FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Vizualizacija podataka

Meteorološki podatci RH

Projektni zadatak

Student: Marin Stević, DRD

Sadržaj

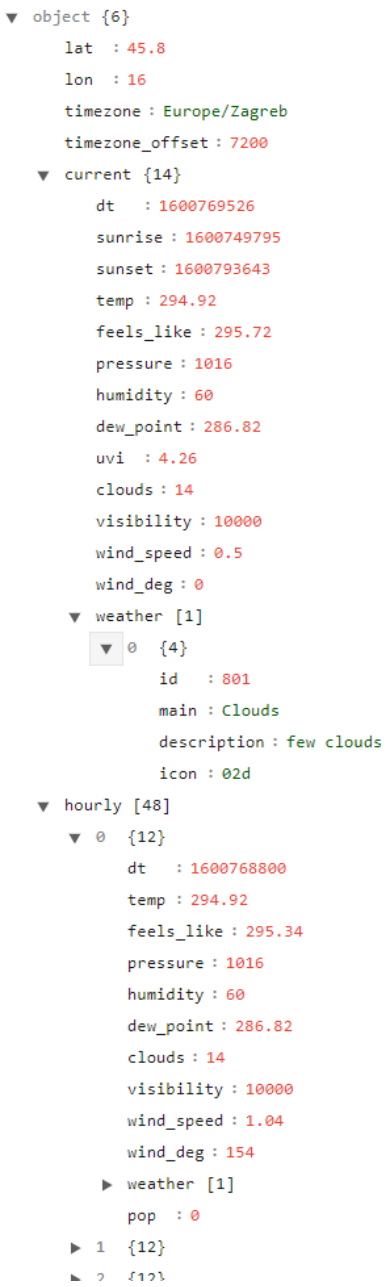
1. Opis projekta

Ideja projekta je prikaz meteoroloških podataka na području Republike Hrvatske. Podatci koji se prikazuju su: temperatura, brzina i smjer vjetra, oblik vremena i UV indeks. Zbog toga što se koristi besplatni API podatci koji se prikupljaju su trenutni osim za temperaturu za koju se prikupljaju podatci za proteklih 48 sati u intervalu od sat vremena.

Vizualizacija je ostvarena korištenjem karte i linijskog grafa koji prikazuje temperaturu za protekla 2 dana. Postoje 2 načina korištenja aplikacije: prvi način podrazumijeva odabir lokacije na karti klikom miša i time se otvara prozor s trenutnim meteorološkim podatcima za traženo mjesto. Podatci se uzimaju s najbliže meteorološke postaje od odabranih koordinata. Drugi način korištenja je da se iz izbornika s lijeve strane odabere grad i pritiskom na ciljnik pored imena grada karta se automatski pomiče do traženoga grada i prikazuju se trenutni meteorološki podatci. U oba slučaja ispod karte se pojavljuje graf koji prikazuje temperaturu za protekla 2 dana. Graf se promjenom odabranog mjesta dinamički prilagođava podatcima.

2. Prikupljanje podataka

Podatci se prikupljaju pomoću OpenWeatherMap API-a[1] koji ima besplatni pristup koji podržava 60 API poziva u minuti, što je dovoljno za ovakav projekt. Podatci pristižu u JSON formatu i na slici 2.1. se može vidjeti struktura jednog odgovara API poziva.



Slika 2.1. Primjer JSON odgovora dobivenog pomoću OpenWeatherMap API-a

[1] https://openweathermap.org/api

Ispod je prikazan primjer jednog API poziva koji dohvaća meteorološke podatke za proteklih 48 sati s bazne stanice u Osijeku:

- http://api.openweathermap.org/data/2.5/onecall?units=metric&lat=45.160278&lon=18.015556&exclude=minutely,daily,alerts&APPID=6e56fb3433479d82ffab33418b64eb3b

Iz ovog poziva može se primijetiti da su predane koordinate mjesta, a ne ime grada što se isto može. To je napravljeno s razlogom da se izbjegne korištenje nepodržanih znakova. Na ovaj način API automatski povezuje koordinate s najbližom baznom postajom i šalje nazad rezultat.

