Documentatie proiect - Mersul trenurilor (A)

Ghidoarca Mihail-Adrian 2E4

Facultatea de Informatica Iasi

Decembrie 2022

1 Introducere

In proiectul *"Mersul trenurilor"* vorbim despre o aplicatie utilitara in stil *client/server* care trimite/actualizeaza informatii despre functionarea trenurilor si traseele acestora. Astfel, utilizatorii aplicatiei vor avea acces la urmatoarele 2 actiuni:

1. Sa ceara informatii in legatura cu mersul trenurilor:
- plecari/sosiri trenuri din ziua respectiva.
- plecari/sosiri trenuri din ora respectiva.
- plecari/sosiri trenuri (conform programului, cu intarziere de x minute, cu x minute mai devreme).

2. Sa ofere informatii serverului despre posibile intarzieri sau plecari/sosiri desfasurate mai devreme/mai tarziu.

Toata logica va fi realizata in server, clientul doar cere informatii despre plecari/sosiri sau trimite informatii la server despre posible intarzieri si estimarea sosirilor.

2 Tehnologii utilizate

Pentru realizarea conexiunii server/client, vom utiliza modelul *TCP (Transmission Control Protocol) concurent*, modelul orientat-conexiune care realizeaza transportul fara pierdere de informatii, asigurand primirea acestora in ordinea in care au fost transmise de catre server. Am facut alegerea sa utilizez acest protocol deoarece asigura atat precizie cat si integritatea datelor transportate. Avand in vedere prioritatea ridicata a exactitatii in contextul venirilor si plecarilor trenurilor, o abordare ce foloseste *UDP* este total inconvenabila.

Asiguram concurenta prin folosirea *thread-urilor*, aceasta fiind o metoda rapida si eficienta de a diviza procesele pentru clienti. De altfel, independenta acestor *thread-uri* intre ele ofera posibilitatea clientului de a face request-uri fara a afecta actiunile celorlalti clienti.

Odata cu tehnologiile mentionate mai sus, am ales sa folosesc si o baza de date de tip *XML* ce va contine informatii cendespre mersul trenurilor. Folosirea unei baze de date ne faciliteaza gestionarea si modificarea informatiilor stocate in cadrul acesteia.

3 Arhitectura aplicatiei

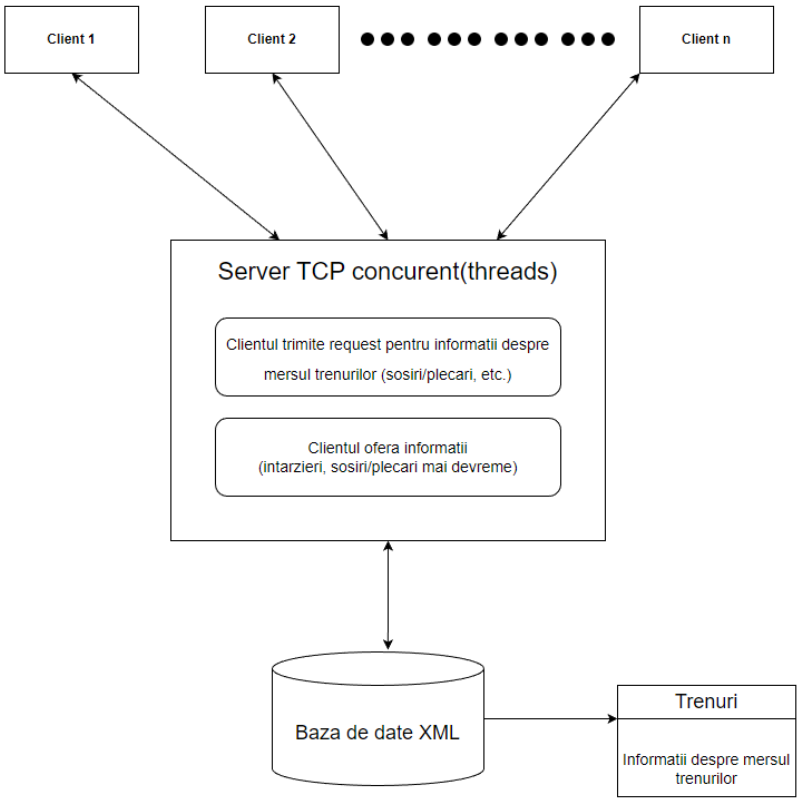

Fig. 1. Diagrama reprezentativa arhitecturii aplicatiei

Aplicatia functioneaza dupa urmatoarea logica:

1. Clientul va trimite o comanda din meniul de comenzi al serverului care va fi afisat odata cu conectarea acestuia.

2. In functie de comanda clientului, acesta va trebui sa adauge anumite informatii aditionale pentru realizarea comenzii.

