Studentų 50 – 213

Kompiuterių katedra

Informatikos inžinerijos studijų programa

Inžinerinis projektas

T120B172 Sistemų integracijos technologijos

Sistemų API technologijos

**ATLIKO:**

**Dominykas Pašukys**   **IFB-7**

(Vardas Pavardė) (Parašas) (Grupė)

**DĖSTYTOJAS:**

**Vaidas Jukavičius**

(Vardas Pavardė) (Parašas)

**DARBAS ATIDUOTAS:**

\_\_\_d.\_\_\_\_\_\_\_mėn. 2020

**KAUNAS 2020**

# Užduotis

Sugalvoti ir sukurti kokį nors taikymą, panaudojant bent tris esamus (jau sukurtus) REST API.

# 3 sprendimo architektūra/aprašymas

Šis sprendimas realizuotas panaudojant keturių jau esamų API gražinamus duomenis. Realizuota tik klientinė projekto dalis.

**Naudoti API:**

* Facebook Graph API
* Google Maps Javascript API
* Google Geocoding API
* OpenWeatherMap API.

**Naudoti įrankiai bei platformos:**

* **Facebook for developers** – sukurtų Facebook aplikacijų valdymo, API priežiūros, prieigos teisių, vartotojų valdymo platforma.
* **Facebook Login** įskiepis, skirtas prisijungimui prie šios aplikacijos su Facebook paskyros duomenimis iš nurodytos vietos (šiuo atveju localhost).
* **Graph API Explorer įrankis (Facebook for developers platformos dalis)**, skirta rašomų API URL užklausų build‘inimui, debug‘inimui, taip pat ir nurodytų prieigos prie prisijungusio asmens duomenų leidimų pridėjimui bei prieigos token‘o generavimui pridėtų leidimų pagrindu. Šis sugeneruotas token‘as paskui yra naudojamas visose sistemėlės API URL užklausose.
* **Google Cloud** platforma, šiame darbe naudota susikurti API raktą prieigai prie jų API gražinamų duomenų (Google Maps Javascript API ir Google Geocoding API). Šis raktas po to yra naudojamas visose šių API URL užklausose.
* **XAMPP -** lokalaus serverio paleidimo bei valdymo įrankis (localhost).
* **Notepad++ -** kodui rašyti.
* **FB PHP SDK** biblioteka su visais reikiamais metodais darbui su Graph API.

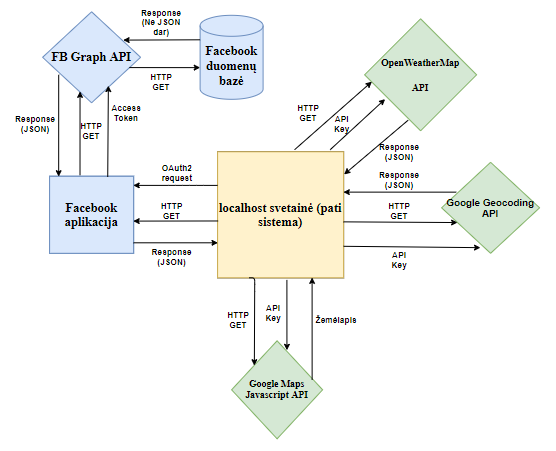
**Naudojamos programavimo kalbos:**

* **PHP -** visoms API užklausoms bei logikai.
* **Javascript –** Google žemėlapiui atvaizduoti bei žemėlapio markerio parametrams bei išvaizdai keisti.
* **HTML bei CSS -** akiai patraukliam gautų duomenų atvaizdavimui.

**API užklausų gražinamų duomenų formatas – JSON** (negalioja Google Maps Javascript API, nes tik buvo pasinaudota jo metodais, kad optimizuoti bei atvaizduoti žemėlapį kartu su vietos žymekliais).

Taip pat verta paminėti, kad visuose panaudotuose API yra reikalingi sugeneruoti API raktai arba prieigos token‘as (Graph API) gražinamų duomenų apsaugai. Iš viso du API Keys (OpenWeatherMap vienas, Google Maps Javascript API ir Google Geocoding API bendras raktas) taip pat vienas prieigos token‘as (Facebook Graph API), sugeneruotas pagal nurodytus prieigos prie Facebook profilio duomenų leidimus.