Još jedan bitan skup podataka korišten u ovom projektu je popis većih gradova s njihovim koordinatama, koji je preuzet s ove stranice: https://simplemaps.com/data/hr-cities. Pomoću ovoga skupa moguće je bilo prikupiti meteorološke podatke svih gradova pomoću njihovih koordinata i OpenWeatherMap API-a.

3. Programsko rješenje

Projekt se sastoji od više datoteka, ali glavne su: index.html, code.js i hr.json. Index.html sadrži osnovni izgleda stranice i podjelu stranice na osnovne elemente: lijevi izbornik, kartu i graf. Unutar code.js datoteke nalazi se sva logika oko prikaza karte i grafa, te učitavanja gradova iz hr.json datoteke.

Vizualizacija karte je realizirana uz pomoć leaflet.js biblioteke koja upravo i služi za rad nad kartama. Karta je smještena u div element s id-em „map“. Mapa se učitava pokretanjem sljedećega koda:

```
//Creating the map
var map = L.map( 'map', {
    center: [44.404582, 16.573679],  //Zagreb: 45.8150269, 15.9818139
    minZoom: 2,
    zoom: 7
});

L.tileLayer( 'http://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
    attribution: '&copy; <a href="https://www.openstreetmap.org/copyright">OpenSt
reetMap</a>',
    subdomains: ['a','b','c']
}).addTo( map );
```

Primjer 3.1. učitavanje karte

Iz primjera 3.1. se vidi da se karta učitava pomoću „tile-ova“ koji predstavljaju manje dijelove karte dobivenih pomoću OpenStreetMap API-a, također se može koristiti i google maps API.

Kada se karta učita možemo početi s unosom podataka u lijevi izbornik koji sadrži pretraživu listu gradova u Hrvatskoj. Izbornik sadrži tablicu u koju se mogu dinamički dodavati elementi odnosno redci i to se ostvaruje pomoću sljedeće funkcije:

```
//Add cities to the map sidebar table
function sidebarTableAddCity(city,table){
    var sidebarTable  = document.getElementById(String(table));

    var sidebarTableRow = document.createElement('tr');
    sidebarTableRow.className = String(table) + "-row";

    var sidebarTableCell_name = sidebarTableRow.insertCell();
```

```
    var name = document.createElement('p');
    name.innerHTML = city.data.name;
    name.className = "sidebar-table-cell";
    sidebarTableCell_name.appendChild(name);

    var sidebarTableCell_location = sidebarTableRow.insertCell();
    sidebarTableCell_location.innerHTML = '<i class="fas fa-crosshairs fa-
2x" onclick="locateAndShowCity(' + city.data.lat + ',' + city.data.lon + ',\'' +
city.data.name + '\')"></i>';

    //Sort the cities while adding them to the map sidebar table
    var rows = sidebarTable.getElementsByTagName("TR"), rowCell, i;
    if (rows.length == 1) {
        sidebarTable.appendChild(sidebarTableRow);
    }
    else {
        for (i = 1; i < (rows.length); i++) {
            rowCell = rows[i].getElementsByTagName("TD")[0];
            if (sidebarTableCell_name.firstChild.innerHTML.toLowerCase() < rowCel
l.firstChild.innerHTML.toLowerCase()){
                sidebarTable.insertBefore(sidebarTableRow, rows[i]);
                return;
            }
        }
        sidebarTable.appendChild(sidebarTableRow);
    }
}
```

Primjer 3.2. dodavanje gradova u izbornik

Iz primjera 3.2. funkcija prima objekt oblika grad i id tablice u koju je potrebno dodati navedeni grad. Objekt „city“ sadrži naziv i koordinate grada. Imena gradova i njihove koordinate se dobivaju pomoću XMLHttpRequest za učitavanje hr.json datoteke i njenim parsiranjem se podatci šalju funkciji iz prethodnoga primjera koja ih dodaje u izbornik.

```
//Load cities
var xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
    var myObj = JSON.parse(this.responseText);
    for (var i = 0; i < myObj.length; i++){
        var city = {
            'data': {
                'name': myObj[i].city,  //city name
                'lat': myObj[i].lat,    //city lat
```

```
                'lon': myObj[i].lng,    //city lon
            },
        }
        sidebarTableAddCity(city,"city-list-table");
    }
  }
};
xmlhttp.open("GET", "../hr.json", true);
xmlhttp.send();
```

