

# Proiect Retele - Mersul Trenurilor

Andra Simion

December 2019

## 1 Introduction

Proiectul ales este intitulat *Mersul Trenurilor(A)*. Acesta reprezinta o platforma de comunicare prin terminal intre *server* si *client*. Clientii pot fi de 3 tipuri:

- **Calatorii** - cer informatii despre mersul trenurilor (de exemplu la ce ora soseste trenul, daca trenul are intarzieri, ce trenuri sunt pe un anumit traseu etc)
- **Panoul** - se afla in gara si isi actualizeaza informatiile in timp real atat cu ora de plecare si de sosire a trenurilor din ziua respectiva cat si cu posibilele intarzieri.
- **Trenul** - trimite date despre intarzieri in timpul traseului pentru a putea fi actualizate si trimise mai departe la clientii de mai sus

Clientii trebuie sa se logheze pentru a avea acces la platforma. In server vor fi implementate toate functionalitatile, iar acesta va trimite rezultatele la client.

## 2 Tehnologii utilizate

Am decis sa implementez un server concurent TCP deoarece clientii trebuie sa fie executati in paralel pentru a nu astepta dupa cel dinaintea lui sa se termine si este nevoie de o comunicare ce se bazeaza pe confirmarea primirii datelor. Este importanta ordinea deoarece vrem ca raspunsurile pachetelor trimise de un client sa ajunga fix la clientul respectiv si nu la altul.(trebuie sa fie posibil ca doi calatori sa primeasca raspunsul cerut pentru mersul trenurilor in acelasi timp).




Serverul TCP va crea un copil nou cu *fork()* pentru fiecare client conectat astfel incat sa existe posibilitatea servirii in mod simultan a acestora. Clientul se va conecta la ip-ul si portul pe care ruleaza serverul. Clientul va fi cel care va afisa rezultatele trimise de server pe ecran.

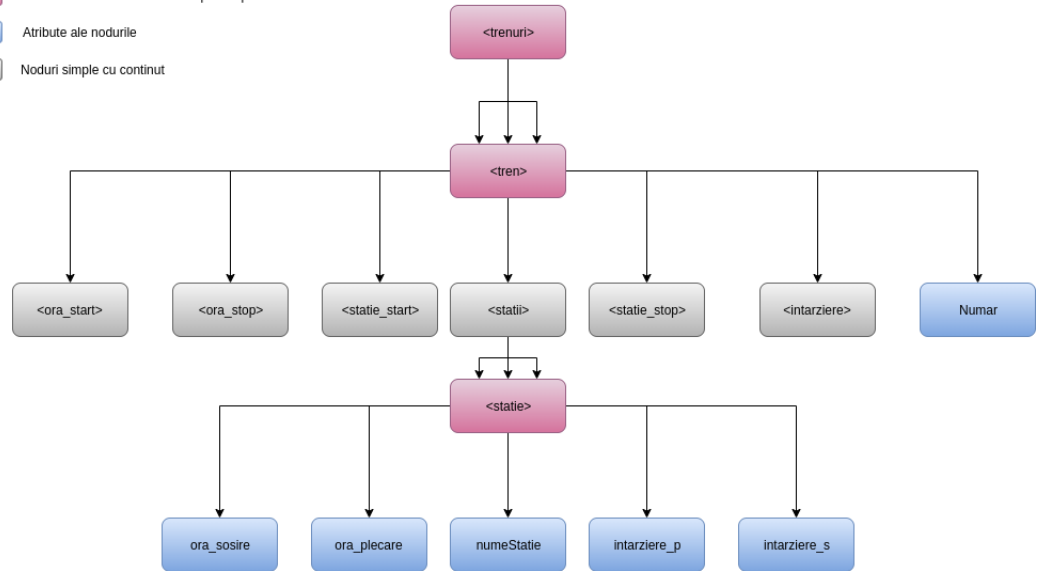
Folosesc fisiere xml pentru a retine datele. Am doua fisiere xml, unul cu datele de autentificare a clientilor(de orice fel) si unul cu mersul trenurilor, oficial postat de CFR, in care sunt descrise toate traseele de pe parcursul anului impreuna cu trenurile asignate si alte detalii.



## Trenuri.xml




### Legenda

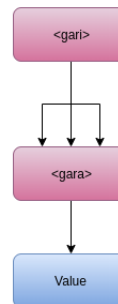
-  Noduri cu unul sau mai multi copii de tipul nodului indicat
-  Atribute ale nodurilor
-  Noduri simple cu continut



## Gari.xml




### Legenda

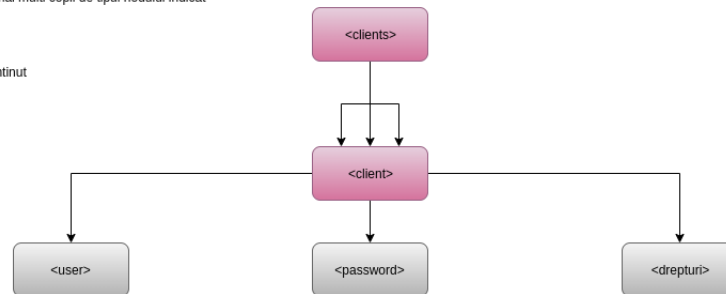
-  Noduri cu unul sau mai multi copii de tipul nodului indicat
-  Atribute ale nodurilor
-  Noduri simple cu continut



