

# Proiect Console Shopper

Jitarasu Catalina-Larisa, Grupa X3

Facultatea de Informatica, Iasi

**Abstract.** Redactarea raportului tehnic al Proiectului Console Shopper  
in cadrul materiei Retele de Calculatoare

## 1 Introducere

Proiectului prezinta o aplicatie client/server realizata in limbajul C/C++, sub Linux care ofera functionalitatea unui magazin online de produse. Aplicatia prezinta un magazin de electrocasnice, unde produsele sunt sortate pe categorii:

-electrocasnice mari(frigider, congelator, masina de spalat vase, masina de spalat haine, uscator de haine etc)

-electrocasnice mici(fier de calcat, aspirator, masina de cusut, espresso, cafetiera etc.)

-gaming(console PS5, console Xbox, jocuri PS5 etc.) etc.

Pentru a putea accesa magazinul, clientul trebuie sa fie autentificat. Autentificarea se va face pe baza unui nume de utilizator care este verificat in baza de date. Clientul poate vizualiza categoriile de produse, produsele din diferite categorii precum si pretul lor. Acesta isi poate adauga produse in cos. El poate plasa comanda, sterge produse din cos etc.

## 2 Tehnologii utilizate

### 2.1 Protocolul TCP

Pentru acest proiect se foloseste Transmission Control Protocol (TCP).

Un protocol, in contextul retelelor de calculatoare, este un set de reguli si proceduri care guverneaza modul in care datele sunt transmise. Scopul unui protocol este acela de a furniza un mod de comunicare pentru oricine din intregul WWW (world wide web), independent de locatie, limbaj, hardware. A fost ales protocolul TCP deoarece este orientat spre conexiune, adica clientul nu se poate conecta daca serverul nu este deschis (deci mereu trebuie sa deschidem serverul prima data). Un alt motiv pentru aceasta alegere este faptul ca legatura este realizata si mentinuta pana cand serverul cat si clientul termina de trimis mesaje. Serverul acestui protocol poate satisface cererile provenite de la clienti in mod iterativ (la un moment dat serverul va deservi cereri provenite de la un singur client, secvential pentru toti clientii lui) sau concurent (mai multe cereri provenite de la clienti multipli vor fi procesate simultan).

Avantajele protocolului TCP sunt:

- asigurarea realizării unei comunicări stabile, permanente în rețea;
- siguranța că informațiile trimise de un proces vor fi recepționate corect, complet și în ordine la destinație sau va fi semnalată o excepție în caz contrar (spre deosebire de UDP unde pachete sunt independente unul față de altul)

Pentru acest proiect am folosit varianta de server TCP concurent pentru a asigura dinamicitatea aplicației. Concurența este asigurată prin `fork()` pentru a eficientiza executia programelor. Serverul TCP va crea câte un proces copil (cu ajutorul primitivei `fork()` pentru fiecare client care dorește să intre pe el, astfel asigurând conectarea mai multor clienți în același timp). Scopul este de a le permite utilizatorilor să poată adăuga produse în cos sau a trimite comenzi în același timp.

Varianta veche: Pentru salvarea informațiilor despre clienți, produse, dar și despre cosurile de cumpărături ale utilizatorilor se vor folosi fișiere text, deci se va lucra cu operații pe fișiere.

Varianta finală: Pentru salvarea informațiilor despre clienți, produse, dar și despre cosurile de cumpărături ale utilizatorilor se vor folosi operații pe baza de date.

## 2.2 MySQL

Sistemul folosit în cadrul proiectului în gestionarea bazei de date este MySQL.

Funcții folosite:

- `mysql_real_connect(conn,server,user,password,database,0,NULL,0)`- folosită pentru a conecta baza de date
- `mysql_query(conn,char*interogare)` - pentru a realiza o interogare în baza de date
- `mysql_error(conn)`
- `mysql_use_result(conn)`
- `mysql_fetch_row(res)`
- `mysql_free_result(conn)`

## 3 Arhitectura Aplicației

La această aplicație se poate conecta un număr oarecare de clienți după deschiderea serverului.

Utilizatorii trebuie să se conecteze pe baza unui nume de utilizator.

