Univerzitet u Beogradu

Fakultet organizacionih nauka

Katedra za elektonsko poslovanje

Primena IoT u detekciji vremenskih nepogoda (projekat)

Seminarski rad iz Elektronskog poslovanja

Nastavnik: dr Dušan Barać Studenti: Ognjen Gvozdenović 314/17 Saradnik: Luka Đurđević Bojana Dimitrić 421/17

Beograd, 2021.

Contents

[Uvod 1](#_Toc65098220)

[Pregled literature 2](#_Toc65098221)

[Prednosti i mane IoT-a 2](#_Toc65098222)

[Potrošačke i poslovne IoT primene 3](#_Toc65098223)

[Primene IoT-a u nekim kompanijama 4](#_Toc65098224)

[Bosch:Inovator za praćenje 5](#_Toc65098225)

[John Deere: Samovozeći traktori 5](#_Toc65098226)

[Maersk: Pametna logistika 5](#_Toc65098227)

[IoT u detekciji vremenske prognoze 6](#_Toc65098228)

[API (application programming interface) 6](#_Toc65098229)

[JSON 8](#_Toc65098230)

[Openweathermap 8](#_Toc65098231)

[Bootstrap 9](#_Toc65098232)

[Alternativni API-ji 10](#_Toc65098233)

[The Dark Sky API 11](#_Toc65098234)

[Simple Weather API 11](#_Toc65098235)

[AIR Quality API 11](#_Toc65098236)

[AccuWeather API 12](#_Toc65098237)

[Razvoj aplikacije 12](#_Toc65098238)

[Kreiranje aplikacije 13](#_Toc65098239)

[Javascript i API pozivi 14](#_Toc65098240)

[Zaključak 23](#_Toc65098241)

[Korišćena i referentna literatura 24](#_Toc65098242)

# Uvod

Sve više stvari oko nas su povezane na internet, od očekivanog kompjutera do neočekivanih: automobila, šporeta, frižidera, satova, pejsmejkera, uređaja za merenje šećera u krvi i mnogih drugih stvari. Sve te stvari čine internet inteligentnih uređaja. Ova tehnologija je toliko napredovala u prethodnih nekiliko decenija i postala veoma rasprostranjena da ne postoji budućnost bez njenog svakodnevnog mešanja u naš život, olakšavajući ga i čineći ge lakšim, zabavnijim i jedostavnijim.

Svakodnevni dosadni poslovi mogu postati uživanje za ljude, čineći neke zamorne aktivnosti veoma prostim. Dodavanjem senzora raznim alatima i uređajima ih pretvara u pametne uređaje, još ako se povežu na internet onda postaju i moćni uređaji. Internet inteligentnih uređajaje budućnost svačijeg života.

Osim pojedinačnih ljud, internet inteligentnih uređaja pravi preduzećima velike uštede i neverovatnu kompetetivnu prednost transformišući kompanije u pionire budućnosti. Nekoliko načina na koji su određene kompanije stekle takav status je opisano u daljem radu.

U ovom radu biće objašnjene neke od najbnitnijih tehnologija potrebnih za kreiranje kompetetivnih veb aplikacija na osnovu eksternih API-ja, njihov način primene i korišćenje kao i tip podatka sa kojim komuniciraju.

Podaci o vremenskoj prognozi su jedni od najbitnijih podataka u svetu, mnogo kompanija i ogroman broj ljudi baziraju svoje aktivnosti na dobrim vremenskim uslovima. Tačne i besplatne informacije o vremenskoj prognozu moraju biti svima dostupne, bilo gde u svetu. Zato smo se i opredelili za ovu temu, svaki čovek mora da zna da li se bliži oluja, nevreme ili je pak napolju pretoplo za izlaženje. Pored korisničke, ova aplikacija i ovaj rad pružaju uvid drugim programerima u svet javnih API-ja, način na koji oni funkcionišu, kako se koriste i moć koju poseduju.

# Pregled literature

Internet inteligentnih uređaja *(eng. Internet of Things – IoT)* predstavljaju milijarde fizičkih uređaja koji su povezani na internet, ti uređaji zajedno skupalju i dele podatke preko interneta. Zahvaljujući veoma jeftiniim kompjuterskim čipovima današnjice i raspostranjenošću bežičnog interneta, moguće je pretvoriti bilo šta, od obične pilule do čitavog aviona u deo IoT mreže. Povezivajući sve ove uređaje i dodavanjem senzora svakom od njih, pruža ovim uređajima takav nivo digitalne inteligencije što ih pretvara u veoma pametne uređaje, omogućavajući im da razmenjuju podatke u realnom vremenu bez mešanja čoveka. Internet inteligentnih uređaja čini svet oko nas mnogo pametnijim i mnogo responzivnijim, povezivajući digitalni i fizički univerzum. [1]

IoT pomaže ljudima da rade i žive pametnije omogućavajući im da imaju potpunu kontrolu nad svojim životom. Pored toga što daje pametne uređaje za automatizaciju domova, IoT je od suštinskog značaja za poslovanje. IoT pruža preduzećima uvid u realnom vremenu o radu njihovih sistema, od performansi mašina do lanca snabdevanja o logističkih operacija [2]

IoT omogućava kompanijama da automatizuju procese i smanje troškove rada. Takođe pomaže u smanjenju otapada i poboljšava pružanje usluga čime se smanjuju troškovi proizvodnje i transporta.[2]

Kao takva, IoT je jedna od najvažnijih tehnologija svakоdnevnog života i nastaviće da raste što više kompanija shvati potencijal i kompetetivnu prednost povezanih uređaja. [2]

## Prednosti i mane IoT-a

Neke od prednosti korišćenja IoT-a: [2]

* Mogućnost pristupa informacijama bilo gde, bilo kada i sa bilo kog uređaja
* Poboljšana komunikacija između konekotovanih uređaja
* Transferovanje podatak u povezanoj mreži štedi i vreme i novac
* Automatizovanje zadataka pomaže u poboljšanju kvaliteta usluga

Mane koriščenja IoT-a: [2]

* Što se više povećava broj povezanih uređaja i količina informacija koja se između njih deli, tako raste i šansa da haker ukrade potencijalno poverljive informacije
* Preduzeća će u međuvremenu morati da se suoče sa ogromnim brojem IoT uređaja, šta više, možda čak i milionima, što će im znatno otežati sakupljanje i upravljanje tolikom količinom podataka
* Ako postoji greška u sistemu, velika je šansa da će svi uređaji biti kompromizovani
* S obzirom da ne postoji internacionalni standard za kompatibilnost IoT uređaja, veoma je teško uskladiti uređaje od različitih proizvođača

