Komunikator internetowy typu GG - Sprawozdanie

Justyna Frączek 145307

Filip Kokosza

25 stycznia 2021

Spis treści

1 Informacje ogólne

Celem projektu była implementacja systemu w architekturze klient-serwer z użyciem protokołu TCP.

Zaimplementowaliśmy komunikator typu GG, umożliwiający prowadzenie konwersacji między użytkownikami.

2 Opis protokołu komunikacyjnego

W programie używamy protokołu komunikacyjnego TCP. Połączenie klienta z serwerem jest realizowane za pomocą gniazd serwerowych - sockets.

2.1 Rejestracja i logowanie

Na początku po uruchomieniu klienta wyświetla nam się panel z dwiema opcjami - Login i Register.

Powyżej znajdują się cztery pola do wypełnienia: Username, Password, IP Address i Port. Będą one wypełnione domyślnymi wartościami, które oczywiście można zmienić.

Port ustawiony w naszym serwerze jest na 1234, tak więc żeby połączenie się powiodło, ta wartość portu musi zostać wpisana.

Jeżeli dany klient jeszcze nie został zarejestrowany, to przy próbie logowania otrzymamy stosowny komunikat, że należy się najpierw zarejestrować.

Po kliknięciu Register, jeśli rejestracja się powiedzie, użytkownik zostanie automatycznie zalogowany i wyświetlony zostanie następny panel - panel chatu.

2.1.1 Struktury komunikatów

Rejestracja:

Przy próbie rejestracji klient wyśle do serwera następujący komunikat w postaci linii tekstu, która zaczyna się literką **R**, po której następują username i hasło klienta. Wszystko jest oddzielone od siebie znakami tabulacji.

Logowanie:

Przy próbie logowania klient wyśle do serwera następujący komunikat w postaci linii tekstu, która zaczyna się literką **L**, po której następują username i hasło klienta. Wszystko jest oddzielone od siebie znakami tabulacji.

2.2 Prowadzenie konwersacji

Znajdujemy się teraz w panelu chatu. Na górze jest widoczne wąskie pole, w którym użytkownik może wpisać nazwę innego użytkownika w celu dodania, bądź usunięcia go ze swojej listy znajomych.

Istotne jest to, że jak dany użytkownik doda innego, to u tego innego automatycznie on też się pojawi jako jego znajomy. Przy usuwaniu znajomego tak samo.

Pod przyciskiem Add new friend znajduje się lista znajomych, do której będą dopisywani użytkownicy.

W prostokącie na dole jest pole, w którym wpisujemy naszą wiadomość. Następnie w celu wysłania jej do danego użytkownika musimy najpierw kliknąć na jego nazwę w naszej liście znajomych. Po kliknięciu na użytkownika możemy kliknąć Send i nasza wiadomość zostanie wysłana.

Aby użytkownik zobaczył wiadomość, którą dostał od danego użytkownika, musi kliknąć jego nazwę w swojej liście znajomych i kliknąć button Load chat. Spowoduje to załadowanie ich wspólnej konwersacji.

Tak więc zawsze po wysłaniu wiadomości, aby odbiorca je zobaczył u siebie, musi najpierw kliknąć na username nadawcy wiadomości, a następnie kliknąć Load chat. Zaktualizuje to ich konwersację.

2.2.1 Struktury komunikatów

Dodanie znajomego:

Przy próbie dodania znajomego klient wyśle do serwera następujący komunikat w postaci linii tekstu, która zaczyna się literką **A**, po której następują username klienta, który ma być dodany. Wszystko jest oddzielone od siebie znakami tabulacji.

Po otrzymaniu przez serwer tego komunikatu i dodaniu obu użytkowników wzajemnie do swoich tablic znajomych (friends_list), serwer wysyła do użytkownika, który właśnie został dodany przez innego, literkę **a** i username użytkownika, który go dodał. Użytkownik tę wiadomość odbiera u siebie asynchronicznie w wątku i automatycznie też dodaje użytkownika, który go dodał do swojej listy znajomych.

Usunięcie znajomego:

Analogicznie jak przy dodawaniu do znajomych, tylko zamiast literki A klient wysyła literkę **D**, a w wątku przy odebraniu literki **d** od serwera i nazwy użytkownika, który go usunął wie, że też ma go usunąć i usuwa.

