

Prosty serwer HTTP zgodny z RFC 2616 co najmniej w zakresie żądań GET, HEAD, PUT, DELETE

Opis protokołu:

Nasz projekt wykorzystuje protokół transmisji TCP, który jest połączeniowy, niezawodny i szeroko wykorzystywany w komunikacji klient-serwer. W tym podejściu serwer oczekuje na nawiązanie połączenia na określonym porcie, a klient inicjuje połączenie do serwera. Zaletą tego protokołu jest gwarancja dostarczenia wszystkich pakietów w całości do odbiorcy z zachowaniem kolejności i bez duplikacji pakietów.

W nawiązaniu połączenia wykorzystywana jest procedura three-way handshake. Najpierw host A wysyła segment SYN do hosta B. Następnie jeśli B chce nawiązać połączenie wysyła do hostu A segment SYN z informacją o dolnym numerze sekwencyjnym wykorzystywanym do numerowania segmentów. Następnie jeśli A otrzyma segment SYN przechodzi w stan ESTABLISHED oraz potwierdza hostowi B odebranie sygnału SYN. W tym momencie może zostać już nawiązana transmisja.

Socket to wewnętrzne gniazdo w serwerze bądź kliencie wykorzystywane do wysyłania i odbierania danych. Każde gniazdo posiada trzy główne właściwości: typ gniazda określający protokół wymiany danych, lokalny adres IP oraz numer portu na którym dana usługa będzie pracować. Komunikacja może zostać zestawiona jeśli klient nawiąże połączenie z serwerem podając adres IP oraz numer portu serwera.

Opis implementacji:

Naszym zadaniem było wykonanie prostego serwera HTTP zdolnego wykonywać żądania GET, HEAD, PUT, DELETE.

W naszym rozwiązaniu każda z powyższych metod została zaimplementowana w osobnej funkcji. W części main zawarliśmy podstawowe elementy tworzenia serwera TCP, a także zaimplementowaliśmy obsługę wielowątkową. Zamki założone są na dostępie do plików współdzielonych to jest books.json i endpoints_url.txt. Dzięki temu mamy pewność, że edycja i odczyt tych plików jest atomowy i każdy z klientów przegląda bądź edytuje stabilne źródło danych. Serwer jest w pełni współbieżny. Nowy klient dodawany jest do tablicy klientów serwera, a wcześniej sprawdzany czy jego deskryptor nie jest już utworzony w tablicy klientów. Jeżeli jest to klient nie wysyła kolejnego requesta, w przeciwnym przypadku request jest wysyłany a klient otrzymuje odpowiedź od serwera. Po przeprowadzeniu takiej operacji, pamięć dynamiczna używana przez klienta jest zwalniana, a miejsce jego deskryptora w tablicy zastąpione jest zerem, dzięki temu, kolejny klient będzie mógł dodać się na zwolnione miejsce w tablicy.

Funkcja build_request parsuje adres URL oraz odczytuje typ zapytania i w zależności od parametrów uruchamia odpowiednią funkcję, bądź zwraca stronę błędu.

GET:

Pozwala wylistować całą listę książek, bądź konkretny element z listy. W naszym podejściu, przy kompletowaniu odpowiedzi korzystamy z pliku tymczasowego w którym umieszczamy poszczególne części odpowiedzi, tj. nagłówek i ciało, a następnie wysyłamy całą odpowiedź do odbiorcy.

HEAD

Po wpisaniu adresu niezależnie czy odwołując się do konkretnej książki czy do całego zbioru, zwraca nagłówek informujący o powodzeniu bądź błędzie wykonania.

PUT

Pozwala dodawać nowe książki do zbioru, a także modyfikować już istniejące. Jeśli podamy w body zapytania rekord o nowym id zostanie on dodany do bazy, jeśli podamy rekord o numerze id istniejącym w bazie, taki rekord zostanie zmodyfikowany.

DELETE

Usuwa daną pozycję z bazy danych oraz zwraca zaktualizowaną listę wszystkich książek bez usuniętej pozycji.

Dodatkowo zaimplementowaliśmy obsługę POST oraz no_implemented.

Post - pozwala na dodanie nowego rekordu do bazy danych. Niezależnie od wejścia ta metoda zawsze zwróci ten sam wynik, na takich samych danych.

No_implemented – zwraca stronę błędu serwera, w przypadku gdy użytkownik wykona niezaimplementowaną na serwerze usługę HTTP.

Opis sposobu kompilacji i uruchomienia projektu:

W celu kompilacji należy otworzyć folder z projektem. Następnie, będąc w katalogu głównym projektu, należy wykonać w terminalu komendę `cd /src`

Kolejnym krokiem jest kompilacja projektu poprzez wykonanie polecenia:
`gcc -pthread server.c -Wall -o server`

Następnie można uruchomić server przy pomocy polecenia `./server`

Uruchomienie klienta

Klientem naszej aplikacji jest aplikacja Postman bądź przeglądarka internetowa. W aplikacji klienta wystarczy podać adres ip serwera, dla lokalnego serwera jest to adres 127.0.0.1 bądź inny w przypadku komunikacji poprzez sieć oraz wskazać numer portu. Nasza aplikacja działa na porcie nr. 8080.

Nasza aplikacja wykonuje polecenia na zbiorze rekordów, zawierających opis książek, które znajdują się w pliku .json.

W pliku README.md na gitlabie umieściliśmy sposoby wykonania zapytań przy użyciu aplikacji Postman oraz lokalnego adresu IP.