MersulTrenurilor*

Cucuteanu L. Lucian-Andrei(2B3)

Facultatea de Informatică, Strada General Henri Mathias Berthelot 16, Iași 700483
lucian.cucuteanu@info.uaic.ro
https://www.info.uaic.ro/

Abstract. Proiectul constă într-o aplicație client-server în care utilizatorul poate cere sau edita detalii despre anumite rute feroviare sau ale trenului propriu-zis.

Keywords: TCP · Server · Client · XML · CommandQueue · Concurent · Sqlite3 · Threads.

1 Introducere

Proiectul conține un client ce se poate conecta la un server TCP în mod concurent. Clientul citește de la linia de comandă diferite comenzi precum: /**help** pentru a primi instrucțiuni legate de comenzi, **/register id parola** pentru a te inregistra, **/login id parola** pentru a te loga, /**logout** pentru a te deloga, /**quit** pentru a inchide clientul, **/getInfo oras1 oras2** pentru a primi informații despre posibilele rute feroviare dintre cele 2 orașe introduse ca parametri și **/update IDTren StatusTren OraSesizarii** ce actualizează în fișierul XML datele după sesizarea clientului. Pentru parametrii comenzii update, IDTren este de forma "IRxxxx", statusul poate fi: IS (Intarziere Sosire statie plecare) sau IP (Intarziere Plecare), iar oraSesizarii este de forma "xx:xx" sau "x:xx". Comenzile sunt transmise către server, ce le prelucrează pe acestea și trimite înapoi clientului diferite mesaje în funcție de comanda introdusă.

2 Tehnologii Aplicate

Tehnologiile folosite în construcția proiectului sunt:

ServerTCP - Acesta este un sistem ce utilizează Transmission Control Protocol(TCP) pentru a se conecta cu clientul și pentru a trimite și primi date. Totodată, TCP asigură retransmiterea pachetelor în caz de pierdere a acestora. Am folosit tehnologia TCP pentru a avea o conexiune fiabilă între client și server și pentru a mă asigura că toate datele sunt transmise cu succes. Spre deosebire de TCP, User Datagram Protocol(UDP) trimite fără a fi interesesat de starea datelor, unde unele dintre ele se pot pierde.

XML(**ExtensibleMarkupLanguage**) - Această tehnologie este un metalimbaj de marcare ce stocheaza și organizează date într-un mod ușor de înțeles

* Proiect realizat pentru materia Rețele de Calculatoare

atât de computere, cât și de oameni. Am folosit biblioteca libxml2 (XML Parsing Library) pentru a citi/scrie date dintr-un/într-un document XML.

Baza de Date Sqlite3 - Tehnologia Sqlite3 este o bibloteca in limbajul de programare C care implementeaza un motor de baze de date SQL. Am folosit aceasta tehnologie pentru a stoca date ale conturilor clientilor (id si parola criptata).

3 Structura Aplicației

În figura de mai jos (**Fig.1.**) este prezentată diagrama detaliată a aplicației. Conceptele folosite în modelarea aplicației sunt:

Model Multi – Threaded - Acesta constă în utilizarea a două sau mai multe thread-uri care rulează simultan (un server și un client sau un server și mai mulți clienți). Serverul poate gestiona mai mulți clienți simultan, fiecare client fiind deservit de un thread separat.

Server Concurent - Concurența implică execuția mai multor sarcini într-o perioadă de timp suprapusă. În această aplicație, sarcinile sunt concepute să poată fi rezolvate indiferent dacă sarcinile precedente au fost rezolvate sau nu.

Procesarea cererilor trimise de clienți - Clientul trimite către server una dintre cele 7 comenzi implementate (/help, /register, /login, /logout, /quit, /getInfo sau /update) alături de parametrii necesari funcționării. Serverul primește mesajul și verifică dacă comanda este validă. În caz de invaliditate, serverul trimite către client un mesaj corespunzator. Dacă comanda este validă, serverul continuă să proceseze sarcina primită și trimite rezultatul către client. În cazul comenzii update, clientul va primi doar o înștiințare legată de succesul actualizării documentului XML, in timp ce serverul va notifica ceilalti clienti interesati de acel tren. În final, clientul afișează rezultatul primit și este pregătit pentru o nouă sarcină!

