Kompilacja

Program się kompiluje przez napisanie `gcc *.c -o serwer -pthread`
Serwer się włącza przez napisanie: `./serwer <port>`
Klient się włącza przez napisanie: `python main.py <ip> <port>`

Struktura wiadomości

Wiadomość wysyłana między klientem, a serwerem (i na odwrót) to tablica char o następującej strukturze: “\t header \t content \t”. Header jest to nazwa sygnału, natomiast content jest to jego treść. W przypadku wysłania wielu sygnałów pod rząd serwer łatwo rozróżnia początek i koniec każdego sygnały, ponieważ początek i koniec sygnału ma postać “\t”. Oznacza to, że dwa sygnały (wiadomości) oddzielone są od siebie zawsze dwoma znaki tabulacji.

Opis protokołu komunikacyjnego

Obsługa sygnałów o header (operation) ustawionym na:

1. SIGNUP - rejestracja nowego użytkownika w systemie.
   Treść wiadomości od klienta: “username password”.
   Sukces: Wysyłana jest wiadomości o wartości operation ustawionej na “SIGNUP_R SUCCESS”, a message na “SUCCESS” do klienta.
   Błędy:
   - Próba zarejestrowania nowego klienta, przy osiągnięciu wcześniej maksymalnej liczby.
     Wysyłana jest wiadomości o wartości operation ustawionej na “SIGNUP_R FAILED”, a message na “LIST_IS_FULL” do klienta.
   - Próba zarejestrowania klienta o username istniejącym już w bazie.
     Wysyłana jest wiadomości o wartości operation ustawionej na “SIGNUP_R FAILED”, a message na “USERNAME_TAKEN” do klienta.
   - Pole password różni się od pola confirm_password - obsługiwane po stronie klienta.
2. LOGIN - logowanie użytkownika.
   Treść wiadomości od klienta: “username password”.
   Sukces: Wysyłana jest wiadomości o wartości operation ustawionej na “LOGIN_R SUCCESS”, a message na “SUCCESS” do klienta.
   Błędy:
   - Użytkownik o podanym username jest już zalogowany.
     Wysyłana jest wiadomości o wartości operation ustawionej na “LOGIN_R FAILED”, a message na “ALREADY_LOGGED” do klienta.
   - Użytkownik podaje username, który istnieje w systemie, ale hasło nie zgadza się.
     Wysyłana jest wiadomości o wartości operation ustawionej na “LOGIN_R FAILED”, a message na “WRONG_USER_OR_PASSWORD” do klienta.
   - Użytkownik o podanym username nie istnieje w systemie.
     Wysyłana jest wiadomości o wartości operation ustawionej na “LOGIN_R FAILED”, a message na “NO_SUCH_USER” do klienta.
3. ADD_FRIEND - dodanie innego użytkownika do znajomych.
   Treść wiadomości od klienta: “username”.
   Sukces: Wysyłana jest wiadomości o wartości operation ustawionej na “ADD_FRIEND_R SUCCESS”, a message na “SUCCESS” do klienta.
   Błędy:
   - Użytkownik chce dodać do znajomych użytkownika o username, który nie istnieje w bazie.
     Wysyłana jest wiadomości o wartości operation ustawionej na “ADD_FRIEND_R FAILED”, a message na “NO_SUCH_USER” do klienta.
   - Użytkownik chce dodać do znajomych użytkownika, z którym już jest znajomym.
     Wysyłana jest wiadomości o wartości operation ustawionej na “ADD_FRIEND_R FAILED”, a message na “ALREADY_FRIENDS” do klienta.
4. REMOVE_FRIEND - usunięcie użytkownika ze znajomych.
   Treść wiadomości od klienta: “userid”.
   Sukces: Wysyłana jest wiadomości o wartości operation ustawionej na “REMOVE_FRIEND_R SUCCESS”, a message na “SUCCESS” do klienta.

Błędy:
- Użytkownik chce usunąć użytkownika ze znajomych, który nie jest na jego liście znajomych. Wysyłana jest wiadomości o wartości operation ustawionej na “REMOVE_FRIEND_R FAILED”, a message na “NOT_FRIENDS” do klienta.

5. FRIENDS_LIST - wysłanie listy znajomych do klienta.
   Treść wiadomości od klienta: “” (pusta).
   Sukces: Wysyłana jest wiadomości o wartości operation ustawionej na “FRIENDS_LIST_R SUCCESS”, a message jest tablicą char, która składa się z listy znajomych klienta. Ma ona następującą strukturę: [userid1 is_active1 username1] [userid2 is_active2 username2] … Cała wiadomość wysyłana jest do klienta. W przypadku braku znajomych przesyłana jest pusta wiadomość (“”).
6. SEND_MESSAGE - klient1 chce wysłać do klienta2 wiadomość.
   Treść wiadomości od klienta: ”klient2_id message”.
   Sukces: Wysyłana jest wiadomości o wartości operation ustawionej na “SEND_MESSAGE_R SUCCESS”, a message na “SUCCESS” do klienta1. Natomiast do klienta2 jest wysyłana wiadomości o wartości operation ustawionej na “GET_MESSAGE_R SUCCESS”, a message na “message” (treść wiadomości od klienta1).
   Wszystkie możliwe błędy (wysłanie wiadomości do użytkownika, który nie jest naszym znajomym) są zablokowane po stronie klienta.
7. LOGOUT - wylogowanie się.
   Treść wiadomości od klienta: “”.
   Sukces: Wysyłana jest wiadomości o wartości operation ustawionej na “LOGOUT_R SUCCESS”, a message na “SUCCESS” do klienta.
   Wszystkie możliwe błędy są zablokowane po stronie klienta.

*message w funkcji (void) send_message zastąpiony jest nazwą content.

Struktura programu

Nasz program składa się z trzech części:
- klienta (*.py)
- serwera(*.c)
- opisu interfejsu graficznego (*.ui)

Klient:

Klient oparty jest na dwóch plikach:
- main.py - główna część klienta zawiera obsługę komunikacji z serwerem i GUI. Klasy, które wchodzą w jej skład:
  - MainWindow - główna klasa. Wszystkie klasy poniżej są zawarte w niej.
  - Receiver - klasa, która przy jej aktywacji tworzy dodatkowy wątek, którego zadaniem jest wysłuchiwanie wiadomości od serwera i ich obsługa.
  - GUI
    - Welcome Page - pierwsza strona aplikacji. Może doprowadzić do strony logowania i rejestracji
    - LoginPage - strona obsługująca logowanie klienta
    - Registration Page - strona obsługująca rejestracje klienta
    - MenuPage- strona obsługująca usuwanie/dodawanie znajomych oraz komunikację z nimi poprzez serwer
    - Message Widget - widget, która pomaga w obsłudze wyświetlania wiadomości
- utils.py - plik zawierający funkcję pomocnicze, nie mające związku z działaniem komunikacji, czy GUI

Opis interfejsu graficznego:

Program pobiera cztery strony z czterech plików: login_page.ui, menu_page.ui, signup_page.ui, welcome_page.ui

Serwer:

Cały serwer znajduję się w pliku main.c