3. Serverul primeste request-ul si interogheaza baza de date pentru a prelua sau modifica informatiile expediate de client, in functie de request.

4. Dupa interogarea bazei de date, serverul transmite inapoi clientului informatiile cerute, sau un mesaj de confirmare a modificarii acestuia.

5. Clientul are posibilitatea de a mai trimite request-uri catre server, odata ce indeplinirea request-ului anterior a avut succes.

4 Detalii de implementare

Pentru comunicarea dintre client si server, alegem folosirea unui *socket* acesta dovedindu-se mult mai util decat un *fifo* sau un *pipe* deoarece este bidirectional.

Concurenta aplicatiei in implementarea serverului este oferita de uzul *thread-urilor* care se vor crea individual odata cu aparitia fiecarui client nou pentru a-i satisface cererile.

In server vom avea functii dupa urmatoarea structura:
- **sendDelay** : modifica baza de date cu informatii despre intarzierea unui tren pe ruta acestuia
- **getTodayRoutes** : cere informatii despre mersul trenurilor in ziua respectiva, intr-o din statie aleasa de client
- **getHourDeps** : cere informatii despre plecarile din urmatoarea ora dintr-o statie aleasa de client
- **getHourArvs** : cere informatii despre sosirile din urmatoarea ora intr-o statie aleasa de client
- **quit** : clientul cere inchiderea aplicatiei

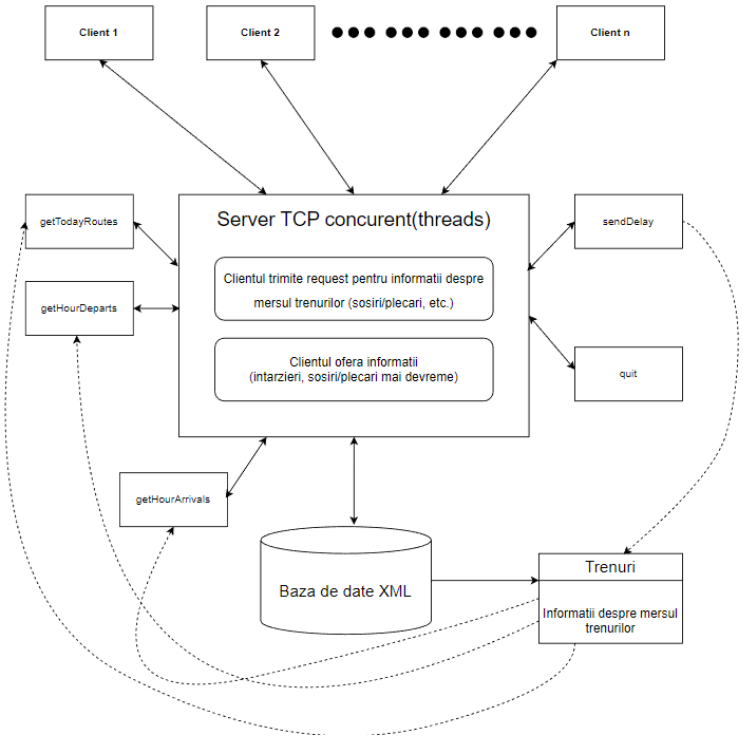

Fig. 2. Diagrama reprezentativa arhitecturii aplicatiei cu detaliile finale de implementare

```cpp
void getTodayRoutes(void *arg) //functia care ne arata ce trenuri vor trece prin
                               // statie in ziua respectiva
{
  struct thData tdL;
  tdL= *((struct thData*)arg);

  char delay[200];
  bool isDelay = false;
  char station[100];
  int count=0;
  time_t raw;
  time(&raw);

  struct tm *time_ptr;
  time_ptr = localtime(&raw);

  char current_time [20];
  strftime(current_time, sizeof(current_time), "%H:%M", time_ptr);

  pugi::xml_document doc;
  pugi::xml_parse_result result = doc.load_file("Trenuri.xml");

  pugi::xml_node trenuri = doc.child("Trenuri");
  pugi::xml_attribute attr = trenuri.attribute("time");

  attr.set_value(current_time);
  doc.save_file("Trenuri.xml");

  if (!result)
  {
    std::cout << "Failed to parse XML: " << result.description() << std::endl;
    perror ("Eroare la deschidere XML\n");
  }
```

