count++;
}
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 această secvență deoarece reprezintă realizarea conexiunii cu serverul și trimiterea automată a id-ului clientului către server. În funcție de acest Id i se va afișa vremea personalizată de ultimele setări făcute. De exemplu, dacă clientul avea orașul Iași setat inițial, se va afisa vremea din acea locație. De asemenea, se poate observa și un prim prototip de meniu și o verificare pentru alegerea opțiunii. Dacă clientul alege o opțiune inexistentă, acesta va trebui să reintroducă un caracter până când acesta reprezintă o opțiune validă.

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 (verfYId(msg) == true){
   bzero(msgrasp,100);
   importWeather(msgrasp,msg);
   if (write (client, msgrasp, 100) <= 0){
      perror ("[server]Eroare la write() catre client.\n");
      continue;
   }
}
if(verifyExit(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;
   }
}
//verifyId reprezinta o functie care verifica daca datele primite de la
     client sunt de forma Id. Daca sunt de aceasta forma aducem vremea
     prestabilita din fisierul acelui user
//importWeather -> aduce vremea personalizata pentru client
//verifExit -> daca este 0 inchide consexiunea
```

Am ales la server această secvență deoarece reprezintă acceptarea clientului, crearea copilului și închiderea conexiunii în procesul părinte și de asemenea primirea inițială a id-ului clientului. Pe baza acestuia se importă din baza de date / fișier(e) ultimele setări și se afișează vremea. După acceptarea id-ului, verificarea existenței acestuia și afișarea datelor meteorologice predefinite de utlizator, acesta așteaptă o opțiune de la client. Dacă este 0 serverul închide conexiunea cu clientul.

6 Concluzii

Până în momentul de față avem realizate atât arhitectura aplicației (UML) cât și scenariile de utilizare a aplicației (USE CASE). Mai mult, avem și un prototip de client și server, urmând să completăm cu funcțiile de verificare și parsare a textului. De îmbunătățit ar fi securitatea, viteza și rezolvarea buggurilor din aplicație.

7 Bibliografie

În realizarea acestui document s-au utilizat următoarele 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