Wysłanie wiadomości:

Przy próbie dodania znajomego klient wyśle do serwera następujący komunikat w postaci linii tekstu, która zaczyna się literką **M**, po której następują username klienta, do którego ma zostać wysłana wiadomość oraz wiadomość. Wszystko jest oddzielone od siebie znakami tabulacji.

Serwer po odebraniu tego komunikatu, zapisuje w tablicach wiadomości (messages) obu użytkowników tę wiadomość. Wiadomości z tej tablicy zostaną dopiero wysłane przy załadowaniu/aktualizacji konwersacji z danym użytkownikiem.

Załadowanie/zaktualizowanie konwersacji:

Przy próbie załadowania konwersacji klient wyśle do serwera następujący komunikat w postaci linii tekstu, która zaczyna się literką **l**, po której następuje username klienta, z którym konwersacja ma zostać załadowana. Wszystko jest oddzielone od siebie znakami tabulacji.

Serwer po odebraniu tego komunikatu, wysyła wiadomości do klienta, który wysłał komunikat, jakie ma zapisane w tablicy messages z klientem, na którego kliknął w swojej liście znajomych. Jak wiadomości z klientem się skończą, to serwer wysyła klientowi koniec linii i 'stop' zakończony końcem linii, dzięki czemu klient wie, że ma przestać odczytywać.

Pole z konwersacją jest ustawiane na tekst wiadomości odczytany przez klienta.

Wylogowanie (zamknięcie okna):

Gdy klient będzie chciał zamknąć panel i potwierdzi zamknięcie, to przed zamknięciem programu klienta zostanie wywołana metoda log_out().

Wysyłany jest w niej komunikat do serwera w postaci linii tekstu, która zaczyna się literką **Q**, a kończy znakiem tabulacji.

Serwer po otrzymaniu literki Q zmienia status klienta na 0 (if_logged_in) co będzie oznaczać, że ten klient nie jest aktualnie zalogowany.

3 Opis implementacji

3.1 Implementacja serwera

Program serwera zaimplementowaliśmy w języku C. W funkcji main inicjalizujemy gniazda serwera. Sprawdzamy, czy nie jest już podłączonych za dużo klientów, jeśli tak, to nowy klient nie będzie w stanie się zalogować ani zarejestrować - zostanie do niego wysłana stosowna informacja.

Jeśli liczba podłączonych klientów nie osiągnęła maksimum, dane nowego klienta są zapisywane w tablicy clients należącej do struktury thread_data_t przy rejestracji, po wcześniejszym sprawdzeniu, czy klient o takiej nazwie już jest zarejestrowany.

Następnie przy logowaniu, jeśli przebiegnie ono poprawnie, ustawiany jest unikalny indeks dla klienta, który będzie przechowywany w strukturze index. Dane te zostają przydzielone do wątku klienta.

W funkcji wątku ThreadBehavior serwer wchodzi w nieskończoną pętle i czeka na odpowiednie komunikaty od klienta (mogą się zaczynać na A, l, M, Q lub D), które zostały opisane wyżej w protokole komunikacyjnym.

3.2 Implementacja Klienta

Program klienta zaimplementowaliśmy w języku Java, z użyciem biblioteki Java Swing do wykonania GUI.

Klasa, którą uruchamiamy, zawierająca główną funkcję main to Test.java.

Po jej uruchomieniu uruchamiana jest klasa Application.java. Zostaje w niej utworzony obiekt start klasy StartWindow, który zwróci nam pierwszy startowy panel GUI z opcją rejestracji bądź logowania (zostało opisane tutaj).

Kolejny panel, jaki zostanie wyświetlony, jeśli rejestracja i logowanie bądź samo logowanie przebiegną pomyślnie, to panel chatu (opisany tutaj). Sposób jego tworzenia jest podobny do poprzedniego panelu - w klasie StartWindow.java jest tworzony obiekt klasy Chat.java i obiekt ten zwraca nam panel chatu.

W klasie Chat.java zostaje również uruchomiony wątek klienta.

4 Kompilacja i uruchamianie projektu

4.1 Serwer

Do kompilacji serwera należy użyć polecenia:

```
gcc Server.c -Wall -pthread -o serwer
```

które wygeneruje wykonywalny plik o nazwie `serwer` w bieżącym katalogu. Nazwa jest oczywiście opcjonalna i może być dowolna.

W celu uruchomienia serwera należy użyć polecenia:

```
./serwer
```

4.2 Klient

Aby uruchomić klienta, można to zrobić w środowisku Intellij. Należy wtedy skompilować i uruchomić klasę Test.java.