Podstawy komunikacji HTTP i ograniczenia komunikacji dla treści międzydomenowych (CORS)

*Zadania do wykonania na udostępnionym obrazie systemu **Mint** (użytkownik: bsi, hasło: isb)*

A. Podstawy komunikacji HTTP

1. W przeglądarce Chrome wchodzimy na stronę **https://httpbin.org/**. Jest to serwis HTTP umożliwiający testowanie różnych rodzajów komunikacji (m.in różne metod HTTP). W dalszej części ćwiczenia wykorzystamy umieszczone na stronie przykłady.
    a. Sprawdzamy **każdy** kolejny przykład (**Try It Out** -> **Execute**). Podczas pracy używamy debuggera (wybieramy w nim zakładkę Network). Porównujemy efekty zapytań wyświetlone na stronie z wynikami w debugerze. Należy dokładnie przeanalizować jak wyglądają:
        i. Request URL
        ii. Request Headers
        iii. Response Headers

        W przypadku problemu z interpretacją nagłówków należy zapoznać się z ich dokładniejszymi opisami, np. na **MDN**[1]
    b. Dla wybranego przez prowadzącego zestawu przykładów sprawdzamy komunikację:
        i. z użyciem narzędzia **curl**[2] (raz złożyć polecenie ręcznie, drugi raz pobrać gotowy request z debugera Chrome analizując komunikację sieciową z poprzedniego punktu, zakładka Network)
        ii. z użyciem narzędzia **Isomnia**, ręcznie składając zapytania (zwrócić uwagę na nagłówki) i interpretując odpowiedzi. Spróbować wstawić zapytanie kopiując polecenie curl z debugera Chrome, zakładka Network.
        iii. w konsoli Chrome z użyciem funkcji **fetch** (zwrócić uwagę na wykorzystanie promisów)[3]
        iv. w konsoli Chrome z użyciem obiektu **XMLHttpRequest**[4]
2. Stworzyć plik **html**, zawierający skrypt z wywołaniem **fetch** dla trzech przykładów z https://httpbin.org. Plik umieścić w osobnym katalogu. Zawartość katalogu udostępnić za pomocą gotowego serwera plików (z wykorzystaniem **nodejs**, **python** bądź **ruby**) oraz niezależnie, na innym porcie, za pomocą własnego prostego serwera plików napisanego w **nodejs**[5]. Załadować stronę w przeglądarce, w osobnych kartach z różnych portów. W debugerze (zakładka Network) dokładnie przeanalizować przebieg komunikacji.

