Documentație: Offline Messenger

Leagăn Dan Adrian 2A5

Ianuarie 2022

1 Introducere

Proiectul Offline Messenger constă in crearea unei aplicații în care se pot loga mai mulți utilizatori, putând să trimită mesaje unui alt utilizator.

Funcționalitățile aplicației sunt vizualizarea comenzilor disponibile, crearea unui cont, logarea, vizualizarea listei cu utilizatorii înregistrați, trimiterea unui mesaj, vizualizarea conversației cu un anume utilizator, vizualizarea mesajelor necitite, raspunderea la un mesaj specific, delogarea și închiderea aplicației.

2 Tehnologii utilizate

2.1 Modelul arhitecturii

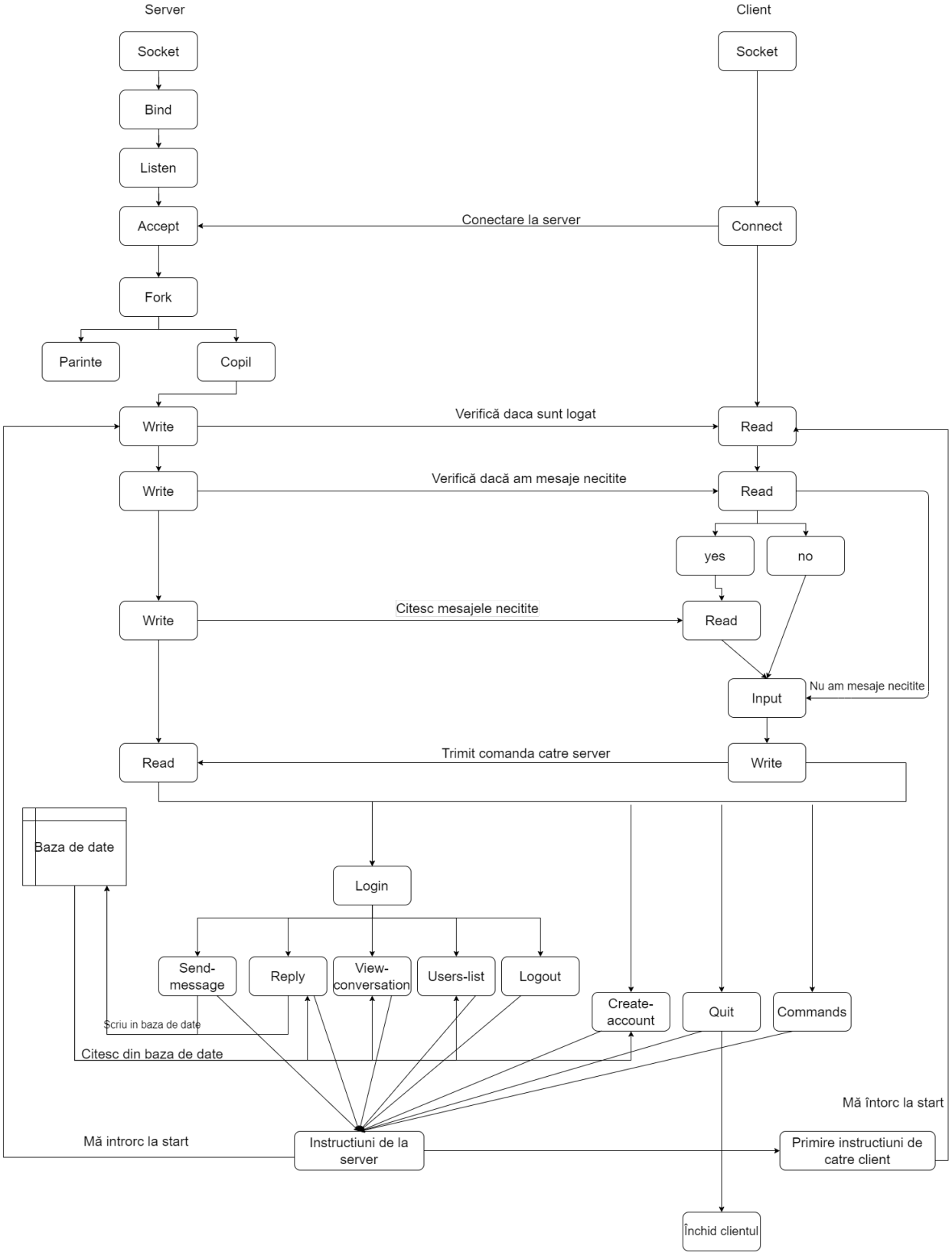
Realizarea conexiunii dintre server si client este constituită de modelul TCP/IP (Transmission Control Protocol / Internet Protocol). Acest model permite conectarea a mai multor utilizatori simultan, servindu-i în mod concurent, verificând datele primite de către clienți, asigurând astfel protecție împotriva pierderii datelor.