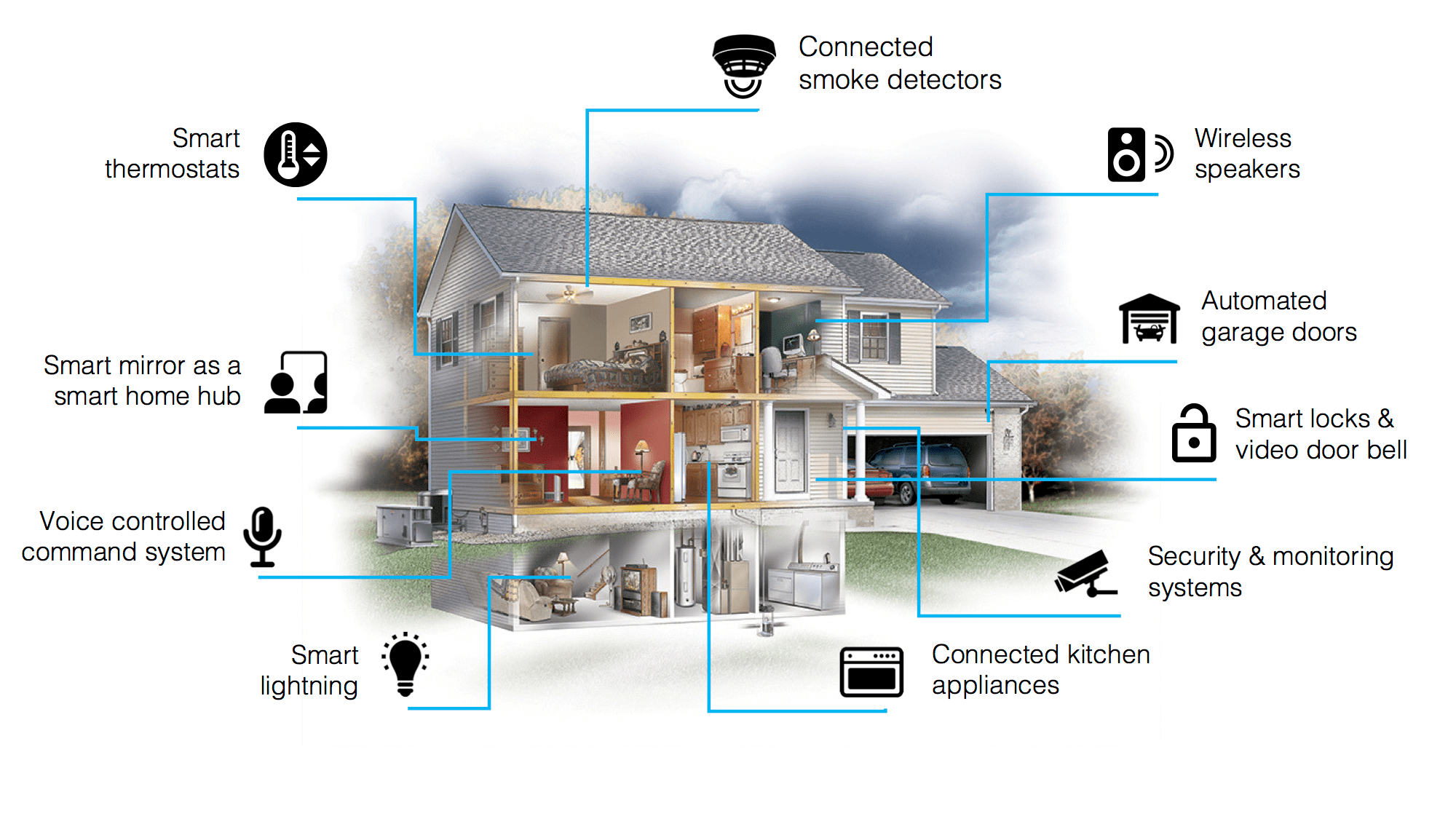
## Potrošačke i poslovne IoT primene

Postoji veoma mnogo načina na koji se IoT koristi, počevši od potrošačkog i poslovnog pa sve do industrijskog.

U potrošačkom segmentu postoje, na primer, pametne kuće koje su opremljene pametnim termostatima, pametnom belom tehikom, grejanjem,strujom i još raznoraznim pametnim uređajima koji mogu da se kontrolišu u bilo kom trenutku ili mestu pomoću pametnih telefona ili kompjutera. [2]

Nosivi uređaji *(eng. wearable devices)* sa velikim brojem senzora koji mogu da sakupljaju i analiziraju podatke korisnika i zatim da šalju poruke ostalim uređajima i tehnologijama o samom korisniku, trudeći se da olakšaju njegov život. To su na primer: Fit narukvice, uređaji za praćenje rada srca, pametni satovi ili čak uređaj za praćenje nivoa šećera u krvi. Na razne načine ovi uređaji pomaži i olakšavaju živote ljudi. [2]

Pametni dom *(eng. smart home)* je takvo okruženje u kome je ambijent inteligentan i automatizovan što omogućava ukućanima da komuniciraju sa njim i prilagode ga svojim potrebama. [3]



Slika 1 – Pametni dom

Standardan pristup kreiranju pametnih domova je njihovo kompjuterizovanje i dodavanje senzora raznim uređajima koji bi prikupljali različite podatke. Kompjuteri ili mikrkokontroleri analiziraju ove podatke radi identifikovanja radnji samih ukućana ili određenih događaja. Oni zatim odgovaraju na te radnje to jest događaje kontrolisanjem određenih mehanizama koji su ugrađeni u kuću. Najjednostavniji primer je uključivanje svetla kada ukućanin uđe u sobu. [3]

Ovo su neke od najrasprostranjenijih i najpoznatihijh primena IoT uređaja u potrošačke svrhe.

## Primene IoT-a u nekim kompanijama

IoT Ima mnoštvo primena u proizvodnim pogonima i ostalim delovima poslovanja raznih kompanija. Može da olakša proces proizvodnje jer IoT uređaji automatski prate proizvodne cikluse i upravljaju skladištima i zalihama. To je jedan od razloga zbog čega je ulaganje u IoT uređaje neverovatno skočilo u poslednjih nekoliko decenija. IoT u proizvodnji, logistici i transportu je poraslo na 40 milijardi dolara u 2020. godini. [4]

### Bosch:Inovator za praćenje

U 2015. godini Bosh je pokrenuo čuveni ,,Prati i pronađi“ program *(eng. Track and Trace)*. Glavni povod za ovakav program je to što su radnici trošili ogromnu količinu vremena na tražeći gde su ostavili svoj alat. Zbog toga je kompanija dodala senzore na određeni alat da bi ih radnici lakše nalazili. Daljim napretkom ove tehnologije Bosch planira da je koristi i u proizvodnim pogonima. [4]

