VirtualSoc

Cazacu Ion

Facultatea de Informatică, Universitatea "Alexandru Ioan Cuza" Iași

1 Introducere

1.1 Viziunea Generala:

Proiectul VirtualSoc propune dezvoltarea unei aplicații client-server ce simulează funcționalitățile unei rețele sociale. Cu o gamă variată de caracteristici, sistemul oferă utilizatorilor posibilitatea de a se înregistra în diferite tipuri de conturi (utilizatori obișnuiți și administratori) și de a interacționa într-un mediu virtual social. Această aplicație vine să răspundă nevoilor utilizatorilor moderni de a comunica, partaja și interacționa într-un spațiu virtual.

1.2 Obiectivele Proiectului:

1. **Inregistrarea si Autentificarea Utilizatorilor:** Dezvoltarea unui sistem eficient pentru înregistrarea și autentificarea utilizatorilor în aplicație.
2. **Postari si Vizibilitate:** Permite utilizatorilor să-și creeze propriile postari și să le stabilească nivelul de vizibilitate (public sau privat).
3. **Relații de Prietenie și Tipuri de Prietenii:** Implementarea unui sistem de adăugare și gestionare a prietenilor, cu posibilitatea de a specifica tipuri diferite de relații (apropiat, cunoștință, etc.).
4. **Comunicare Privată:** Dezvoltarea unui mecanism de comunicare privată între doi sau mai mulți utilizatori.

2 Tehnologii Aplicate

Pentru a realiza funcționalitățile propuse și pentru a asigura o experiență fluidă pentru utilizatori in proiectul VirtualSoc se utilizează urmatoarele tehnologii:

2.1 Comunicație între Client și Server

În cadrul proiectului VirtualSoc, comunicarea între client și server este esențială pentru transferul eficient și sigur al datelor. Protocolul TCP (Transmission Control Protocol) este ales pentru această comunicare, oferind un cadru fiabil și orientat pe conexiune pentru schimbul de informații între cele două entități.

Motivația Alegerii TCP: Protocolul TCP este ales datorită fiabilității sale. Este un protocol orientat pe conexiune, care asigură livrarea secvențială și corectă a datelor între client și server. TCP oferă control de flux, retransmisie a datelor în caz de pierdere și confirmare a primirii, făcându-l potrivit pentru aplicații care necesită integritate și securitate în transmiterea datelor.

2.2 Bază de Date SQLite3

Pentru gestionarea eficientă a datelor în cadrul proiectului VirtualSoc, se utilizează baza de date SQLite3. SQLite3 este un sistem de gestionare a bazelor de date relaționale, ușor de integrat în aplicații datorită caracterului său compact și fără necesitatea unui server separat.
Motivația Alegerii SQLite3: Baza de date SQLite3 oferă o soluție eficientă și compactă pentru stocarea și gestionarea datelor utilizatorilor. Fiind o bază de date integrată, nu necesită configurarea unui server separat și este ideală pentru aplicații care doresc să gestioneze datele într-un mod simplu și local.

