**PubSub Service**

**Opis:**

Ovo je multi threadovani server koji omogucava komunikaciju izmedju **publishera i subscribera**, on odrzava komunkaciju baziranu na sistemu gde se **publisher salje poruke** na odredjenu temu i onda **subcriber** koji je prijavljen na tu temu **prima te poruke**, sistem takodje sadrzi storage servis za belezenje poruka.

**Mogucnost:**

Prihvata veci broj klijentskh konekcija koristeci **thread pool**, podrzava komunikaciju zasnovanu na temama izmedju publshera i subscribera, koristi custom **hash mapu** za upravljanje mapiranjem teme na subscribere, implementira **cirkularni buffer** za belezenje publishovanih poruka, podrzava **perzistenciju** **poruka** tako sto ih prosledjuje u storage service, koristi **mutexe** da obezbedi konzistentnost niti.

**Inicijalizacija:**

Windows socket initialization() – server inicijalizuje **Winsock** koristeci InitializeWindowsSockets() funckiju da bi obezbedio komunikaciju preko mreze.

Global Data Initialization() - inicijalizuje **kljucne strukture podataka**: topicSubscribers() je custom hash mapa koja mapira temu sa subscriberom, publishedMessagesBuffer je cirkularni buffer za belzenje poruka pre procesiranja, publishesMessagesMutex je mutex koji sluzi za sinhronizaciju pristupa bufferu.

Thread Pool Initialization:

InitializeThreadPool() - **kreira worker threadove** koji obradjuju zahteve od klijenata, IntitializeMessageThreadPool() – kreira niti za procesiranje poruka da obavestavaju subscribere i beleze poruke

**Porketanje servera:**

Server kreira TCP socket, binduje ga na port (DEFAULT\_PORT) i postavlja socket u mod slusanja (listenSocket).

Zatim se prihvata klijentova konekcija, kada se **novi klijent konektuje** socket od tog klijenta se **doda u red** (clientQueue), ako je server prekoracio svoj kapacitet konekcija se zatvara

**Obradjivanje klijentovih poruka:**

ProcessClientMessages() izcitava poruke sa klijenta i preusmerava ih na osvnou tipa, publisher poruke su u formatu **PUBLISHER:topic:message**, Subscriber salje poruke u formatu **SUBSCRIBER:topic**, Storage service se takodje konektuje sa predefinisanim idenentitetom.

**Procesiranje poruka:**

ProcessPublisherMessage(clientSocket, message) – uzima temu i poruku i stavlja ih u **cirkularni buffer**.

ProcessSubscriberMessage(clientSocket, message) – vezuje klijentov socket na tu temu u **hash mapi**.

**Raspodela poruka:**

MessageWorkerFunctikon() konstantno uzima poruka iz cirkularnog buffera i onda: poziva notifySubscribers(topic, message) da salje poruke subskribovanim klijentima, poziva SendToStorage(message) to prosledi poruku ka storage servisu.

**Ciscenje:**

CleanupGlobalData() – Oslobadja memoriju koju su zauzeli cirkulrani buffer i hash mapa.

ClaenupThreadPool() – terminira worker threadove.

CleanupMesageThreadPool() – terminira threadove za procesiranje poruka.

Closesocket(listenSocket) – zatvara server socket

WSACleanup() – cisti Winsock

**Strukture podataka**

**Cirkularni buffer:**

Predstavlja dinamicki alciran niz TopicMessagePair sturktura, on sadrzi:

TopicMesagePair\* bufer – pokazivac na buffer koji sadrzi topic-message parove,

size\_t capacity, - trenutni kapacitet buffera,

size\_t head – Index **sledece slobodne pozicije** u bufferu, pokazuje na mesto gde ce sledeca poruka biti zabelezena,

size\_t tail – index najstarije poruke u bufferu, pokazuje na sledecu poruku koja ce biti procitana,

size\_t size – trenutni broj poruka u bufferu

TopicMessage pair je sturkura koja sadrzi:

Char\* topic – tema poruke

Char\* message – sadrzaj poruke

**HashMapa:**

Predstavlja hash mapu koja belezi topic-subscriber mapiranja.

HashMapNode\* buckets[HASH\_MAP\_SIZE] je niz bucketa od kojih svaki sadrzi povezanu listu HashMapNode struktura

HashMapNode predstavlja temu u hash mapi, svaki node sadrzi temu i listu subscribera za tu temu, node sadrzi:

char topic[MAX\_TOPIC\_LENGTH] – sama tema,

SubscriberNode\* subscribers – povezana lista subscribera za tu temu,

HashMapNode\* next – pokazivac na sledeci node u bucketu (ako postoji)

Subscriber node predstavlja samog subscribera, on sadrzi informaciju o njegovom socketu i pokazivac na sledeceg u listi:

SOCKET socket – socket koji je povezan sa subscriberom,

SubscriberNode\* next – pokazivac na sledeceg subscribera u listi.

**Publisher Client**

**Opis:**

Ovaj klijent se konektuje na TCP server i dozvoljava korisnicima da salju poruke za odredjene teme, poruke se salju u formatu **PUBLISHER:<topic>:<message>**. Klijent konstantno trazi od korisnika teme i poruke sve dok korisnik ne unese “exit”, nakon cekga se klijent diskonektuje sa servera.

