Zajęcia Projektowe - Programowanie Sieciowe		
Imię i nazwisko: **Miłosz Janik** **Antoni Kijania** **Kacper Czepiec**	Temat projektu: **Studencki pokój do rozmów (chatroom wielowątkowy za pomocą serwera współbieżnego)**	
Grupa laboratoryjna: 3	Data: 14.01.2021 r.	Dokumentacja Projektu

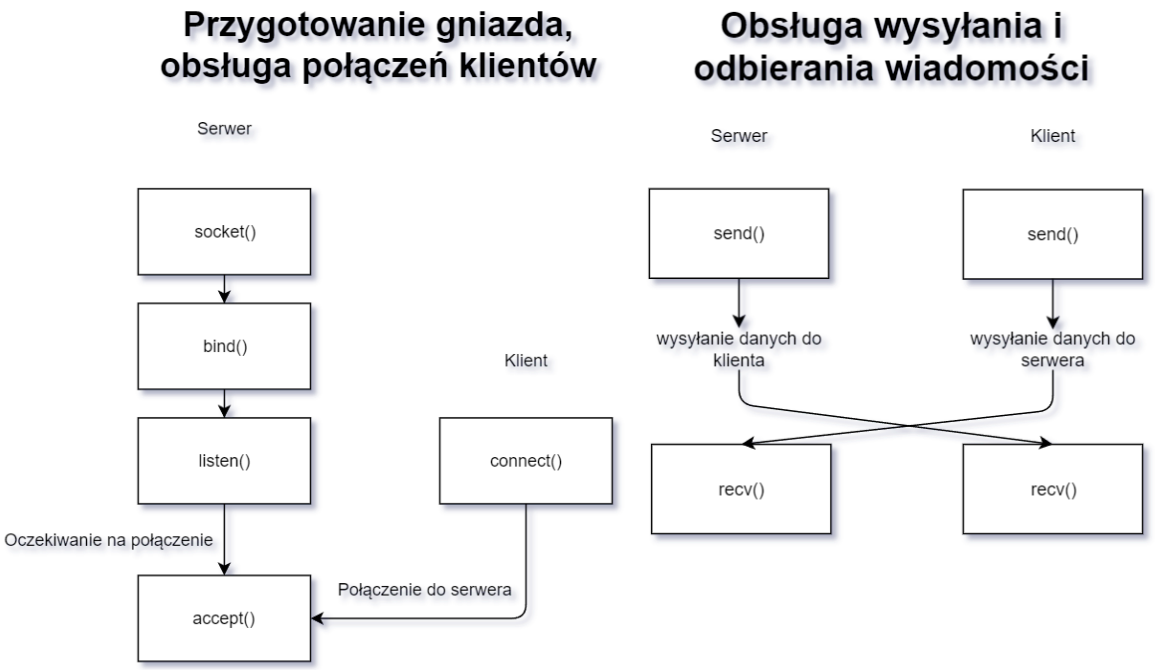
1. Funkcjonalność projektu

a) Serwer

Serwer ustanawia połączenie pomiędzy określoną liczbą klientów, umożliwiając im wysyłanie wiadomości tekstowych. Wymaga wybrania portu do nasłuchiwania w oczekiwaniu na połączenia klientów oraz określenia maksymalnej liczby klientów. Jeśli liczba ta zostanie potencjalnie przekroczona, to użytkownik próbujący dostać się na serwer, nie dostanie zaakceptowany.
Serwer wyświetla na bieżąco w oknie terminala informacje o nowych połączeniach, zmianach nazw użytkownika oraz treść wiadomości publicznych. Umożliwia to sprawne zarządzanie serwerem i monitorowanie jego stanu.