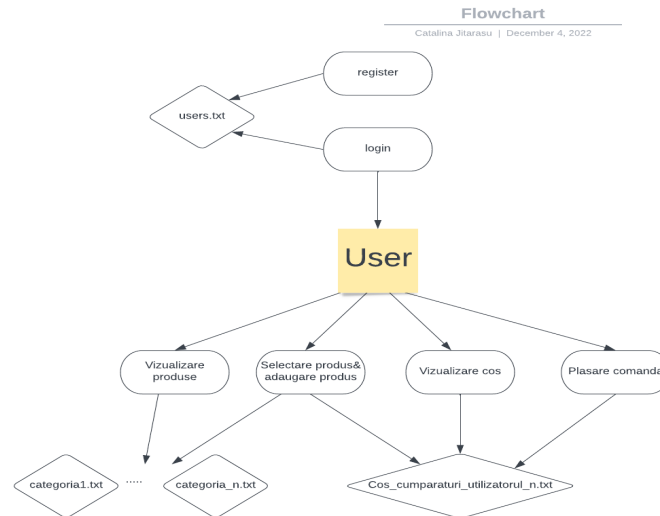
Funcționalitățile oferite de server sunt:

1. logare 'nume\_utilizator'
2. vizualizare categorii disponibile în magazinul online
3. vizualizare produse dintr-o anumită categorie 'nume\_categorie'
4. vizualizarea cosului de cumpărături
5. adăugarea unui produs în cos 'nume\_produs'
6. salvarea cosului de cumpărături atunci când se apelează comanda 'exit'

7. plasarea comenzii
8. exit
9. quit

Serverul TCP al aplicatiei este unul concurrent, deci acesta creaza pentru fiecare utilizator un proces copil in asa fel incat va exista posibilitatea servirii mai multor utilizatori in mod simultan si eficient. Acesta va avea rolul de a prelua datele de la client, de a le procesa si ulterior sa transmita rezultatul inapoi utilizatorului.

Pentru a putea fi folosit efectiv, un socket va trebui atașat la portul mașinii la care serverul va aștepta cereri de conexiune din partea posibililor clienți. Acest lucru se realizează prin intermediul primitivei `bind()`. Pentru oferirea de servicii nestandard, se recomandă folosirea porturilor mai mari decat 1024.



**Fig. 1.** Functionalitatile programului

## 4 Detalii de implementare

Varianta veche:

Funcțiile implementate vor respecta cele două diagramele de mai sus.

Funcția de **login** are rolul de a introduce clientul în aplicație. Clientul se conectează la magazinul online pe baza unui nume de utilizator care este verificat în fișierul `users.txt`. Dacă numele de utilizator nu se regăsește în fișierul `users.txt`, atunci este posibilă înregistrarea noului client prin funcția **register** (care permite adăugarea unui nou utilizator în fișierul `users.txt`).

Aplicația are, de asemenea și comanda de **"exit"** -această comandă va deloga orice client a cerut acest lucru, dar înainte de a-l deloga va apărea pe ecran un mesaj (ex: "Sunteți sigur că doriți să vă deconectați înainte de a salva cosul?" "YES/NO" → "NO" → apelează **funcția de salvare a cosului de cumpărături** pentru că atunci când clientul se conectează din nou la magazin, cosul acestuia este neschimbat; "YES" → **quit**)

Produsele disponibile în magazin pot fi afișate pe ecran prin comenzile:

- vizualizareCategorii()** - care va afișa toate categoriile disponibile
- vizualizareProduse( nume\_categorie )** - care va afișa toate produsele dintr-o anumită categorie

Produsele vor putea fi adăugate în cos prin funcția:

- selectProdus( nume\_categorie, nume\_produs )** - care va adăuga acel produs în cos

Cosul poate fi salvat prin apelarea comenzii:

- salvareCos( nume\_utilizator )** - această funcție are rolul de a salva cosul de cumpărături într-un fișier text (`cosCumparaturi_nume_utilizator.txt`) pentru că acesta să poată fi reutilizat atunci când un client se reloghează pe aplicația magazinului sau pentru a plasa o comandă

Plasarea comenzii în cazul aplicației magazinului nostru este doar o funcție:

- plasareComanda( cosCumparaturi\_nume\_utilizator )** - care plasează comanda clientului.

