INTRODUCERE

Ideea de baza a proiectului “Offline Messenger” consta in crearea unei aplicatii de tip server/client care sa faca posibila comunicarea intre doi sau mai multi utilizatori(clienti) conectati la server. Transmiterea mesajelor se poate efectua fie atunci cand ambii utilizatori sunt conectati la server, fie atunci cand unul este conectat, iar celalalt nu. In cel de-al doilea caz, in momentul in care utilizatorul s-ar conecta, acesta va vedea mesajele primite cat timp a fost offline. Un alt feature al acestei aplicatii de tip messenger ar fi acela ca fiecare utilizator(client) sa poata raspunde unui anumit mesaj primit de la oricare alt utilizator(client). Arhivarea conversatiei dintre un utilizator(client) si orice alt utilizator(client) este esentiala, de aceea aplicatia ar trebui sa faca posibila afisarea de catre server a unui “istoric” al mesajelor.

Cu totii ne folosim zilnic de comunicarea de acest tip, indiferent ca este vorba de retele de socializare sau aplicatia de mesagerie. Adesea ne intalnim cu probleme datorita aplicatiei sau a internetului si ne este usor sa-i criticam pe cei ce ne pun la dispozitie acea aplicatie. Am ales sa imi creez propria aplicatie de tip “Offline Messenger” pentru a descoperi cum se pun bazele acesteia, ce probleme apar si, mai ales, cum se rezolva ele.

TEHNOLOGII UTILIZATE

Am ales sa utilizez protocolul TCP/IP pentru a realiza comunicarea intre un server concurent si client deoarece:

-Este nevoie sa transmitem mesaje text(comenzi,mesaje de la utilizatorul x la utilizatorul y) si date de utilizator(username,parola), de aceea este nevoie de protocolul TCP, protocol ce ne asigura transferul octetilor in ordinea in care au fost trimisi.

Sa presupunem ca serverul asteapta numele de utilizator al clientului de la acesta si, mai apoi, parola. Clientul va trimite mai intai numele de utilizator si apoi parola, iar protocolul TCP ne ofera siguranta ca acestea vor ajunge in aceeasi ordine in care au fost trimise. In cazul mesajelor, nimic nu se schimba. Daca un mesaj de 100 caractere este trimis catre server in 5 mesaje de cate 20 de caractere fiecare, este nevoie ca serverul sa le primeasca in ordinea in care le-a trimis clientul pentru a avea inteles.

-TCP utilizeaza un mecanism de control al fluxului pentru a evita ca expeditorul să trimită date prea repede spre receptor care trebuie sa le primeasca si sa le proceseze fiabil. Dispunerea unui mecanism de control al fluxului este esențială într-un mediu în care comunica mașini cu viteze variabile de rețea. De exemplu, daca sunt transmise date de la un calculator catre altul care proceseaza greu datele, cel din urma trebuie sa regleze fluxul de date. Astfel, controland fluxul de date, TCP ne ofera siguranta ca nu se pierd date si nici nu ajung la destinatar intr-o alta ordine fata de cea in care sunt trimise de catre expeditor.

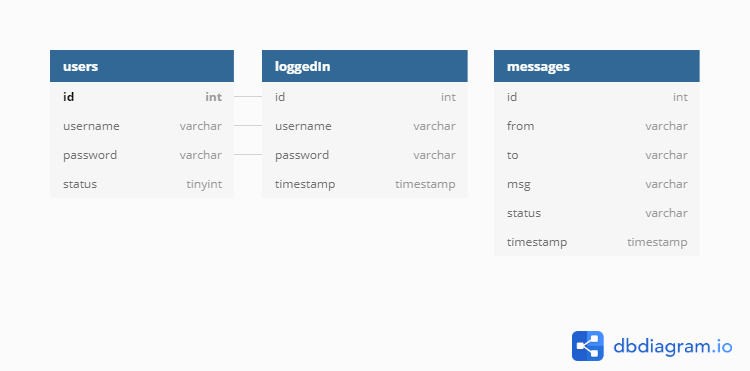
-Comunicarea intr-o aplicatie de tip messenger trebuie sa se faca in ambele sensuri. TCP, avand adresa IP si PORT-ul fiecarui client, poate realiza o comunicare full-duplex(bidirectionala).

