Offline messenger

Maxim Maria-Valeria,2A3

January 10, 2024

1 Introducere

1.1 Descriere proiect

Proiectul Offline Messenger reprezintă o aplicație client/server dedicată schimbului de mesaje între utilizatorii conectați, dar și trimiterii mesajelor utilizatorilor online.

Utilizatorii vor avea opțiunea de a răspunde în mod specific la mesajele primite și de a vizualiza istoricul conversațiilor.

1.2 Viziune și obiective

Offline Messenger este o platformă esențială de comunicare, ce va deveni rapid un serviciu de rețea extrem de apreciat pentru modul în care facilitează interacțiunea dintre utilizatori. Această aplicație se distinge prin funcționalități inovatoare, punând un mare accent pe conectivitate. O caracteristică remarcabilă este capacitatea de a permite utilizatorilor să trimită mesaje celor care nu sunt conectați în acel moment, asigurându-se că acestea sunt livrate imediat ce destinatarii se reconectează.

Printre funcționalitățile valoroase ale Offline Messenger se numără și accesul la un istoric complet al mesajelor, facilitând astfel o mai bună organizare a conversațiilor. Mai mult, aplicația oferă o opțiune ingenioasă de răspuns direct la mesaje specifice.

2 Tehnologii aplicate

Protocolul TCP concurent

Implementarea protocolului TCP concurent în cadrul proiectului Offline Messenger aduce o serie de avantaje esențiale. Printre acestea se numără garantarea fiabilității în livrarea mesajelor, asigurându-se astfel că fiecare mesaj ajunge la destinație. De asemenea, protocolul contribuie eficient la gestionarea fluxului de date și la prevenirea congestiei rețelei, dar si reasamblarea mesajele în mod corect la nivelul aplicației, esențială pentru menținerea integrității comunicării. Comunicarea bidirecțională este asigurată prin socketuri. Mai mult, TCP asigură ordinea garantată a livrării mesajelor.

Threadurile

Pentru a asigura concurența la nivelul serverului vom utiliza threaduri. Acest mecanism permite execuția simultană a mai multor sarcini.De asemenea, threadurile pot fi utilizate și pentru separarea activităților la nivelul serverului .De exemplu, citirea continuă în client care permite si trimiterea instantă a mesajelor de la un user către altul se va face separat în client, dar și tratarea concurentă a clienților în server.

Baze de date SQL

Pentru a simplifica gestionarea datelor și dezvoltarea aplicație am implementat un motor de baze de date SQL, prin intermediul bibliotecii SQLite. SQL oferă un limbaj puternic pentru interogarea bazelor de date. Acesta facilitează căutarea ,filtrarea și extragerea informațiilor într-un mod eficient, ceea ce este necesar într-o aplicație de mesagerie.