2.2 Alegerea modelului TCP în favoarea modelului UDP

Am ales modelul TCP/IP deoarece informațiile trimise de client către server, și invers, trebuie să fie precise și să nu sufere modificări, acest model asigurând că datele nu sunt corupte sau pierdute în timpul trimiterii, verificând pachetele pentru erori.

3 Arhitectura aplicatiei

3.1 Diagrama aplicației



3.2 Stocarea datelor

Aplicația stochează datele intr-o bază de date, folosind sqlite3, instalat pe sistemul de operare. Aplicația deschide baza de date în timpul rulării, făcând modificări sau extrăgând date necesare pentru realizarea funcționalităților sale.

Baza de date rămâne cu modificările făcute de aplicație, și după inchiderea acesteia, salvând conversațiile și conturile utilizatorilor până la următoarea accesare. Baza de date conține două tabele, "utilizatori", care conține username-urile și parolele utilizatorilor care și-au creat cont până in momentul respectiv, și "mesaje" care stochează mesajele utilizatorilor, destinatarul și expeditorul mesajului respectiv, statusul mesajului (dacă a fost văzut de către destinatar sau nu) și mesajul anterior pentru care s-a dat reply (dacă mesajul curent nu a fost dat ca reply, această coloana va avea în baza de date valoarea NULL).

3.3 Conceptele aplicației

După ce sunt deschise serverul și clientul, clientul așteaptă de la tastatură o comandă (ex: Login, Send-message). Unele comenzi sunt disponibile doar după ce a fost efectuată logarea (ex: Reply) iar altele sunt disponibile oricând (ex: Quit). După ce a fost introdusă comanda, clientul trimite această comandă la server, aceasta recunoscând-o și prelucrând corespunzător informațiile la care are acces prin baza de date. După prelucrare, trimite înapoi la client instrucțiuni, clientul afișând pe ecran în funcție de instrucțiunile primite.

4 Detalii de implementare

4.1 Conectarea clientului

După ce clientul se conectează la server, acestuia i se creează un proces copil care se va ocupa de client, urmând ca acesta să trimită înapoi datele cerute, sau sa modifice baza de date. Clientul primește de la tastatură o anumită comanda, pe care serverul o va recunoaște și va deschide apoi baza de date. Dacă un al doilea client se conectează la server, în timp ce primul client incă este conectat, i se va crea un nou proces copil care va primi comenzile sale.

4.2 Conectarea la baza de date

Pentru conectarea la baza de date sqlite3, s-au folosit următoarele funcții:

```
rc = sqlite3_open("data6.db", &db);
sql = "SELECT * from utilizatori";
rc = sqlite3_exec(db, sql, callback, (void *)data, &zErrMsg);
```

4.3 Comanda "Login"

După ce a fost introdusă comanda "Login", aplicația va aștepta de la tastatură un nume de utilizator, după care se va conecta la server și va verifica în baza de date dacă există acest username. După acest pas, clientul va aștepta de la tastatura parola, și va verifica dacă aceasta corespunde numelui de utilizator introdus anterior. Dacă aceasta corespunde, utilizatorul va fi logat în aplicație.

```
if (strcmp(msg, "Login") == 0)
{
  printf("Intrduceti un nume de utilizator:\n");
  printf("  >>");
  bzero(msg, 100);
  scanf("%s", msg);
  write(sd, msg, 100);
  read(sd, &raspunsint, 4);
  if (raspunsint == 0)
    printf("Nu a fost gasit un utilizator cu acest nume!\n");
  else
  {
    printf("Introduceti parola:\n");
    printf("  >>");
    bzero(msg, 100);
    scanf("%s", msg);
    write(sd, msg, 100);
    read(sd, &raspunsint, 4);
    if (raspunsint == 1)
    {
      printf("\nBine ati revenit in contul dumneavoastra!\n\n");
    }
    else
      printf("\nParola gresita. Incercati din nou!\n\n\n");
  }
}
```