**Sukurtos sistemos architektūros schema (žr. 1 pav.):**



**1 pav. Sukurtos sistemos architektūros schema.**

**Facebook Graph API**

Šiame API, URL užklausa gali gražinti ieškomus duomenis keliais būdais:

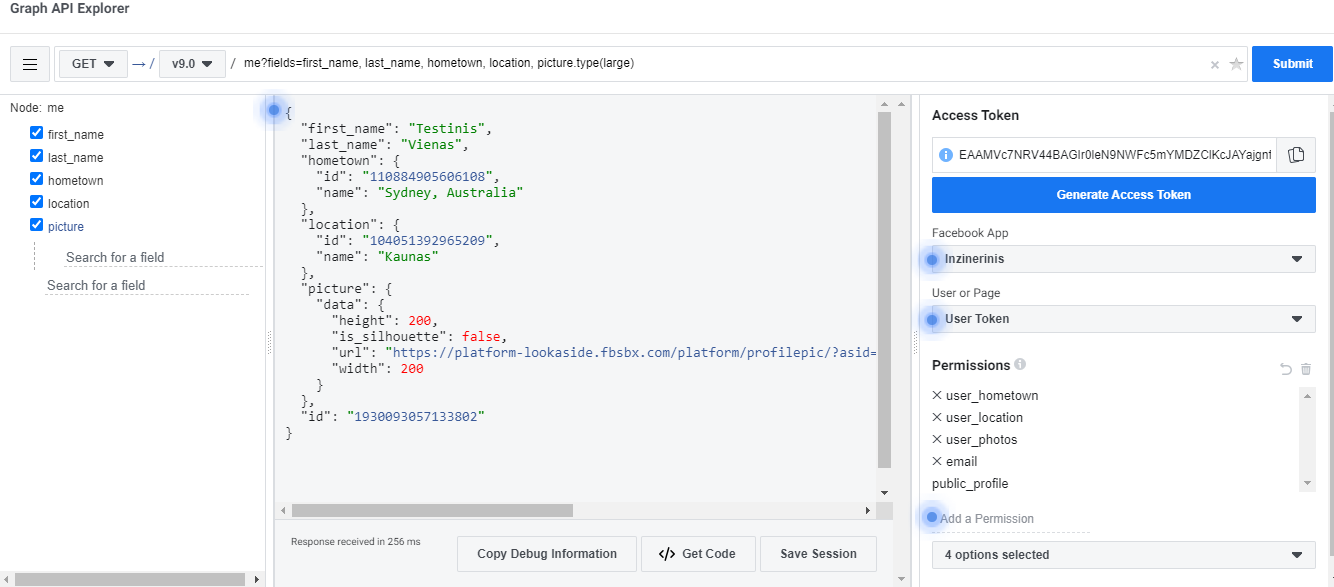
* Mazgais (nodes) – kas yra tiesiog individualūs objektai, tokie kaip vartotojas, nuotrauka, puslapis ar komentaras.
* Kraštais (edges) – kas yra tiesiog ryšiai tarp objektų kolekcijos ir vieno objekto, pavyzdžiui komentarai prie vienos specifinės nuotraukos, kelios nuotraukos tam tikrame puslapyje ir t.t.
* Laukais (fields) – tiesiog duomenys apie tą vieną objektą, pavyzdžiui, vartotojo vardas arba gimtadienis.

Šiame darbe buvo kreipiamasi į šį API, kad gražintų tik tam tikrus laukus apie dabar per localhost prisijungusį vartotoją (žr. 2 pav.):



**2 pav. API GET užklausa, naudojama kurtoje sistemoje.**

**Paaiškinimas:** Nurodomas mazgas ‚/me‘, kas reiškia, kad bus imami mano, kaip prisijungusio vartotojo paskyros duomenys, po klaustuko tiesiog nurodomi kokie laukai bus paimami, šiuo atveju vardas, pavardė, gimtasis miestas, dabartinis miestas bei profilio nuotrauka. Žinoma, dar reikia nurodyti prieigos token‘ą FB PHP SDK bibliotekos metode get(), kas šioje nuotraukoje (žr. 1 pav.) yra $accessToken PHP kintamasis. **Svarbu, kad token‘as būtų teisingas, t.y, sugeneruotas su visais leidimais prie prašomos informacijos.**