### John Deere: Samovozeći traktori

Kako poljoprivreda postaje više nauka nego umetnost koja se prenosi iz generacije u generaciju, John Deere odgovara primenom interneta inteligentnih uređaja, možda čak i najviše u sferi samovozećih traktora. Traktori više nisu ono što su bili, sada su prepuni raznorarznih senzora, kompjutera i mikrokontrolera. John Deere je takođe kompanija koja je mnogo usavršila GPS (eng. Global Position System) tehnologiju. Najnapredniji sistemi su precizni i do neverovatnih dva centimetra. [4]

### Maersk: Pametna logistika

Danska brodarska kompanija, isporučivši milione kontejnera u 121 zemlju šitom sveta, prihvatila je internet inteligentnih uređaja kako bi pratila svoju imovinu i optimizovala potrošnju goriva i rute svojih brodova. Pokazalo se da je tehnologija posebno korisna za rashladne kontejnere čiji bi se sadržaj mogao pokvariti bez adekvatnog hlađenja. Budući da kompanija godišnje troši oko milijardu dolara na transport praznih kontejnera za transport, Maersk je ugradio senzore i aplikacije za analitiku podataka kako bi bolje optimizovali korišćenje i lociranje kontejnera. [4]

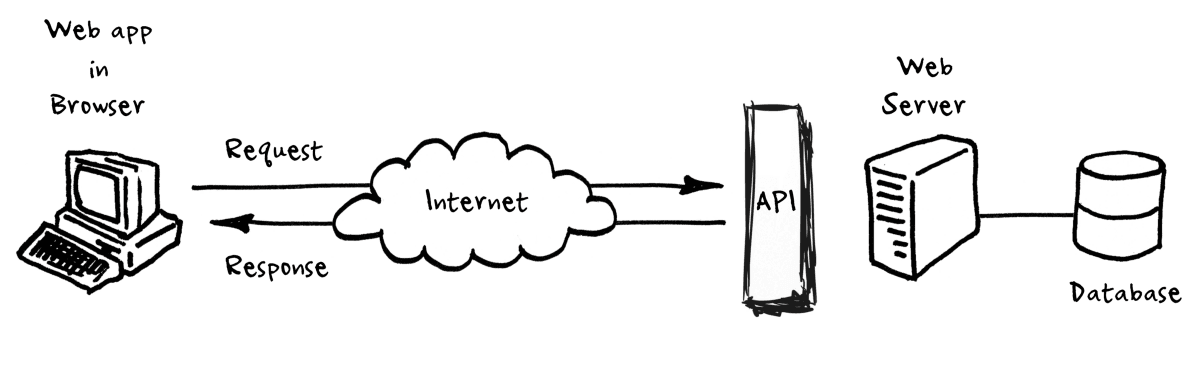
## IoT u detekciji vremenske prognoze

Ne postoji mnogo tipova podataka koji mogu da utiču na poslovne odluke i lične svakodnevne planove u istoj meri kao podaci o vremenskoj prognozi. Tačni i dostupni vremenski podaci su jedni od možda najbtinijih podataka za svakodnevni život i normalno poslovanje određenih kompanija. [5]

Sistem za praćenje vremenske prognoze se bavi otkrivanjem i prikupljanjem različitih vremenskih parametara na različitim lokacijama, koji se zatim mogu analizirati i koristiti za prognozu vremena. Cilj ovog sistema postiže se pomoću tehnologije kao što je IoT i Cloud. Korišćenjem interneta, informacije IoT uređaja se lako mogu preneti na cloud a zatim iz cloud-a do krajnjeg korisnika. Praćenje vremenske prognoze je osnovna i praktična implementacija IoT-a. To uključuje prikupljanje raznih podataka o vremenskoj prognozi i njihovo korišćenje za upozoravanje, slanje obaveštenja, prilagođavanje određenih uređaja u skladu sa dobijenim podacima i za dugotrajnu analizu. [5]

### API (application programming interface)

API je programski kod koji omogućava softverskim proizvodima da razmenjuju podatake među sobom. On takođe sadrži uslove pod kojim se ova razmena podataka vrši. [6]



Slika 2-API [6]

API funkcioniše tako što omogućava programeru da šalje određene zahteve tj pozive da bi dobio ili poslao određene podatke. Ova komunikacije se vrši korišćenjem programskog jezika JSON (JavaScript Object Notation). On takođe može da ažurira ili čak da briše podatke prilikom poziva. Postoji četiri tipa zahteva koji se mogu koristiti preko API-ja:

* GET – prikuplja informacije
* PUT- ažurira informacije
* POST – kreira nove informacije na serveru
* DELETE – briše određene informacije

Primer GET zahteva OpenWeather API-ja:

api.openweathermap.org/data/2.5/weather?q={city name}&appid={API key}

Odgovor u JSON formatu:

{

"coord": {

"lon": -122.08,

"lat": 37.39

},

"weather": [

{

"id": 800,

"main": "Clear",

"description": "clear sky",

"icon": "01d"

}

],

"base": "stations",

"main": {

"temp": 282.55,

"feels\_like": 281.86,

"temp\_min": 280.37,

"temp\_max": 284.26,

"pressure": 1023,

"humidity": 100

},

"visibility": 16093,

"wind": {

"speed": 1.5,

"deg": 350

},

"clouds": {

"all": 1

},

"dt": 1560350645,

"sys": {

"type": 1,

"id": 5122,

"message": 0.0139,

"country": "US",

"sunrise": 1560343627,

"sunset": 1560396563

},

"timezone": -25200,

"id": 420006353,

"name": "Mountain View",

"cod": 200

}

### JSON

JavaScript Object Notation(JSON) je jedan od lakših tekstualnih otvorenih standarda dizajniranih za čitljivu razmenu podataka. Ekstenzije datoteke sa podacima u JSON-ovom formatu je .json. [7]

JSON format polako zamenjuje XML jer ima prednost u odnosu na njega jer ne koristi tagove pa ima kraći kod koji je čitljiviji i lakši za pisanje. Najbitnija prednost JSON je što se parsira kroz standardnu JavaScript funckiju. JSON ima strukturu objekta u Java programskom jeziku. [7]

### Openweathermap

Openweathermap i njihov API pružaju jedno od najkorišćenijih načina da se dođe do tačnih podataka o vremenskoj prognozi i mnogim drugim podacima o vremenu. Prikupljaju podatke na preko 40000 lokacija u svetu. Dokumentacija je odlično objašnjena i dostupna stoga i jeste toliko korišćen, naročito kod početnika i amatera u kreiranju aplikacija. Jedna od najbitnijih stvari je to što API šalje podatke u JSON formatu olakšavajući njegovu upotrebu drastično.