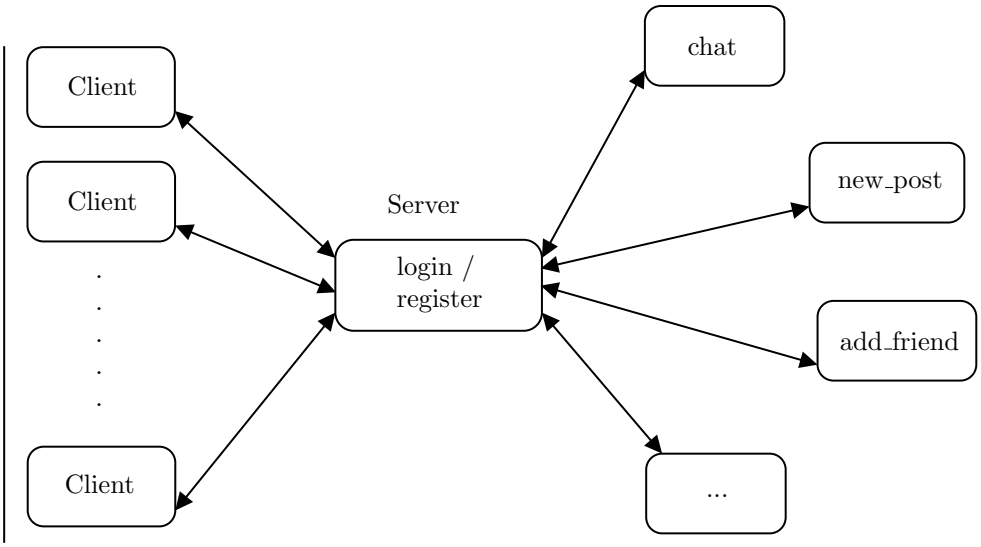
3 Structura Aplicatiei

3.1 Comenzile implementate

- `register` → Înregistrarea unui nou utilizator în baza de date cu toți utilizatorii deja existenți.
- `login` → Logarea utilizatorului prin identificarea acestuia în baza de date ce conține toți utilizatorii înregistrați.
- `logout` → Delogarea utilizatorului.
- `new_post` → Crearea unei noi postări pentru a putea fi ulterior afișată pe pagina proprie a utilizatorului.
- `posts` → Extragerea tuturor postărilor unui utilizator în dependență de statutul profilului si postarilor (privat sau public).
- `setprofile_private` → Setarea profilului in stare privată (nimeni nu poate vedea postările utilizatorului).
- `setprofile_public` → Setarea profilului în stare publică (toți pot vedea postările publice a utilizatorului).
- `setpost_private` → Setarea postării în stare privată (postarea este vizibilă doar utilizatorului care a creat-o).
- `setpost_public` → Setarea postării în stare publică (toți utilizatorii pot vedea această postare).
- `ban` → Blocarea utilizatorului, utilizatorul nu se mai poate autentifica pe contul propriu, comanda disponibilă doar administratorilor.
- `unban` → Deblocarea utilizatorului, comanda disponibilă doar administratorilor.
- `add_friend` → Adăugarea unui alt utilizator în lista de prieteni.
- `rm_friend` → Ștergerea unui prieten din lista de prieteni.
- `friends` → Afișarea tuturor prietenilor utilizatorului.
- `add_admin` → Acordă privilegii de administrator unui utilizator, comanda disponibilă doar administratorilor.
- `rm_admin` → Retrage privilegiile de administrator, comanda disponibilă doar administratorilor.
- `chat` → Deschiderea unui chat pentru transmiterea mesajelor între 2 sau mai mulți utilizatori.
- `exit` → Ieșire din aplicație.

* comenzile new_post, add_friend, rm_friend, friends, setprofile_private, setprofile_public, setpost_private, setpost_public, chat, ban, unban, add_admin și rm_admin nu pot fi utilizate de către utilizatorii neautorizati (guests), însă ei pot: urmări postările(publice) a altor utilizatori, să se înregistreze, să se logheze sau să iasă din aplicație.

3.2 Diagrama



4 Aspecte de Implementare

4.1 Secțiuni de cod specifice

Comenzi implementate in server:

```
void CreateTable()
{
    char *sql = "CREATE TABLE IF NOT EXISTS users (id INTEGER PRIMARY KEY, email TEXT
    NOT NULL, password TEXT NOT NULL, username TEXT NOT NULL UNIQUE, friends TEXT,
    admin_key INTEGER, profile_type INTEGER, block INTEGER);";

    int rc = sqlite3_exec(db, sql, 0, 0, 0);
    if (rc != SQLITE_OK)
    {
        fprintf(stderr, "Cannot create table: %s\n", sqlite3_errmsg(db));
        exit(1);
    }

    sql = "CREATE TABLE IF NOT EXISTS posts (id INTEGER PRIMARY KEY, user_id INTEGER,
    content TEXT, post_type INTEGER, created_at DATETIME DEFAULT CURRENT_TIMESTAMP);";

    rc = sqlite3_exec(db, sql, 0, 0, 0);

    if (rc != SQLITE_OK)
    {
        fprintf(stderr, "Cannot create posts table: %s\n", sqlite3_errmsg(db));
        exit(1);
    }

    if (isAdminUserExisting() > 0) {
        printf("Admin user already exists.\n");
        return;
    }

    char password[] = "admin";
    Encrypt(password, encrypt_key);
    char insertAdminUserSQL[200];
    sprintf(insertAdminUserSQL, "INSERT INTO users (email, password, username, admin_key,
    profile_type) VALUES ('admin', '%s', 'admin', 1, 1);", password);

    rc = sqlite3_exec(db, insertAdminUserSQL, 0, 0, 0);

    if (rc != SQLITE_OK)
    {
        fprintf(stderr, "Cannot insert admin user: %s\n", sqlite3_errmsg(db));
        exit(1);
    }
}
```