Toate aceste comenzi/ funcții vor conține și scenarii de eroare, cazuri în care vor apărea mesaje pe ecran (de ex: "Produsul cerut nu se află în magazinul nostru sau în categoria selectată. Reintroduceți un produs: ")

**Varianta noua:**

Clientul introduce pe rand una din urmatoarele comenzi:

1. logare 'nume\_utilizator'

Care verifica daca nume de utilizator introdus de client se regaseste in baza de date deja creata. Daca acesta se regaseste in baza de date atunci acest user poate continua si are acces la mai multe comenzi (acest lucru este verificat prin existenta variabilei status)

2. quit

Prin introducerea acestei comenzi clientul paraseste conexiunea cu serverul.

Dupa conectarea la platforma magazinului, clientul are acces la urmatoarele comenzi:

1. Vizualizare categorii

Prin aceasta comanda se vor afisa toate categoriile din baza de date a magazinului;

2. Vizualizare produse din 'categorie'

Aceasta comanda afiseaza produsele dintr-o anumita categorie dorita de client, precum si pretul fiecarui produs;

3. Adaugare in cos 'nume\_produs'

Prin apelarea acestei comenzi clientul adauga in cosul sau un produs. Adaugarea produsul in baza de date se face in tabela CosCump sub forma 'nume\_produs' si 'nume\_utilizator'.

4. Vizualizare cos

Afiseaza toate produsele adaugate in cos de client, validat prin numele de utilizator. Astfel, din baza de date se extrag produsele care au fost adaugate in tabele CosCump de un anumit 'nume\_utilizator';

5. Plasare comanda

Este o functie care plaseaza comanda la nivel teoretic. Clientul trebuie sa introduca o adresa. Dupa ce este plasata comanda, cosul de cumparaturi al acelui user este golit.

6. Stergere din cos 'nume\_produs'

Stergerea unui anumit produs din cos se realizeaza prin aceasta comanda.

7. exit

Prin aceasta comanda clientul se deconecteaza de la aplicatie; Atunci cand este data aceasta comanda, este oferita userului intrebarea daca doreste sa isi salveze cosul de cumparaturi.

8. quit

Aceasta comanda deconecteaza automat clientul fara a salva cosul de cumparaturi.

Pentru partea de logare a unui utilizator:

Pentru partea de logare a utilizatorilor se foloseste un status pentru a putea tine cont daca un user este sau nu conectat.

#### 4.1 Scenarii de utilizare

Pentru a reda modul in care un client gestioneaza aplicatia magazinului descriem cateva scenarii de utilizare:

```

if(strstr(msgClient,"logare")!=NULL && status==0)
{
    bzero(numa_utilizator,100);
    char *token;
    const char sep[2]=" ";
    token=strtok(msgClient,sep);
    while(token!=NULL){
        if(strstr(token,"logare")==NULL)
            strcpy(numa_utilizator,token);
        token=strtok(NULL,sep);
    }

    if(mysql_query(conn, "select UserName from User")){
        fprintf(stderr, "%s\n", mysql_error(conn));
    }

    res=mysql_use_result(conn);
    while((row=mysql_fetch_row(res))!=NULL)
    {
        if(strstr(numa_utilizator, row[0]))
            {status=1; break;}
        else
            status=0;
    }

    if(status==1){
        strcpy(rspClient,"n\nBine ati venit pe platforma magazinului nostru online! n Aveti la dispozitie urmatoarele comezi:
\n1.Vizualizare categorii n2.Vizualizare produse din 'categoria' n3.Adaugare in cos 'nume_produs' n 5.Stergere din cos
'nume_produs' n5.Vizualizare cos n6.Plasare comanda n7.exit n 8.quit n");
    }
    else {
        strcpy(rspClient, "Ne pare rau! Nu sunteti utilizator al magazinului. Introduceti alt nume de utilizator\n");
    }
}

