Wprowadzenie

W ćwiczeniu wykorzystuje się platformę node.js i dodatkowe pakiety języka Javascript, w związku z czym rudymentarna znajomość tego języka jest przydatna, choć nie jest niezbędna.

Przebieg ćwiczenia obejmuje uruchomienie i eksperymenty z czterema wersjami bardzo prostego serwera. W każdym etapie po uruchomieniu serwera (poprzedzonym ewentualnym doprowadzeniem środowiska do stanu początkowego, np. poprzez restart) należy przeprowadzić kilka operacji, w tym ewentualnie dokonać modyfikacji serwisu. Wyniki w postaci kopii znakowej z terminala należy wykorzystać do udokumentowania przebiegu działań.

Dla obserwowania komunikacji należy wykorzystać konsole (terminale znakowe) - pomimo, iż użycie bardziej zaawansowanych narzędzi (np. *Postman*) do obserwowania zachowania serwisów jest dopuszczalne, jednak zalecane jest korzystanie z dwóch terminali lub ewentualnie przeglądarki; terminal przeznaczony jest do uruchomienia serwera, obserwacji logu i dla wysyłania zleceń i obserwowania wyników, za pomocą programu *curl*. Pozwala to łatwo i bardzo szybko kopiować znakowo fragmenty konwersacji dla umieszczenia w raporcie ćwiczenia (zamiast używać zrzutów z ekranu, które zwykle wymagają przycięcia do rozsądnych rozmiarów).

Etap 1: Katalog 01_HttpServer

Działania:

- Posługując się drugim terminalem przy pomocy komendy **curl** z parametrem **-X** wysłać żądanie wykonania metody GET dla głównego URLa serwisu a następnie dla podstrony "/hello" (należy łączyć się na port 3000).
- Zaobserwować wyprowadzane przez serwer komunikaty i przeanalizować kod aplikacji; sprawdzić która część kodu jest odpowiedzialna za raporty i zmienić ją tak, aby raportowała również czas obsługiwanego wywołania (metoda *Date.now()* w JavaScript).
- Dodać obsługę metody GET dla ścieżki URL */time*, która zwróci aktualny czas.
- Zamieścić w raporcie zmieniony fragment kodu (z zakomentowaną starą wersją) i przykładowe wyniki.

Etap 2: Katalog 02_HttpServer

Komentarz:

- Drugi przykład rozszerza poprzedni przez dodanie parametrów, przekazywanych do serwera po znaku pytajnika kończącego część główną URLa. Parametry stanowią część wzorca URLa (używanego przez pakiet Express), w którym sygnalizujemy ich obecność za pomocą prefiksowania nazwy dwukropkiem, np: "/hello/:**name**".
  Komponent z takim wzorcem będzie przechwytywał i obsługiwał URL zaczynający się od "/hello/", po którym pojawi się jeden człon ścieżki, którego wartość będzie udostępniona w zmiennej o nazwie *request.params.**name***.
- Kwerendy reprezentują dodatkowe dane przekazywane w części URLa po pytajniku. Dane zawsze są dostarczane w URLu w postaci par: *nazwa=wartość*, które mogą sie powtarzać, jak w przykładzie: "/patient?name=John&surname=Doe". Należy pamiętać o tym, że jeśli para, specyfikująca wartość parametru, nie wystąpi, to parametr przy próbie pobrania otrzyma wartość **null**; odpowiedni test wykluczający odwołanie z użyciem wartości **null** jest niezbędny.

Działania:

- Wyprowadzić (najlepiej w przeglądarce) i przeanalizować wynik metody GET dla bazowego URLa i innych wariantów wywołań.
- Dodać obsługę metody GET dla ścieżki URLa składającego się z trzech parametrów, która losowo zwraca jedną z części URLa (tj. jeden z parametrów). Aby uzyskać losową wartość całkowitą z przedziału [min..max] można użyć funkcji
  ```
  function getRandomInt(min, max) { return Math.floor(Math.random() * (max - min + 1)) + min; }
  ```
- Zamieścić w raporcie nową wersję fragmentu kodu i przykładowe wyniki.

Etap 3: Katalog 03_HttpServer

Komentarz:

- Trzeci przykład pokazuje użycie metod innych niż GET. Serwer przechowuje listę nazw obiektów, która jest uzupełniana wskutek wywołań metody PUT z URLem "/item/:name" (co powoduje dodanie obiektu o wskazanej identyfikacji za pierwszym razem, zaś za każdym następnym odwołaniem z tą samą nazwą obiektu w URLu tylko wyprowadzenie komunikatu).
- Aby w języku Javascript zmienić zawartość elementu tablicy zawierającego określoną starą zawartość można użyć konstrukcji:

```
items[items.indexOf(itemName)] = newItemName;
```

Działania:

- Wyprowadzić i przeanalizować wynik metody GET dla bazowego URLa i innych wariantów wywołań.
- Zmodyfikować kod, tak aby dodawanie nowego elementu odbywało się przy pomocy metody HTTP POST, a modyfikacja - przy pomocy metody PUT (nową nazwę dla elementu należy podawać przy pomocy kwerendy).
- Zamieścić w raporcie wyniki testów, nową wersję fragmentu kodu i przykładowe wyniki jej testów.

Etap 4: Katalog 04_HttpServer

Komentarz:

- Ten przykład pokazuje implementację prostego API. Po starcie serwer, wykorzystując pakiet *lowdb* tworzy bazę w postaci pliku (o nazwie *db.json*).

Działania:

- Wyprowadzić i przeanalizować wynik metody GET dla bazowego URLa.
- Zaobserwować rezultaty dla URLa zawierającego numer (*id*) pacjenta w zależności od użytej metody HTTP.
  Dla testowania wygodnie jest użyć wywołania **curl -X**
  Przeanalizować różnice w logice poszczególnych implementacji kodu dla obsługi tych metod.
  Uwaga: Zwrócić uwagę na fragment kodu pomiędzy db i średnikiem, który odwołuje się do kodu pakietu *lowdb*, zainicjowanego na początku.
- Zamieścić w raporcie przykładowe wyniki (i komentarz do nich).