Fig. 3. Functia GetTodayRoutes

```xml
<?xml version="1.0"?>
<Trenuri time="15:30">
        <Tren nume="123">
                <Statii>
                        <Statie id="1" loc="Iasi" pOra="05:07" tIntarziere="0" ifIntarziere="false" />
                        <Statie id="2" loc="Pascani" sOra="06:00" pOra="06:15" tIntarziere="0" ifIntarziere="false" />
                        <Statie id="3" loc="Bacau" sOra="07:30" pOra="07:40" tIntarziere="0" ifIntarziere="false" />
                        <Statie id="4" loc="Buzau" sOra="09:20" pOra="09:30" tIntarziere="0" ifIntarziere="false" />
                        <Statie id="5" loc="Ploiestl" sOra="11:00" pOra="11:15" tIntarziere="0" ifIntarziere="false" />
                        <Statie id="6" loc="Bucurestl" sOra="13:00" tIntarziere="0" ifIntarziere="false" />
                </Statii>
        </Tren>
        <Tren nume="234">
                <Statii>
                        <Statie id="1" loc="Bucurestl" pOra="12:07" tIntarziere="0" ifIntarziere="false" />
                        <Statie id="2" loc="Ploiestl" sOra="12:45" pOra="13:00" tIntarziere="0" ifIntarziere="false" />
                        <Statie id="3" loc="Campina" sOra="13:25" pOra="13:30" tIntarziere="0" ifIntarziere="false" />
                        <Statie id="4" loc="Sinaia" sOra="14:00" pOra="14:05" tIntarziere="0" ifIntarziere="false" />
                        <Statie id="5" loc="Predeal" sOra="14:20" pOra="14:40" tIntarziere="0" ifIntarziere="false" />
                        <Statie id="6" loc="Brasov" sOra="15:00" tIntarziere="0" ifIntarziere="false" />
                </Statii>
        </Tren>
        <Tren nume="345">
                <Statii>
                        <Statie id="1" loc="Constanta" pOra="08:25" tIntarziere="0" ifIntarziere="false" />
                        <Statie id="2" loc="Bucurestl" sOra="10:45" pOra="11:00" tIntarziere="0" ifIntarziere="false" />
                        <Statie id="3" loc="Craiova" sOra="16:45" pOra="16:55" tIntarziere="0" ifIntarziere="false" />
                        <Statie id="4" loc="Drobeta" sOra="18:00" pOra="18:25" tIntarziere="0" ifIntarziere="false" />
                        <Statie id="5" loc="Lugoj" sOra="19:00" pOra="19:10" tIntarziere="0" ifIntarziere="false" />
                        <Statie id="6" loc="Timisoara" sOra="20:50" tIntarziere="0" ifIntarziere="false" />
                </Statii>
        </Tren>
        <Tren nume="456">
                <Statii>
                        <Statie id="1" loc="Iasi" pOra="15:25" tIntarziere="0" ifIntarziere="false" />
                        <Statie id="2" loc="Pascani" sOra="16:10" pOra="16:55" tIntarziere="0" ifIntarziere="false" />
                        <Statie id="3" loc="Targu Neamt" sOra="17:35" tIntarziere="0" ifIntarziere="false" />
                </Statii>
        </Tren>
```

Fig. 4. Baza de date XML a aplicatiei

5 Concluzii

Solutia ar putea fi imbunatatita prin:
- implementarea unui **blacklist system** care va intra in uz in momentul in care un user logat incearca sabotarea bazei de date prin transmiterea de informatii false (alternativ, un **administrator al serverului** poate tine evidenta acestor cazuri)
- implementarea unui **GUI atractiv si usor de utilizat**
- **informatii despre situatia capacitatii trenurilor** la ora respectiva (cat de pline sunt)

References

1. https://profs.info.uaic.ro/~computernetworks/cursullaboratorul.php
2. https://app.diagrams.net/
3. https://www.geeksforgeeks.org/tcp-server-client-implementation-in-c/
4. https://www.overleaf.com/learn/latex
5. https://www.crazyengineers.com/threads/user-login-and-registration-using-files-in-c.55378
6. https://igotopia.ro/cum-citesti-si-cum-scrii-fisiere-folosind-limbajul-c/
7. https://dev.mysql.com/doc/