```

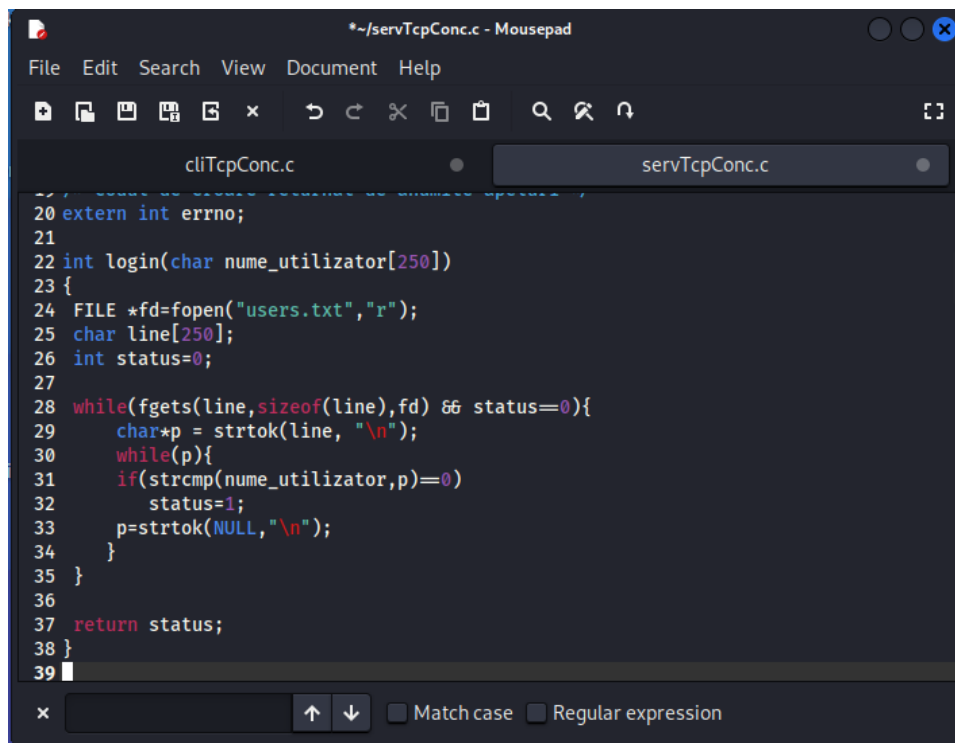
Fig. 2. Noua modalitate de a se loga un client

```

#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <errno.h>
#include <unistd.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <fcntl.h>

```

Fig. 3. Biblioteci folosite

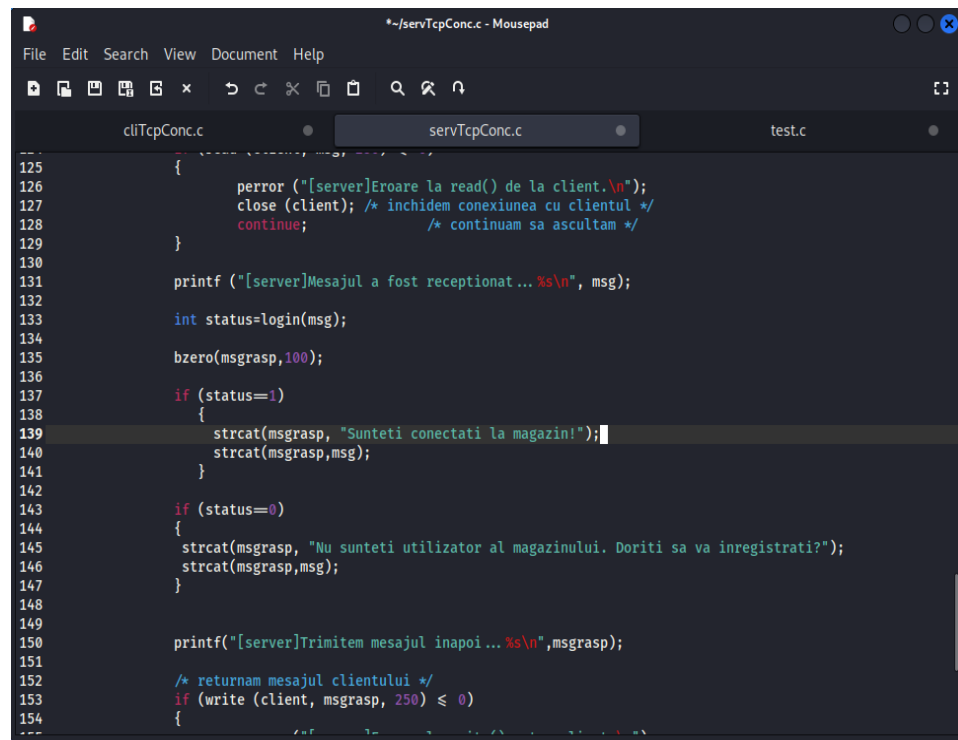


The image shows a screenshot of a text editor window titled "\*/servTcpConc.c - Mousepad". The window has a menu bar with "File", "Edit", "Search", "View", "Document", and "Help". Below the menu bar is a toolbar with various icons for file operations and editing. The editor displays the source code for the `login` function in `servTcpConc.c`. The code is as follows:

```
20 extern int errno;
21
22 int login(char nume_utilizator[250])
23 {
24     FILE *fd=fopen("users.txt","r");
25     char line[250];
26     int status=0;
27
28     while(fgets(line,sizeof(line),fd) && status==0){
29         char*p = strtok(line, "\n");
30         while(p){
31             if(strcmp(nume_utilizator,p)==0)
32                 status=1;
33             p=strtok(NULL, "\n");
34         }
35     }
36
37     return status;
38 }
39
```