Primjer 3.3. učitavanje datoteke hr.json

Za pretraživanje izbornika koristi se sljedeća jednostavna funkcija koja nakon svakog unosa u polje za pretraživanje osvježava tablicu i prikazuje samo gradove koji sadrže uneseni niz znakova.

```
//Search cities in the map sidebar table
function sidebarSearch(input,table) {
    var input, filter, table, tr, td, i;
    input = document.getElementById(String(input));
    filter = input.value.toUpperCase();
    table = document.getElementById(String(table));
    tr = table.getElementsByTagName("tr");
    for (i = 0; i < tr.length; i++) {
        td = tr[i].getElementsByTagName("td")[0];
        if (td) {
            if (td.innerHTML.toUpperCase().indexOf(filter) > -1) {
                tr[i].style.display = "";
            }
            else {
                tr[i].style.display = "none";
            }
        }
    }
}
```

Primjer 3.3. pretraživanje liste gradova

Odabirom pojedinog grada ili klikom na bilo koje mjesto na karti prikupljamo podatke s API-a za to mjesto koristeći AJAX. Sljedeća funkcija prikazuje način dohvaćanja podataka s OpenWeatherMap API-a pomoću predanih koordinata. Ova funkcija se pokreće odabirom grada iz lijevog izbornika i karta se automatski približava na taj grad i prikazuje se skoči prozor sa svim meteorološkim podatcima za taj grad.

```
//Showing the selected city
function locateAndShowCity(lat, lon, name) {
```

```
    $.ajax({
        url: 'https://api.openweathermap.org/data/2.5/onecall?units=metric&lat='+
lat+'&lon='+lon+'&exclude=minutely,daily,alerts&appid=6e56fb3433479d82ffab33418b6
4eb3b',
        async: false,
        dataType: 'json',
        success: function (response) {
            //console.log(response);
            var station = response.current;

            var txt = '<div class="owm-popup-name">';
            txt += name;
            txt += '</div>';
            if (typeof station.weather != 'undefined' && typeof station.weather[0
] != 'undefined') {
                if (typeof station.weather[0].description != 'undefined') {
                    txt += '<div class="owm-popup-description">'
                        + station.weather[0].description
                        + '</div>';
                }
            }
            var imgData = getImageData(station);
            txt += '<div class="owm-popup-
main"><img src="' + imgData.url + '" width="' + imgData.width
                    + '" height="' + imgData.height + '" border="0" />';
            if (typeof station != 'undefined' && typeof station.temp != 'undefine
d') {
                txt += '<span class="owm-popup-temp">' + station.temp
                    + ' °C</span>';
            }
            txt += '</div>';
            txt += '<div class="owm-popup-details">';
            if (typeof station != 'undefined') {
                if (typeof station.humidity != 'undefined') {
                    txt += '<div class="owm-popup-detail">'
                        + 'Humidity'
                        + ': ' + station.humidity + ' %</div>';
                }
                if (typeof station.pressure != 'undefined') {
                    txt += '<div class="owm-popup-detail">'
                        + 'Pressure'
                        + ': ' + station.pressure + ' hPa</div>';
                }
                if (true) {
                    if (typeof station.temp_max != 'undefined' && typeof station.
temp_min != 'undefined') {
                        txt += '<div class="owm-popup-detail">'
                            + 'Temp. min/max'
                            + ': '
                                + station.temp_min
                            + ' / '
                            + station.temp_max
```

```
                            + ' °C</div>';
                    }
                }
            }
            if (station.rain != null && typeof station.rain != 'undefined' && typ
eof station.rain['1h'] != 'undefined') {
                txt += '<div class="owm-popup-detail">'
                    + 'Rain (1h)'
                    + ': ' + station.rain['1h'] + ' ml</div>';
            }
            if (typeof station.wind_speed != 'undefined' && typeof station.wind_d
eg != 'undefined') {
                if (typeof station.wind_speed != 'undefined') {
                    txt += '<div class="owm-popup-detail">';
                    txt += 'Wind' + ': '
                        + station.wind_speed + ' '
                        + 'm/s';
                    txt += '</div>';
                }
                if (typeof station.wind_deg != 'undefined') {
                    txt += '<div class="owm-popup-detail">';
                    txt += 'Direction' + ': ';
                    txt += station.wind_deg + '°';
                    txt += '</div>';
                }
            }
            if (typeof station.dt != 'undefined' && true) {
                txt += '<div class="owm-popup-timestamp">';
                txt += '(' + convertTimestamp(station.dt) + ')';
                txt += '</div>';
            }
            txt += '</div>';
            var popup = L.popup()
                .setLatLng([lat, lon])
                .setContent(txt)
                .openOn(map);
            map.setView([lat, lon],13);

            var data = [];

            for (var i = 0; i < response.hourly.length; i++){
                var reading = {
                    "temperature": response.hourly[i].temp,
                    "timestamp": convertTimestamp(response.hourly[i].dt)
                }
                data.push(reading);
            }
            draw(data);
        }
    });
}
```