**3 pav. Iš mazgo „/me“ grąžinami prašyti laukai JSON formatu Graph API Explorer įrankio pagalba.**

**OpenWeatherMap API**

Šiame API galima gražinti orų duomenis tiek JSON formatu, tiek XML formatu (šiame darbe naudotas JSON formatas). Galima gražinti orų informaciją tiek pagal miestą, tiek pagal koordinates, tiek pagal šalį, tiek pagal pašto kodą. Galima nurodyti URL žingsnį, kas kiek laiko reikia grąžinti orų prognozes, galima gražinti tiek šios dienos, tiek bet kokio laikotarpio orų prognozes.

Šiame darbe naudota URL struktūra (žr. 4 pav.):

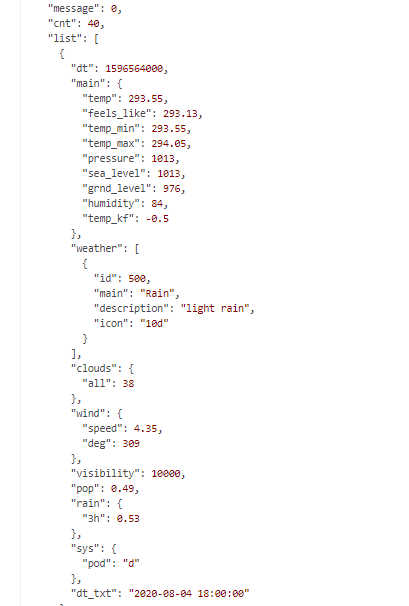


**4 pav. Darbe naudota OpenWeatherMap API URL stuktūra.**

**Paaiškinimas:** Vietoje „forecast“ galima naudoti „weather“ tuomet bus grąžinta tik orų informacija šiuo metu. „q“ dalyje čia nurodytas vartotojo pasirinktas miestas iš savo profilio. Vietoje „q“ galima naudoti „zip“ (pašto kodui), „lat=....long=“ – ilgumos ir platumos koordinatėms įvesti ir daug kitų kombinacijų.

„lang“ – norima kalba (šiuo atveju anglų), „units“ – norima matavimo vienetų sistema, pavyzdžiui vėjo greitis ar kilometrais per valandą ar myliom per valandą (galima rinktis tik imperial ir metric). „APPID“, tai openweathermap.org sugeneruotas privatus API raktas (reikia užsiregistruoti).

URL užklausos grąžintų pavyzdinių duomenų rinkinys JSON formatu (žr. 5 pav.):



**5 pav. URL užklausos grąžintų pavyzdinių duomenų rinkinys JSON formatu.**

**Google Geocoding API**

Šis API gražina bet kokio adreso koordinates bei kitą informaciją tiek JSON, tiek XML formatais (šiame darbe JSON).

Darbe naudojama URL struktūra (žr. 6 pav.) ir gražinamų pavyzdinių duomenų fragmentai (žr. 7-9 pav.):



**6 pav. Šiame darbe naudojama API užklausos URL.**

**Paaiškinimas:** Vietoje „json?“ galima nurodyti „xml“ ir duomenys bus gražinti ne JSON, o XML formatu. „address“ parametre nurodome, kokio norime tikslumo adresą. Tai gali būti nuo šalies iki pat buto numerio. „key“ parametre nurodome Google Cloud platformoje susigeneruotą asmeninį privatų API raktą.



**7 pav. URL gražinami pavyzdiniai duomenys JSON formatu (1-as fragmentas).**



**8 pav. URL gražinami pavyzdiniai duomenys JSON formatu (2-as fragmentas).**



**9 pav. URL gražinami pavyzdiniai duomenys JSON formatu (3-as fragmentas).**

Google Maps Javascript API buvo naudojamas tik žemėlapio atvaizdavimui, markerio savybių keitimui bei jo padėjimui į teisingą vietą pagal Geocoding API gražintas koordinates. Kodas yra žemiau esančiame skyrelyje.