Aceasta comunicare full-duplex intre server si client se face prin intermediul socket-urilor.

Serverul trebuie sa fie unul concurent deoarece trebuie sa permita conectarea simultana a doi sau mai multi client si, de exemplu, daca este conectat un ClientX la server, ClientY se poate conecta si poate utiliza anumite functionalitati fara sa fi terminat ClientX. Intre timp, poate aparea si un ClientZ si ClientW…

Intr-o aplicatie de tip messenger este nevoie ca utilizatorii sa fie inregistrati si sa se logheze cu datele sale(username si parola), de aceea trebuie sa le memoram. Pentru acest lucru, am ales sa utilizez o baza de date pentru a gestiona datele tuturor utilizatorilor inregistrati, datele utilizatorilor conectati la server si mesajele trimise de la expeditor la destinatar, ora la care s-a transmis mesajul si statusul acestuia(trimis,vazut).   
Pentru aceasta am considerat ca este folositor sistemul de gestiune a bazelor de date “SQLITE”, SGBD inclus intr-o librarie C. Spre deosebire de alte sisteme de gestiune a bazelor de date, SQLITE nu necesita un server separat.

SCHEMA BAZEI DE DATE



ARHITECTURA APLICATIEI

Draw.io

--Serverul concurent asteapta conexiuni ale clientilor;

--Clientul cere sa se conecteze la server si asteapta acceptarea/respingerea cererii sale;

--In cazul in care cererea de conectare a clientului este acceptata de catre server, cel din urma va crea un proce copil care sa efectueze comunicarea dintre server si client;

--Din momentul in care este conectat, clientul va avea la dispozitie doua comenzi: login si register

--In cazul in care clientul cere sa faca login, atunci serverul ii va cere sa introduca username-ul si password-ul pentru login. La login se va verifica daca in baza de date exista perechea username->password. In caz afirmativ, se va face update in baza de date, tabelul loggedIn. Altfel, exista posibilitatea de a crea un cont nou, prin comanda register.

--In cazul in care clientul cere sa faca register, atunci serverul ii va cere sa introduca username-ul si password-ul pentru register. Se va insera in tabela users perechea username-password doar in cazul in care username nu exista deja in tabela. Daca exista, se va intoarce o eroare spre client.

--Odata conectat, utilizatorul va avea acces la writeto,reply,history,logout,check

-Comanda “writeto” => aceasta comanda permite comunicarea intre doi utilizatori.

=> PANA CAND ESTE TRIMISA COMANDA END:

se va deschide un thread in procesul copil care verifica continuu

tabela messages daca sunt mesaje cu statusul=0(mesaj primit, dar

nu si vazut.

🡪in cazul in care apare un mesaj cu statusul=0, se va trimite intr-un alt thread, catre server

-Comanda “reply” => aceasta comanda permite trimiterea unui raspuns pentru un anumit mesaj primit

-Comanda “history” => aceasta comanda permite vizualizarea istoricului mesajelor din comunicarea cu un anumit client.

-Comanda “logout” => aceasta comanda permite inchiderea conexiunii clientului la server

-Comanda “check” => aceasta comanda permite verificarea mesajelor primite (cand acesta era offline). Se va apela atunci cand un client se conecteaza la server.

DETALII DE IMPLEMENTARE

COD relevant proiectului-comunicare

Scenarii: caz general,caz serverul pica, clientul pica

CONCLUZII

CUM AS PUTEA IMBUNATATI?

BIBLIOGRAFIE

[www.Dbdiagram.io/d](http://www.Dbdiagram.io/d)

[www.draw.io/](http://www.draw.io/)

<https://profs.info.uaic.ro/~computernetworks/>