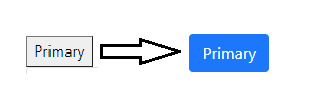
Njihov API omogućava pristup raznim podacima kao što su: [8]

* Istorija vremenske prognoze u zadnjih 40 godina sa satnom granulacijom
* Trenutna vremenska prognoza sa desetominutnim ažuriranjem
* Vremenska prognoza u narednih 4 dana sa satnom granulacijom
* Vremenska prognoza u narednih 16 dana koja daje podatke 4 puta dnevno: uveče,u toku dana,predveče i ujutru
* Vremenska prognoza za narednih 30 dana
* Mape sa globalnim padavinama zasnovanim na radarskim podacima i satelitskim slikama koje se kreiraju pomoću mašinskog učenja

## Bootstrap

Jedan od najpopularnijih front-end fremjworkova otvorenog koda na svetu, bootstrap, koristi se za kreiranje dinamičnih veb-sajtova i veb aplikacija. Baziran je na HTML i CSS šablonima za tipografiju, kreiranju formulara, dugmadi, navigacionim i ostalim komponentama interfejsa kao i opcionim javascript dodacima.

Bootstrap omogućava jednostavno i brzo kreiranje veoma privlačnih i unikatnih veb-sajtova. On se dodaje na sam vebsajt skidanjem saimih fajlova i povezivanjem sa datim projektom ili običnim likovanjem u html dokumentu.

Bootstrap je veoma lak i jednostavan za korišćenje. Dodavanjem određenim html elementima određene bootstrap klase, oni menjaju svoj oblik, boju, veličinu, bez ikakvog pisanja css-a. Da bi smo dugme (slika 3 - levo) transformisali u dugme (slika 3 - desno) potrebno je dodati datom dugmetu samo klase: btn, btn-primary.  
Moguće je kreirati čak i takozvane jumbotrone sa svega nekoliko html tagova i par klasa. Tako nešto bi oduzelo dosta vremena kada bi moralo da se kreira pomoću čistog css-a.

Slika 3 – transformacija pomoću bootstrapa

<div class="jumbotron">

<h1 class="display-4">Hello, world!</h1>

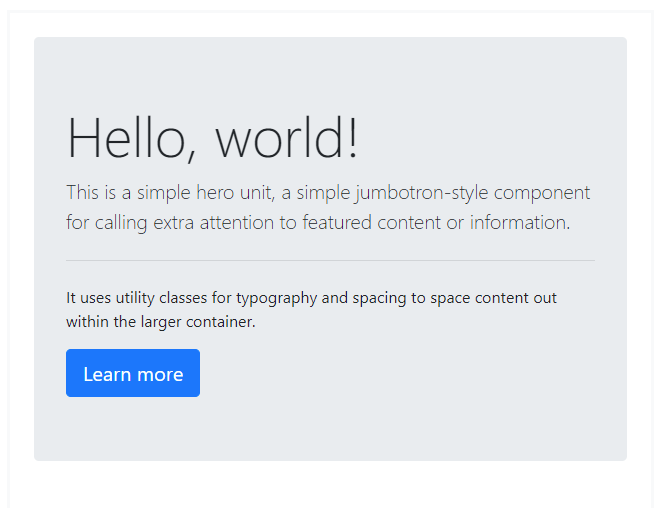
<p class="lead">This is a simple hero unit, a simple jumbotron-style component for calling extra attention to featured content or information.</p>

<hr class="my-4">

<p>It uses utility classes for typography and spacing to space content out within the larger container.</p>

<a class="btn btn-primary btn-lg" href="#" role="button">Learn more</a>

</div>



Slika 4 - Jumbotron

Prednost bootstrepa je što ima jako detaljnu, lako razumljivu i obimnu dokmunetaciju, jednostavno je pronaći ono što je potrebno i veoma je lako primeniti već kreirane stvari. Mana toga je što veb-sajtovi kreirani preko običnog, nemenjanog bootstrapa izgledaju nekako poznato redundantno.

# Alternativni API-ji

Postoji nekoliko raziličitih javnih API-ja za vremensku prognozu poput OpenWeatherMap API, Dark Sky API, Visual Crossing Weather API i ostali, svaki ima određene prednosti i mane, ali svaki može da napravi moćnu aplikaciju za vremensku prognozu

### The Dark Sky API

The Dark Sky API je robusni API koji pruža veoma kompleksne podatke o vremenskoj prognozi o bilo kom mestu u svetu. API vraća veliku količinu podataka: kakvo će vreme biti svaki naredni minut u trajanju od sat vremena, vremensku prognozu za svaki sat u narednih 48 sati i svakodnevnu prognozu u narednih nedelju dana. [9]

Neke od prednosti ovog API-ja su to što je veoma precizan na nekim malim mestima, podaci o vremenskoj prognozi se ažuriraju veoma često, postoji baza podataka o vremenskoj prognozi u prethodnih nekoliko decenija i podržava čak 39 različitih jezika. [9]

Nedostaci ovog API-ja su to što je besplatno korišćenje API-ja dopušta samo 100 poziva mesečno a svaki sledeći košta 0.001 $. [9]

### Simple Weather API

Potpuno besplatni Simple Weather API je veoma lak za korišćenje radi dobijanja trenutne vremenske prognoze za određene lokacije na svetu. Ovaj API je omogućio da skoro svako može da dobije informacije o vremenskoj prognozi potpuno besplatno. Ovaj API ne daje kompleksne podatke o vremenskoj prognozi već samo najjednostavnije i najbitnije podatke za većinu korisnika. [9]

### AIR Quality API

Ovaj API je unikatan i specijalizovan konkretno za čistoću vazduha. Nakon što mu se pošalju geografska širina i dužina API šalje pouzdane podatke o čistoći vazduha koji su veoma bitni u današnje vreme globalnog zagrevanja i emisije štetnih gasova, omogućavajući ljudima da unapred planiraju putovanja i ekskurzije izbegavajući veoma zagađene predele. [9]

Prednosti ovog API-ja su to što poseduje izuzetno tačne podatke o kvalitetu vazduha koji su zasnovani na najboljim svetskim modelima vremenske prognoze i internim tehnologijama. API vraća parametre čistoće vazduha u naredna 3 dana, parametri su izraženi takozvani AQI ( air qualitz index) skorom. [9]

### AccuWeather API

Najveći konkurent OpenWeather API-ju, AccuWeather API pruža slične podatke o vremenskim uslovima. Pruža pouzdane podatke sa bilo koje lokacije na Zemlji. Sa ovim API-jem se može dobiti vremenska prognoza svaki minut sa video prezentacijama koje se mogu prikazivati na raznim uređajima. [9]

Veliki plus ovog API-ja je što ima besplatnu verziju za testiranje i kreiranje jednostavnih aplikacija. Ali velika mana je što se većina tih aplikacija nikada neće pustiti u promet jer je plaćanje korišćenja API-ja veoma skupa i van budžeta većini. [9]

