Sprawozdanie - sieci komputerowe 2 (SK2)	
Prowadzący: *mgr inż. Michał Boroń*	Grupa: *3.2*
Temat Ćwiczeń: *Komunikator typu GG*	
Autorzy: *Daniel Zdancewicz[145317] i Monika Zielińska [143719]*	

1 Wstęp

Projekt implementujący prosty czat opierający się na komunikacji klient-serwer.

1.1 Technologia

Zastosowane technologie

- CMake w wersji 3.22
- Język C w standardzie C23 wraz z rozszerzeniami:
  - OpenSSL - biblioteka nadająca podstawowe interfejsy do paczek szyfrujących
  - Crypto - biblioteka współdziałająca z biblioteką OpenSSL
  - PThreads - biblioteka pozwalająca na tworzenie osobnych wątków
- Websocket
- Język Typescript wraz z rozszerzeniami:
  - React - framework budujący i przetrzymujący stan klient'a
  - React-DOM - rozszerzenie framework'u React o obsługę przeglądarki
  - Lodash - biblioteka pomocniczych metod programowania funkcyjnego
  - MUI - biblioteka zawierająca stylizowane komponenty do biblioteki React
  - Emotion/Styled - biblioteka wspierająca komponenty stylizowane w bibliotece React
  - Classnames - biblioteka pomocnicza zawierająca możliwość konkatenacji nazw klas języka css
  - vite - Technologia wspierająca proces tworzenia i budowania aplikacji
  - Node - Technologia obsługująca połączenie z przeglądarką

2 Opis protokołu

Protokoły wykorzystane w projekcie to HTTP oraz Websocket:

- Websocket - technologia pozwalająca na tworzenie obustronnych sesji między użytkownikami przeglądarki i aplikacji zewnętrznych. Interfejs ten pozwala na przesyłanie wiadomości i ich odczytywanie za pomocą mechanizmu notyfikacji serwera
- HTTP - wykorzystany do wstępnego utworzenia komunikacji za pomocą klasycznego handshake'a przy wykorzystaniu protokołu Websocket. Działa na zasadzie żądań i odpowiedzi w modelu klient-serwer. Serwer po otrzymaniu żądania stara się odpowiedzieć zawierając różnego typu nagłówki i dane niezbędne do zrozumienia odpowiedzi

2.1 Struktura

- Serwer rozpoczyna pracę na porcie 9090
  - W między czasie rozpoczyna się wątek zbierający statystyki dotyczące mechanizmu notyfikacji i aktualnych połączeń zawartych w pamięci serwera
  - próbuje na oddzielnym wątku utworzyć połączenie z systemowym mechanizmem notyfikacji
  - Wątek główny oczekuje na zakończenie pracy wątku mechanizmu notyfikacji by bezpiecznie zakończyć pracę aplikacji
- Po rozpoczęciu pracy przez serwer istnieje możliwość połączenia się z poziomu przeglądarki przy pomocy utworzenia klienta za pomocą websocket skierowanego na port 9090 z odpowiadającym mu nagłówkiem wskazującym na jego unikalna nazwę
- Przy próbie połączenia się klienta z serwerem, tworzy się socket na serwerze odpowiedzialny za komunikację w stronę klienta. Ta przy pierwszym wydarzeniu odczytu wysyłana żądanie HTTP o przywitanie się. Te zawiera imię klienta i klucz szyfrujący połączenie websocket. Klient oczekuje na zaszyfrowaną w SHA1 i zaszyfrowaną w technologii base64.
- Po zatwierdzonym połączeniu socket klienta przechodzi w tryb nasłuchiwania na datagramy nadesłane w protokole websocket obsługując kilka wypadków.
  - Wiadomość: Przesłanie wiadomości z jednego klienta do drugiego
  - Żądanie informacji zawartych na serwerze: Klient może żądać informacje o aktualnych użytkownikach serwera
  - Zakończenie połączenia: Klient żąda zakończenia połączenia websocket przez co z serwera są wykasowane jego dane i zamykany jest jego używany socket i jest usuwany z kolejki oczekiwania na wydarzenie
  - Błędny datagram: Datagram jest ignorowany i serwer jest informowany o jego błędności

3 Sposób uruchomienia

3.1 Wymagania

- System operacyjny oparty na architekturze UNIX
- CMake 3.22
- npm + node 16+

3.2 Polecenia

- Kompilacja i uruchomienia aplikacji obsługującej serwer
  - cmake CmakeLists.txt
  - cd bin && ./server
- Utworzenie aplikacji klient
  - npm install
  - npm start