Jakub, Glazik 156018
Kacper, Szymański 155860

Sprawozdanie sk2

Komunikator Internetowy

1. Opis projektu (0.5 strony)

Projekt to prosty komunikator internetowy umożliwiający użytkownikom rejestrację, logowanie oraz komunikację w czasie rzeczywistym. Aplikacja obsługuje zarówno rozmowy indywidualne, jak i grupowe. Użytkownicy mogą dodawać znajomych, wysyłać i odbierać zaproszenia do znajomych, a także tworzyć grupy do wspólnych rozmów. Dane o użytkownikach zapisywane są w dedykownym pliku umieszczonym na serwerze.

Użyte technologie:

- Tkinter – biblioteka do tworzenia GUI po stronie klienta.
- Socket – moduł do obsługi komunikacji sieciowej TCP/IP.
- JSON – format do przesyłania danych między klientem a serwerem oraz zapisu danych użytkowników na serwerze.
- biblioteka Threading – zapewnia wielowątkowe działanie serwera i klienta.

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

W naszym projekcie komunikacja między serwerem a klientem odbywa się w oparciu o architekturę klient-serwer. Klient nawiązuje połączenie z serwerem, który przechodzi w tryb nasłuchiwania i obsługuje żądania w dedykowanych dla poszczególnych klientów wątkach. W celu zachowania responsywności aplikacji oraz zapewnienia wymiany danych w czasie rzeczywistym, każdy klient uruchamia również osobny wątek służący nasłuchiwaniu na odpowiedzi ze strony serwera, aby bez opóźnień reagować na zmiany zachodzące na serwerze.

Sama wymiana danych opiera się na formacie JSON, co umożliwiło tworzenie prostej i dość wygodnej struktury wiadomości zawierającej typ żądania (np. logowanie, wysyłanie wiadomości, dodawanie nowych znajomych) i dodatkowe informacje konieczne do poprawnej komunikacji między klientem a serwerem (np. nazwa użytkownika wysłającego zapytanie, nazwy grup na których należy wykonać operacje). Po otrzymaniu żądania serwer wykonuje odpowiednie operacje (np. przesyłanie wiadomości, zarządzanie grupami, modyfikacja odpowiednich struktur w kodzie) i odsyła nadawcy informację o sukcesie lub porażce wykonanej operacji, jeśli to konieczne informuje o tym również pozostałych użytkowników (np. podczas wysyłania wiadomości lub tworzenia grup)

3. Podsumowanie (0.5-1 strona)

1. Implementacja serwera oraz klienta

   Serwer napisany w C++ zarządza połączeniami sieciowymi wykorzystując bibliotekę `socket`. Korzysta z wielowątkowości (`thread`), abu obsługiwać kilku klientów w tym samym czasie.

   Klient naspisany został zaś w języku Python, do stworzenia czytelnego oraz dynamicznie zmieniającego się GUI posłużyła nam biblioteka `Tkinter`.

   Klient korzysta z osobnych wątków do odbierania wiadomości, aby interfejs użytkownika pozostał responsywny podczas komunikacji.

2. Komunikacja:

   Komunikaty są przesyłane w formacie JSON, co zapewnia łatwę przypisywanie typów dla poszczególnych zapytań.Obsługiwane są różne typy akcji, np. `login, send_message, create_group, accept_friend_request`, itp. Struktura tego formatu znacznie ułatwia również odczytywanie kluczowych informacji potrzebnych do wykonania koniecznych operacji.

3. Główne funkcjonalności:

   Rejestracja i logowanie: Użytkownicy mogą zakładać konta oraz logować się na istniejące.

   Zarządzanie znajomymi: Dodawanie znajomych, akceptowanie zaproszeń oraz synchronizacja list znajomych.

   Wiadomości prywatne oraz dostęp do czatów (kanałów) grupowych: Wysyłanie wiadomości między dwoma wybranymi użytkownikami oraz wiadowmości wysyłane do wszystkich użytkowników obecnie połączonych na czacie grupowym.

4. Trudności:

   Dość problematyczna była obsługa jednoczesnego wysyłania i odbierania wiadomości w kliencie oraz serwerze. Wymagało to zastosowania wielowątkowości po obu stronach połączenia. Szczególnie istotne było to w konteksicie unikanięcia blokowania głównego wątku interfejsu graficznego.

   Implementacja protokołu komunikacyjnego, która wymagała precyzyjnego zaplanowania formatów komunikatów JSON, by serwer i klient mogły poprawnie rozumieć swoje żądania i odpowiedzi. Wymagało to wiele debugowania, w poszukiwaniu błędów w komunikacji.

   Powiązanie dynamicznej logiki serwera (np. aktualizacja listy znajomych lub wiadomości) z elementami GUI w Tkinterze było całkiem skomplikowane, szczególnie w przypadku asynchronicznego odbierania wiadomości czy aktualizcji interfejsu.