**Apibendrintas sukurtos sistemėlės kūrimo planas (etapai):**

1. Užsiregistruojame developers.facebook.com ir susikuriame tuščią aplikaciją.
2. Pasileidžiame XAMPP localhost serverį.
3. Developers platformoje nurodome svetainės adresą (šiuo atveju <http://localhost>).
4. Įtraukiame Facebook login įskiepį į aplikaciją.
5. Parsisiunčiame FB PHP SDK, išarchyvuojame aplanką į public\_html aplanką.
6. Padarome prisijungimo puslapį savo svetainėje (localhost).
7. Su Graph API Explorer įrankiu nurodome kokios informacijos leidimų mums reikės ir šiuo pagrindu susigeneruojame prieigos token‘ą.
8. Paimame reikalingą informaciją (prisijungusio žmogaus nuotrauką, vardą, pavardę, gimtąjį miestą, dabartinį miestą) ir atvaizduojame puslapyje.
9. Padarome, kad prisijungęs žmogus galėtų pasirinkti, kokio miesto orų informaciją jis nori peržiūrėti (ar gimtojo, ar dabartinio).
10. Užsiregistruojame openweathermap.org ir susigeneruojame asmeninį API raktą.
11. Užsiregistruojame Google Cloud platformoje ir susikuriame tuščią projektą bei pridedam savo „Billing“ informaciją (kreditinės kortelės duomenis), nes kitaip Geocoding API gražins ne JSON duomenis, o klaidą ACCESS\_DENIED.
12. Google Cloud platformoje susikuriame nuosavą API raktą be jokių apribojimų (nustačius references dalyje localhost, kažkodėl neveikia).
13. Pasinaudodami atitinkamais API gauname šio miesto orų prognozes ir koordinates.
14. Apdorojame JSON formatu gautus duomenis ir susirašome į kintamuosius (patogumui).
15. Pasirašome CSS stilius gražiam atvaizdavimui.
16. Atvaizduojame viską (įskaitant ir žemėlapį su markeriu).
17. Padarome, kad keistųsi atvaizdavimo stilius ir žemėlapio markeris priklausomai nuo orų prognozės.

# 3 sprendimo programinis kodas

Šioje aplikacijoje, kaip ir buvo minėta, buvo panaudoti keturių jau esamų REST API metodai bei jų gražinami duomenys. Žemiau pateikiamas sistemėlės failų sąrašas su jų paskirties paaiškinimais (failai yra įtraukti į šį dokumentą kaip .txt failai). Šie žemiau pateikiami failai yra visas sukurtos sistemėlės programinis kodas.

PASTABA: Norint, kad sistema veiktų ant bet kokio domeno, reikia atsisiųsti FB PHP SDK biblioteką ir ją patalpinti į public\_html direktoriją kartu su visais sistemos failais, taip pat nurodyti teisingą kelią iki šios bibliotekos fb\_api.php faile.

Norint matyti failuose esantį kodą, reikia du kartus pele spustelėti ant kiekvieno failo šiame sąraše.