At the bottom of the window, there is a search bar with a close button (X), up and down arrow buttons, and checkboxes for "Match case" and "Regular expression".

Fig. 4. Functie care verifica nume de utilizator in fisierul users.txt



```

125     {
126         perror("[server]Eroare la read() de la client.\n");
127         close(client); /* inchidem conexiunea cu clientul */
128         continue;      /* continuam sa ascultam */
129     }
130
131     printf("[server]Mesajul a fost receptionat... %s\n", msg);
132
133     int status=login(msg);
134
135     bzero(msgrasp,100);
136
137     if (status==1)
138     {
139         strcat(msgrasp, "Sunteti conectati la magazin!");
140         strcat(msgrasp,msg);
141     }
142
143     if (status==0)
144     {
145         strcat(msgrasp, "Nu sunteti utilizator al magazinului. Doriti sa va inregistrati?");
146         strcat(msgrasp,msg);
147     }
148
149     printf("[server]Trimitem mesajul inapoi... %s\n",msgrasp);
150
151     /* returnam mesajul clientului */
152     if (write (client, msgrasp, 250) <= 0)
153     {
154         perror("[server]Eroare la write() de la client.\n");
155         close(client);
156         continue;
157     }
158 }

```

Fig. 5. Apelul functiei login(nume.utilizator) in server



### 1. Scenariu de succes

Clientul se logheaza la aplicatie printr-un nume\_utilizator care se regaseste in fisierul users.txt. Acesta doreste sa vizualizeze categoriile magazinului (apelam functia vizualizareCategorii()) , dupa care doreste sa vizualizeze si produsele dintr-o categorie (vizualizareProduse(ume\_categorie)). Selecteaza un produs (selectProdus(ume\_categorie,nume\_produs)) pe care il va adauga in cosul sau (un fisier text care insa nu se salveaza pana cand nu este apelata functia de salvareCos). Clientul decide sa iasa din aplicatie (folosind comanda exit) moment in care pe ecran va apare un mesaj in care clientul trebuie sa decida daca doreste sau nu sa salveze cosul (in cazul nostru il va salva). Dupa o perioada, clientul se relogheaza iar cosul sau este neschimbat si el poate adauga in continuarea produse in cos sau poate plasa comanda (plasareComanda (cosCumparaturi\_ume\_utilizator)).

### 2. Scenariu de esec

Clientul nu se logheaza la magazin si el incearca sa apeleze functii pe care le poate apela doar un client logat la aplicatie, momente in care vor tot apare mesaje de eroare.

## 5 Concluzii

Varianta veche:

Proiectul poate fi imbunatatit atat pe partea de client, cu implementarea unei interfete grafice, dar si pe partea de server. Toate datele legate de utilizatori, cumparaturi ar putea fi salvate intr-o baza de date pentru imbunatatire.

Varianta noua:

Cea mai vizibila imbunatatire este implemenetarea unei interfete grafice, insa pe partea de client ar mai putea fi imbunatatita comanda de "Plasare comanda", astfel incat sa genereze un numar de comanda, sa afiseze mai multe informatii care sa poata fi introduse.

## References

1. [https://profs.info.uaic.ro/computernetworks/files/6rc\\_ProgramareaInReteaII\\_Ro.pdf](https://profs.info.uaic.ro/computernetworks/files/6rc_ProgramareaInReteaII_Ro.pdf).
2. [https://profs.info.uaic.ro/computernetworks/files/7rc\\_ProgramareaInReteaIII\\_Ro.pdf](https://profs.info.uaic.ro/computernetworks/files/7rc_ProgramareaInReteaIII_Ro.pdf).
3. <https://profs.info.uaic.ro/ioana.bogdan/>.
4. <https://profs.info.uaic.ro/andrei/index.php/computernetworks/>.
5. Diagramele au fost construite aici: <https://lucid.app/documents#/dashboard>.