

## 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 .....	6
Podsumowanie .....	7

### 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 „służbowy” klucz: `7ded80d91f2b280ec979100cc8bbba94`. **UWAGA!** Klucz zostanie dezaktywowany niedługo po zajęciach. Musisz wygenerować swój własny.

W przypadku blokady twórczej można posiłkować 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...

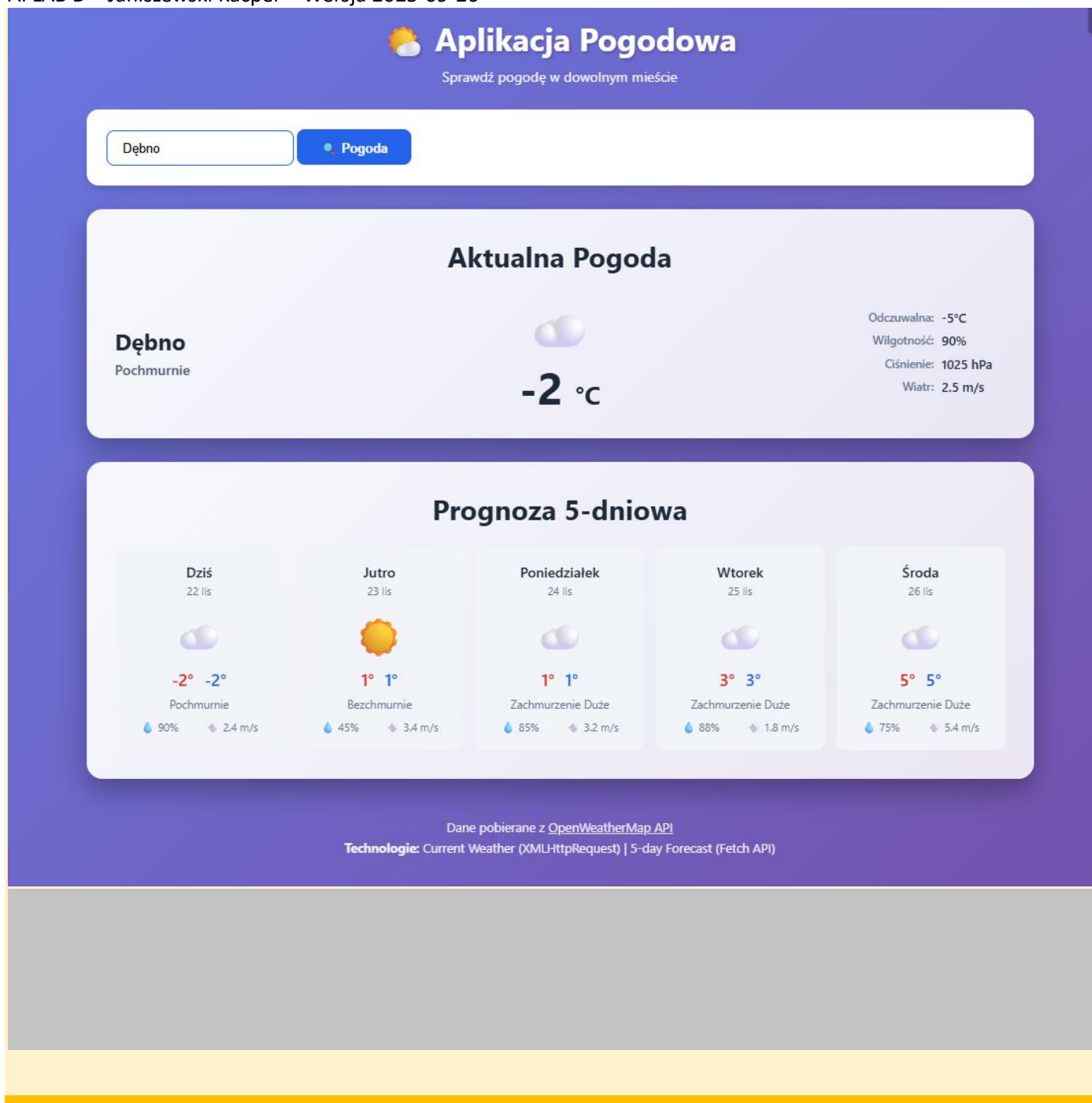
## BADANIE API

Poświęć kilka minut na wykonanie przykładowych zapytań do API z poziomu pasku 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:

```
function getCurrentWeatherXHR(city) { Pin selection to current chat prompt (Ctrl+Alt+X) | Don't show this again
  return new Promise((resolve, reject) => {
    const xhr = new XMLHttpRequest();
    const url = `${API_BASE_URL}/weather?q=${encodeURIComponent(city)}&appid=${API_KEY}&units=metric&lang=pl`;

    xhr.open('GET', url, true);

    xhr.onload = function() {
      if (xhr.status >= 200 && xhr.status < 300) {
        try {
          const data = JSON.parse(xhr.responseText);
          console.log('XMLHttpRequest - Dane z API Current Weather:', data);
          displayCurrentWeather(data);
          resolve(data);
        } catch (error) {
          reject(new Error('Błąd parsowania danych'));
        }
      } else if (xhr.status === 404) {
        showError('Nie znaleziono miasta. Sprawdź poprawność nazwy.');
```

