Szymon Skowroński 151885
Patryk Osiak 151744

Sprawozdanie SK2

Komunikator

1. Opis projektu

Krótki opis

Przedmiotem projektu jest aplikacja symulująca komunikator internetowy.

Użytkownik może zarejestrować swoje konto w systemie oraz korzystać z niego do wymieniania wiadomości tekstowych z dowolnym innym użytkownikiem. Każdej parze kont związanych znajomością przypisywany jest osobny pokój do rozmów, w którego obrębie rejestrowana jest konwersacja. Program umożliwia współbieżne prowadzenie wielu rozmów naraz.

Wszystkie dane użytkowników, takie jak ich loginy, hasła, znajomi oraz rozmowy, są zapisywane poza programem.

Użyta technologia

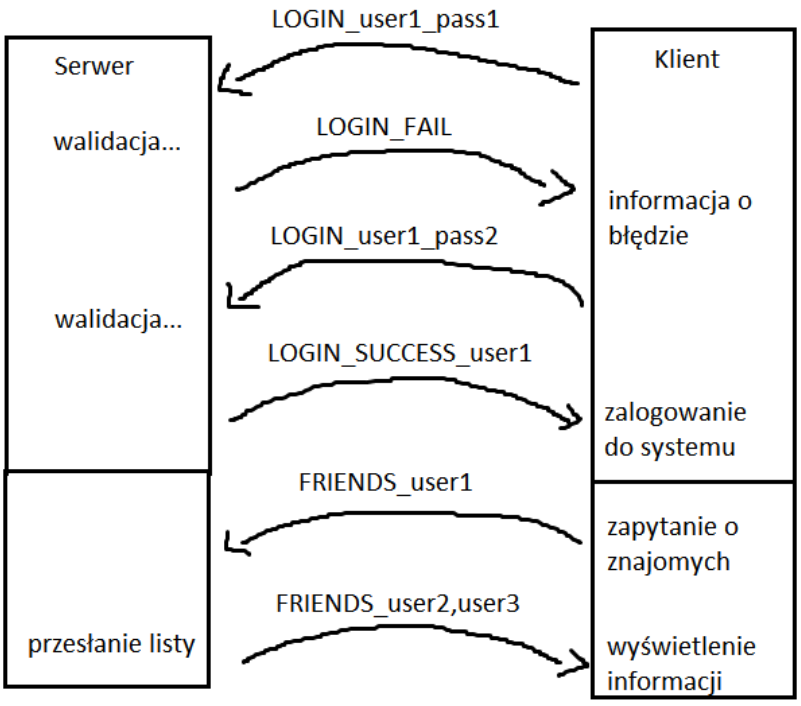
Do implementacji serwera użyto języka C++. Usługi sieciowe zostały zrealizowane za pomocą BSD sockets.

Klient napisany został w języku Python. Użyto tutaj tej samej technologii do komunikacji poprzez sieć co w przypadku serwera, GUI natomiast zrealizowano za pomocą biblioteki tkinter. Projekt kompatybilny jest z wersją Python 3.6.

2. Opis komunikacji pomiędzy serwerem i klientem

Serwer oparty jest na standardzie TCP i korzysta z wielowątkowości do obsługi dowolnej liczby klientów jednocześnie.

Najważniejszą cechą charakterystyczną komunikacji między klientem a serwerem jest system nagłówków nadawanych przesyłanym sobie wiadomościom. Dzięki nim klient i serwer są w stanie delegować zapytania do odpowiednich dla siebie funkcji i metod, co zapewnia przejrzystą i sprawną wymianę informacji. Ponadto, system ten skutecznie unika kolizji danych w sytuacjach, w których wiele okien dialogowych naraz może otrzymywać informacje.



Powyższy przykład przedstawia wymianę zapytań pomiędzy klientem i serwerem w procedurze logowania.

W pierwszym kroku klient podaje dane logowania oczekując autentykacji od serwera i tym samym autoryzacji do dostępu do listy znajomych. Serwer waliduje otrzymane dane i przesyła z powrotem informację o błędzie, którą klient wyświetla.

Po podaniu prawidłowych danych logowania użytkownik jest o tym poinformowany i wysyła zapytanie o listę swoich znajomych. Serwer odpowiada i przesyła żądane dane, a klient wyświetla je w oknie dialogowym.

3. Podsumowanie

Najważniejsze informacje o implementacji

Zarządzanie pamięcią

W projekcie zadbano o unikanie wycieków pamięci dzięki spójnym i jasno określonym procedurom zamykania aplikacji.

Zamknięcie okna logowania lub okna wyświetlającego listę znajomych kończy działanie klienta oraz informuje o tym serwer. Po otrzymaniu informacji o zamknięciu serwer zamyka gniazdo klienta oraz pozwala jego wątkowi się zakończyć.

Zamknięcie serwera natomiast wysyła sygnał EXIT do każdego z klientów, który kończy ich działanie. Następnie na każdym wątku klienta wykonywana jest operacja join(), po czym zamykane jest gniazdo serwera i program się kończy.

Komunikacja między klientami

Przy zawiązaniu znajomości między dwoma użytkownikami, tworzony jest pokój przypisany ich parze. Wszystkie wiadomości wysłane przez którąkolwiek ze stron są zapisywane w pokoju o danym numerze. Na bieżąco monitorowany jest również stan aktywności użytkownika. Jeżeli rozmówca rozłączy się z pokoju lub do niego dołączy, zostaniemy o tym poinformowani.

Przechowywanie danych poza programem

Program do przechowywania danych korzysta z trzech rodzajów plików tekstowych.

Pierwszy z nich to pojedynczy plik, który zawiera informacje o uformowanych znajomościach i przechowuje rekordy w postaci `użytkownik1:użytkownik2:numerPokoju`

Drugi z nich to również pojedynczy plik, przechowujący dane kont użytkowników w postaci `login hasło`.

Trzeci rodzaj pliku to plik dotyczący rozmów w danym pokoju. Na zamianę linia pod linią zapisywani są tam użytkownicy oraz wiadomość, którą wysłali.

Trudności

Jedną z najbardziej znaczących trudności podczas procesu rozwoju aplikacji było ograniczenie przepływu danych od serwera tylko do jednego właściwego okna dialogowego. Początkowo klient korzystał tylko z jednej kolejki danych, co stanowiło kłopot w momencie, w którym więcej niż jedno okno dialogowe było otwarte naraz. Wówczas dochodziło do przechwytywania danych przez obiekty, którym te dane nie były przeznaczone, co skutkowało brakiem zachowania kolejności wiadomości lub brakiem danych w ogóle.

Problem rozwiązano, rozszczepiając kolejkę na wiele części i implementując mechanizm delegacji danych w zależności od nagłówka do odpowiednich dla siebie okien.