# Razvoj aplikacije

Aplikacija za vremensku prognozu ,,Weather“ treba da obezbedni pregled vremeneske prognoze na veoma jednostavan i pregledan način uz visok nivo funkcionalnosti. Ova aplikacija treba da zadovolji potrebe korisnika koji želi da vidi samo kakvo je vreme napolju, da li pada kiša, da li je hladno ili čak pada sneg. Takvog korisnika zanima samo trenutno vreme, on se neće zadržati na aplikaciji duže od jednog minuta. Drugi tip korisnika koji nas interesuje je onaj koji planira neko putovanje ili ga čak zanima kakvo je vreme u nekom drugom gradu i kakvo će vreme biti u naredna 3 dana. Najvžnije od svega je da aplikacija bude besplatna. Niko ne voli da prilikom dolaska na neki sajt autmatski izlazi neka notifikacija o tome koliko košta data usluga, uostalom, svet sve više prelazi na model otvorenog koda.

Veoma je bitno da kada korisnik pronađe našu aplikaciju da se na njoj i zadrži,što duže. Zato boje koje su korišćene za dizajniranje aplikacije moraju biti blage i obične. Aplikacija mora biti responzivna da bi korisnik mogao da je poseti na svakom uređaju, bilo daje kompjuter, tablet, ili pak pametni frižider, pametno ogledalo ili bilo koji drugi uređaj.

Da bi se sve to ispoštovalo, korišćeni su javascript, css i bootstrap. Sa ove 3 tehnologije moguće je napraviti jako moćnu veb aplikaciju kojoj je dovoljan samo veb čitač da bi funkcionisala na svakom uređaju.

## Kreiranje aplikacije

Veb aplikacija ,,Weather”, je kreirana korišćenjem javascript, css i bootstrap programskih jezika. Aplikacija poziva API koji nudi kompanija openweathermap i na jednostavan način, na jednoj veb stranici prikazuje sve potrebne informacije o vremenskoj prognozi. Aplikacija je projektovana tako da bude što jednostavnija za korišćenje. Veb stranicu čini 5 bootstrap kartica koje su kreirane koristeći bootstrap ,,card“ klasu. Responzivnost je obezbeđena korišćenjem bootstrap grid sistema. Prilikom smanjivanja medijuma za korišćenje aplikacija, dizajn aplikacije se menja. Raspored kartica se menja tako da umesto u dva, kartice su raspoređene u pet redova jedne ispod drugih. (slike 5 i 6)

Informacije koje ovih pet kartica pružaju su:

1. Dve kartice daju mogućnost biranja bilo kog grada u svetu i dobijanje vremenske prognoze za konkretan grad, to podrazumeva: temperaturu, vlažnost vazduha i brzinu vetra
2. Prikazivanje trenutno vremenske prognoze za korisničku lokaciju
3. Prikazivanje broj stepeni u naredna 3 dana
4. Slajder koji omogućava prikazivanje temperature svakog sata za korisničku lokaciju

## Javascript i API pozivi

Jedan od najvećih izazova ove aplikacije je bilo napraviti korektne i tačne API pozive. Iako je dokumentacija veoma obimna i dobro objašnjena, ovo se pokazalo kao najveći problem pogotovo zbog asinhronog ponašanja javascripta.

Prva i možda najbitnija javscript funkcija u ovoj aplikaciji je geo():

function geo() {

if ('geolocation' in navigator) {

navigator.geolocation.getCurrentPosition(function(position){

const lat = position.coords.latitude;

const lon = position.coords.longitude;

getAsync(`https://api.openweathermap.org/data/2.5/onecall?lat=${lat}&lon=${lon}&units=metric&appid=db30a0d6b8effaec3310a7703223e2bd`).then(data =>{

visitorWeather(data);

nextDays(data);

hourly(data);

})

})

}else{

alert("geolocation unavailable");

}

}

Ova funckija nam popunjava čak 3 od 5 kartica. Ona sama i poziva različite druge funkcije i najkompleksnija je u celom projektu stoga će ona prva biti objašnjena. Ona služi da se dobije lokacija korisnika i na osnovu nje napravi API poziv korišćenjem aync await forme na osnovu koga API vraća odgovarajuće podatke koji se zatim ubacuju u konkretne kartice.

Da bi se testiralo da li postoji mogućnost dobijanja korisničke lokacije, proverava se da li postoji geolocation objekat. Ako ne postoji, korisnik se obaveštava odgovarajućim alertom u suprtonom se konačna lokacija dobija pozivanjem .getCurrentPosition metode. Nakon što se dobije lokacija,izvršava se call back funkcija i geografska širina i dužina se čuvaju u lat i lon promenljivama radi lakšeg API poziva.

Asinhrono programiranje je programiranje koje omogućava neki tip paralelnog programiranja koje omogućava da se neki deo programa izvršava na zasebnoj niti. API poziv se vrši pomoću async funkcije i *fetch()* metode. Async reč se dodaje ispred funkcije i pretvara je u asinhronu funkciju. To je funkcija koja zna kako da barata sa ,,await“ rečju kojim se priziva asinhroni kod. Asinhrona funkcija je bazirana na takozvanim obećanjima *(eng. promises)* u javascriptu i olakšava baratanje sa njima. Obećanje je javascript objekat koji predstravlja konačno izvršavanje ili prekid izvršavanja asinhronih operacija i njihovih konačnih rešenja. Da bi se zapravo iskorsitila vrednost koja je vraćena kada se obećanje izvršilo koristi se .then() blok. Prednosti asinhrone funkcije se vidi tek kada se kombinuje sa await rečju. Await može da se stavi ispred bilo kog asinhrone funkcije da zaustavi kod na toj liniji dok se obećanje ne ispuni i tek onda vraća rešenje. U nastavku je data funkcija koja vrši API poziv.

