Raport Tema 2 - Monitorizarea Traficului

Coteanu Andra Maria
grupa 2A4

decembrie 2020

Cuprins

1 Introducere

În această fișă de raport tehnic voi prezenta toate informațiile necesare pentru implementarea proiectului Monitorizarea Traficului (A) pe care am ales să îl fac în limbajul C/C++. Aceasta este o aplicație de tip client-server, ce are rolul de a prelucra datele reale și/sau obținute din mediul online și de la șoferi pentru a-i ajuta pe aceștia să evite evenimente neplăcute în trafic, dar și pentru menținerea siguranței șoferilor, pasagerilor, pietonilor și a oricărui alt participant la trafic.

Sistemul trebuie să fie capabil să gestioneze traficul și să ofere informații tuturor șoferilor conectați la aplicație. Informațiile ,,real time" despre trafic vor fi furnizate de șoferi (aceștia pot raporta orice eveniment sau incident la care au fost martori în trafic).

Alte funcționalități ale aplicației:

- clientul va fi obligat să se autentifice fie cu un cont (formatul: utilizator paroală) existent fie să își creeze unul nou de fiecare dată când dorește să acceseze aplicația;

- aplicația va înregistra automat viteza cu care circulă autovehiculul și locația acestuia (și va anunța șoferul dacă a depășit limita de viteză pe porțiunea de drum pe care se află) (datele vor fi actualizate cu frecvența de 1 minut);

- sistemul va notifica șoferii de evenimente în trafic precum blocaje, accidente, drumuri închise, drumuri avariate sau drumuri în reparații și despre restricțiile de viteză aferente;

- sistemul poate oferi informații despre vreme, sport sau prețurile pentru combustibil la cererea utilizatorului; în plus, stațiile peco afișate vor fi cele mai aproape 3 stații de locația clientului la momentul solicitării (se va încerca pe cât posibil ca cele 3 benzinării să fie diferite firme pentru a oferi clientului mai multe opțiuni de intervale de prețuri pentru combustibil).

2 Tehnologiile utilizate

Sistemul poate fi considerat ca fiind format din 2 părți:

1. Aplicația în sine și funcționalitățile ei

Acest lucru poate fi realizat prin implementarea unui server care să asigure legătura între baza de date, graf-uri și alte forme de a stoca, sorta și face rost de informațiile necesare participanților la trafic, dar și prin implementarea unui client care va comunica cu serverul.

2. Utilizatorii

Aceștia, când se vor conecta la aplicație, se vor conecta de fapt la serverul care asigură integritatea structurală sistemului. Astfel utilizatorii pot trimite mesaje spre server (prin intermediul clientului), unde informația va fi procesată și utilizatorul va primi un răspuns conform solicitării făcute. Dacă un utilizator a raportat un incident, atunci serverul va produce un răspuns care îi vizează pe toți utilizatorii conectați atunci.

Pentru a realiza aceste lucruri este necesară o conexiune client-server, deci voi alege să implementez modelul TCP concurent care să asigure acest lucru. Serverul TCP va crea câte un fir de execuție pentru fiecare client așa încât să existe posibilitatea servirii mai multor utilizatori simultan. Se vor lua datele de la client, se vor procesa și se va trimite răspunsul

spre acesta. Clientul va avea stabilit un port de conexiune de unde se vor prelua datele de către server. În client se va produce (decodifica mesajul de la server) răspunsul final primit de utilizator.

Serverul va comunica și cu o bază de date și două grafuri pentru a reține utilizatorii, dacă s-au abonat la știri, harta orașului și locațiile stațiilor peco.

Clientul va avea 3 thread-uri ce vor rula în paralel unul față de celălalt:

- threadul aplicației: va asigura rularea corectă a întregului program;
- threadul vitezei: va afișa la un interval de aproximativ 1 minut viteza cu care circulă utilizatorul, locația sa și atentionare legată de limita de viteza;
- threadul incidentelor: va aștepta mereu un mesaj de la server, când îl va primi îl va afișa instant pe ecranele utilizatorilor conectați.