Primjer 3.4. dohvaćanje podataka s API-a i stvaranje skočnog prozora na karti

Identična funkcija se koristi za stvaranje skočnog prozora na klik miša jedina razlika je što se njoj predaje event ostatak funkcije je jedna funkciji iz primjera 3.4.

Slika 3.1. Primjer skočnog prozora na karti

Na kraju kada smo dohvatili podatke i prikazali skočni prozor na karti s osnovnim meteorološkim podatcima pokrećemo prikaz grafa. Graf je prvobitno skriven do prvoga dohvaćanja podataka. Prvo postavljamo osnovne vrijednosti grafa kao što su: imena osi, margine, vrsta podataka, te visina i širina grafa.

```
const xValue = d => d.timestamp;
const xLabel = 'Past 2 days';
const yValue = d => d.temperature;
const yLabel = 'Temperature';
const margin = { left: 120, right: 30, top: 20, bottom: 120 };

var elmnt = document.getElementById("graph");
const svg = d3.select('svg');
const width = elmnt.offsetWidth;
const height = elmnt.offsetHeight;
const innerWidth = width - margin.left - margin.right;
const innerHeight = height - margin.top - margin.bottom;
```

Primjer 3.5. postavljanje osnovnih vrijednosti grafa

Nakon toga definiran je <g> element koji je dodan glavnom <svg> elementu koji je definiran u index.html datoteci. Također su <g> elementu dodane x i y os i njihovi atributi.

```
const g = svg.append('g')
    .attr('transform', `translate(${margin.left},${margin.top})`);
const xAxisG = g.append('g')
    .attr('transform', `translate(0, ${innerHeight})`);
const yAxisG = g.append('g');

const xScale = d3.scaleTime();
const yScale = d3.scaleLinear();

const xAxis = d3.axisBottom()
    .scale(xScale)
    .tickPadding(15)
    .ticks(10)
    .tickSize(-innerHeight);

const yTicks = 5;
const yAxis = d3.axisLeft()
    .scale(yScale)
    .ticks(yTicks)
    .tickPadding(15)
    .tickSize(-innerWidth);

const line = d3.line()
    .x(d => xScale(xValue(d)))
    .y(d => yScale(yValue(d)))
    .curve(d3.curveBasis);

const row = d => {
    d.timestamp = new Date(d.timestamp);
    d.temperature = +d.temperature;
    return d;
};
```