B. Komunikacja międzydomenowa (CORS)

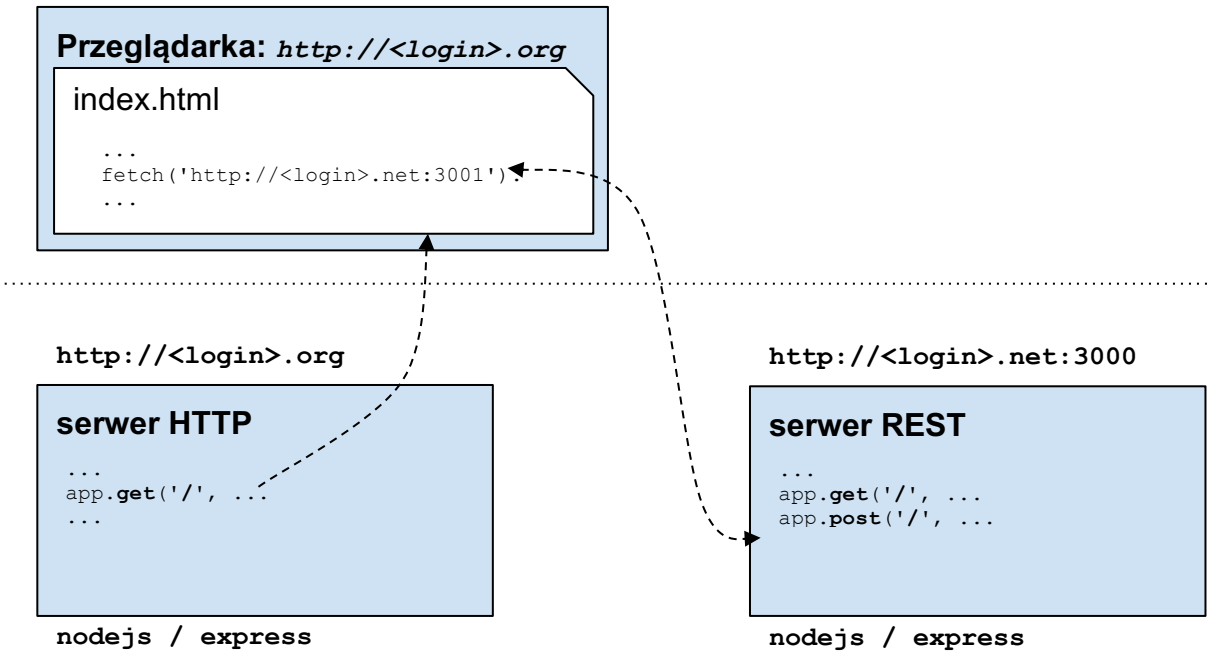
1. Wykorzystując plik **/etc/hosts** zamapować dwie nazwy domeny **<login>.org** i **<login>.net** (np. **wmackow.org**) na lokalne IP maszyny
2. Utworzyć nowy katalog na projekt, zainicjować w nim npm (**npm init**) i zainstalować lokalnie **Express** (npm install express --save). Zweryfikować zawartość pliku **package.json**
3. Korzystając z **Express** (uwaga: nie wykorzystujemy generatora express, a jedynie biblioteki) napisać prosty serwer REST, działający na HTTP, który będzie pozwalał na wywołania GET i

---

[1] https://developer.mozilla.org/pl/docs/Web/HTTP/Headers
[2] https://flaviocopes.com/http-curl/
[3] https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch
[4] https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/Using_XMLHttpRequest
[5] https://adrianmejia.com/blog/2016/08/24/building-a-node-js-static-file-server-files-over-http-using-es6/

POST[6]. Serwer będzie przechowywał zmienną całkowitą, której wartość będzie można pobierać za pomocą GET (bezparametrowe) a ustawiać za pomocą POST (w ciele polecenia wysyłamy JSON z nową wartością)[7]. W przypadku obydwu metod serwer zwraca aktualną wartość zmiennej w postaci JSON. Działanie serwera przetestować np. za pomocą narzędzia **Insomnia**. Serwer uruchomić np. na porcie 3000.

4. Napisać prostą stronę HTML, która będzie zawierała dwa przyciski i pole tekstowe. Przyciski będa służyły wywołania metod GET i POST naszego serwera REST (użyć np. fetch). Pole tekstowe będzi pokazywało orzymana wartość całkowitą jak i służyło do utworzenia żądania REST. Plik strony umieścić w podkatalogu **public** naszego projektu. Do serwera rest odwołujemy się za pomocą adresu **<login>.net:<port>** (np. wmackow.net:3000).
5. Korzystając **z Express** napisać prosty serwer plików udostępniający zawartość katalogu **public**[8]. Uruchomić serwer na porcie 80. Zwróć uwagę, że w systeme Linux do zajęcia portów o numerze poniżej 1024 potrzebne są uprawnienia administracyjne.



Rys.1 Schemat komunikacji pomiędzy przeglądarką a serwerami (wykonanie ćwiczenia do kroku 9).

6. W przeglądarce otworzyć udostępnioną przez serwer plików stronę odwołując się do adresu **<login>.org**. Spróbować wykorzystać przyciski na stronie.
7. Włączyć podstawową obsługę **CORS**, która pozwoli na korzystanie przez naszą stronę z naszego serwera REST[9]. Który z serwerów musimy zmodyfikować?
8. Spróbować uzyskać ten sam efekt bez wykorzystania ***middleware'u* cors** z poprzedniego punktu a jedynie odpowiedni preparując nagłówki żądań i odpowiedzi.

[6] https://expressjs.com/en/starter/hello-world.html
[7] https://expressjs.com/en/4x/api.html#req.body
[8] https://expressjs.com/en/4x/api.html#router.use
[9] https://medium.com/@alexishevia/using-cors-in-express-cac7e29b005b