## Clients.xml

### Legenda

-  Noduri cu unul sau mai multi copii de tipul nodului indicat
-  Atribute ale nodurilor
-  Noduri simple cu continut



## 4 Detalii de implementare

### 1. Scenarii de utilizare

- In cazul in care utilizatorul se autentifica si greseste username-ul sau parola va avea posibilitatea sa incerce din nou
- In cazul in care utilizatorul vrea sa se deconecteze tasteaza *exit* sau doar *CTR-C*, fara a ramane vreun proces zombie ulterior
- In cazul in care clientul a dat o comanda nerecunoscuta de server, acesta il va avertiza si il va pune sa dea o noua comanda
- Daca serverul pica, sau se intrerupe, utilizatorii vor vi cu totii deconectati fara a li se mai permite sa dea comenzi
- Functia `verify_auth(root, username, password, Read_var, Write_var)` ia ca si parametri username-ul si parola trimise de client si cauta in fisierul `clients.xml` daca exista vreo astfel de pereche salvata pentru a se putea autentifica. Functia returneaza 1 daca o astfel de pereche a fost gasita, 0 in caz contrar. Aceasta mai ia 2 parametri si anume `Read_var` si `Write_var` pentru a putea identifica ce fel de client s-a autentificat, daca are drept de citire, scriere sau ambele.
- Functia `mersul_trenurilor(root, resp)` returneaza prin parametru `resp`, raspunsul pe care il va trimite serverul mai departe la client. Functia parcurge xml-ul cu trenuri si trasee si selecteaza cele mai importante informatii despre fiecare tren si le concateneaza in `resp`.
- Functia `cautaTren(statie_plecure, statie_sosire, response)` primeste ca si parametri statia de unde vrea calatorul sa plece si destinatia si in `response` se vor returna date despre trenurile care merg pe respectivul traseu.

- Functia `infoPlecari(response)` returneaza clientului informatii despre mersul trenurilor care trebuie sa plece in urmatoarea ora intr-o anumita statie.
- Functia `infoSosiri(response)` returneaza clientului informatii despre mersul trenurilor care trebuie sa soseasca in urmatoarea ora intr-o anumita statie.
- Functia `statii_tren(root, nr_tren, msgrasp)` returneaza o lista cu toate statiile prin care trece un anumit tren.
- Functia `infoEstimareSosire(root, nr_tren, statie_initiala, msgrasp)` calculeaza si returneaza ora estimata la care ajunge un tren in statia in care se afla calatorul in functie de intarzieri. (mai devreme sau mai tarziu)
- Functia `update_user(root3, username, password, document3)` introduce un nou user cu username-ul si parola date in fisierul `clients.xml`. Clientii introdusi sunt doar calatori.
- Functia `verify_gara(root2, statie_initiala)` verifica daca statia introdusa este existenta.
- Functia `verif_statie(root, statie_initiala, resp, root)` trimite informatii despre mersul trenurilor care trec prin statia initiala.
- Functia `updateIntarzieri_sosire(root, x + 5, i, intarziere1, document, root)` updateaza fisierul `trenuri.xml` cu intarzierile la sosire pentru statiile de pe traseul unui tren.
- Functia `updateIntarzieri_plecure(root, x + 5, i, intarziere2, document, root)` updateaza fisierul `trenuri.xml` cu intarzierile la plecare pentru statiile de pe traseul unui tren.
- Functia `int updatedData(filename, oldMTime)` primeste ca si parametru fisierul pe care dorim sa-l verificam daca s-a actualizat si cand s-a facut ultima modificare in acesta. O sa returneze 1 daca ultima modificare este mai recenta decat variabila retinuta anterior.
- Conexiunea nu va fi acceptata daca clientul incearca sa acceseze alt ip sau port decat cele pe care ruleaza serverul
- Clientul nu poate da nici o comanda daca nu s-a autentificat
- Clientul trebuie sa se inregistreze cu un username unic, dupa care se poate loga cu acesta.

## 2. Descrierea protocolului

Utilizatorul se conecteaza prin intermediul clientului avand acces la ip-ul si portul pe care ruleaza server-ul. Acesta se logheaza triminand la server username-ul si parola cu care vrea sa se autentifice. Dupa ce s-a autentificat cu succes va fi identificat ca un panou care poate doar citi data din fisierul xml, sau un tren care poate doar scrie in respectivul fisier eventualele intarzieri de la diferite trenuri sau un calator ce poate cere

detalii despre mersul trenurilor. După ce clientul și-a terminat treaba se va deconecta și procesele vor fi omorate.

Toată comunicarea server-client este descrisă astfel: clientul primește comanda, o trimite la server, serverul își folosește funcțiile pentru a obține un răspuns pe care îl trimite ulterior înapoi clientului și acesta îl afișează pe ecran.

