

REST API CLIENT

SPIS TREŚCI

| | |
|------------------------------|---|
| Spis treści | 1 |
| Cel zajęć | 1 |
| Rozpoczęcie..... | 1 |
| Uwaga | 1 |
| Wymagania | 2 |
| Badanie API | 2 |
| Implementacja | 2 |
| Commit projektu do GIT | 4 |
| Podsumowanie..... | 4 |

CEL ZAJĘĆ

Celem głównym zajęć jest zdobycie następujących umiejętności:

- pobieranie danych z zewnętrznych zasobów za pomocą REST API
- zdobywanie wiedzy na temat zewnętrznych API za pomocą dokumentacji typu Swagger
- wysyłanie asynchronicznych żądań z wykorzystaniem XMLHttpRequest i Fetch API

W praktycznym wymiarze uczestnicy stworzą dynamiczną stronę HTML pozwalającą na wyświetlanie bieżącej informacji pogodowej oraz prognoz dla zadanej przez użytkownika miejscowości.

ROZPOCZĘCIE

Rozpoczęcie zajęć. Powtórzenie wykonywania połączeń synchronicznych i asynchronicznych z poziomu JS na stronie.

Wejściówka?

UWAGA

Ten dokument aktywnie wykorzystuje niestandardowe właściwości. Podobnie jak w LAB A wejdź do Plik -> Informacje -> Właściwości -> Właściwości zaawansowane -> Niestandardowe i zaktualizuj pola. Następnie uruchom ten dokument ponownie lub Ctrl+A -> F9.

WYMAGANIA

W ramach LAB D przygotowane powinny zostać:

- pojedyncza strona HTML ze skryptem ładowanym z zewnętrznego pliku JS
- pole tekstowe (input typu „text”) do wprowadzania adresu
- przycisk „Pogoda”, po kliknięciu którego wykonywane jest zapytanie asynchroniczne:
 - do API Current Weather: <https://openweathermap.org/current> za pomocą XMLHttpRequest
 - do API 5 day forecast: <https://openweathermap.org/forecast5> za pomocą Fetch API
- obsługa zwrotki z obu API – wypisanie pogody bieżącej oraz prognoz poniżej pola wyszukiwania.

Wygeneruj klucz do API. Ponieważ aktywacja może chwilę potrwać, na czas trwania laboratorium możesz wykorzystać „ślużbowy” klucz: 7ded80d91f2b280ec979100cc8bbba94. **UWAGA!** Klucz zostanie dezaktywowany niedługo po zajęciach. Musisz wygenerować swój własny.

W przypadku blokady twórczej można posilić się filmem: <https://www.youtube.com/watch?v=WoKp2qDFxKK> jednakże spróbuj rozwiązać ten problem samodzielnie!

Prowadzący omówi powyższe wymagania. Upewnij się, czy wszystko rozumiesz.

Tu umieść swoje notatki:

...notatki...

BADANIE API

Poświęć kilka minut na wykonanie przykładowych zapytań do API z poziomu paska adresu przeglądarki. Podaj wymagane parametry dla osiągnięcia różnych wyników. Zbadaj odpowiedzi API, aby uzyskać pełen obraz wymagań i możliwości API.

IMPLEMENTACJA

Tradycyjnie implementację należy zacząć od zbudowania w HTML + CSS wszystkich wymaganych elementów / placeholderów na te elementy. Następnie krok po kroku należy implementować poszczególne zachowania.

Wstaw zrzut ekranu zawierającego stronę ze wszystkimi elementami, tj. pole tekstowe, przycisk, miejsce do wyświetlenia pogody i prognozy:

| | | |
|---------|---|---|
| Punkty: | 0 | 1 |
|---------|---|---|

Wstaw zrzut ekranu kodu odpowiedzialnego za wysyłanie żądania do current za pomocą XMLHttpRequest:

```
getCurrentWeather(query){  
    let url = this.currentWeatherLink.replace("{query}", query);  
    let req = new XMLHttpRequest();  
    req.open("GET",url,true);  
    req.addEventListener("load",() => {  
        this.currentWeather = JSON.parse(req.responseText);  
        console.log(JSON.parse(req.responseText));  
        this.drawWeather();  
    });  
    req.send();  
}
```

Wstaw zrzut ekranu pokazujący otrzymaną odpowiedź za pomocą `console.log()` w przeglądarce.

```

{
  "coord": {
    "lon": 14.553,
    "lat": 53.4289
  },
  "base": "stations",
  "weather": [
    {
      "id": 3083829,
      "main": "Clouds",
      "description": "few clouds",
      "icon": "02d"
    }
  ],
  "clouds": {
    "all": 100
  },
  "dt": 1764174481,
  "id": 2034200,
  "name": "Szczecin",
  "sys": {
    "type": 2,
    "id": 2034200,
    "country": "PL",
    "sunrise": 1764139605,
    "sunset": 1764168710
  },
  "main": {
    "temp": 3.57,
    "feels_like": -0.17,
    "temp_min": 2.77,
    "temp_max": 4.12,
    "pressure": 1016,
    "humidity": 85
  },
  "wind": {
    "speed": 4.47,
    "deg": 330
  },
  "visibility": 10000,
  "status": "OK"
}
  