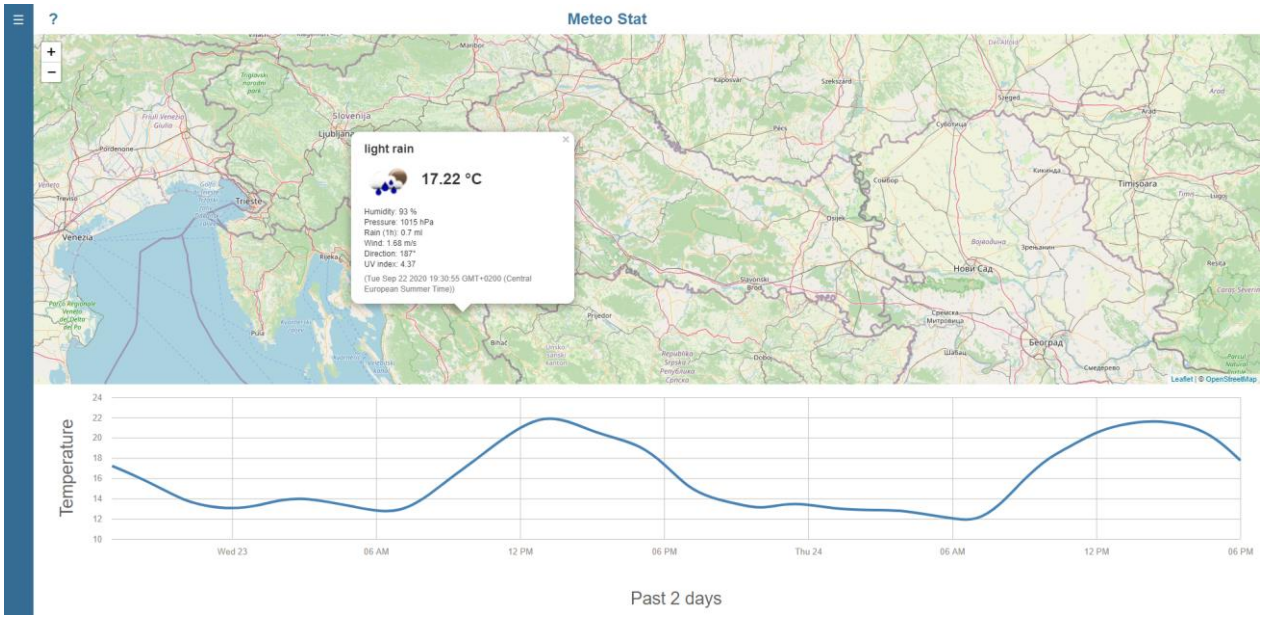
Primjer 3.6. dodavanje osnovnih elemenata grafa i njihovih atributa

Jedina funkcija koja je preostala je funkcija za crtanje grafa koja prima podatke u obliku niza objekata koji sadrže vrijeme i izmjerenu temperaturu u tom vremenu. Ova funkcija se poziva nakon svakog API poziva i iznova crta graf tako da se stari podatci preoblikuju u nove i tako graf izgleda kao da se prilagođava novim podatcima.

```
function draw(data) {
    initGraph();

    const t = d3.transition().duration(1000)

    xScale
```

```
    .domain(d3.extent(data, xValue))
    .range([0, innerWidth]);

    yScale
    .domain(d3.extent(data, yValue))
    .range([innerHeight, 0])
    .nice(yTicks);

    g.append('path')
        .attr('fill', 'none')
        .attr('class', 'line')
        .attr("stroke", 'steelblue')
        .attr('stroke-width', 4);

    const chart = g.select('.line')
        .datum(data);

    chart.exit().remove()

    const enter = chart.enter();

    enter.merge(chart)
        .transition(t)
        .attr("d", line(data));

    xAxisG.call(xAxis);
    yAxisG.call(yAxis);
}
```

Primjer 3.7. funckija za crtanje grafa



Slika 3.2. izgled aplikacije

4. Literatura

- https://openweathermap.org/api - openweather API dokumentacija
- https://simplemaps.com/data/hr-cities - skup podataka o gradovima u RH
- https://leafletjs.com/examples/quick-start/
- https://d3js.org/
- https://stackoverflow.com/
- https://github.com/MarinStevic/MeteoStat - Projektni repozitorij