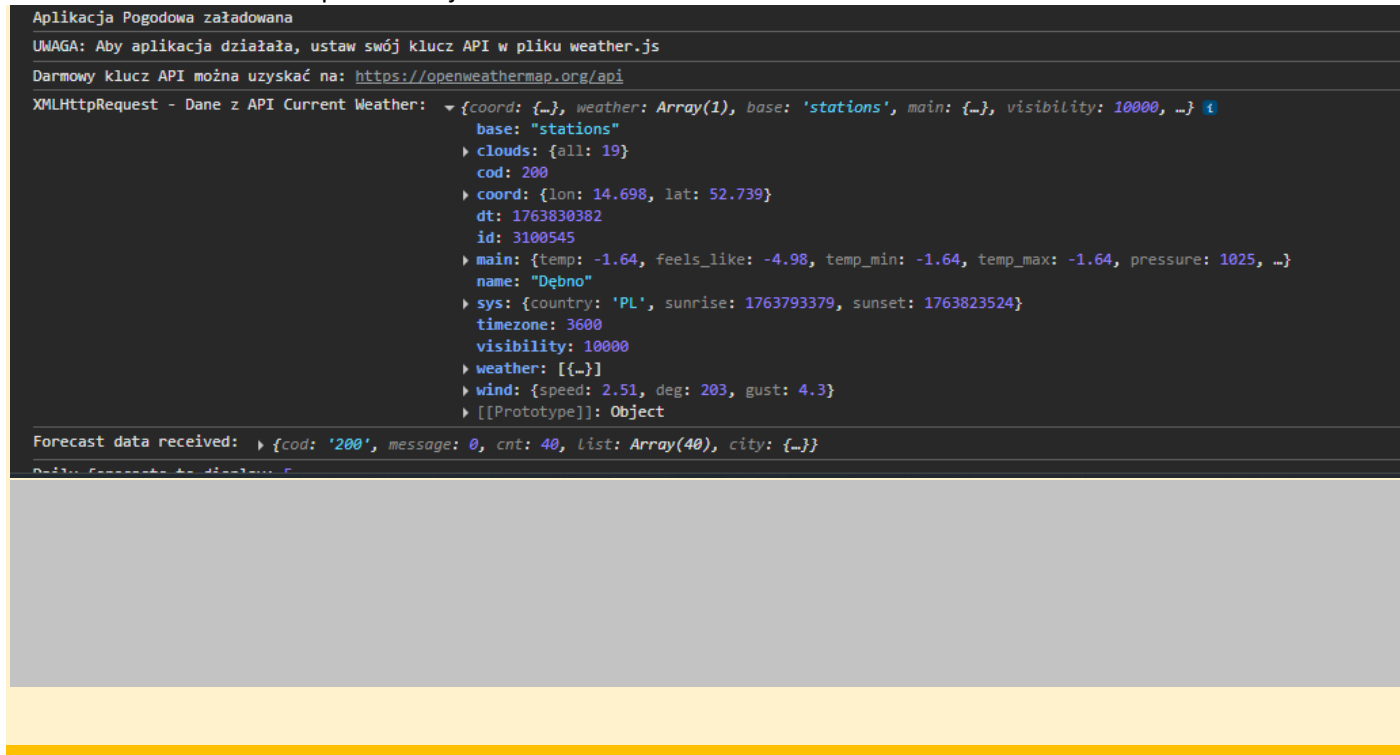
```
        reject(new Error('City not found'));
      } else {
        showError(`Błąd HTTP: ${xhr.status}`);
        reject(new Error(`HTTP Error: ${xhr.status}`));
      }
    };

    xhr.onerror = function() {
      showError('Błąd połączenia. Sprawdź połączenie internetowe.');
```

```
      reject(new Error('Network error'));
    };

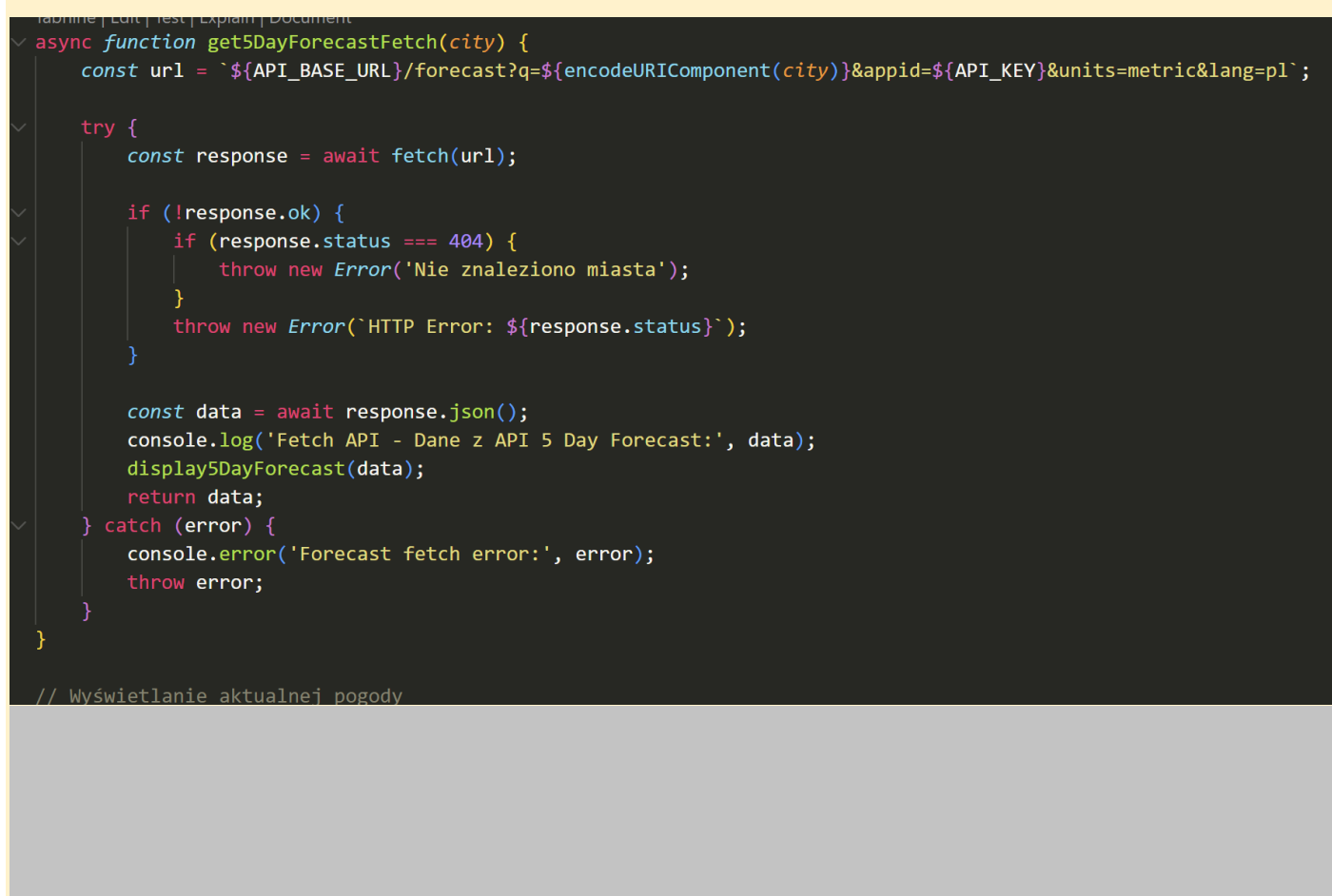
    xhr.send();
  });
}
```

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



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

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



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



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

Wstaw zrzut ekranu przedstawiającego wizualizację prognoz 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:

...link, np. <https://github.com/inazwisko/ai1-lab/tree/lab-d...>

<https://github.com/Griniuu/Aplikacje-Internetowe.git>

## PODSUMOWANIE

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

Zadanie fajne do rozwoju umiejętności JS, przypomnienie HTML i CSS i nauka używania zewnętrzne API

Zweryfikuj kompletność sprawozdania. Utwórz PDF i wyślij w terminie.