3 Structura aplicației

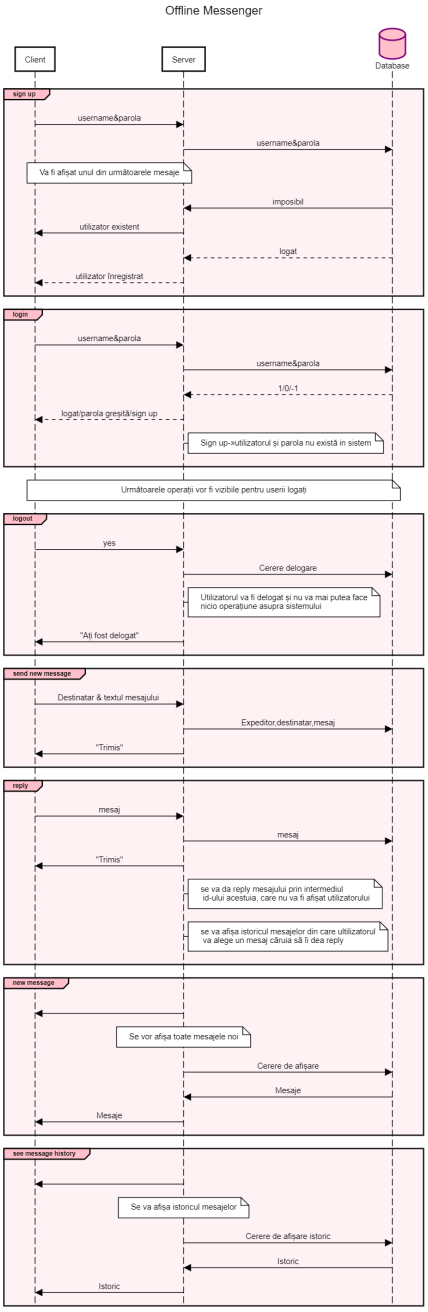


Figure 1: Structura aplicatiei

La nivelul aplicației am utilizat următoarele concepte:

1.**Users(Utilizatorii)**→reprezintă entitatea de bază a aplicației, fiecare utilizator având un username unic și o parolă care sa îi permită conectarea la server.

2.**Conversații**→O conversație reprezintă schimbul de mesaje între 2 utilizatori ,fiecare conversație va avea un identificator unic și trebuie păstreze și mesajele anterioare.

3.**Mesaje**→Mesajele sunt entități care conțin textul transmis între utilizatori. Acestea ar trebui să conțină informații despre expeditor, destinatar, conținut și timestamp (pentru implementarea acestora vom folosi un struct care sa conțină aceste informații și vom stoca mesajele în baza de date pe baza acestor criterii) .

4.**Server**→Serverul este componenta ce gestionează comunicarea dintre clienți și stocheaza mesajele în baza de date.

5.**Client**→Clientul este cel care se ocupă de cererile utilizatorilor, fiecare utilizator se va conecta prin intermediul unui nou client la server și va alege din meniu de comenzi succesiv câte o comandă.

5.**Baza de date**→O bază de date este utilizată pentru a stoca permanent datele, inclusiv utilizatori(username&parola), conversații și mesaje.

Structura bazei de date:

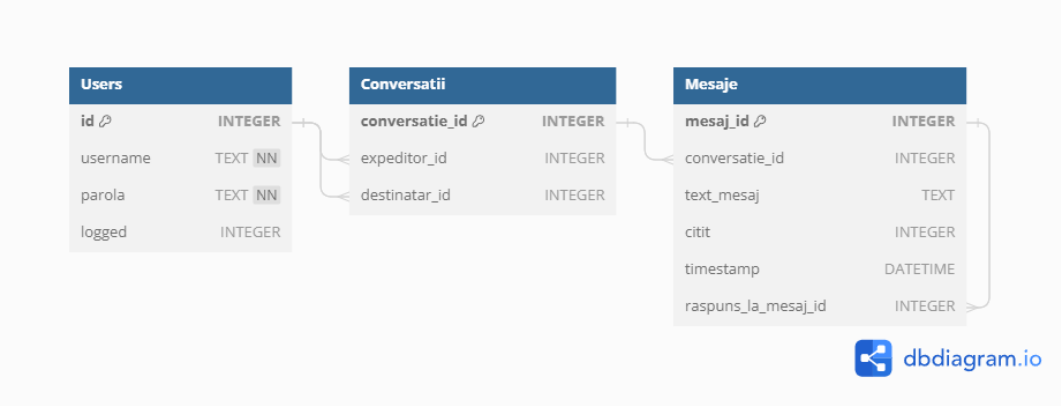


Figure 2: Structura bazei de date

4 Aspecte de implementare

Conexiunea dintre clienți și server se face prin intermediul socketurilor. Socketurile permit comunicații bidirecționale "full-duplex", ceea ce înseamnă că datele pot fi trimise și primite în ambele direcții simultan.

