

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	7
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 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

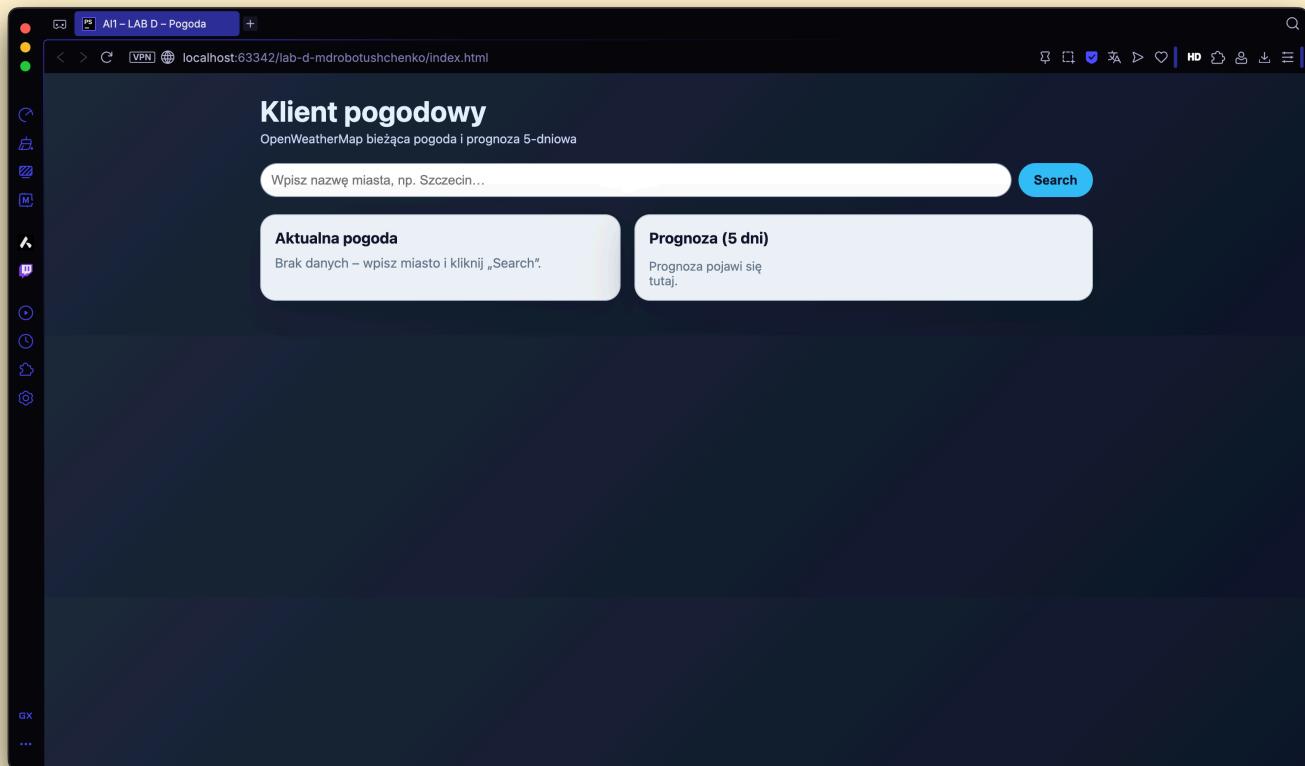
Poświęć kilka minut na wykonanie przykładowych zapytań do API z poziomu paska adresu przeglądarki. Podaj wymagane parametry dla osiągnięcia różnych wyników. Zbadaj odpowiedzi API, aby uzyskać pełen obraz wymagań i możliwości API.

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:

The screenshot shows a code editor with a project named "lab-d-mdrobotushchenko". The file "app.js" is open, containing JavaScript code for a weather application. The code includes functions for getting current weather by city and by coordinates, using the OpenWeatherMap API. It also handles errors and logs data to the console. A sidebar provides tips for geolocation implementation and a snippet of alternative code for handling geolocation permission.

```

geoBtn.addEventListener("click", () => {
  ...
  // CURRENT WEATHER po nazwie miasta (XMLHttpRequest)
  Show usages
  function getCurrentWeatherByCity(city) : void {
    const url : string = `https://api.openweathermap.org/data/2.5/weather?q=${encodeURIComponent(city)}&appid=${API_KEY}&units=metric&lang=pl`;

    const xhr : XMLHttpRequest = new XMLHttpRequest();
    xhr.open( method: "GET", url);

    xhr.onload = function () : void {
      if (xhr.status === 200) {
        const data = JSON.parse(xhr.responseText);
        console.log("CURRENT WEATHER (city):", data);

        renderCurrentWeather(data);
      } else {
        console.error(`Błąd, status: ${xhr.status}, ${xhr.statusText}`);
        currentBox.innerHTML = `

Nie udało się pobrać aktualnej pogody (błąd ${xhr.status})

`;
      }
    };

    xhr.onerror = function () : void {
      console.error(`Błąd połączenia (XMLHttpRequest)`);
      currentBox.innerHTML = `

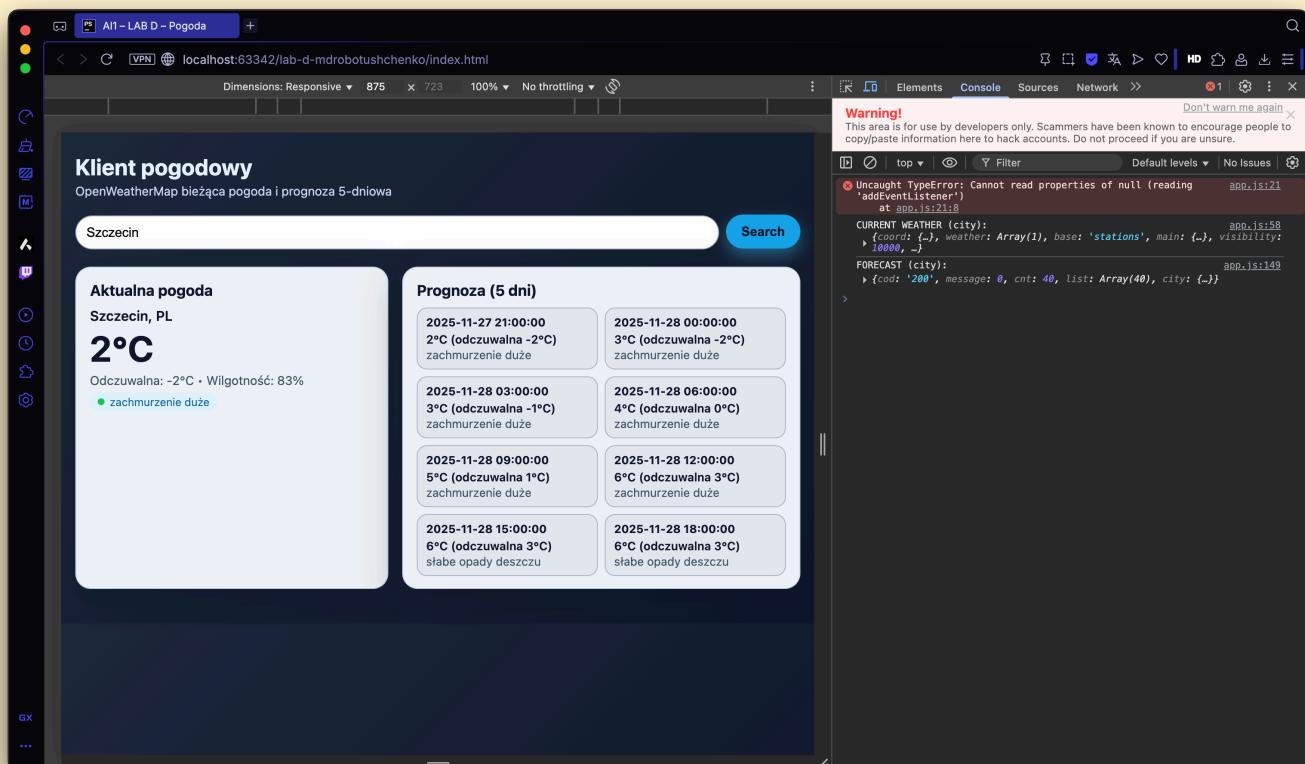
Błąd połączenia po stronie klienta

`;
    };
    xhr.send();
  }

  // CURRENT WEATHER po współrzędnych (XMLHttpRequest)
  Show usages
  function getCurrentWeatherByCoords(lat, lon) : void {
    const url : string = `https://api.openweathermap.org/data/2.5/weather?lat=${lat}&lon=${lon}&appid=${API_KEY}&units=metric`;
  }
});

```

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:

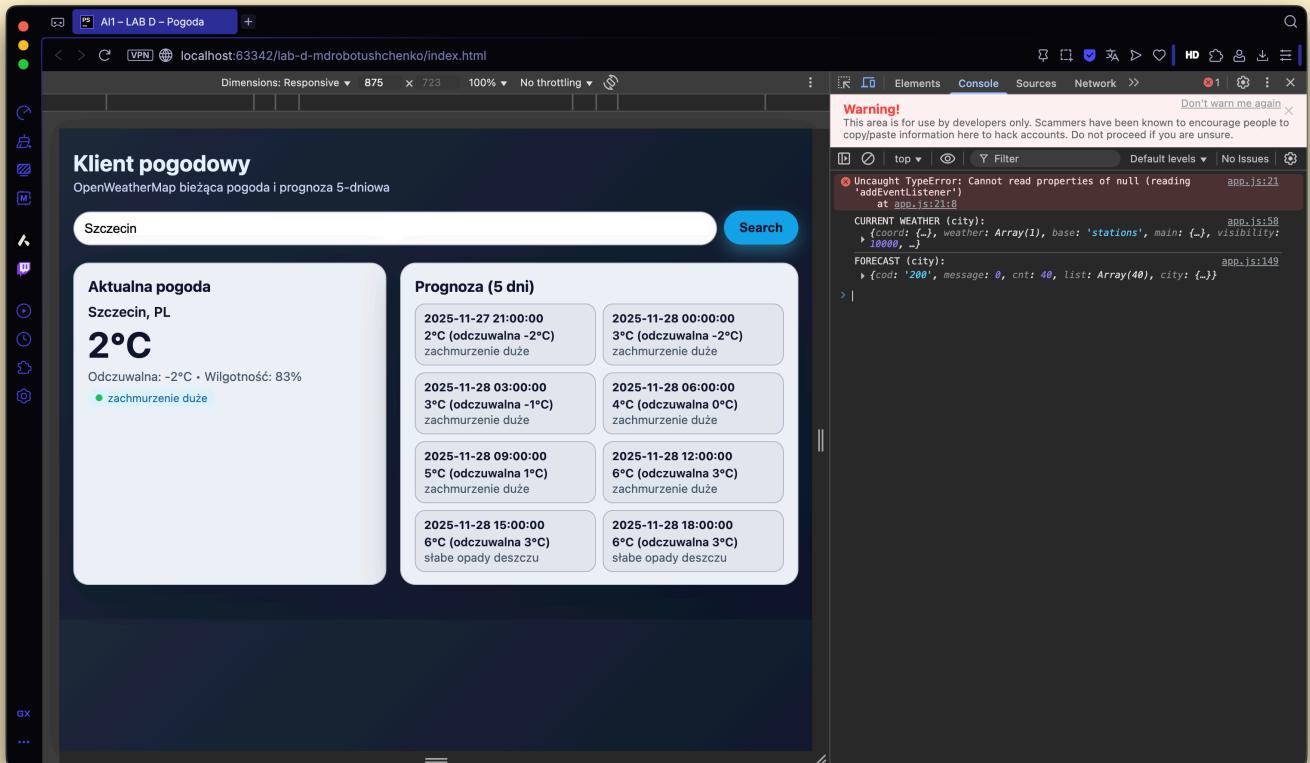
```

140     function fetchForecast(url, logLabel) : void {
141         fetch(url).Promise<Response>
142             .then((res : Response) => {
143                 if (!res.ok) {
144                     throw new Error("HTTP " + res.status);
145                 }
146                 return res.json();
147             }).Promise<any>
148             .then((data) : void => {
149                 console.log(logLabel, data);
150
151                 if (!data.list || !Array.isArray(data.list)) {
152                     forecastBox.innerHTML =
153                         '<p class="placeholder">Brak danych prognozy</p>';
154                     return;
155                 }
156
157                 const items : unknown[] = data.list.slice(0, 8);
158                 let html : string = "";
159
160                 items.forEach((item) : void => {
161                     const date = item.dt_txt;
162                     const t : number = Math.round(item.main.temp);
163                     const feels : number = Math.round(item.main.feels_like);
164                     const desc : AllowSharedBufferSource | string = item.weather[0].description;
165
166                     html += `
167                         <article class="forecast-item">
168                             <strong>${date}</strong>
169                             <div class="forecast-temp">${t}°C (odczuwalna ${feels}°C)</div>
170                             <div class="forecast-desc">${desc}</div>
171                         </article>
172                     `;
173                 });
174
175                 forecastBox.innerHTML = html;
176             }).Promise<void>
177             .catch((err) : void => {
178                 console.error("Blad Fetchforecast:", err);
179                 forecastBox.innerHTML =
180                     '<p class="placeholder">Nie udało się pobrać prognozy pogody</p>';
181             });
182     }

```

The screenshot shows a code editor with a dark theme. The main pane displays a JavaScript file named 'app.js' containing code for fetching weather data from a URL using the Fetch API. The code handles errors and logs the received data to the console. A tooltip on the right side of the editor provides a brief explanation of the logic, mentioning permission handling and error handling. The bottom status bar shows PHP: 5.6, 180:77, LF, UTF-8, and other system information.