**Sistemos failų sąrašas:**

*  - šiame faile, naudojantis OAuth2 metodu kreipiamasi į sukurtą Facebook aplikaciją, kad ši suteiktų prieigos token‘ą, kuriame užšifruoti visi tam vartotojui, kuris jungiasi, aplikacijos prašomi suteikti leidimai. Šiuos leidimus vartotojas sutinka (arba nesutinka) duoti. Kai vartotojas patvirtina, kad suteikia leidimą prie prašyme nurodytos informacijos tuomet pateikiamas URL kreipinys Graph API iš kurio gaunami prašomi duomenys JSON formatu (jei galioja duotas prieigos token‘as ir jame tikrai yra nurodyti tokie leidimai prie prašomos informacijos). JSON duomenys apdorojami panaudojant tokius FB PHP SDK metodus kaip GetGraphObject() ir GetGraphProperty() ir po to, šie apdoroti duomenys surašomi į atitinkamus PHP sesijos kintamuosius tolimesniam jų naudojimui prisijungimo sesijoje.
* Šiame faile yra kreipiamasi į FB PHP SDK biblioteką ir ją panaudojant sukuriamas $fb objektas, kuriame yra sukurtos Facebook aplikacijos ID (prie kurios reikia jungtis), App secret (saugumui užtikrinti) bei graph\_api\_version (tiesiog nurodoma dabartinė versija). Taip pat sukuriamas $handler, kuris skirtas gauti prieigos token‘ą iš aplikacijos.
* - Šiame faile yra aprašyta PHP funkcija Geocode($address), kurią galima panaudoti bet kuriame kitame faile, įvedus norimą miestą gaunamas masyvas $data\_arr, kuriame gražintos yra šio miesto koordinatės. Veikimo principas paprastas: suformuojamas URL su nurodytu miestu. Tada gaunamas atsakymas iš Geocode API JSON formatu. Apdorojamas atsakymas iš JSON formato ir surašoma visa reikalinga informacija į atskirus kintmuosius (šiuo atveju reikalinga informacija yra miesto ilgumos bei platumos koordinatės). Po to šie kintamieji yra surašomi į $data\_arr masyvą, kuris yra gražinamas panaudojus šią funkciją.
* - Pats pirmas svetainės puslapis, kuriame yra prisijungimo prie sukurtos Facebook aplikacijos (prieigos token‘ui gauti, kurio pagrindu sukuriamas URL, kuriuo nukreipiama į localhost/apdorojimas.php puslapį) nuorodos mygtukas.
* - CSS (Cascading Style Sheets) failas. Kuriame yra aprašytas stilius orų informacijai atvaizduoti.
* - Šiame faile atvaizduojami iš Graph API gauti prisijungusio vartotojo duomenys (Vardas, pavardė, gimtasis miestas, dabartinis miestas ir profilio nuotrauka). Taip pat yra du pasirinkimų sąrašai. Viename iš jų yra vartotojo profilyje aptikti gimtasis bei dabartinis miestai, kitame – galima pasirinkti kelių dienų prognozę norima peržiūrėti. Pasirinkus vieną iš miestų bei pasirinkus, kelių dienų prognozę norima peržiūrėti bus pateiktas žemėlapis su atitinkamu dabartinio oro paveiksliuku bei dabartine oro temperatūra vietoje žemėlapio vietos markerio, dabartinių orų informacija bei prognozių informacija.
* - Šiame faile iš pradžių suformuojamas URL OpenWeatherMap API vartotojo nurodyto miesto orų duomenims gauti (apsaugai naudojamas privatus man suteiktas OpenWeatherMap API raktas). Tuomet pasinaudojant Geocode($address) funkcija gaunamas duomenų masyvas, kuriame yra vartotojo nurodyto miesto ilgumos bei platumos koordinatės (apsaugai naudojamas Google Cloud platformos suteiktas privatus raktas tinkantis visiems jų API, kadangi neuždėjau jokių aprobojimų). Gaunami orų duomenys iš OpenWeatherMap API JSON formatu. Atvaizduojamas vietovės žemėlapis su markeriu (pasinaudojant Google Maps API javascript). Atvaizduojama dabartinių vartotojo nurodytos vietovės orų informacija pagal sukurtą stiliaus šabloną (kartu ir pasirinkto dienų kiekio prognozės). Realizuotas minimali logika: Jeigu dienos maksimali oro temperatūra yra didesnė negu penki laipsniai Celsijaus, tai pasikeičia atvaizdavimo stilius (iš mėlyno rėmelio ir fono pasidaro oranžinis ir atvirkščiai). Taip pat padaryta, kad jeigu dabartinė maksimali oro temperatūra yra didesnė negu 5 laipsniai nurodytoje vietovėje, tai žemėlapio rėmelis nusidažo oranžine spalva, jei mažesnė – mėlyna. Taip pat skiriasi žemėlapio vietovės markeris, kuris susideda iš dabartinių orų paveiksliuko (pvz. saulė, debesis ir t.t.) ir dabartinės maksimalios oro temperatūros šalia jo.

