Node.js

Laboratorium 3

1. Wykorzystując zdobytą wiedzę z poprzednich zajęć skorzystaj z zewnętrznej biblioteki 'yargs' (https://www.npmjs.com/package/yargs) oraz wbudowanego modułu file system(fs) i stwórz aplikację która zapisze przekazane argumenty

> node app.js --name Jan --lastname Nowak

efektem końcowym powinien być plik z zawartością:
{"name":"Jan","lastname":"Nowak"}

podpowiedź: aby możliwe było zapisanie naszego obiektu musimy go przekonwertować do postaci tekstowej(`string`) funkcją `JSON.stringify()`

2. Wykorzystując plik wynikowy z zadania 1 odczytaj plik i wyświetl jedynie nazwisko (`lastname`) w konsoli.

Końcowym efektem jest wyświetlenie na konsoli nazwiska zapisanego w pliku.

podpowiedź: aby wczytać dany plik wykorzystujemy asynchroniczną funkcję `readFile()` z modułu wbudowanego(`fs`). Kolejnym elementem jest przekonwertowanie wczytanego ciągu znaków do obiektu wykorzystując funkcję `JSON.parse()`

3. Bazując na zadaniu 2 zabezpieczmy naszą aplikację tak aby w przypadku błędu odczytu pliku(zła nazwa pliku/inny rodzaj błędu) poinformowała użytkownika o problemie w konsoli.

4. Stwórz aplikację która pobierze informację o użytkowniku podając jego id z API: https://jsonplaceholder.typicode.com/users/{ID}
   i wyświetl w konsoli współrzędne geograficzne skąd dany użytkownik pochodzi oraz jego imię.

Endpoint z przykładowym ID: https://jsonplaceholder.typicode.com/users/2

Uruchomienie aplikacji:
> node app.js

Wynik w konsoli

lat -43.9509
lng -34.4618

5. Rozszerzmy zadanie 4 tak aby nasza aplikacja poinformowała użytkownika o błędzie pobierania danych lub braku szukanego użytkownika w bazie.

Każdy `response` w swoim `callback` posiada informacje niezbędne do sprawdzenia poprawności pobranych danych, taki callback wygląda następująco:

```
function callback(error, response, body) {
    // … ciało funkcji
}
```

error – jest błędem który informuję nas iż jest problem z połączeniem do serwera

response – jest naszą odpowiedzią która zawiera informacje jakie serwer nam zwrócił(w naszym zadaniu istotny jest `statusCode`, w przypadku zwrócenia poprawnie danych otrzymamy kod z numerem 200 (https://pl.wikipedia.org/wiki/Kod_odpowiedzi_HTTP) w innym przypadku dostaniemy inny kod.

body – dane zwrócone z naszego API.

W tym zadaniu pobawmy się w debugowanie! Zobaczmy kiedy i jakie dane przychodzą z serwera!

6. Dodajmy do zadania 5 możliwość dynamicznego wprowadzania ID poprzez wczytanie zewnętrznej biblioteki `yargs`

> node app.js --id=2

Wynik w konsoli:

Ervin Howell
lat -43.9509
lng -34.4618

> node app.js --id=20

Wynik w konsoli:

User not found.

7. Wykorzystując zadanie 6 dodajmy opcję pobrania pogody dla wczytanego użytkownika

Endpoint wygląda następująca:
https://api.openweathermap.org/data/2.5/weather?appid=0ed761300a2725ca778c07831ae64d6e&lat={LAT}&lon={LNG}

Pod kluczem {LAT} i {LNG} powinniśmy podać wczytane dane z naszego wcześniejszego endpointu.

8. Podobnie jak w zadaniu 5 zabezpieczmy naszą aplikację z zadania 7 tak aby poinformować użytkownika o przypadkowych problemach z naszym API.

9. Podzielmy naszą aplikację z punktu 8 na moduły odpowiednio:

app.js – plik główny który uruchomi naszą aplikację i wywoła funkcje z naszego modułu user.js

user.js – plik z pobieraniem danych użytkownika

weather.js – plik z pobieraniem danych o prognozie pogody

DROBNE WSKAZÓWKI

1. Konwersja obiektu do postaci tekstowej(string)

```
const obj = {
   name: 'Jan',
   lastname: 'Nowak'
};
const objString = JSON.stringify(obj);

console.log(objString);
```

wynik: {"name":"Jan","lastname":"Nowak"}

2. Konwersja tekstu typu JSON do obiektu:

```
const objString = '{"name":"Jan","lastname":"Nowak"}';
const obj = JSON.parse(objString);

console.log(obj.name)
```

Wynik Jan