

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.....	5
Podsumowanie.....	5

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 stornie.

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 własny lub wykorzystaj gotowy klucz do API: 7ded80d91f2b280ec979100cc8bbba94

W przypadku blokady można poszukać 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:

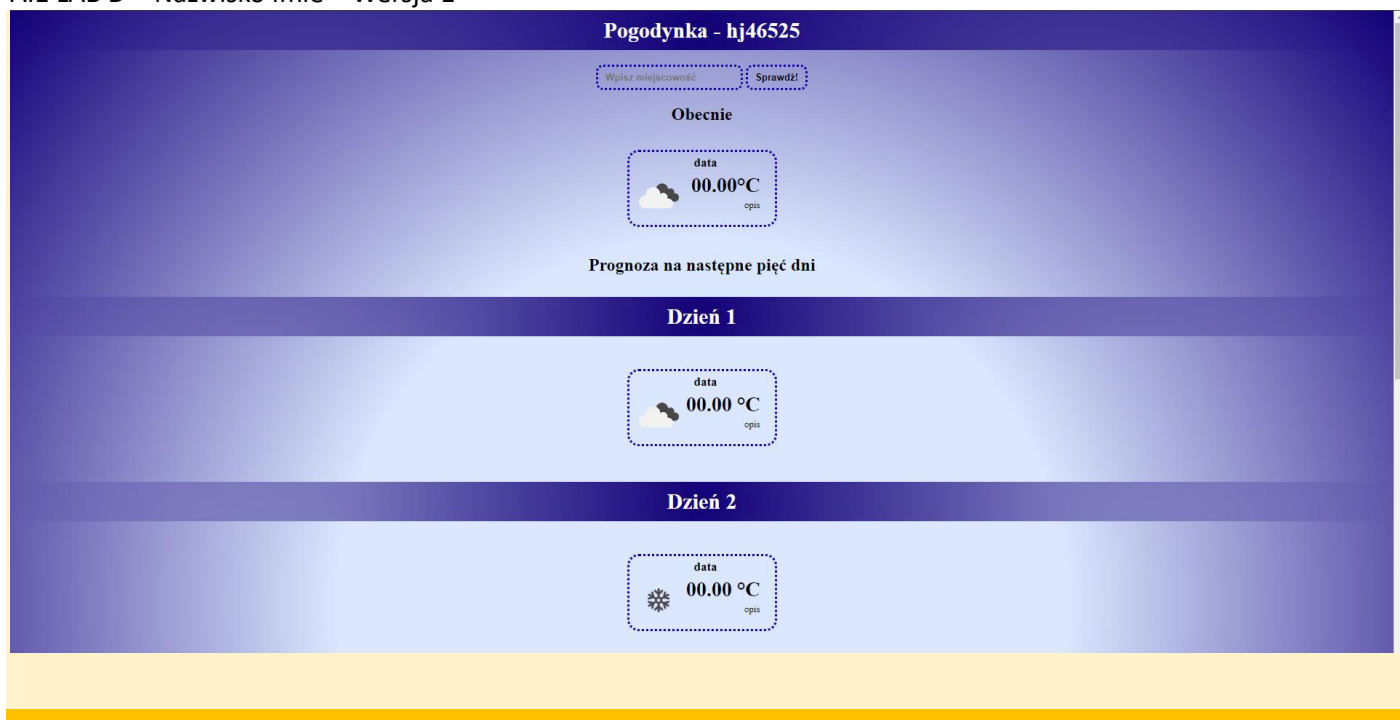
BADANIE API

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:

```

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

```

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

```

▼ Object 1
  base: "stations"
  ► clouds: {all: 100}
  cod: 200
  ► coord: {lon: 14.553, lat: 53.4289}
  dt: 1704312393
  id: 3083829
  ► main: {temp: 8.41, feels_like: 6.26, temp_min: 5.61, temp_max: 9.12, pressure: 987, ...}
  name: "Szczecin"
  ► sys: {type: 2, id: 19799, country: 'PL', sunrise: 1704266231, sunset: 1704293664}
  timezone: 3600
  visibility: 10000
  ► weather: [{-}]
  ► wind: {speed: 3.58, deg: 160}
  ► [[Prototype]]: Object