3 Arhitectura aplicației

Informațiile cum ar fi utilizatorii deja existenți și dacă s-au abonat sau nu la știri vor fi memorate într-o bază de date sql.

Pentru memorarea străzilor și a limitei legale de viteză pe acele porțiuni de drum voi implementa un graf neorientat cu cost pe muchie (presupunând că nu există drumuri cu sens unic) conform hărții [6] de mai jos:

Locațiile stațiilor peco, precum și numărul acestora de identificare vor fi memorate într-un graf neorientat cu cost pe muchie. Fiecare id va corespunde unei firme diferite de benzinărie.

Știrile meteo sau despre sport vor fi furnizate utilizatorilor cu ajutorul unor funcții în server care va genera informațiile prin mimica unei inteligențe artificiale.

Cele două funcții *sign_in()* și *login()* vor apărea împreună cu funcția *quit()* în meniul principal. Clientul va fi nevoit să se autentifice fie cu un cont existent, fie cu un cont nou creat pentru a putea accesa toate funcționalitățile aplicației.

Un exemplu de instanță sql este următoarea, unde am verificat dacă userul și parola introduse de client există. Dacă există, atunci se va putea loga, iar serverul va mai accesa încă odată baza de date pentru a verifica dacă utilizatorul proaspăt logat este sau nu abonat la știri.

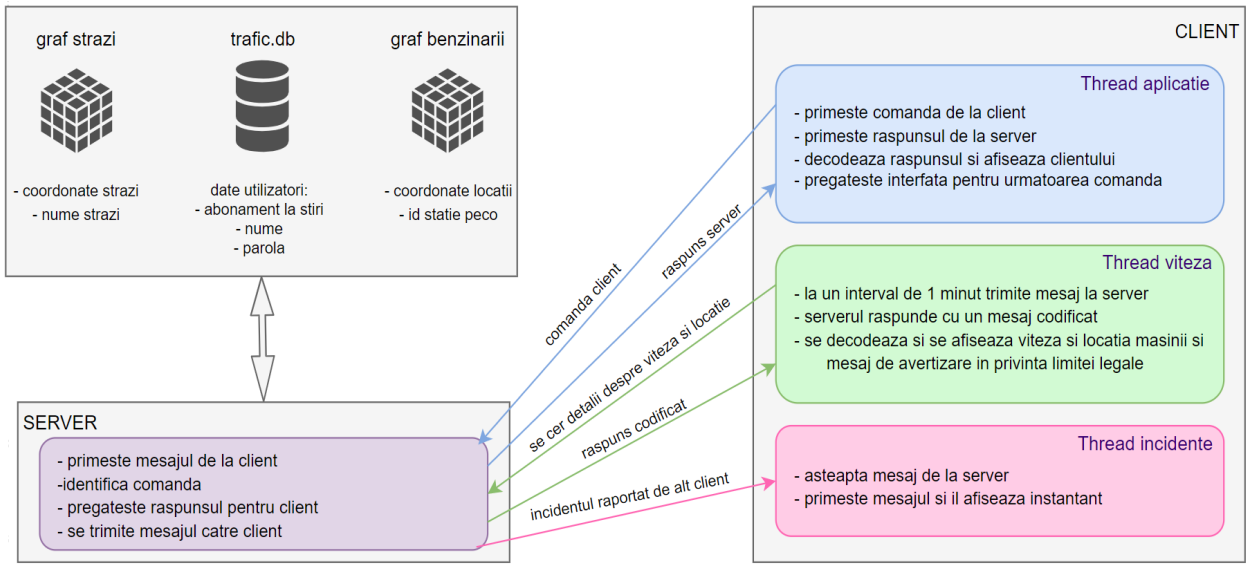
```
sql = (char *)malloc(256);
/* se creaza o instanta sql */
sprintf(sql, "SELECT count(nume) from users WHERE nume='%s' and paswd='%s'", user, paswd);

/* se executa instanta sql */
rc = sqlite3_exec(db, sql, callback_login, (void*)mesaj_callback, &ErrMsg);

/* daca se produce o eroare atunci: */
if(rc != SQLITE_OK)
{
    fprintf(stderr, "Acest user nu exista.\n");
    logat=0;
    sqlite3_free(ErrMsg);
}
/* daca nu se produce nicio eroare => userul exista */
else
{
    /* verificam daca userul este abonat la stiri */
```