4.4 Comanda "Logout"

La comanda "Logout", serverul va schimba variabila ce reține dacă clientul este logat, și va transmite o înștiințare către client.

```
if (strcmp(msg, "Logout") == 0)
{
    if (loggedin == 1)
    {
        raspunsint = 1;
        loggedin = 0;
        da = 0;
        da2 = 0;
    }
    else
        raspunsint = 0;
    write(client, &raspunsint, 4);
}
```

4.5 Comanda "Users-list"

Pentru fiecare comandă, serverul reține numele și parolele utilizatorilor într-o matrice de caractere (actualizând-o la fiecare comandă introdusă). La comanda "Users-list", acesta va forma un șir de caractere cu toți utilizatorii din matrice, trimițandu-l înapoi în client pentru afișare.

```
if (strcmp(msg, "Users-list") == 0)
{
    if (loggedin == 1)
    {
        raspunsint = 1;
        write(client, &raspunsint, 4);
        bzero(msgrasp, 100);

        for (int k = 0; k <= lungime_username; k++)
        {
            strcat(msgrasp, username[k]);
            strcat(msgrasp, "\n");
        }
        write(client, msgrasp, 100);
    }
    else
    {
        raspunsint = 0;

        write(client, &raspunsint, 4);
    }
}
```

4.6 Comanda "Send-message"

Comanda "Send-message" așteaptă de la tastatură numele unui utilizator căruia dorim să-i trimitem un mesaj. Clientul trimite numele către server, pentru verificare în baza de date. Dacă există un utilizator, clientul va aștepta mesajul, trimițandu-l în server, pentru a-l introduce în baza de date în tabelul "mesaje".

```
if (strcmp(msg, "Send-message") == 0)
{
  read(sd, &raspunsint, 4);
  if (raspunsint == 1)
  {
    printf("\nIntroduceti destinatarul: \n");
    printf("  >>");
    bzero(destinatar, 100);
    scanf("%s", destinatar);
    write(sd, destinatar, 100);
    read(sd, &raspunsint, 4);
    if (raspunsint == 1)
    {
      printf("\nIntroduceti mesajul:\n");
      printf("  >>");
      bzero(mesaj, 500);
      read(0, mesaj, 500);
      write(sd, mesaj, 500);
    }
    else
    {
      printf("\nNu exista un destinatar cu acest username\n");
    }
  }
  else
    printf("Trebuie sa fiti logat pentru a trimite un mesaj\n\n");
}
```

4.7 Comanda "Create-account"

Această comandă primește de la tastatura un username, și verifică baza de date pentru a nu exista un utilizator cu același nume. Dacă nu există, aplicația așteaptă o parolă, și apoi inserează în baza de date, trecând prima dată prin server, noul cont creat.

```
if (strcmp(msg, "Create-account") == 0)
{
create:
  printf("Introduceti username nou:\n");
  printf("  >>");
  bzero(nou, 100);
  scanf("%s", nou);
  write(sd, nou, 100);

  read(sd, &raspunsint, 4);
  if (raspunsint == 0)
  {
    printf("Exista deja un utiliazator cu acest nume\n");
    goto create;
  }
  else
  {
    printf("\nIntroduceti parola noua:\n");
    printf("  >>");
    bzero(nou, 100);
    scanf("%s", nou);
    write(sd, nou, 100);
    bzero(nou, 100);
    read(sd, nou, 100);
    printf("%s\n", nou);
    goto _start;
  }
}
```

4.8 Comanda "View-conversation"

La această comandă, serverul, după ce este contactat de client, deschide baza de date, și își formează/actualizează o matrice în care reține mesajele din baza de date, având grijă ca destinatarul și expeditorul să fie cei indicați de către client. După acest pas, trimite fiecare mesaj înapoi la client pentru afișare, și actualizează coloana "VĂZUT" în "1" pentru fiecare mesaj trimis și afișat.

```
k1 = 0;
rc = sqlite3_open("data6.db", &db);
sql = "SELECT * from mesaje";
rc = sqlite3_exec(db, sql, callback_conversatie, (void *)data, &zErrMsg);

write(client, &lungime_mesaje, 4);
for (int k1 = 0; k1 < lungime_mesaje; k1++)
{
    write(client, mesaje[k1], 500);
}

rc = sqlite3_open("data6.db", &db);
sql = sqlite3_mprintf("UPDATE mesaje set VAZUT = 1 where DESTINATAR= '%s' AND EXPEDITOR= '%s'", nume_user, nume_user_aux);
rc = sqlite3_exec(db, sql, callback_insert, (void *)data, &zErrMsg);
```