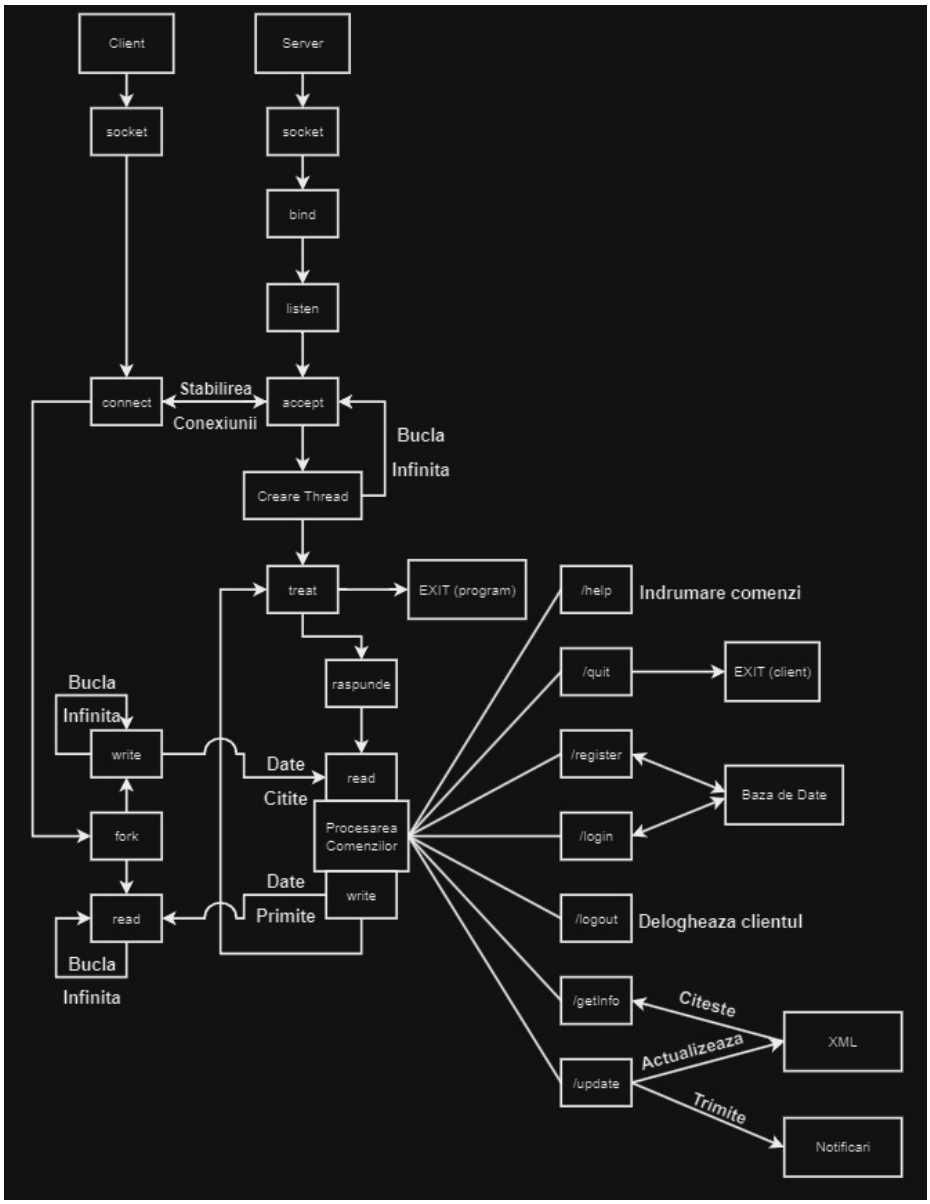

Fig. 1. Diagrama aplicației.

4 Aspecte de Implementare

Doua exemple de trenuri în format XML.

```
<listaTrenurilor>
    <tren id="IR1883">
        <plecare>Iasi</plecare>
        <destinatie>Suceava</destinatie>
        <oraPlecarii>8:22</oraPlecarii>
        <oraSosirii>11:10</oraSosirii>
        <durata>2h 48m</durata>
        <intarziereSosirePlecare>0</intarziereSosirePlecare>
        <intarzierePlecare>0</intarzierePlecare>
    <intarziereSosireDestinatie>0</intarziereSosireDestinatie>
    </tren>
    <tren id="IR1111">
        <plecare>Bucuresti</plecare>
        <destinatie>Cluj-Napoca</destinatie>
        <oraPlecarii>7:00</oraPlecarii>
        <oraSosirii>13:40</oraSosirii>
        <durata>6h 30m</durata>
        <intarziereSosirePlecare>0</intarziereSosirePlecare>
        <intarzierePlecare>0</intarzierePlecare>
        <intarziereSosireDestinatie>0</intarziereSosireDestinatie>
    </tren>
</listaTrenurilor>
```

Definire, deschidere și citire a unui document XML și parcurgerea detaliilor fiecărui $<tren>$ **din** $<listaTrenuri>$**.**

```
if (strcmp(comanda, "getInfo") == 0)
{
    ...
    xmlDoc *document;
    xmlNode *root, *node;
    char *filename;
    filename = "mersulTrenurilor.xml";
    document = xmlReadFile(filename, NULL, 0);
    root = xmlDocGetRootElement(document);
    ...
    for (node = root->children; node != NULL; node = node->next)
    {
        if (node->type == XML_ELEMENT_NODE)
        {
            bzero(trenInfo, 1000);
            strcpy(trenInfo, xmlNodeGetContent(node));
```

```
            ...
        }
    }
    xmlFreeDoc(document);
    xmlCleanupParser();
    ...
}
```