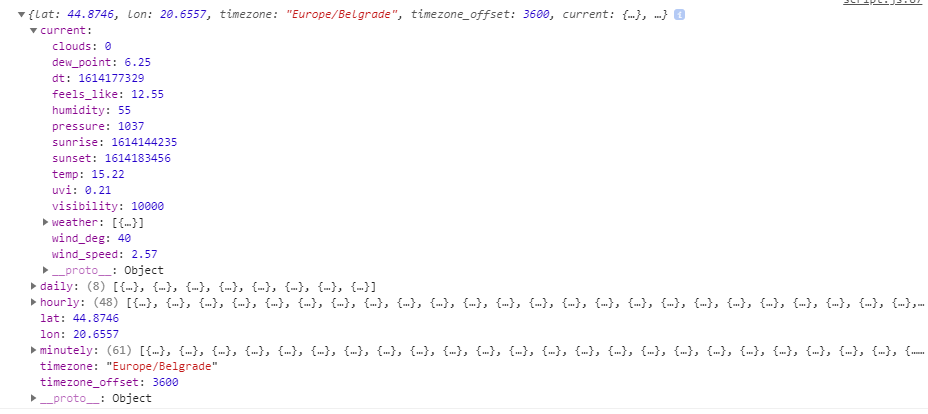
async function getAsync(url) {

let response = await fetch(url);

let data= await response.json();

return data

}



Slika 5 – API odgovor

Ova funkcija prima API link specificiran u dokumentaciji i pomoću fetch() metode prihvata odgovor i pomoću .json() metode ga ,,parsuje” u javascript objekat. Veoma je lako pristupini svakom podatku iz datog odgovora koristeći objektnu notaciju. Nakon što getAsync() vrati podatke, pozivaju se tri funkcije: visitorWeather(data), nextDays(data), hourly(data);

Funkcija visitorWeather(data)prima podatke dobijene od API-ja i na osnovu njih kreira karticu sa podacima o vremenskoj prognozi u gradu u kome korisnik živi. Ti podaci su: trenutna temperatura, brzina vetra i vlažnost vazuduha sa propratnom slikom i kratkim opisom vremena.

function visitorWeather(data) {

const cityName=document.getElementById('yourLocation');

const userWeather = document.getElementById('userWeather');

const dataCityName = data.timezone.split('/');

crtImgFinal('locationCard','newIMG', data.current.weather[0].icon,'hr2','rounded mx-auto d-block','150px','150px');

let desc = data.current.weather[0].description;

desc = desc[0].toUpperCase() + desc.slice(1);

cityName.innerHTML= `<h1>${dataCityName[1]}</h1><hr><small>${desc}</small>`;

userWeather.innerHTML = `Current: <b>${Math.floor(data.current.temp)}</b> &#8451;<br> Feels like: ${Math.floor(data.current.feels\_like)} &#8451; <br> Wind speed: ${data.current.wind\_speed} km/h <br> Humidity: ${data.current.humidity}%`;

}

Ova funkcija je krajnje jednostava i njen posao je da prikaže dobijene podatke tako što uzima konkretne HTML elemente preko njihovog id-a i u njih upisuje dobijene podatke. Zanimljiva funkcija koja se poziva ovde je crtImgFinal() koju ćemo posebno objasniti.

function crtImgFinal(container,newId,imgName,insertBeforeId,classNames,height,width) {

const imgCointainer=document.getElementById(container);

let newIMG = document.createElement('img');

newIMG.id=newId;

newIMG.src=`http://openweathermap.org/img/wn/${imgName}@2x.png`;

newIMG.className=classNames;

newIMG.style.height= height;

newIMG.style.width= width;

const before =document.getElementById(insertBeforeId);

imgCointainer.insertBefore(newIMG,before);

}

Data funkcija prima dosta parametara zato što se koristi na više mesta i na svakom mestu su potrebni slični parametri da bi se dobio željeni rezultat. Ova funkcija prima kontejener u kome slika treba da se prikaže, njen id, konkretno ime ikonice, HTML element pre koga treba ona da se ubaci, bootstrap klase koje treba da dobije i njenu visinu i širinu. Slika se dodaje u datu karticu tako što API prilikom odgovora daje konkretno ime ikonice koja trebe da se koristi, i to ime se zatim ubacuje u standardni deo već datog linka koji će ga razlikovati od ostalih slika. Na samom kraju ove funkcije korišćena je .insertBefore() metoda. Ova metoda prima kao prvi parametar novi HTML element koji želi da se ubaci, a kao drugi prima postojeći HTML element pre koga treba da se doda novi.

function nextDays(data) {

const dailyTxt = document.getElementById('dailyTxt');

for (let index = 1; index < 4 ; index++) {

switch (index) {

case 1:

dailyTxt.innerHTML=`${dayOfTheWeek(data.daily[index].dt)}: <h1>${Math.floor(data.daily[index].temp.day)} &#8451;</h1><br>`;

break;

case 2:

dailyTxt.innerHTML+=`${dayOfTheWeek(data.daily[index].dt)}: <h1>${Math.floor(data.daily[index].temp.day)} &#8451;</h1><br>`;

break;

case 3:

dailyTxt.innerHTML+=`${dayOfTheWeek(data.daily[index].dt)}: <h1>${Math.floor(data.daily[index].temp.day)} &#8451;</h1>`;

break;

default:

break;

}

}

}

Sledeća funkcija koja se poziva je nextDays(data)ona takođe prima odgovor API-ja i služi da prikaže vremensku prognozu za naredna tri dana. To vrši kroz jednu for petlju sa jednim switchem. Veliki problem za ovakvo ispisivanje je što openWeather API daje datum u UNIX vremenu *(eng. timestamp)* što nije korisno za svakodnevnog korisnika stoga mora da se konvertuje u dane. Za to je potrebna zasebna funkcija koja će kao parametar da primi dati timestamp i pretvoriti ga u nešto poznato korisniku.

function dayOfTheWeek(unixTime) {

var a = new Date(unixTime\*1000);

var days = ['Sunday','Monday','Tuesday','Wednesday','Thursday','Friday','Saturday'];

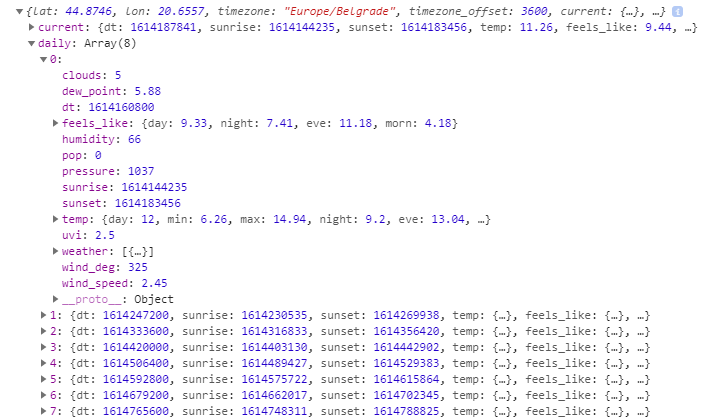
var dayOfWeek = days[a.getDay()]

return dayOfWeek;

}

Najpre je potrebno kreirati novi vremenski objekat, a da bi smo to uradili moramo timestamp da pretvorimo u milisekunde jer takav tip podatka prima njegov konsturktor. Da bi se dobio konkretan dan u nedelju potrebno je napraviti niz sa stringovima imena dana, počveši od nedelje koja se nalazi na poziciji 0 do subote koja je na poziciji 6 u nizu. Upravo sama metoda nad objektom Date pod nazivom .getDay() vraća broj od 0 do 6 u zavisnosti od toga koji je dan u nedelji. Ovo se upravo može iskoristiti da bi se vratio onaj dan iz niza koji je zapravo današnji.

