Komunikator typu GG

Filip Bończyk, Kamil Burdziński

Grudzień 2018

1 Wstęp

Tematem naszego projektu był komunikator internetowy typu "Gadu-Gadu". Podstawową funkcjonalnością programu jest wysyłanie i odbieranie wiadomości między dwoma wcześniej zarejestrowanymi użytkownikami. Możliwe jest także wyszukiwanie, dodawanie i usuwanie przyjaciół. Serwer został napisany w C++, natomiast klient w Javie.

2 Funkcjonalność klienta

Po uruchomieniu programu pierwszy ukaże się **ekran łączenia z serwerem**. Podajemy tutaj adres IP serwera oraz port. Jeżeli nie uda się połączyć z serwerem wyświetlony zostanie komunikat "Can't reach server", w przeciwnym razie zostaniemy przeniesieni do **ekranu logowania**. Tutaj możemy zalogować się na istniejące konto wpisując odpowiednie dane i zatwierdzając przyciskiem *sign in* . Jeżeli nie mamy konta przechodzimy do **ekranu rejestracji** klikając przycisk *sign up*, a następnie wypełniamy odpowiedni formularz. Gdy zalogowanie już się powiedzie, wówczas pokaże się **główny ekran** . Zaczynając od góry widzimy wyświetlony nasz nick, niżej znajduje się pole, w którym możemy ustawić opis, zatwierdzając go klawiszem *enter*. Jeszcze niżej znajdują się dwa przyciski *Add* i *Delete* kolejno do dodawania i usuwania znajomych. Większą część zajmuje jednak lista znajomych, gdzie możemy rozpocząć konwersację ze znajomym klikając dwa razy w jego pole, a także sprawdzić jego stan aktywności i opis. Jeżeli przejdziemy do **wyszukiwania znajomych** to wpisujemy w input login szukanego znajomego, zaznaczamy go, a następnie klikamy przycisk *add.*

3 Funkcje sieciowe klienta

Klient do połączenia wykorzystuje klasę *Socket*. Wiadomości wysyłane są za pośrednictwem klasy *PrintWriter* zaś odczytywane za pomocą BufferedReader. Jeżeli będziemy chcieli połączyć się, zalogować lub zarejestrować, a serwer będzie niedostępny, to wówczas zostanie wyświetlony odpowiedni komunikat. Przy

połączeniu występuje pięciosekundowy *timeout*. Jeżeli w trakcie działania klienta serwer zakończy swoją pracę to otrzymuje od niego wiadomość o treści 600—FIN, a następnie wyświetla informacje o utraconym połączeniu.

4 Serwer

Przy uruchomieniu, stan serwera, tj. listy klientów i konwersacji, jest deserializowany. Administrator serwera może używać konsoli do wyświetlania bieżącej listy zarejestrowanych klientów i ich usuwania poprzez wpisywanie odpowiednich liter (odpowiednio k i d). Drugi z wątków serwera w pętli oczekuje na nowe połączenia. Po otrzymaniu połączenia tworzy nowy wątek *Responder.cpp* - programu obsługującego zapytania - i przypisuje go do odpowiedniego gniazda. Wątek ten w pętli odczytuje wiadomości z gniazda i odpowiednio reaguje na przychodzące kody. Aby chronić program przed niepożądanymi efektami współbieżności, globalna lista użytkowników i listy znajomych każdego z użytkowników są chronione mutexami. Wątek *Responder* nie ma przypisanego jednego klienta - może się on zmieniać w trakcie działania wątku, tak długo jak istnieje połączenie. Gdy połączenie zostaje zerwane, *Responder* upewnia się, że przypisany do niego klient nie jest zalogowany i kończy działanie, a jego zasoby są zwalniane. Serwer należy wyłączyć poprzez wpisanie w konsolę litery *c*. Serwer powiadamia wówczas wszystkich zalogowanych klientów o zakończeniu działania i uruchamia serializację poprzez klasę *ServerSerializer*. Po zakończeniu serializacji program jest wyłączany.

5 Kody

Komunikacja miedzy klientem a serwerem opiera się na poniższych kodach:

- 100 - wysłanie wiadomości
- 101 - logowanie
- 102 - wylogowanie
- 103 - rejestracja
- 104 - wyszukiwanie znajomego po loginie
- 105 - dodanie do znajomych
- 106 - wysyłanie listy znajomych
- 107 - wysyłanie historii konwersacji
- 108 - nowa konwersacja
- 109 - zmiana opisu

- 110 - usuniecie ze znajomych
- 111 - prośba o ID konwersacji z danym użytkownikiem
- 500 - otrzymanie wiadomości
- 501 - zaktualizowanie przez znajomego opisu bądź zmiana stanu aktywności
- 502 - poinformowanie o tym, że użytkownik utworzył ze mną konwersację oraz otrzymanie jej danych
- 600 - serwer zakończył pracę w sposób bezpieczny

Wiadomości mają przykładowo postać: 100|ID-konwersacji|wiadomość
Żądania o numerach od 100 do 111 wysyła klient, a w odpowiedzi otrzymuje wiadomości o odpowiednich numerach od 400 do 411, a zawierają żądane dane lub informacje o niepowodzeniu. Wiadomości 500, 501 i 502 wysyła tylko serwer i klient nie odpowiada na nie.