4.9 Comanda "Reply"

Codul de mai jos reprezintă momentul în care serverul a primit de la client mesajul pentru care se va da reply și mesajul cu care se va da reply. Serverul își extrage destinatarul prin for, având apoi toate datele neccesare pentru a actualiza tabela mesaje cu noul mesaj.

```
int i = 0;
for (i = 0; i < strlen(mesaje_reply[raspunsint]); i++)
{
    if (mesaje_reply[raspunsint][i] == ':')
        break;
    destinatar[i] = mesaje_reply[raspunsint][i];
}
destinatar[i] = '\0';

rc = sqlite3_open("data6.db", &db);
sql = sqlite3_mprintf("INSERT INTO mesaje (EXPEDITOR,DESTINATAR,MESAJ,VAZUT,REPLIED_TO) VALUES ('%s', '%s', '%s', 0,'%s');", nume_user, destinatar, mesaj_aux, mesaje_reply[raspunsint]);
rc = sqlite3_exec(db, sql, callback_insert, 0, &zErrMsg);
goto _start;
```

4.10 Scenarii de utilizare

4.10.1 Scenariul 1: Utilizator nu are cont

Dacă utilizatorul nu are cont, acesta poate să își creeze unul ("Create-account"), să vizualizeze comenzile aplicației ("Commands") sau să închidă aplicația ("Quit"). Va avea acces la comanda "Login", însă nu se va putea loga, și nu va avea acces la comenzile: Users-list, Send-message, Reply, View-conversation, Logout.

4.10.2 Scenariul 2: Utilizatorul are cont

Utilizatorul se poate loga în cont ("Login") pentru pentru a utiliza una din comenzile: Users-list, Send-message, Reply, View-conversation, Logout, sau poate folosi, fără a fi logat, comenzile: Create-account, Commands, Quit.

4.10.3 Scenariul 3: Utilizatorul abia logat are mesaje necitite

Utilizatorului îi va apărea o notificare despre faptul că are mesaje necitite, și va fi întrebat dacă dorește să le vizualizeze. Acesta poate introduce de la tastatură comanda "y" pentru vizualiza mesajele, sau "n" pentru a continua utilizarea aplicației.

4.10.4 Scenariul 4: Utilizatorul abia logat nu are mesaje necitite

Utilizatorul va fi informat cum că nu are mesaje necitite.

4.10.5 Scenariul 5: Utilizatorul introduce o comandă necunoscută

Va fi afișat pe ecran faptul că nu există o comandă precum cea introdusă.

4.10.6 Scenariul 6: Utilizatorul introduce un username greșit la comenzile "Send-message", "View-conversation"

Se va înștiința faptul că nu există un utilizator cu username-ul introdus.

4.10.7 Scenariul 7: Utilizatorul delogat introduce comanda "Logout"

Utilizatorul va fi informat că trebuie să fie logat într-un cont pentru a se putea deloga.

5 Concluzii

În concluzie, pentru a sumariza codul, clientul se conectează la server prin TCP/IP. După conectare, clientul primește o comandă de la tastatură și îl trimite serverului, această comandă fiind recunoscută atât de server cât și de client. Mesajele și utilizatorii sunt stocați într-o bază de date, unde se fac interogări și modificări în funcție de comandă. După ce comanda a fost efectuată, serverul se întoarce, așteptând o altă comandă de la client, iar clientul așteaptă o comandă introdusă de la tastatură.

Soluția ar putea fi îmbunătățită prin:
-O interfață grafică destinată clientului care să-i permită deschiderea mai multor conversații simultan.
-Posibilitatea de a șterge un mesaj trimis.
-Posibilitatea de a modifica un mesaj trmis.
-Posibilitatea unui utilizator de a-și șterge contul.

6 Bibliografie

https://www.tutorialspoint.com/sqlite/sqlite_c_cpp.htm

https://www.razorsql.com/articles/sqlite_linux.html

https://profs.info.uaic.ro/~computernetworks/cursullaboratorul.php

https://profs.info.uaic.ro/~gcalancea/Laboratorul_7.pdf

https://stackoverflow.com/questions/2942370

https://www.sqlite.org/datatype3.html

https://unix.stackexchange.com/questions/84813

https://stackoverflow.com/questions/24194961