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

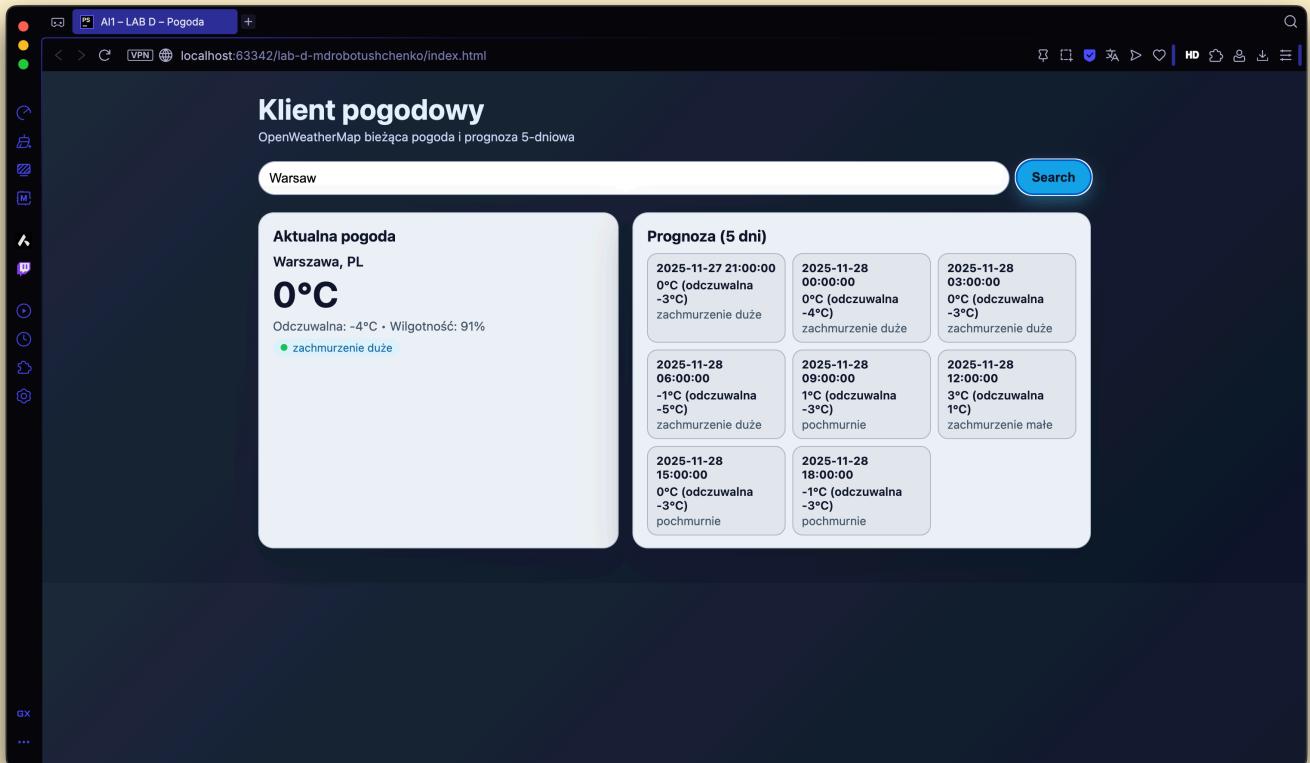


Punkty:

0

1

Wstaw zrzut ekranu przedstawiającego 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:

...link, np. <https://github.com/Rakinary/lab-d>

PODSUMOWANIE

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

...podsumowanie

Laboratorium okazało się całkiem praktyczne i pozwoliło mi lepiej zrozumieć działanie REST API oraz różnice między użyciem XMLHttpRequest i Fetch API. Udało mi się poprawnie pobrać zarówno aktualną pogodę, jak i 5-dniową prognozę, a następnie zaprezentować je w czytelnej formie kart. Próbowałem również zaimplementować funkcję pobierania pogody z wykorzystaniem geolokalizacji użytkownika. Sama część odpowiedzialna za wywołanie API działała poprawnie, jednak przeglądarka nie zwracała współrzędnych mimo że pojawiał się komunikat o udzieleniu dostępu.

Zweryfikuj kompletność sprawozdania. Utwórz PDF i wyślij w terminie.