# Offline Messenger

Bucătaru Andreea A2

Facultatea de Informatică, Iași

14 Decembrie 2018

## 1 Introducere

Offline Messenger este o aplicație care permite schimbul de mesaje între utilizatori care sunt conectați și oferă funcționalitatea trimiterii mesajelor și către utilizatorii offline, acestora din urmă apărându-le mesajele atunci când se vor conecta la server. De asemenea, utilizatorii au posibilitatea de a trimite un răspuns (reply) în mod specific la anumite mesaje primite. Aplicația oferă și istoricul conversațiilor pentru și cu fiecare utilizator în parte.

## 2 Tehnologii utilizate

Aplicația trebuie să țină evidența userilor conectați. Având în vedere acest lucru, vom folosi protocolul de comunicare TCP, deoarece acesta ne asigură realizarea unei conexiuni între client și server. TCP asigură ordinea și retransmiterea pachetelor în caz de pierdere, oferind siguranță.

Ca limbaj de programare, aplicația este scrisă în C/C++.

## 3 Arhitectura aplicației

Am folosit principiile programării orientate pe obiect pentru o mai bună organizare și modificare a funcționalităților aplicației.

Aplicația este formată din două părți: server și client.

Serverul va gestiona clienții în mod concurent. De asemenea, acesta se va ocupa de transmiterea datelor preluate de la un client la un altul în două moduri: online sau offline.

Partea de client se va ocupa doar de preluarea inputului de la utilizator și trimiterea acestuia către server pentru a fi procesat. Pe lângă acest lucru, va furniza utilizatorului informații primite de la server.

## 4 Detalii de implementare

Modelul TCP este implementat concurent, folosindu-se *socket*-uri pentru comunicarea între procese.

În server, pentru a gestiona clienții în mod concurent, vom crea un proces separat, cu $fork()$, menit să se ocupe de fiecare client:

```
void Server::listenClients() {
    listen(serverDescriptor, 5);
    while (1) {
      clientDescriptor = accept(serverDescriptor, &from, &length);
      handleClient(clientDescriptor); // servim in mod concurent clientii
    }
}
```

```
void Server::handleClient(int clientDescriptor) {
    if (fork() != 0) return; // parintele continua listenClients
    while (1) {
      myRead(clientDescriptor, user.state);
      if(user.state == 1) { // log in
        myRead (clientDescriptor, user.username);
        myRead (clientDescriptor, user.password);
        int found = checkUsernamePassword(user.username, user.password);
        myWrite(clientDescriptor, found);
      }
      else if(user.state == 2) { // sign up
        myRead (clientDescriptor, user.username);
        myRead (clientDescriptor, user.password);
        int found = checkUsername(user.username);
        myWrite(clientDescriptor, found);
        if(!found)
          addNewUser(user.username, user.password); // il adaug in fisier
      }
      else if(user.state == 3) break; // quit
      else if(user.state == 5) { // chat
        userToChat = chooseUserToChat(clientDescriptor, user);
        chat(clientDescriptor, user, userToChat);
      }
      else if(user.state == 6) to be continued... // view history
      else if(user.state == 7) break; // log out
    }
  }
```

Partea de client se va ocupa de citirea comezilor și procesarea acestora:

```
void Client::listenInput() {
    while(!quited) {
        showMenu();
        cin >> state;
        validateInput(input);
        processInput(input);
    }
}
```

```
void Client::processInput(string input) {
    switch (state) {
        case 1: login(); break;
        case 2: signUp(); break;
        case 3: quit(); break;
        case 4: chat(); break;
        case 5: viewHistory(); break;
        case 6: logOut(); break;
    }
}
```

## 5 Concluzii

*fork*() este un mecanism costisitor din punctul de vedere al timpului, de aceea soluția propusă ar putea fi îmbunătățită folosind *thread*-uri.

Aplicația ar fi mult mai ușor de utilizat dacă partea de client ar avea și o interfață grafică.

Pentru stocarea datelor utilizatorilor și pentru ușurința decodificării acestora, am putea salva informațiile într-o bază de date sau în fișiere de tip *xml*.

## Bibliografie

1. Cursurile de la Rețele de calculatoare: https://profs.info.uaic.ro/~computernetworks/cursullaboratorul.php
2. Laboratoarele de la Rețele de calculatoare: https://profs.info.uaic.ro/~eonica/rc/
3. https://stackoverflow.com/