Implemetare Server TCP

```
struct sockaddr_in server;
struct sockaddr_in from;
...
if ((sd = socket(AF_INET, SOCK_STREAM, 0)) == -1)
{
    ...
}
int opt = 1;
setsockopt(sd, SOL_SOCKET, SO_REUSEADDR, (void *)&opt, sizeof(opt));
bzero(&server, sizeof(server));
bzero(&from, sizeof(from));
server.sin_family = AF_INET;
server.sin_addr.s_addr = htonl(INADDR_ANY);
server.sin_port = htons(PORT);
if (bind(sd, (struct sockaddr *)&server, sizeof(struct sockaddr)) == -1)
{
    ...
}
if (listen(sd, 5) == -1)
{
    ...
}
while (1)
{
    ...
    client = accept(sd, (struct sockaddr *)&from, &length);
    if (client < 0)
    {
        ...
    }
    ...
...
```

Căutarea unui tren după ID-ul unic (<*tren* id = "IRxxxx">) în documentul XML și editarea orei sosirii trenului cu ajutorul comenzii update.

```
for (node = root->children; node != NULL; node = node->next)
{
    if (node->type == XML_ELEMENT_NODE)
    {
        char idTrenXml[100];
        strcpy(idTrenXml, xmlGetProp(node, (const xmlChar *)"id"));
        if (strcmp(idTren, idTrenXml) == 0)
        {
            ...
            oraSosirii = xmlFirstElementChild(node);
            while (oraSosirii != NULL &&
            xmlStrcmp(oraSosirii->name, (const xmlChar *)"oraSosirii") != 0)
            {
                oraSosirii = xmlNextElementSibling(oraSosirii);
            }
            xmlNodeSetContent(oraSosirii, (const xmlChar *)oraSesizarii);
            if (xmlSaveFormatFile("mersulTrenurilor.xml", document, 1) == -1)
            {
                fprintf(stderr, "Eroare la salvarea documentului XML.\n");
            }
        }
    }
}
xmlFreeDoc(document);
xmlCleanupParser();
```

Notificarea clientilor interesati de un anumit tren ce nu au actualizat trenul

```
void sendNotification(int client, char *idTren)
{
   ...
    FILE *clientiInteresati;
    clientiInteresati = fopen("clientiInteresati.txt", "r");
    if (clientiInteresati == NULL)
    {
        ...
    }
    int vector[1024] = {0};

    while (fgets(bufferClienti, 4096, clientiInteresati) != NULL)
    {
       ...
        if (vector[atoi(clientID_din_Fisier)] == 1)
        {
            continue;
        }
```

```
            vector[atoi(clientID_din_Fisier)] = 1;
           ...
            if (strcmp(idT, idTren) == 0 && client != atoi(clientID_din_Fisier))
            {
               ...
                size_t marimeMesajRaspuns = strlen(msgrasp);
                if (write(clientIDupdate, &marimeMesajRaspuns, sizeof(size_t)) < 0)
                {
                    ...
                }
                if (write(clientIDupdate, msgrasp, marimeMesajRaspuns) < 0)
                {
                   ...
                }
               ...
            }
        }
        fclose(clientiInteresati);
}
```

Un scenariu real de utilizare: O persoană dorește să călătorească cu trenul din Iași spre Suceava. Aceasta poate verifica de la ce oră pleacă trenurile pe această rută și care este durata călătoriei. Persoana observă că trenul nu este ajuns la timp în stație și poate sesiza întârzierea trenului ce va modifica atât pentru el cât și pentru ceilalți oameni ora plecării și ora sosirii.

5 Concluzii

Posibilă îmbunătățire: Dacă pe o rută (ex. Iași Bacău) nu există un tren care să circule, atunci aplicația va căuta cel mai scurt drum (din punct de vedere al timpului de așteptare și de călătorie al trenului) alcătuit din 2 sau mai multe rute (ex. Iași Vaslui - Vaslui Bacău) care să ducă spre acea destinație.

În concluzie, MersulTrenurilor este o aplicație complexă și interesantă care poate ajuta clienții cu detalii legate de anumite rute feroviare.

References

1. Pagina Cursului - https://profs.info.uaic.ro/~computernetworks/cursullaboratorul.php
2. Linux Man7 - https://man7.org/linux/man-pages/index.html
3. TCP vs UDP - https://www.geeksforgeeks.org/differences-between-tcp-and-udp/
4. XML - https://www.geeksforgeeks.org/xml-full-form/
5. libxml2 - https://gitlab.gnome.org/GNOME/libxml2/-/wikis/home
6. Sqlite3 - https://www.tutorialspoint.com/sqlite/sqlite_c_cpp.htm
7. Threads - https://www.cs.cmu.edu/afs/cs/academic/class/15492-f07/www/pthreads.html