**Zavisnosti**: Bibiloteke koje se koriste su: Ws2\_32.lib, Mswsock.lib, AdvApi32.lib.

**Struktura koda i funkcije:**

Main() – prikazuje pocetnu poruku, ceka korisnikovu ulaznu poruku, poziva connect da bi se uspostavila konekcija sa serverom

Connect() – Inicijalizuje Winsock koristeci InitializeWindowsSockets(), kreira TCP socket, pokusava da se konektuje na server, otvara se petlja koja prihvata input korisnika za temu i poruku, salje podatke koristeci Publish(), zatvara sockete i cisti Winsock pre izlaza

InitializeWindowsSockets() – poziva WSAStartup da bi inicijalizovala Winsock, vraca true ako je uspesno, a ako nije onda printa error i vraca false.

FlushInputBuffer() – Cisti input buffer da bi preventirala da se neki nezeljeni karakteri ne dospeju u korisnikov input.

Publish(SOCKET connectSocket, const char\* topic, const char\* message) – formira poruku kao **PUBLISHER:topic:message**, salje poruku koristeci send(), upravlja greskama u prenosu i izlazi iz programa ako send() ne uspe.

**Subscriber Client**

**Opis:**

Ovaj klijent se konektuje na TCP server, subskrajbuje se na odredjene teme i konstantno slusa poruke koja se publishovane iz tih tema za koje je prijavljen, poruke sa servera su prikazane u realnom vremenu. Subskrajber moze da se: subskrajbuje za temu tako sto unese njeno ime, prima i prikazuje poruke za tu temu, izlazi iz aplikacije tako sto unese “exit”

**Zavisnosti**: Bibiloteke koje se koriste su: Ws2\_32.lib, Mswsock.lib, AdvApi32.lib.

**Struktura koda i funkcije:**

Main() – prikazuje pocetnu poruku, ceka korisnicki input, poziva connect() da bi se uspostavila konekcija sa serverom.

Connect() – Inicijalizuje Winsock koristeci InitializeWindowsSockets(), kerira TCP socket, pokusava da se konektuje na server, kreira odvojeni thread (ReceiveMessages) da obradjue pristigle poruke sa servera, pravi se petlja koja prihvata korisnikov input za prijavljivanje za temu, salje zahtev za prijavu koristeci Subscribe(), zatvara konekciju nakon zahteva korisnika.

InitializeWindowsSockets() – poziva WSAStartup() da inicijalizuje Winsock, vraca true ako je uspesno, ako nije printa error i vraca false.

FlushInputBuffer() – Cisti input buffer da bi preventirala da se neki nezeljeni karakteri ne dospeju u korisnikov input.

ReceiveMessages(LPVOID lpParam) – pokrenut je kao poseban thread, ceka na pristigle poruke sa servera, prikazuje pristigle poruke, obavlja diskonektovanje servera i primanje gresaka.

Subscribe(SOCKET connectSocket, const char\* topic) – formira zahtev za prijavu kao: SUBSCRIBER:<topic>, salje formirani zahtev serveru, obradjuje greske u prenosu i izlazi iz programa ako send() ne uspe.

**Storage Service**

**Opis:**

Storage service se konektuje na TCP server, prima poruke i obezbedjuje interfejs za korisnika i omogucava mu da pregleda, pretrazuje i upravlja zabelezenim porukama. Klijent se registruje kao storage service nakon konekcije i odrzava primljene poruke u povezanoj listi.

Ovaj servis omogucavaa: Primanje i belzenje poruke sa servera, prikazivanje svih primljenih poruka, pretrazivanje poruka po temi, upravljanje memorijskom alokacijom i ciscenjem zabelezenih poruka, upravlja gasenjem servera i greskama.

**Zavisnosti:** Bibiloteke koje se koriste su: Ws2\_32.lib, Mswsock.lib, AdvApi32.lib.

**Struktura koda i funkcije:**

Main() – inicijalizuje Winsock koristeci InitializeWindowsSockets(), kreira TCP socket i konektuje ga na server, salje storage service identifirer (STORAGE\_IDENTIFIER) da obavesti server o njegovoj ulozi, pokrece poseban thread (ReceiveMessages) da obradi pristigle poruke sa servera, prikazuje interaktivni meni korisnicima koji im omogucava da: vise sve zabelezene poruke (ReadAllMessages()), pretragu poruka po tebi (SearchMessages(topic)), izlaz i ciscenje svih zabelezenih poruka (CleanupMessages()).

InitializeWindowsSockets() – poziva WSAStartup() da inicijalizuje Winsock, vraca true ako je uspesno, ako nije ispisuje error i vraca false.

ReceiveMessages(LPVOID lpParam) – pokrenut je kao poseban thread, prima pristigle poruke sa servera i belezi ih koristeci funkcije za obradu poruka, parsira poruke u “topic:message formatu, poziva SaveMessge(const char\* topic, const char\* message) – alocira memorije za nove poruke, belzi temu, poruku i vremenski trenutak pristizanja poruke, ubacuje poruku u povezanu listu.

ReadAllMessages() – uzima i prikazuje sve zabelezene poruke

SearchMessages(const char\* topic) – omogucava korisniku da pretrazuje poruke po temi

CleanupMessages() – cisti zabelezene poruke i zatim gasi storage servis