b) Klient

Klient wymaga podania IP serwera oraz port do połączenia. Użytkownik jest proszony również o podanie unikalnej nazwy użytkownika oraz grupy dziekańskiej /laboratoryjnej, aby wyróżniać się w pokoju do rozmów indywidualną nazwą. W przypadku wybrania już zajętej nazwy użytkownik zostanie o tym poinformowany. Klient wysyła na czat swoją wiadomość, a każdy inny klient otrzymuje informację o nadawcy i treść wiadomości. Istnieje możliwość użycia dodatkowych funkcji, których lista zostanie wyświetlona po wysłaniu polecenia **/pomoc**:
/msg - wysłanie wiadomości prywatnej do wyłącznie jednego, wybranego użytkownika. Inni członkowie czatu oraz serwer nie zostaną o tym poinformowani.
/changename - pozwala użytkownikowi na zmianę wybranego wcześniej pseudonimu. Wciąż musi on być oryginalny, więc nie jest możliwa obecność dwóch uczestników o tych samych nazwach.
/online - wyświetla listę aktualnie zalogowanych użytkowników.
/wyjdz - pozwala opuścić pokój chatu.



2. Użyte funkcje i biblioteki

Wszystkie funkcjonalności zostały szczegółowo opisane za pomocą obszernych komentarzy w kodzie.

- Język programowania - Python 3.9 z użyciem biblioteki socket
- Protokół TCP i adresacja IPv6
- Wielowątkowość po stronie klienta i serwera (biblioteka *threading*)
- blok `if __name__ == "__main__":`
- Wykorzystujemy przy wysyłaniu i odbieraniu system kodowania Unicode `"utf8"`.

Klient:

- Zawiera bibliotekę *datetime* dopisująca bieżącą godzinę do wysyłanej wiadomości.
- Tworzy osobne wątki do odbierania i do wysłania wiadomości - dopóki oba są “żywe”, klient działa.
- Klient łączy się z serwerem za pomocą socket.connect(), do wysyłania używa funkcji socket.send(), a do odbierania wiadomości socket.recv().
- Nie posiada logiki umożliwiającej odpowiednią reakcję na wprowadzone dane przez użytkownika - wysyła wszystkie dane do serwera, który odpowiednio je obsługuje.

Serwer:

- Serwer nasłuchuje na porcie wybranym przez administratora. Klientowi zostaje przydzielony inny, określony port zdalny (remote).
- Aplikacja serwera tworzy wątek przyjmujący połączenia od nowych klientów ***connectionThread.*** On z kolei, po przyjęciu nowych połączeń (sock.accept()), każdemu klientowi przydziela osobny wątek ***clientThread*** do jego obsługi.
- Dane klientów są przechowywane w dwóch słownikach: ***users*** oraz ***addresses.*** Pierwszy z nich przechowuje obiekt gniazda danego klienta jako klucz oraz nazwę użytkownika jako wartość. Drugi słownik również posiada gniazdo jako klucz, lecz adres IP klienta jako wartość.
- Komunikacja z klientami odbywa się za pomocą funkcji **send()** i **recv()**.
- Serwer wypisuje na bieżąco komunikaty w terminalu za pomocą funkcji **print().**
- W przypadku problemów z komunikacją z danym klientem, zostaje on usunięty ze słowników, a jego połączenie zostanie zamknięte za pomocą **close()**.
- Obsługuje wszystkie dodatkowe funkcjonalności czatu (poprzedzone "/") - sprawdza treść wiadomości wysłanej przez użytkownika.
- Wiadomości publiczne ***sendtoall*** są zrealizowane przez iterowanie po słowniku users. Wiadomości prywatne natomiast wykorzystują wyszukiwanie konkretnego użytkownika w tym słowniku.
- Po opuszczeniu programu przez klienta, następuje zamknięcie wszystkich połączeń na używanym przez niego gnieździe przez zaimportowany moduł *atexit.*

Źródła:

- https://docs.python.org/3/library/socket.html#module-socket
- https://realpython.com/intro-to-python-threading/
- https://realpython.com/python-sockets/
- https://docs.python.org/3/tutorial/datastructures.html
- https://docs.python.org/3/library/codecs.html
- https://stackoverflow.com/questions/41250805/how-do-i-print-the-local-and-remote-address-and-port-of-a-connected-socket