```

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

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

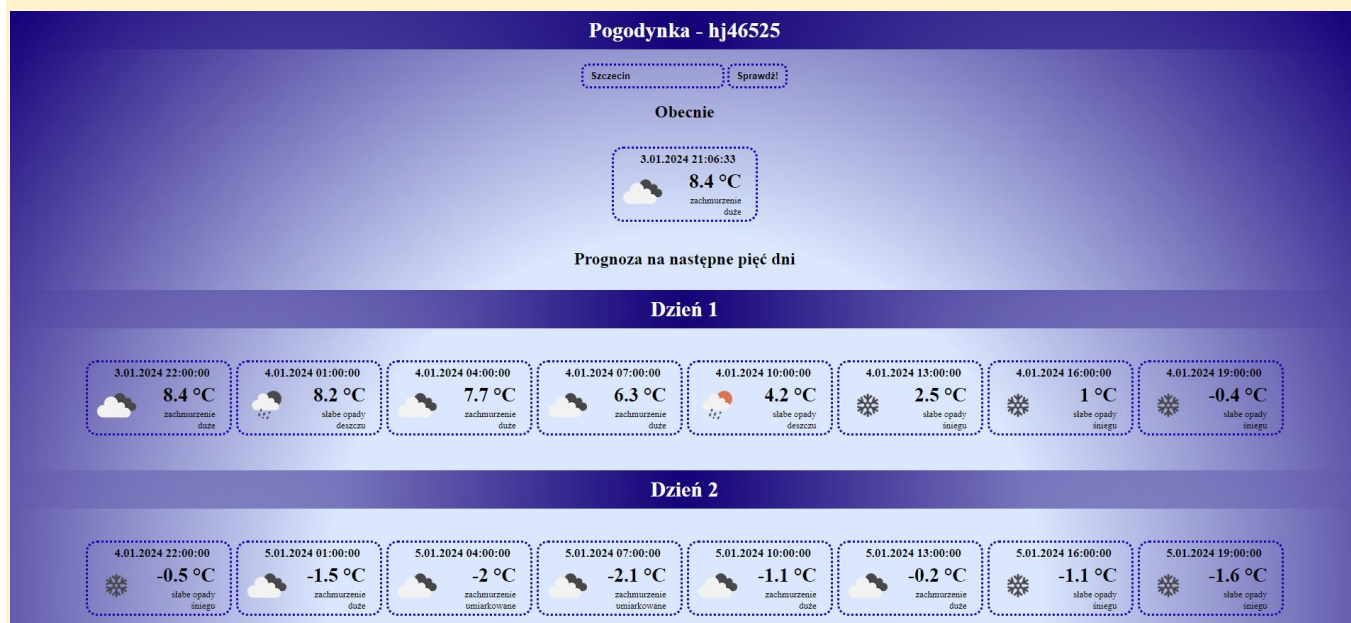
```
getForecast(cityQuery) {
  let url = this.forecastLink.replace("{cityQuery}", cityQuery);
  fetch(url)
    .then((response) => {
      return response.json();
    })
    .then((data) => {
      console.log(data);
      this.forecast = data.list;
      this.drawWeather(cityQuery);
    });
}
```