Socketurile permit aplicațiilor să comunice între ele, indiferent dacă rulează pe aceeași mașină sau pe mașini diferite conectate la o rețea. În cazul Offline Messenger, aceasta înseamnă posibilitatea de a trimite mesaje între server și clienți diferiți.

```
/* crearea unui socket */
if ((sd = socket (AF_INET, SOCK_STREAM, 0)) == -1)
  {
    perror ("[server]Eroare la socket().\n");
    return errno;
  }
```

Figure 3: Crearea unui socket

Pentru aplicațiile care funcționează pe modelul client/server, socketurile sunt mecanismul ideal de comunicare. Serverul poate gestiona simultan conexiuni multiple de la clienți, fiecare având un socket asociat.

Funcția bind() este folosită pentru a stabili portul și adresa la care serverul va accepta conexiuni de la clienți. Fără această etapă, clienții nu ar ști la ce adresa sau port să se conecteze pentru a comunica cu serverul.

```
/* atasam socketul */
if (bind (sd, (struct sockaddr *) &server, sizeof (struct sockaddr)) == -1)
  {
    perror ("[server]Eroare la bind().\n");
    return errno;
  }
```

Figure 4: Legarea de server

De asemenea, pentru ascultarea conexiunilor cu clienții am utilizat funcția listen().

```
/* punem serverul sa asculte daca vin clienti sa se conecteze */
if (listen (sd, 2) == -1)
  {
    perror ("[server]Eroare la listen().\n");
    return errno;
  }
```

Figure 5: Ascultarea conexiunilor cu clienții

Serverul utilizează thread-uri pentru a gestiona multiple conexiuni de la clienți în mod concurent. Codul relevat în server.c arată cum se creează un thread nou pentru fiecare client conectat:

```
pthread_create(&th[i], NULL, &treat, td);
```

De asemenea, clientul utilizează un thread separat pentru a citirea mesajelor, dar și a răspunsurilor primite de la server.Codul relevant pentru crearea acestui thread este:

```
if (pthread_create(&thread_id, NULL, continuouslyReadFromSocket, (void*)&sd) != 0) {
    perror("Failed to create thread");
    return 1;
}
```

Cea mai importantă funcționalitate a aplicație este cea de trimitere instant a mesajulor între utilizatorii conectați. Pentru a implementa acest lucru, am contruit o listă în care am păstrat id-ul acestora și socket-ul la care sunt conectatați.

```c
typedef struct {
    int sd;     // Socket descriptor
    int userId; // User ID
} LoggedUser;//structura utilizata pentru salvarea intr-o lista a userilor conectati la sistem

typedef struct Node {
    LoggedUser user;
    struct Node* next;
} Node;//definirea unui nod din lista de useri
```

Iar funcția de trimitere a mesajelor în cazul în care un utilizator este conectat este următoarea:

```c
/*FUNCTIA DE TRIMITERE A MESAJELOR PRIN INTERMEDIUL SOCKETULUI*/

void sendMessageBySocket(Node* User,char *mesaj)
{
    int sd=User->user.sd;
    ssize_t bytes_written = write(sd, mesaj, strlen(mesaj));
    if (bytes_written < 0) {
        perror("Failed to write to socket");

    }

}
```

Restul comenzilor au fost efectuate utilizând diferite interogări utilizate asupra bazei de date în care au fost salvate atât mesajele cât și informațiile utilizatorilor. Spre exemplu, acesta este modul în care am manipulat efectuarea interogărilor în program:

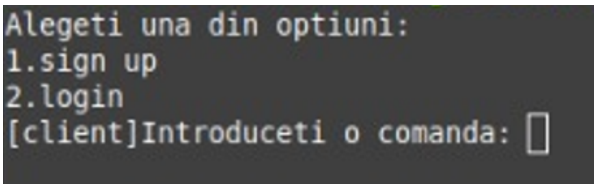
```c
sqlite3 *db;
int rc, found = 0;
sqlite3_stmt *stmt;

rc = sqlite3_open("MessagingDB.db", &db);

const char *sql = "INTEROGARE CE CONTINE ? ";/*INTEROGARE*/

rc = sqlite3_prepare_v2(db, sql, -1, &stmt, NULL);//pregatirea interogarii

/*legarea valorilor la parametri intr-o declaratie pregatita*/
sqlite3_bind_text(stmt, 1, username, -1, SQLITE_STATIC);//pentru valori de tipul siruri de caractere
sqlite3_bind_double(stmt, 4, var);//pentru valori float/double
sqlite3_bind_int(stmt, 3, value);

if (sqlite3_step(stmt) == SQLITE_ROW) {
    ...
}

sqlite3_finalize(stmt);
sqlite3_close(db);
```

Pentru a oferi o experiența plăcută utilizatorului după logarea in baza de acesta va primi un meniu de comenzi din care poate alege acțiunea dorită.

Meniul de comenzi

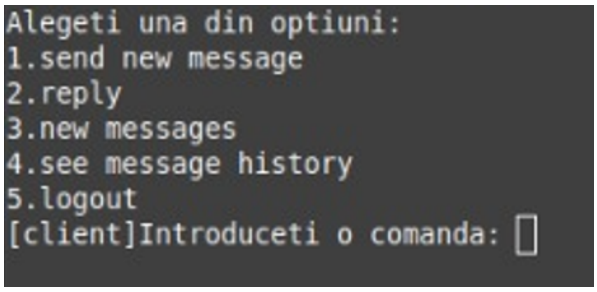
În aplicația Offline Messenger, meniul oferit utilizatorilor pe partea de client include o serie de opțiuni care permit navigarea și utilizarea diferitelor funcționalități ale sistemului.

Meniul de logare



- **Sign up**→Această comandă permite utilizatorului să își creeze un nou cont în aplicație. Dacă acesta introduce un nume existent i se va sugera să îl modifice deoarece este deja existent.
- **Login**→Acestă comandă îi va cere utilizatorului să introducă username-ul si parola pentru a se putea conecta la sistem.

Meniul de comenzi



- **Send new message**→Această opțiune permite utilizatorului să trimită un mesaj unui alt utilizator care este in sistem, indiferent dacă acesta este conectat la server sau nu. Dacă utilizatorul este conectat mesajul apare automat în sesiunea utilizatorului respectiv.
- **Reply**→Prin intermediul acestei opțiuni utilizatorul poate da reply unui mesaj specific al unui anumit utilizator. Acesta va scrie numele utilizatorului caruia vrea sa ii dea reply la mesaj, iar apoi îi va fi prezentată o listă a tuturor mesajelor din acea conversație împreună cu id-ul fiecărui mesaj. Utilizatorul trebuie să introducă id-ul mesajul căruia dorește să răspundă.
- **New messages**→Prezintă mesajele care au fost primite cât timp utilizatorul nu a fost logat.
- **See message history**→Prezintă istoricul mesajelor al tuturor conversațiilor utilizatorului.
- **Logout**→Face delogarea utilizatorului de la server, mai apoi revenindu-se la meniul de logare.

5 Concluzii

Pentru a îmbunătăți soluția dată putem să implementăm o soluție pentru cazul în care utilizatorul își uită parola. Soluția actuală nu permite această acțiune. De asemenea, am putea să adaugăm și opțiunea de ștergere a unui mesaj, lucru care ar îmbunătăți experiența utilizatorilor.

6 Referințe bibliografice

https://profs.info.uaic.ro/ computernetworks/
https://ro.wikipedia.org/wiki/SQLite
https://www.geeksforgeeks.org/
https://www.andreis.ro/teaching/computer-networks
https://www.sqlite.org/cintro.html
https://zetcode.com/db/sqlitec/
https://www.geeksforgeeks.org/multithreading-in-c/