Sprawozdanie - Komunikator IRC

Filip Ciesielski 145257
Michał Ciesielski 145325

5 stycznia 2022

1 Opis protokołu komunikacyjnego

Protokół komunikacyjny pomiędzy klientem a serwerem:

- Komunikaty wysyłane od klienta do serwera w zależności od podjętej akcji
  - Zmiana/nadanie nazwy użytkownika
    "#0%<nowa nazwa użytkownika>$"
  - Stworzenie nowego pokoju
    "#1%<nazwa tworzonego pokoju>$"
  - Dołączenie do pokoju
    "#2%<nazwa pokoju do którego się dołączamy>$"
  - Opuszczenie pokoju
    "#3%<nazwa pokoju, który opuszczamy>$"
  - Wysłanie wiadomości w danym pokoju
    "#4%<nazwa pokoju w którym wysyłamy wiadomość%<czas wysłania wiadomości> <treść wiadomości>$"
  - Usunięcie innego użytkownika przez założyciela(administratora) pokoju
    "#5%<nazwa pokoju, w którym znajduje się usuwany użytkownik>%<nazwa usuwanego użytkownika>$"
- Główny komunikat wysyłany przez serwer do klienta zawiera aktualne dane wszystkich pokoi (nazwy użytkowników, wiadomości) w następującej postaci:

  "#@pokój1%Michał;Filip%Filip;06:44:57;Hej;Michał;06:45:03 ;Cześć;Filip;06:45:07;Co tam?;Michał;06:45:15;Skończyłem projekt z SK2! @pokój2%Michał@pokój3%Filip%Filip;06:45:42;Jestem w tym pokoju sam$"

  Powyższy przykład przekazuje poszczególne dane każdego z pokoi według formatu:

  "#@<nazwa pokoju1>%<użytkownik1>;<użytkownik2>;...; <użytkownikn>%<wiadomosc1>;<wiadomosc2>;...;<wiadomoscn> @...@<nazwa pokojun>%<użytkownik1>;<użytkownik2>;...; <użytkownikn>%<wiadomosc1>;<wiadomosc2>;...;<wiadomoscn>

  Oprócz powyższego komunikatu z danymi, serwer wysyła odpowiedzi na polecenia od klienta, które następnie są wyświetlane na dolnym pasku w aplikacji.

2 Opis implementacji

2.1 Serwer

Implementacja serwera zawiera następujące struktury przechowujące informacje o podłączonych klientach i pokojach:

- User zawierająca nazwę użytkownika i numer kanału podłączonego klienta
- Room przechowująca informacje o nazwie pokoju, tablice struktur User (podłączonych użytkowników do pokoju) i tablice wiadomości w tym pokoju

W celu zapewnienia współbieżności działania serwera wykorzystujemy mutex do obsłużenia odbieranych i wysyłanych wiadomości przez serwer w funkcji obsługującej wątek, który tworzony jest wraz z nowo podłączonym klientem. Po zamknięciu aplikacji przez klienta stworzony wątek zostaje zakończony, a dane klienta zostają zwolnione dla kolejnych użytkowników. Serwer dodatkowo kontroluje aktualną liczbę użytkowników do niego podłączonych, a w przypadku gdy limit ten zostanie przekroczony, serwer zapewnia odesłanie do każdego z nadmiarowych klientów odpowiedniego komunikatu wraz z zamknięciem stworzonego kanału połączeniowego.

2.2 Klient

Program klienta obsługuje w swoim ciele 3 wątki. Pierwszy wątek - główny obsługuje akcje użytkownika związane z GUI aplikacji stworzone przy pomocy JavaFX. Drugi wątek służy tylko do wysyłania komunikatu do serwera, a zatem kończy się po przesłaniu komunikatu. Ostatni, trzeci wątek działa w pętli czekając na odpowiedzi od serwera wraz z aktualnymi danymi, które następnie akutalizuje. Wątek kończy się wraz z zamknięciem aplikacji.

3 Uruchamianie projektu

3.1 Kompilacja i uruchomienie serwera

Serwer kompilujemy wykonując polecenie **gcc -pthread server.c -o server -Wall** w katalogu src/server.

Po pomyślnie wykonanej kompilacji uruchamiamy plik wynikowy poleceniem **./server <numer portu>** lub **./server** (domyślnie port 1234).

3.2 Kompilacja i uruchomienie klienta

Klienta kompilujemy i uruchamiamy poleceniem **mvn clean javafx:run** w katalogu src/client lub przy pomocy środowiska IntelliJ po otworzeniu projektu z katalogu client.