# Rezultatų apibendrinimas

Apibendrinant, šiame projekte buvo realizuota nedidelė orų programėlė, panaudojant keturis jau esamus REST API (Facebook Graph API, Google Maps Javascript API, Google Geocoding API bei OpenWeatherMap API). Darbo realizavimo metu buvo daug sužinota apie API integraciją kuriamai svetainei, metodus, saugumo užtikrinimą, todėl galima teigti, kad darbo tikslas pasiektas.

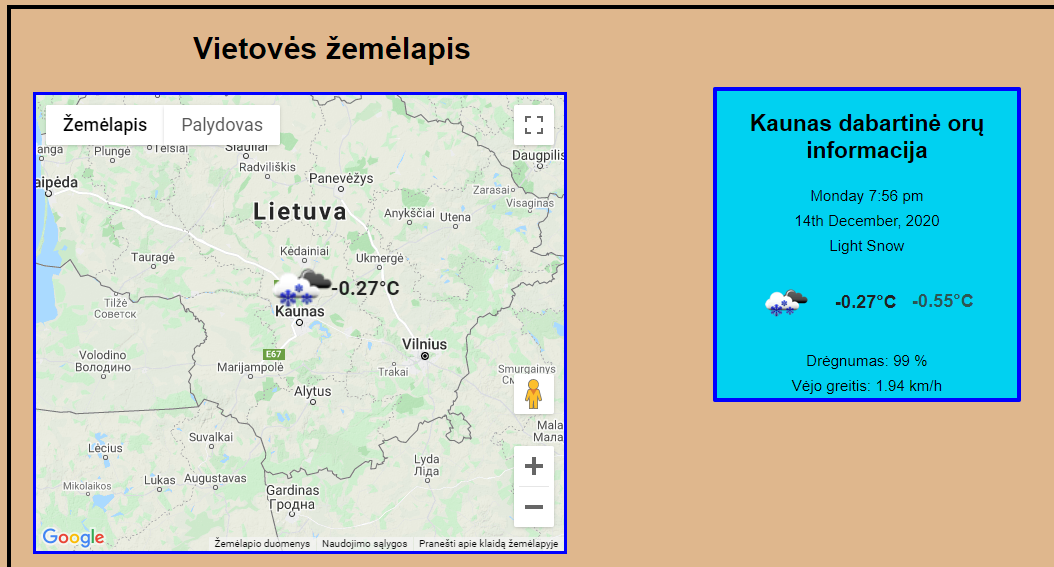
**Keletas nuotraukų iš sistemos (žr. 10-15 pav.):**