```

> *(ctrl)[i] to turn on code suggestions. Don't show again* NEW

| | | |
|---------|---|---|
| Punkty: | 0 | 1 |
|---------|---|---|

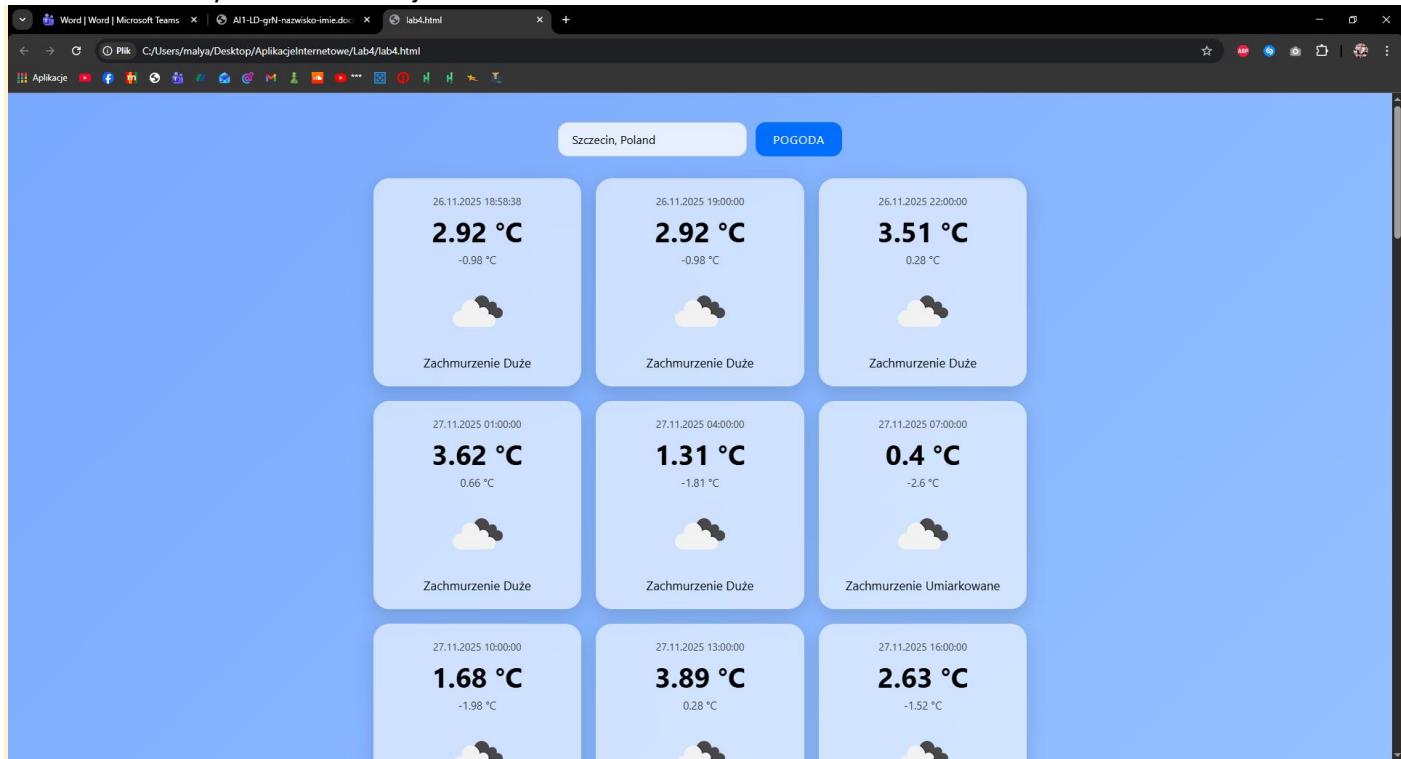
Wstaw zrzut ekranu kodu odpowiedzialnego za wysyłanie żądania do forecast za pomocą Fetch:

```

getForecast(query){
  let url = this.forecastLink.replace("{query}", query);
  fetch(url)
    .then((response) =>{
      //console.log(response);
      return response.json();
    })
    .then((data) => {
      //console.log(data);
      this.forecast = data.list
      this.drawWeather();
    })
;
}
  
```

Wstaw zrzut ekranu pokazujący otrzymaną odpowiedź za pomocą `console.log()` w przeglądarce.

Wstaw zrzut ekranu przedstawiajacego wizualizację prognozy pogody:



Upewnij się, że widoczne są pasek wyszukiwania ze wskazaną miejscowością, a także zarówno pogoda bieżąca jak i prognozy pogody.

Punkty:

0

1

COMMIT PROJEKTU DO GIT

Zacommituj i pushnij swoje rozwiązanie do repozytorium GIT.

Upewnij się, czy wszystko dobrze się wysłało. Jeśli tak, to z poziomu przeglądarki utwórz branch o nazwie `lab-d` na podstawie głównej gałęzi kodu.

Podaj link do brancha `lab-d` w swoim repozytorium:

<https://github.com/anticzektriczek/AplikacjeInternetowe/tree/main/Lab4>

PODSUMOWANIE

W kilku słowach/zdaniach napisz swoje przemyślenia odnośnie tego laboratorium. Nie używaj LLM.

Na początku nie wiedziałem jak podejść do tego zadania. Na szczęście pomógł mi film podlinkowany wyżej, co pchnęło moją pracę do przodu. Projekt sprawił mi dużo satysfakcji kiedy widziałem, że wszystko działa jak należy

mogę sprawdzić pogodę na całym świecie. Teraz już wiem jaką jest różnica między XML i fetch API i mam nadzieję, że Pani prowadząca nie będzie musiała robić wejściówek.

Zweryfikuj kompletność sprawozdania. Utwórz PDF i wyślij w terminie.