Odată conectat clientul trimite o alegere făcută (citită de la tastatură) serverului. Dacă această alegere este diferită de "Login" sau "Register", serverul trimite înapoi "alegere gresită", iar clientul trebuie să o introducă din nou. Dacă clientul a trimis "Register", serverul o citește și trimite înapoi mesajul "REGISTER PLEASE", după care clientul va trebui să introducă un username pe care îl va trimite serverului și acesta îl va valida. Dacă username-ul există deja serverul va trimite înapoi mesajul "username exists" iar clientul va trebui să introducă altul. Dacă usernameul este unic atunci serverul îi va trimite mesajul cu "username ok", iar clientul va trebui să introducă o parolă pe care să o confirme. După ce parolă introdusă corespunde cu cea confirmată, este trimisă la server, iar acesta introduce datele în fișierul cu utilizatori. După ce înregistrarea este făcută cu succes clientul trebuie să se logheze folosind un username și o parolă din baza de date. Dacă alegerea făcută este "Login" serverul va răspunde cu mesajul "LOGIN PLEASE" iar utilizatorul va trebui să introducă username-ul și parola corespunzătoare. Dacă username-ul și parola corespund unei autentificări valide serverul trimite înapoi mesajul "autentificat", respectiv "failed" dacă una din cele două nu se potrivește cu ceva existent.

Odată autentificat clientul, în funcție de datele de conectare primește de la server și mesajul "calător", "panou", respectiv "tren", în funcție de drepturile pe care le deține. Calătorul solicită informații de la server prin intermediul clientului, trenul trimite informații la server pentru a le actualiza, iar panoul își actualizează informațiile în funcție de cele din baza de date.

Dacă clientul este un panou, citește prima dată lungimea mesajului pe care trebuie să îl primească de la server, iar ulterior întregul mesaj de lungimea respectivă. Acest lucru se întâmplă într-o buclă infinită, panoul așteptând modificări în informații deoarece serverul va trimite informații doar după ce s-a făcut o modificare.

Dacă clientul este un tren, acesta își începe traseul odată conectat și se deconectează când ajunge la destinație. Prima dată citește numărul de stații date de server, după care începe să le parcurgă și trimite la server eventualele întârzieri la plecarea sau sosirea într-o stație pe care acesta va trebui să le actualizeze.

Dacă clientul este un calător prima dată va trimite stația inițială în care se alfa pentru a primi informații relevante în următoarele comenzi. De fiecare dată când clientul trimite o stație serverul o verifică să corespundă cu una existentă. În caz afirmativ serverul trimite clientului mesajul "gara ok",

altfel trimite "Statie incorecta". Dupa ce statia introdusa a fost corecta serverul trimite clientului informatii generale despre mersul trenurilor in ziua respectiva si asteapta comenzi de la client in continuare. Comenzile acceptate de server sunt "intarzieri", "plecari", "sosiri", "menu", "exit", "estimare sosire", "statii" si "cauta tren". Pentru orice alta comanda inafara de acestea serverul trimite mesajul "Nu exista comanda. Incercati din nou una din comenzile din "menu"." La comanda "exit", clientul isi inchide conexiunea. Pentru comanda "menu", serverul trimite inapoi o lista cu toate comenzile acceptate. Pentru comanda "intarzieri", clientul trimite numarul trenului pentru care vrea sa stie care minute are intarziere. La comanda "estimare sosire" clientul trimite din nou numarul trenului pentru care vrea sa stie la ce ora va sosi trenul in statia in care se afla in functie de intarzieri. Pentru comanda "statii" clientul trimite numarul trenului pentru ai vedea traseul, adica o lista cu toate statiile prin care trece. Pentru comanda "plecari" clientul trebuie sa trimita numele unei statii pentru care se va afisa ce trenuri pleaca in urmatoarea ora. Pentru comanda "sosiri" clientul trebuie sa trimita numele unei statii pentru care se va afisa ce trenuri sosesc in urmatoarea ora. La comanda "cauta tren" clientul trimita statiile de start si stop pentru calatoria sa, iar serverul ii va returna date despre trenurile care parcurg respectivul traseu.

## 5 Concluzii

Fisierul xml pentru clienti ar putea fi populat cu mai multe date, mai detaliate si reale. Sistemul de inregistrare ar putea avea validari de adrese de mail, parole strong cu restrictii, utilizatorul sa fie anuntat cand se conecteaza cineva de pe contul lui din alta parte, o mapa vizuala pentru trenuri, cu mersul trenurilor in timp real etc. De asemenea, s-ar putea implementa o functionalitate de deconectare a utilizatorului fara a se incheia procesul si de a se putea loga cu alt username imediat dupa. O imbunatatire ar putea si logare unui administrator pe langa calatori, panouri si trenuri, care sa adauge sau sa modifice noi trasee din terminal.

## 6 Bibliografie

<http://www.xmlsoft.org/>  
<https://en.wikipedia.org/>  
<https://stackoverflow.com/>  
<https://mersultrenurilor.infofer.ro/ro-RO/Itineraries>  
<http://man7.org/linux/man-pages/>  
<https://www.draw.io/>