Instanțieri similare au fost folosite și pentru funcția de sign in: se verifică dacă utilizatorul există în baza de date (dacă da, se solicită alt nume din partea clientului; daca nu), se inserează utlizatorul în baza de date fără abonament la știri. În cele din urmă acesta este întrebat dacă dorește să se aboneze și baza de date va fi updatată.

În diagrama de mai jos este reprezentat sumar cum comunică serverul și clientul, de ce se ocupă fiecare în parte și informațiile stocate la care are acces doar serverul.



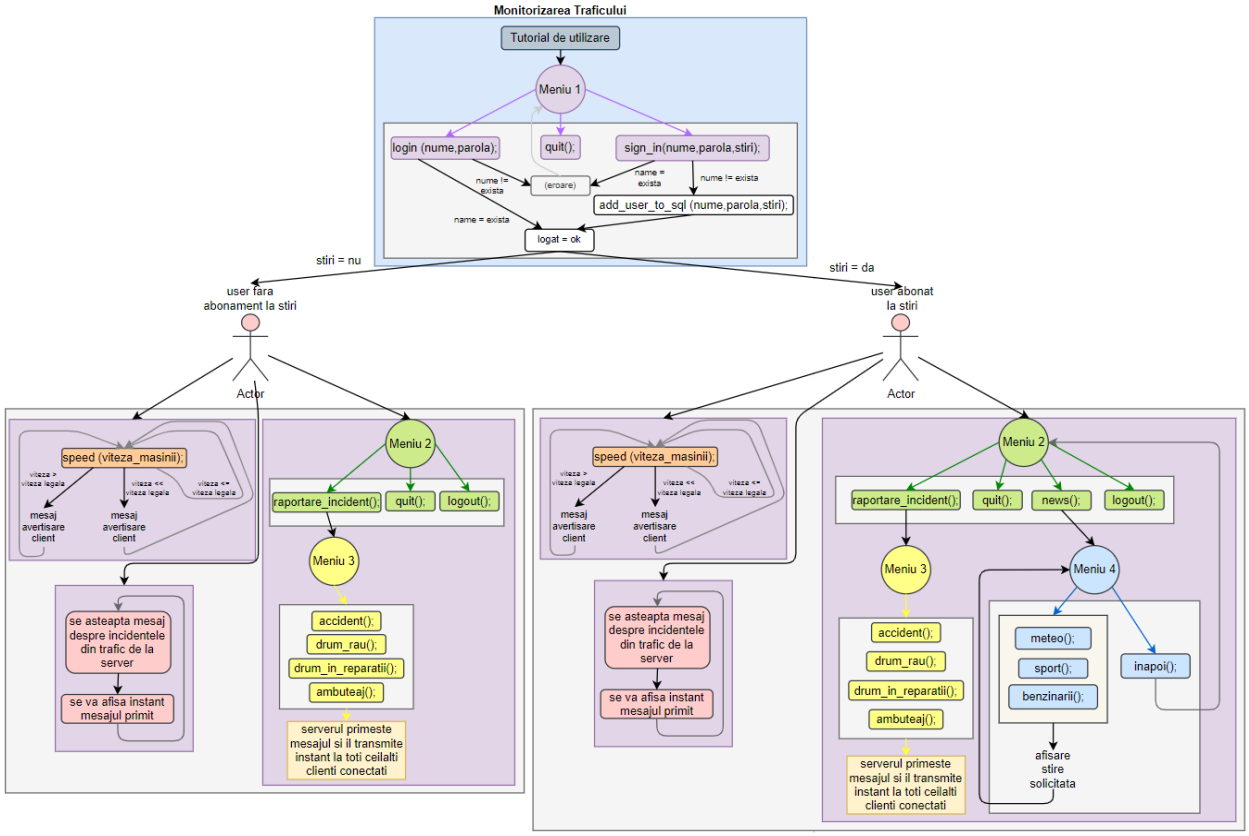
4 Detalii de implementare

Pentru ca aplicația să funcționeze în modul ideal trebuie să se îndeplinească simultan următoarele două condiții:

1. serverul TCP să poată primi și trimite mesaje la și de la client [4,5]

2. funcțiile pe care le voi implementa în program trebuie să respecte diagrama de la capitolul 3.

