Mateusz Chmielewski, 145385
Maciej Kleban, 147031

Sprawozdanie z Sieci Komputerowych 2

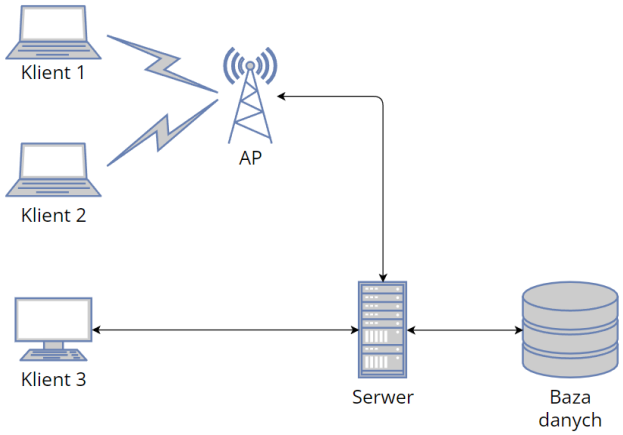
Komunikator internetowy.

Aplikacja pozwalająca na komunikację między klientami.

1. **Opis projektu** (0.5 strony)

Projekt ma na celu utworzenie komunikatora tekstowego między wieloma użytkownikami jednocześnie (na wzór takich komunikatorów jak Messenger).
W projekcie serwer został napisany w języku C za pomocą bsd-sockets, natomiast aplikacja użytkownika została napisana w języku Python przy użyciu biblioteki Kivy i KivyMD.
W celu przechowywania danych projekt korzysta również z bazy danych SQLite3 oraz wątków pthread. Komunikator umożliwia prowadzenie wielu konwersacji jednocześnie na różnych komputerach.
Program będący wynikiem projektu posiada również obsługę błędów (takich jak utworzenie tego samego użytkownika wielokrotnie, wysyłanie wiadomości do nieistniejącego użytkownika).

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



Klient porozumiewa się z serwerem za pomocą bsd-sockets. Serwer ma przypisany stały port i przy każdym połączeniu klienta z serwerem tworzy dla niego osobny wątek, w którym przechowuje również jego deskryptor, z którego później korzysta do komunikacji zwrotnej z klientem. Serwer może się porozumiewać z każdym komputerem w sieci, w której się znajduje.

3. **Podsumowanie** (0.5-1 strona)

Serwer został podzielony na cztery podstawowe funkcjonalności reprezentowane przez osobne pliki w repozytorium projektu. Pierwszym z nich jest podstawowy serwer, który tworzy dla nowy wątek przy każdym połączeniu z nowym użytkownikiem (*serwer.c*), komunikacja z bazą danych w celu zarządzania danymi takimi jak dane logowania, lista znajomych czy wysłane wiadomości (*database.c*), funkcje pomocnicze wykorzystywane przez serwer do zarządzania jego funkcjonalnością (*utils.c*) oraz ekstrakcja pól z danych od klienta (*parser.c*).

Serwer przyjmuje dane w postaci:

<Endpoint> GET / POST <Dane>

Gdzie *<Dane>* są w postaci zwykłej zmiennej typu JSON o określonych polach, opisanych w kodzie stosownym komentarzem w pliku *utils.c*. *GET* oznacza odbieranie danych, a *POST* - wysyłanie. Każde żądanie od klienta uzyskuje informację zwrotną od serwera.

Połączenie z bazą danych odbywa się za pomocą biblioteki SQLite3 poprzez pakiet (*sqlite3-dev*) dostępną na systemach Linux lub poprzez SQLite3 Amalgamation, który umożliwia wykorzystanie bazy danych bez pobierania pakietu. Budowanie projektu odbywa się przy pomocy Makefile.

Przed interakcją z bazą danych należy utworzyć dla niej strukturę (co zostało podkreślone w pliku *README.md*). W repozytorium znajduje się plik *script.sql*, który tworzy odpowiednią strukturę w bazie danych.

Aplikacja klienta została napisana w języku python, przy użyciu bibliotek Kivy 2.0 oraz KivyMD, wprowadzającej szablony GUI w stylu *material design*, obecne w systemie Android. Pozwala to na generowanie responsywnego, łatwego w zrozumieniu i przyjemnego dla oka interfejsu użytkownika.
Do obsługi połączeń aplikacja wykorzystuje bibliotekę sockets oraz json. Pierwsza z nich służy do bezpośredniego łączenia się z serwerem, wysyłania zapytań oraz odbierania odpowiedzi; druga do generowania wiadomości rozumianej przez serwer.

Największą trudność sprawiły wykrycie i obsługa błędów w skrajnych sytuacjach, np. zawieszanie się aplikacji w momencie, kiedy użytkownik nie posiadający przyjaciół wysłał zapytanie o podsumowanie tejże listy.
Na większą uwagę zasługuje problem blokowania komendy *recv* po stronie klienta, która to może doprowadzić do zawieszenia się aplikacji w przypadku nie otrzymania odpowiedzi bądź próbie odebrania pustej wiadomości.

Kolejnym etapem rozwoju byłoby wprowadzenie czatu “w czasie rzeczywistym” (bez konieczności ciągłego odświeżania odebranych wiadomości) oraz uzupełnienie podglądu historii wiadomości o te wysłane przez użytkownika.