Marcin Jaskulski 136560, Paweł Pytel 136786

Sieci Komputerowe 2 Sprawozdanie - Komunikator internetowy typu GG

1. Opis protokołu
   Gdy użytkownik loguje się do serwera dostaje miejsce w pamięci, oraz swój identyfikator. W zaalokowanej pamięci na potrzeby serwera przechowujemy numer deskryptora, oraz, czy dany użytkownik jest obecnie zalogowany. Serwer na raz może obsługiwać 100 użytkowników, dlatego w przypadku podłączenia 101 użytkownika serwer zwraca komunikat z prośbą o próbę ponownego zalogowania za jakiś czas. Po zarezerwowaniu miejsca w pamięci serwer wysyła klientowi informację o jego id, oraz dochodzi do rozesłania wszystkim użytkownikom informacji o obecnym stanie podłączonych do serwera osób.
   Dla każdego użytkownika tworzymy na serwerze wątek reagujący na wysłane wiadomości. Każda wiadomość jest dzielona na tablice char zawierające maksymalnie 100 znaków. Z takiej paczki serwer odczytuje, czy użytkownik chce odczytać wiadomość, czy też zakończyć połączenie. W przypadku zakończenia serwer czyści dane i dochodzi do usunięcia wątku. Następnie zostaje przesłana informacja do wszystkich użytkowników o obecny stanie osób podłączonych do serwera. W przypadku chęci komunikacji serwer wyodrębnia informacje o nadawcy, odbiorcy i wiadomości, następnie opakowuje to w odpowiedni format i wysyła do odbiorcy. Następnie czeka na kolejną paczkę informacji od klienta. W przypadku błędów, lub nieprzewidzianemu rozłączeniu klienta dochodzi do zwolnienia zasobów i usunięcia wątku. Domyślny port serwera to: **`1234.`**

2. Opis poszczególnych formatów do komunikacji podmiotów:
   a. Wysyłanie wiadomości:
      i. Client - dokleja na początku wiadomości skąd i dokąd chce wysłać wiadomość, np: **`001015Cześć!`** (id nadawcy + id odbiorcy + wiadomość)
      ii. Serwer - dokleja informację od kogo została wysłana wiadomość i co zawiera, np: **`#fromId{000}#message{Co tam?}`**
   b. Przepełnienie na serwerze - **`#busySpace`**
   c. Rozłączenie użytkownika:
      i. Jeśli klient chce się rozłączyć - **`#end`** (może też zakończyć połączenie niejawnie)
   d. Rozesłanie informacji o obecnej liczbie osób podłączonych do serwera - **`#friends{0,1,3,4,14}`**

3. Struktura projektu
   Serwer:
   `thread_data_t -` struktura tworzona dla każdego wątku z osobna. Przechowuje takie dane, jak id, nr deskryptora, wskaźnik do pamięci trzymających zalogowanych użytkowników, wskaźnik na pamięć zawierającą deskryptory użytkowników, oraz

dane obecnie przetwarzanej wiadomości, czyli id nadawcy, odbiorcy, oraz treść wiadomości.
whoIs – funkcja rozsyła do wszystkich użytkowników informację o obecnym stanie podłączonych użytkowników.
handleWrite – funkcja odpowiadająca za format, oraz przesłanie wiadomości od nadawcy do odbiorcy.
ReadThreadBehavior – funkcja, w której tworzony jest każdy z wątków. odpowiada za przygotowanie **thread_data_t** , oczekiwanie na wysłaną wiadomość, od podłączonego użytkownika, oraz wykrycie rozłączenia użytkownika z serwerem i zwolnienie zasobów.
handleConnection - rezerwuje pamięć dla wątku, zapisuje w niej podstawowe dane, oraz tworzy wątek.
init – nadaje początkowe dane dla pamięci zalogowanych użytkowników, oraz ich deskryptorów. Używana przy tworzeniu serwera.
main – funkcja startowa serwera. Konfiguruje serwer, następnie oczekuje na logujących się do serwera użytkowników przydzielając im id oraz miejsce w pamięci

Klient:
Main – główna klasa programu
   a. **start** – metoda startowa
   b. **setUpLayout** – przygotowuje GUI
   c. **setUpConnection** – łączy serwerem i pobiera informacje o jego stanie
   d. **deleteThrash** – usuwa zbędne znaki z odebranej wiadomości
   e. **updateFriends** – aktualizuje informacje o zalogowanych użytkownikach
   f. **readMsg** – zapisuje przeczytaną wiadomość
   g. **display** – wyświetla informację o zapełnionym serwerze
   h. **connect** – wyświetla okno pozwalające na podanie nazwy hosta i numeru portu, tworzy instancję klasy Socket
   i. **sendMsg** – wysyła wiadomość do serwera i zapisuje w kliencie
Reader – klasa reprezentująca wątek czytający z serwera
   j. **run** – czyta wiadomości z serwera
LimitTextField – rozszerzenie klasy TextField o możliwość ograniczenia ilości wpisywanych znaków

4. Sposobu uruchomienia
   a. Serwer
      W celu skompilowania serwera należy przejść do folderu zawierającego i wpisać komendę: **gcc threadedServer.c -pthread -o server.out**
      W celu uruchomienia serwera należy w konsoli przejść do folderu zawierającego i wpisać komendę: **./server.out**
   b. Client dla środowiska IntelliJ
      Aby otworzyć projekt: File -> Open..., wybieramy z repozytorium folder klient. Gdy projekt się załaduje, w Run->Edit configurations należy zaznaczyć Allow parallel run (należ mieć już podpiętą konfigurację JDK). Klikając w przycisk Run, uruchamiamy instancję programu.