Vor fi 2 tipuri de utilizatori: cei abonați și cei neabonați la știri. Programul descris va putea fi utilizat conform urmatorului scenariu:



Funcția *login* are rolul de a autentifica utilizatorii ce au introdus un user existent în baza de date. Dacă userul nu există se va afișa un mesaj care să sugereze să se folosească funcția *login* cu alt user, funcția *sign_in* cu userul încercat inițial sau funcția *quit* care închide aplicația.

Funcția *sign_in*, similar cu funcția de *login*, are rolul de a înregistra un nou utilizator în baza de date. În plus acesta este întrebat dacă vrea să se aboneze la știri.

Odată ce clientul este autentificat în aplicație acesta va avea un meniu mai larg de posibilități:

- funcția *raportarea_incidentelor* va primi un mesaj de la client și va avertiza toți ceilalți utilizatori despre pericolul semnalizat;

- funcțiile *meteo*, *sport* și *peco* vor afișa detalii despre vreme, știri din sport sau prețul combustibilului și locația celor mai apropiate 3 stații peco de pe harta;

- funcția *logout* va deconecta utilizatorul și-l va întoarce la meniul principal (*sign_in, login, quit*);

- funcția *quit* va închide aplicația.

Pe lângă aceste funcționalități, odată autentificat, funcția *speed* va oferi șoferului la intervale de aproximativ 1 minut detalii despre viteza cu care circula și dacă a depășit limita legală pe porțiunea de drum pe care se află.

5 Concluzii

Tehnologia aleasă ar fi putut fi îmbunătățită din punctul de vedere al vitezei folosind un server UDP, dar considerând toate caracteristicile celor două tipuri de servere, modelul TCP este mult mai stabil și adecvat situației în cauză.

Soluția prezentată ar putea fi îmbunătățită prin stocarea grafului (ce reprezintă rețeaua stradală) sub forma unei liste, iar limita legală de viteza să fie stocată într-o bază de date care să țină evidența străzilor (nume, coordonate, limita de viteza) și a stațiilor peco de pe acele străzi.

De asemenea știrile meteo și cele din sport ar fi putut fi luate direct de pe site-uri de specialitate cu ajutorul unei crawler web.

GUI-ul pentru client putea fi implementat folosind software-ul Qt în loc de a crea impresia de butoane.

6 Bibliografie

[1] Alboaie, L. (2020a). *Client (TCP concurent).* [online] Uaic.ro. Available at: `https://profs.info.uaic.ro/~computernetworks/files/NetEx/S12/ServerConcThread/cliTcpNr.c`

[2] Alboaie, L. (2020b). *Server TCP concurent.* [online] Uaic.ro. Available at: `https://profs.info.uaic.ro/~computernetworks/files/NetEx/S12/ServerConcThread/servTcpConcTh2.c`

[3] Alboaie, L. and Panu, A. (2015). *Retele de calculatoare — Proiecte propuse.* [online] Uaic.ro. Available at: `https://profs.info.uaic.ro/~computernetworks/ProiecteNet2020.php`

[4] Alboaie, L. and Panu, A. (2020). *Course&Laboratory - Computer Networks 2020-2021.* [online] Uaic.ro. Available at: `https://profs.info.uaic.ro/~computernetworks/cursullaboratorul.php`

[5] Hochstadt, A. (2020). *TCP vs UDP: Understanding the Difference.* [online] vpnMentor. Available at: `https://www.vpnmentor.com/blog/tcp-vs-udp/`

[6] Google maps (2020). *Iași.* [online] Iași. Available at: `https://www.google.com/maps/place/Ia%C8%99i/`