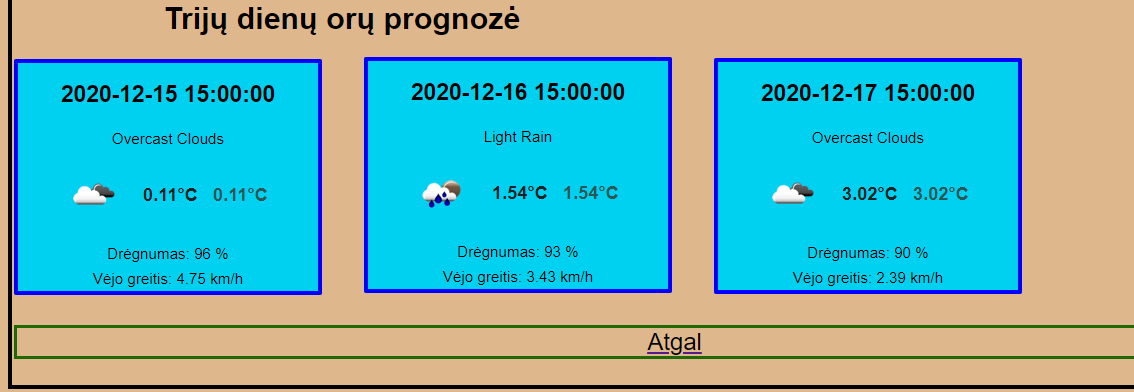
**10 pav. Index.php puslapis.**



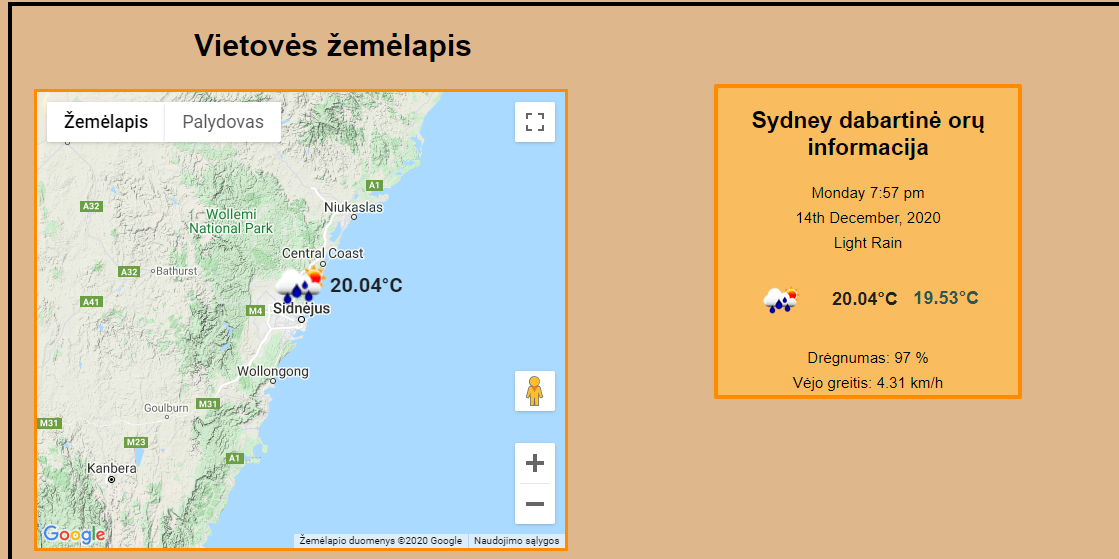
**11 pav. Projektas.php puslapis.**



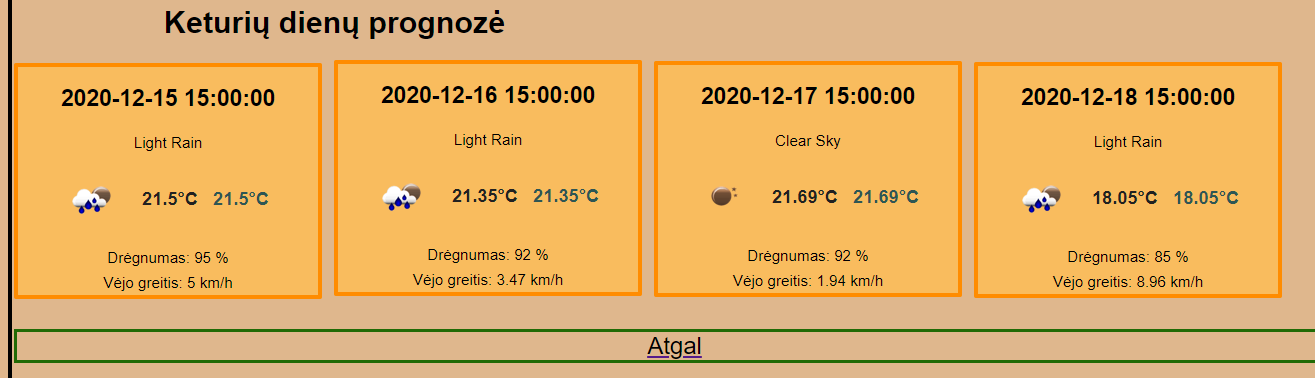
**12 pav. Projektas-orai.php puslapis su Kauno orų informacija, kai pasirinkta trijų dienų prognozė (1-as fragmentas).**



**13 pav. Projektas-orai.php puslapis su Kauno orų informacija, kai pasirinkta trijų dienų prognozė (2-as fragmentas).**



**14 pav. Projektas-orai.php puslapis su Sidnėjaus orų informacija, kai pasirinkta keturių dienų prognozė (1-as fragmentas).**



**15 pav. Projektas-orai.php puslapis su Sidnėjaus orų informacija, kai pasirinkta keturių dienų prognozė (2-as fragmentas).**