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

```
▼ {cod: '200', message: 0, cnt: 40, list: Array(40), city: {}} 1
  city: {id: 3083829, name: 'Szczecin', coord: {_, _}, country: 'PL', population: 407811, _}
  cnt: 40
  cod: "200"
  list: Array(40)
    ► 0: {dt: 1704315600, main: {_, _}, weather: Array(1), clouds: {_, _}, wind: {_, _}}
    ► 1: {dt: 1704326400, main: {_, _}, weather: Array(1), clouds: {_, _}, wind: {_, _}}
    ► 2: {dt: 1704337200, main: {_, _}, weather: Array(1), clouds: {_, _}, wind: {_, _}}
    ► 3: {dt: 1704348000, main: {_, _}, weather: Array(1), clouds: {_, _}, wind: {_, _}}
    ► 4: {dt: 1704358800, main: {_, _}, weather: Array(1), clouds: {_, _}, wind: {_, _}}
    ► 5: {dt: 1704369600, main: {_, _}, weather: Array(1), clouds: {_, _}, wind: {_, _}}
    ► 6: {dt: 1704380400, main: {_, _}, weather: Array(1), clouds: {_, _}, wind: {_, _}}
    ► 7: {dt: 1704391200, main: {_, _}, weather: Array(1), clouds: {_, _}, wind: {_, _}}
    ► 8: {dt: 1704402000, main: {_, _}, weather: Array(1), clouds: {_, _}, wind: {_, _}}
    ► 9: {dt: 1704412800, main: {_, _}, weather: Array(1), clouds: {_, _}, wind: {_, _}}
    ► 10: {dt: 1704423600, main: {_, _}, weather: Array(1), clouds: {_, _}, wind: {_, _}}
    ► 11: {dt: 1704434400, main: {_, _}, weather: Array(1), clouds: {_, _}, wind: {_, _}}
    ► 12: {dt: 1704445200, main: {_, _}, weather: Array(1), clouds: {_, _}, wind: {_, _}}
    ► 13: {dt: 1704456000, main: {_, _}, weather: Array(1), clouds: {_, _}, wind: {_, _}}
    ► 14: {dt: 1704466800, main: {_, _}, weather: Array(1), clouds: {_, _}, wind: {_, _}}
    ► 15: {dt: 1704477600, main: {_, _}, weather: Array(1), clouds: {_, _}, wind: {_, _}}
    ► 16: {dt: 1704488400, main: {_, _}, weather: Array(1), clouds: {_, _}, wind: {_, _}}
    ► 17: {dt: 1704499200, main: {_, _}, weather: Array(1), clouds: {_, _}, wind: {_, _}}
    ► 18: {dt: 1704510000, main: {_, _}, weather: Array(1), clouds: {_, _}, wind: {_, _}}
    ► 19: {dt: 1704520800, main: {_, _}, weather: Array(1), clouds: {_, _}, wind: {_, _}}
    ► 20: {dt: 1704531600, main: {_, _}, weather: Array(1), clouds: {_, _}, wind: {_, _}}
    ► 21: {dt: 1704542400, main: {_, _}, weather: Array(1), clouds: {_, _}, wind: {_, _}}
    ► 22: {dt: 1704553200, main: {_, _}, weather: Array(1), clouds: {_, _}, wind: {_, _}}
    ► 23: {dt: 1704564000, main: {_, _}, weather: Array(1), clouds: {_, _}, wind: {_, _}}
    ► 24: {dt: 1704574800, main: {_, _}, weather: Array(1), clouds: {_, _}, wind: {_, _}}
    ► 25: {dt: 1704585600, main: {_, _}, weather: Array(1), clouds: {_, _}, wind: {_, _}}
    ► 26: {dt: 1704596400, main: {_, _}, weather: Array(1), clouds: {_, _}, wind: {_, _}}
    ► 27: {dt: 1704607200, main: {_, _}, weather: Array(1), clouds: {_, _}, wind: {_, _}}
    ► 28: {dt: 1704618000, main: {_, _}, weather: Array(1), clouds: {_, _}, wind: {_, _}}
    ► 29: {dt: 1704628800, main: {_, _}, weather: Array(1), clouds: {_, _}, wind: {_, _}}
    ► 30: {dt: 1704639600, main: {_, _}, weather: Array(1), clouds: {_, _}, wind: {_, _}}
    ► 31: {dt: 1704650400, main: {_, _}, weather: Array(1), clouds: {_, _}, wind: {_, _}}
    ► 32: {dt: 1704661200, main: {_, _}, weather: Array(1), clouds: {_, _}, wind: {_, _}}
    ► 33: {dt: 1704672000, main: {_, _}, weather: Array(1), clouds: {_, _}, wind: {_, _}}
    ► 34: {dt: 1704682800, main: {_, _}, weather: Array(1), clouds: {_, _}, wind: {_, _}}
    ► 35: {dt: 1704693600, main: {_, _}, weather: Array(1), clouds: {_, _}, wind: {_, _}}
    ► 36: {dt: 1704704400, main: {_, _}, weather: Array(1), clouds: {_, _}, wind: {_, _}}
    ► 37: {dt: 1704715200, main: {_, _}, weather: Array(1), clouds: {_, _}, wind: {_, _}}
    ► 38: {dt: 1704726000, main: {_, _}, weather: Array(1), clouds: {_, _}, wind: {_, _}}
    ► 39: {dt: 1704736800, main: {_, _}, weather: Array(1), clouds: {_, _}, wind: {_, _}}
    ► 40: {dt: 1704747600, main: {_, _}, weather: Array(1), clouds: {_, _}, wind: {_, _}}
```

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-c` na podstawie głównej gałęzi kodu.

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

<https://github.com/huuuuubi/AI1-LA-grN2-Hubkiewicz-Jakub/tree/lab-d/LD>

PODSUMOWANIE

W kilku zdaniach podsumuj zdobyte podczas tego laboratorium umiejętności.

Nauczyłem się podstawowej prezentacji danych uzyskiwanych z zewnętrznego systemu za pomocą API.

Zweryfikuj kompletność sprawozdania. Utwórz PDF i wyślij w terminie.