Anna Dziuba 146393
Alicja Czapla 155959

Sprawozdanie SK2

Komunikator internetowy

1. Opis projektu (0.5 strony)
   - krótki opis
   - użyta technologia

Projekt jest implementacją komunikatora działającego w architekturze klient-serwer. Umożliwia on użytkownikom rejestrację, logowanie oraz komunikację w czasie rzeczywistym. Serwer obsługuje wielu klientów, umożliwiając im prowadzenie wielu konwersacji jednocześnie. Możliwe jest także prowadzenie konwersacji grupowej pomiędzy trzema użytkownikami. Użytkownicy mogą rejestrować się, podając unikalną nazwę użytkownika oraz hasło. Te dane są przechowywane w bazie danych, co umożliwia logowanie. Komunikacji między klientem a serwerem odbywa się za pomocą gniazd sieciowych. Klient zapewnia interfejs graficzny umożliwiający logowanie, rejestrację oraz wysyłanie i odbieranie wiadomości. Po zalogowaniu klient ma dostęp do listy aktywnych użytkowników, która jest aktualizowana przez serwer. Poprzez listę użytkownik wybiera klientów, z którymi może przeprowadzić konwersację.

Użyta technologia:
- Język programowania: C (serwer), Python (klient)
- Baza danych: SQLite
- Protokół komunikacyjny: TCP/IP
- Biblioteka PyQt5 (interfejs graficzny)

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

Komunikacja ma model klient-serwer i opiera się na użyciu gniazd sieciowych (sockets), gdzie serwer ma możliwość obsługi wielu klientów jednocześnie i umożliwia komunikację w czasie rzeczywistym. Serwer nasłuchuje na pojawiających się klientów, a po pojawieniu się nowego akceptuje połączenie i tworzy dla niego osobny wątek. Połączenie między klientem i serwerem nawiązywane jest na wolnym adresie i porcie 1100.

Klient po nawiązaniu połączenia przechodzi proces autoryzacji, który obsługuje logowanie i rejestrację. Dane przesyłane są do serwera, który z wykorzystaniem bazy danych sprawdza ich poprawność. użytkownika Komunikacja wykorzystuje mechanizmy send i recv i odbywa się sekwencyjnie, gdzie recv blokuje wątek do momentu otrzymania nowej wiadomości. Klienci wykorzystują sygnały, które przekazują informacje o odebraniu wiadomości od serwera.

Wątek danego klienta na serwerze ciągle oczekuje na przesłane wiadomości i w zależności od odebranego komunikatu (np. /list w celu uzyskania listy dostępnych użytkowników, /logout w celu wylogowania) wykonuje różne czynności. Każdy użytkownik posiada listę zalogowanych klientów aktualizowaną z każdą zmianą - logowaniem / wylogowaniem użytkownika. W przypadku zalogowania klient dopisywany jest do listy

uzupełnianej przez serwer, a wylogowania usuwany z niej. Całą komunikacja zabezpiecona jest mechanizmem mutex.

3. Podsumowanie (0.5-1 strona)
- Najważniejsze informacje o implementacji
- Co sprawiło trudność

Serwer wykorzystuje wielowątkowość do obsługi wielu klientów jednocześnie. Synchronizacja dostępu do współdzielonych zasobów, takich jak lista aktywnych użytkowników, została osiągnięta za pomocą mechanizmów synchronizacji z użycie mutexów.

Klient ma możliwość prowadzenia prywatnego czatu z jednym innym klientem lub stworzenie czatu grupowego składającego się trzech użytkowników, poprzez zaznaczenie ich nazw na liście dostępnych użytkowników i rozpoczęcie konwersacji. Gdy klient wysyła wiadomość w czacie grupowym, serwer odbierają ją i wysyła do wszystkich klientów, będących członkami konwersacji grupowej.

Baza danych SQLite została wykorzystana do przechowywania danych użytkowników – nazwy użytkownika i hasła, co zapewnia przechowywanie ich poza aplikacją. Umożliwia to zachowanie danych użytkowników niezależnie od działania serwera.

Klient został zaimplementowany w języku Python z wykorzystaniem biblioteki PyQt5, która umożliwia tworzenie interfejsów graficznych. W ramach interfejsu stworzono kilka kluczowych elementów. Pierwszym z nich jest okno logowania i rejestracji, które wyświetla się po uruchomieniu klienta. Po poprawnej autentykacji, użytkownik ma dostęp do listy aktywnych użytkowników, która jest dynamicznie aktualizowana przez serwer i umożliwia wybór innych użytkowników do chatu. Głównym elementem interfejsu jest okno czatu, które pozwala na wysyłanie i wyświetlanie wiadomości w czasie rzeczywistym. Integracja interfejsu graficznego z logiką komunikacji sieciowej wymagała obsługi sygnałów, które są kluczowe dla reakcji interfejsu.

Pierwszym napotkanym problemem był proces logowania oraz ponownego logowania po wylogowaniu. Problem z logowaniem wynikał ze zmieszania wysyłanych przez klienta wiadomości zawierających kolejno interakcję, nazwę oraz hasło w buforze, co uniemożliwiło poprawną weryfikację w bazie danych. Natomiast po pierwszym logowaniu serwer zapamiętywał użytkownika jako już zalogowanego i nie pozwalał na ponowienie czynności.

Podczas implementacji problematyczne okazało się aktualizowanie listy użytkowników, ponieważ informacja przesyłana była z wyprzedzeniem interfejsu graficznego, co skutkowało nie wyświetlaniem się użytkowników. Klient potrzebował odbierać komunikaty przy każdej zmianie użytkowników - zalogowania, wylogowania lub całkowitego zamknięcia.

Kolejną trudnością była możliwość prowadzenia chatów z wieloma użytkownikami. Wiadomość odebrana przez serwer musi zawierać ciąg nazw opisujących adresatów, których odróżnienie mogło wprowadzać błędy.