Konkretnim infromacijama o dnevnoj vremenskoj prognozi se pristupa na ovaj način je re tako definisano u samom API-ju u JSON formatu.



Slika 6 – API deo odgovora za dnevnu prognozu

API vraća odgovor u vidu niza od 0 do 7 tj od današnjeg dana do sedam dana u napred. Zato se ovim podacima i pristupa na ovaj načim, korišćenjem objektne notacije. Takođe, temperatura je zadatau decimalnom zapisu što ne odgovara korisničkim zahtevima stoga je potrebno zaokružiti taj broj korišćenjem Math.floor() metode.

function hourly(data) {

let rangeValue = document.getElementById('myRange');

let hourlyTxt = document.getElementById('hourlyTxt');

let hourlyDeg = document.getElementById('hourlyDegrees');

let getHour=unixHourly(data.hourly[1].dt);

crtImgFinal('testDiv','hourlyImg',data.hourly[1].weather[0].icon,'testhr');

hourlyTxt.innerHTML=`<p class="display-4">${getHour}</p><span class="h5">${data.hourly[1].weather[0].description}</span>`;

hourlyDeg.innerHTML=`<p class="display-4 para">${Math.floor(data.hourly[1].temp)} &#8451;</p>`;

rangeValue.addEventListener('input',function(){

let range=event.target.value;

let getHour=unixHourly(data.hourly[range].dt);

const exists=document.getElementById('hourlyImg');

if (exists!=null) {

exists.remove();

} crtImgFinal('testDiv','hourlyImg',data.hourly[range].weather[0].icon,'testhr');

hourlyTxt.innerHTML=`<p class="display-4">${getHour}</p> <span class="h5">${data.hourly[range].weather[0].description}</span>`;

hourlyDeg.innerHTML=`<p class="display-4 para">${Math.floor(data.hourly[range].temp)} &#8451;</p>`;

})

}

Funkcija hourly(data)služi za dodavanje funkcionalnosti slajderu na petoj bootstrap kartici. Ona služi da prikže temperaturu za svaki čas u narednih 24 sata uz propratnu sliku stanja vremena u tom trenutku. Ova funkcija radi slično kao i nextDays(data) ,jedino što se UNIX timestamp transformiše u časove a ne dane, ostatak same funkcije je veoma slično.

Promenjiva getHour poziva unixHourly(unixTime) funkciju svaki put kada korisnik pomeri slajder i prosleđuje joj UNIX timestamp, ona vraća vreme u časovima koje se prikazuje u kartici uz propratnu sliku. To se vrši korišćenjem .innerhtml svojstva elemenata koje omogućva dodavanje HTML koda na određeno mesto u vidu stringa.

function loadName() {

const cityName=document.getElementById('inputCityName');

const xhr = new XMLHttpRequest();

xhr.open('Get',`https://api.openweathermap.org/data/2.5/weather?q=${cityName.value}&units=metric&appid=db30a0d6b8effaec3310a7703223e2bd`,true);

xhr.onload = function(){

if (this.status===200) {

const response = JSON.parse(this.responseText);

printData(response);

const exists = document.getElementById('newImg');

const icon = response.weather[0].icon;

if(exists===null){ //Ako slika ne postoji, kreiraj, u suprotnom obrisi pa kreiraj

crtImgFinal('insertCard','newImg',icon,'cardResponse');

}else{

exists.remove();

crtImgFinal('insertCard','newImg',icon,'cardResponse');

}

const lrt=document.getElementById('alrtid')

if (lrt!=null) { //ako postoji alert za pogresno upisan grad, obrisi ga

lrt.remove();

}

}

else{

// u suprotnom ga napravi

if (document.getElementById('alrtid')===null) {

const inputNameCard =document.getElementById('cardCityName');

const newAlert = document.createElement('div');

newAlert.classList.add("alert","alert-danger");

newAlert.id='alrtid';

newAlert.textContent='Not a valid name!'

inputNameCard.insertBefore(newAlert,cityName);

