Julia Lamperska 145365

Gabriel Wachowski 145275

Sprawozdanie sk2

Temat

Komunikator tekstowy

1. Opis projektu (0.5 strony)

* krótki opis
  + Aplikacja służy do komunikacji tekstowej pomiędzy kilkoma użytkownikami podłączonymi do jednego serwera. Aby skorzystać z czatu należy się zarejestrować. Serwer przechowuje informację o zarejestrowanych użytkownikach i pozwala na logowanie tylko uprzednio zarejestrowanym użytkownikom. Po zarejestrowaniu należy się zalogować – bez tej operacji klient nie jest uwierzytelniony i nie ma dostępu do listy aktywnych użytkowników oraz nie może wysyłać żadnych wiadomości (serwer odrzuca wiadomość i zwraca klientowi komunikat o błędzie). Zalogowany użytkownik ma wgląd w listę innych użytkowników czatu – zarówno tych aktywnych jak i obecnie niezalogowanych. Ma on również możliwość wysyłania wiadomości i ich odbierania.
* użyta technologia
  + Do implementacji klienta użyto języka Python oraz biblioteki graficznej PyQt5. Komunikacja z serwerem odbywa się poprzez gniazda BSD w protokole TCP.
  + Serwer został napisany w języku C. Również wykorzystuje gniazda BSD. Dzięki multipleksacji we/wy obsługuje on klientów w sposób asynchroniczny i nieblokujący.

2. Opis komunikacji pomiędzy serwerem i klientem (0.5 strony, może być schemat/rysunek)

Klient po podłączeniu do serwera wysyła do niego prośbę o zarejestrowanie użytkownika. Prośba ta opatrzona jest prefiksem informującym o tym, że klient wyraża chęć rejestracji, po nim następuje login oraz hasło. Ostateczna wiadomość wygląda tak: *#REGISTER#USERNAME#PASSWORD#*.

Jeżeli rejestracja przebiegła pomyślnie, serwer odsyła klientowi informację o tym, że udało się zarejestrować, korzystając z tego samego prefiksu. Wiadomość serwera wygląda zatem tak: *#REGISTER#Registered sucessfully#.* W przypadku błędu serwer odsyła wiadomość o błędzie. Opatrzona jest ona prefiksem błędu. Przykładowy błąd: *#ERR#Username or password not provided#.*

Gdy użytkownik jest już zarejestrowany musi się zalogować. W tym celu wysyła wiadomość opatrzoną prefiksem logowania, a następnie podaje login oraz hasło (*#LOGIN#Username#Password#*).

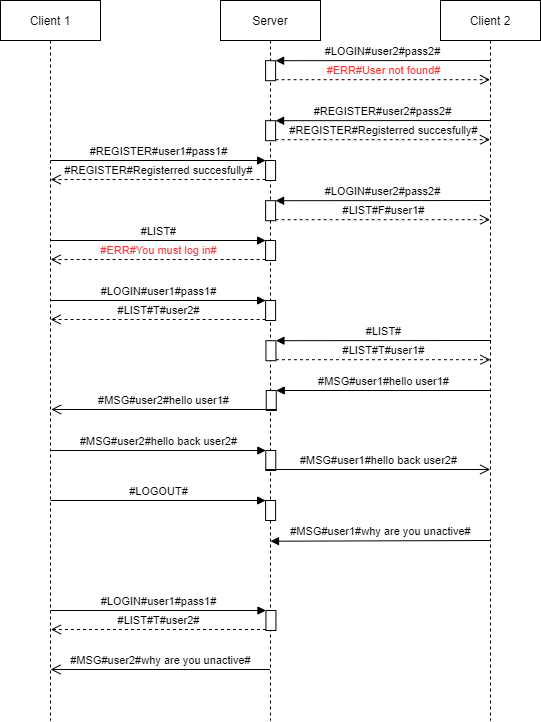
W przypadku błędu zwracana jest informacja o błędzie (podobnie jak przy rejestracji). Jeżeli podane dane są prawidłowe, serwer odsyła klientowi listę dostępnych do konwersacji użytkowników, dodatkowo informując czy dany użytkownik jest w obecnej chwili aktywny. Lista również posiada swój własny prefiks, a informacja o tym czy ktoś jest aktywny przedstawiana jest w postaci tagu przed nazwą użytkownika (*#T#* jeżeli jest aktywny, *#F#* w przeciwnym przypadku). Przykładowa odpowiedź serwera to: *#LIST#F#user1#T#user2#....#*.

Klient chcąc odświeżyć listę użytkowników może wysłać do serwera zapytanie o treści *#LIST#*. Jeżeli użytkownik jest zalogowany, serwer zwróci taką samą listę jak powyżej. W przeciwnym przypadku poinformuje o konieczności zalogowania (*#ERR#You must log in#*).

Przesyłanie wiadomości również wymaga zalogowania. Wiadomość do innego użytkownika musi zostać opatrzona prefiksem #MSG# po którym następuje adresat wiadomości, a następnie jej treść.

(*#MSG#reciever#message#*). Serwer przesyła nadaną wiadomość do odpowiedniego klienta z tą różnicą, że w miejscu adresata umieszcza nadawcę wiadomości. W ten sposób klient wie od kogo otrzymał wiadomość.

W przypadku wysłania wiadomości do nieaktywnego użytkownika serwer przechowuje tą wiadomość i przesyła ją dopiero gdy użytkownik się zaloguje.



3. Podsumowanie (0.5-1 strona)

* Najważniejsze informacje o implementacji
  + Serwer i klient korzystają z prefiksów w celu zadeklarowania pożądanej operacji, dostępne prefiksy to:
    - LOGIN
    - LOGOUT
    - REGISTER
    - ERR
    - LIST
    - MSG
  + Serwer zapisuje w pliku informacje o zarejestrowanych użytkownikach
  + Serwer zapisuje w pliku informacje o wiadomościach do nieaktywnych użytkowników
  + Do obsługi wielu socketów jednocześnie wykorzystano funkcję select()
  + Na gniazdo główne (master\_socket) zgłaszane są prośby o połączenie z serwerem. Nowe połączenie zapisywane jest w tablicy deskryptorów gniazd pod pierwszym wolnym indeksem. Pod tym samym indeksem w tablicy użytkowników (user\_list) zapisywany jest login użytkownika komunikującego się na tym gnieździe (o ile użytkownik się zalogował, w przeciwnym przypadku login nie jest ustawiony a gniazdo nie jest „zautoryzowane”)
  + Klient posiada okienkowy interfejs zaimplementowany w PyQt5
* Co sprawiło trudność
  + Nieznajomość biblioteki PyQt5
  + Brak doświadczenia w tworzeniu GUI
  + Skomplikowane operacje na łańcuchach znaków w C
  + Obsługa wielu klientów jednocześnie