- Serverul crează în baza de date SQLite3 tabelele: users - care v-a stoca toată informația despre utilizatorii înregistrați și tabela posts - care v-a stoca infromațiile despre postările tuturor utilizatorilor
- Serverul automat generează un utilizator admin cu privilegiile de administrator

Comenzi implementate in client:

```
if (strcmp(buffer, "newpost") == 0)
{
    printf("Enter your post: ");
    scanf(" %[^\n]", post);

    int post_size = strlen(post);
    send(client_sock, &post_size, sizeof(int), 0);

    send(client_sock, post, post_size, 0);

    Recv(client_sock, buffer, 1024, 0);
    printf("server: \t%s\n", buffer);
}
else if (strcmp(buffer, "login") == 0)
{
    printf("Enter your username: ");
    scanf("%s", user_buffer);

    send(client_sock, user_buffer, strlen(user_buffer), 0);

    Recv(client_sock, buffer, 1024, 0);
    printf("server: \t%s\n", buffer);
}
```

- Clientul introduce comenzi și le trimite la server prin "send".
- Se verifică tipul comenzii.
- Pentru comanda "login", clientul introduce numele de utilizator și îl trimite la server.
- pentru comanda "newpost", clientul introduce continutul postarii si il trimite la server.
- Răspunsul de la server cu privire la rezultatul functiei este afișat.

4.2 Protocolul TCP

Acest program are un server TCP concurent, ceea ce înseamnă că poate servi mai mulți clienți simultan, fiecare într-un proces sau într-un fir de execuție separat. Iată câteva caracteristici ale serverului concurent:

Server Concurent

- **Multiprocessing sau Multithreading:** Un server concurent utilizează procese sau fire de execuție multiple pentru a manipula clienții simultan. Aceasta permite serverului să răspundă la cereri de la mai mulți clienți în același timp.
- **Performanță Îmbunătățită:** Serverul concurent are potențialul de a oferi performanțe mai bune decât serverul iterativ în situațiile în care există mai mulți clienți care trimit cereri în mod simultan. Astfel, se pot reduce timpii de așteptare pentru clienți.
- **Gestionarea Resurselor:** Serverul concurent poate gestiona eficient resursele, cum ar fi memoria și procesorul, prin împărțirea sarcinilor între mai multe procese sau fire de execuție. Acest lucru poate contribui la optimizarea utilizării resurselor sistemului.

4.3 Scenarii reale de utilizare

Programul este văzut ca un prototip al unei rețele de socializare, fiind capabilă să facă cele mai esențiale procese disponibile precum : crearea contului propriu, crearea postărilor, adăugarea prietenilor, comunicarea prin chat etc.

5 Concluzii

Proiectul VirtualSoc propune o simulare a unei rețele sociale, acoperind o gamă variată de funcționalități. Prin dezvoltarea acestuia, se pot dobândi cunoștințe valoroase în domeniul rețelelor sociale.

Potențiale îmbunătățiri ale proiectului

- adăugarea noilor funcționalități.
- optimizarea codului

6 Bibliografie

https://github.com/nikhilroxtomar/Multiple-Client-Server-Program-in-C-using-fork
https://github.com/nikhilroxtomar/Chatroom-in-C
https://www.sqlitetutorial.net