}

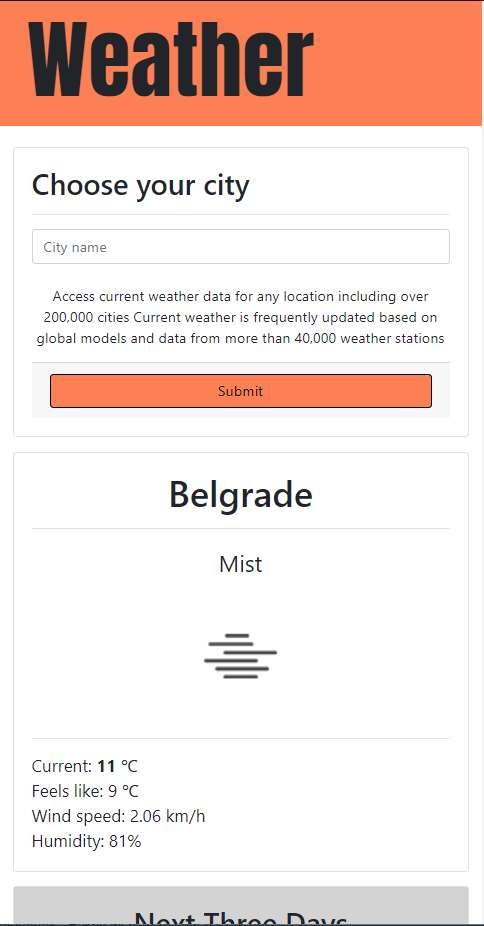
}

}

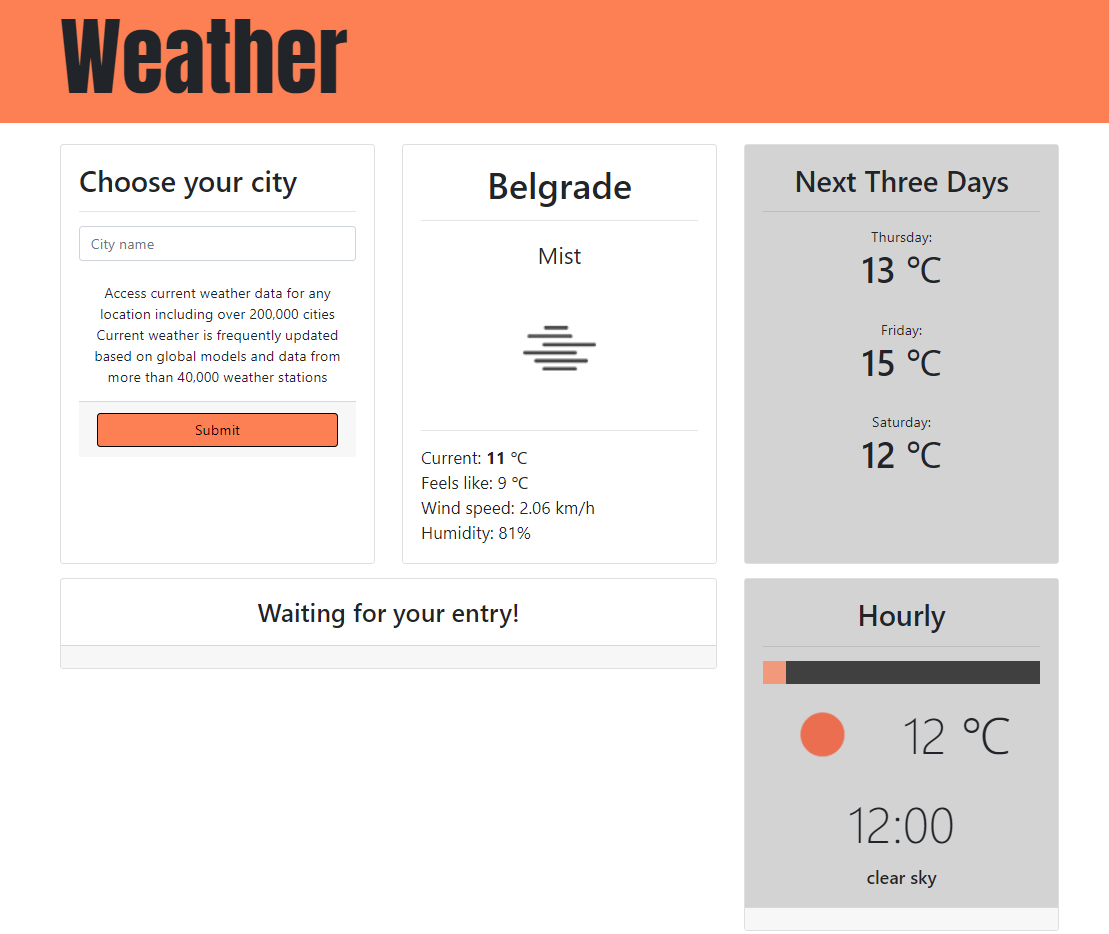
xhr.send();

}

loadName() funkcija služi za prikazivanje vremenske prognoze za bilo koji grad na svetu. Ona, za razliku od ostalih, vrši svoj API poziv zato što je tako specificirano u dokumentaciji. Potreban je specijalan poziv za specjalan tip podataka koji šalje nazad, konkretno, podatke za određeni grad. Ovaj poziv se vrši na drugačiji način od prethodnog, na stari način korišćenjem samo javascripta. XMLHttpRequest (XHR) objekat se koristi za interakciju sa serverom, omogućava komunikaciju sa serverom bez potpunoh osvežavanja stranice. XHR se uglavnom ne koristi za JSON ali funkcioniše besprekorno. XHR vrši novi poziv ka serveru korišćenjem .open() metode koja kao prvi parametar prima HTTP request metodu: GET, POST, PUT, DELETE. Drugi parametar koji prima je sam url sa koga treba da pribavi podatke. Nakon pribavljenih podataka izvršava se .onload() metoda u kojoj se proverava da li je server odgovorio pozitivno, sa satus kodom 200, ili je doslo do neke greške. Ako dođe do greške stvoriće se bootstrap alert sa propratnim klasama koji će obavestiti korisnika da unese tačan naziv grada.



Slika 7 – Izgled aplikacije na malom ekranu



Slika 8 – Izgled aplikacije na velikom ekranu

# Zaključak

Pošto se IoT sve manje može odvojiti od svakodnevnih aktivnosti čoveka, treba ga što više razviti i usavršiti i navići se na njegovo postojanje na svakoj lokaciji u svetu. IoT menja industrije neverovatnom brzinom, menja čitave načine poslovanja, olakšava ih stoga njihova integracija u svakodnevni život mora biti što jednostavnija.

Aplikacija ,,Weather“ koja je razvijena na ovu temu nije direktno povezana sa IoT uređajima već samo predstavlja način da se prikupljeni podaci prikazuju. Trenutno je aplikacija napravljena samo za kompjuter i veb čitače, ali veoma lako se može prilagoditi i za neki od IoT uređaja kao što su pametni frižider, pametni sat ili bilo koji drugi IoT uređaj.

Od samog koda obrađene aplikacije, uglavnom je obrađen javascript jer se smatra da čitalac ovakvog rada poznaje osnovne tehnologije veb dizajna i programiranja, stoga je objašnjena funkcionalnost samo određenih delova javascripta. Bootstrap koji je korišćen u ovoj aplikaciji je najjednostavniji HTML framework, ima jako dobru dokumentaciju i učinio je ovu aplikaciju onim što jeste. Najbitnije je da bilo koja veb aplikacija bude responzivna i da podržava razne veličine ekrana i razne uređaje jer onda aplikacija nije limitirana običnim uređajem.

# Korišćena i referentna literatura

1. Steve R. (2020). The Internet of Things explained. What the IoT is, and where it's going next. Pristupljeno 20. februara, 2020 sa <https://www.zdnet.com/article/what-is-the-internet-of-things-everything-you-need-to-know-about-the-iot-right-now/>
2. Alexander S. G. (2020). Internet of Things (IoT). Pristupljeno 20. februara, 2020 sa [https://internetofthingsagenda.techtarget.com  
   /definition/Internet-of-Things-IoT](https://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT)
3. Liyanage S. , Chamin M., Iskandar P (2012): Engineering Applications of Artificial Intelligence: State of the art of smart homes. IFAC, the International Federation of Automatic Control.
4. Brian B. (2017). The Top 20 Industrial IoT Applications. Pristupljeno 24. februara, 2020 sa <https://www.iotworldtoday.com/2017/09/20/top-20-industrial-iot-applications/>
5. Mohit T., Deepak N., Priya G. Anupma G., Abhinav G., Ankush C. (2020). Weather Monitoring system using IoT and cloud computing. Journal of Advanced Science and Technology Vol. 29, No. 12s, pp. 2473-2479.
6. Perry E. (2017). What exactly IS an API? Pristupljeno 24. februara, 2020 sa <https://medium.com/@perrysetgo/what-exactly-is-an-api-69f36968a41f>
7. Introducing JSON. Pristupljeno 24. februara, 2020 <https://www.json.org/json-en.html>
8. <https://openweathermap.org/>
9. Shyam P. (2018). Top 10 Best Weather APIs: Open Weather Map, Dark Sky, and Others. Pristupljeno 24. februara, 2020 sa <https://blog.api.rakuten.net/top-weather-apis/>