Web aplikacija za prilagodljivo prikazivanje turističkih sadržaja

Ivoš, Goran

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
ELEKTROTEHNIČKI FAKULTET

Sveučilišni studij

Web aplikacija za prilagodljivo prikazivanje turističkih sadržaja

Diplomski rad

Goran Ivoš

Osijek, 2016.

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac D1: Obrazac za imenovanje Povjerenstva za obranu diplomskog rada

Osijek, 07.09.2016.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za obranu diplomskog rada

Ime i prezime studenta:	Goran Ivoš
Studij, smjer:	Diplomski sveučilišni studij Računarstvo, smjer Procesno računarstvo
Mat. br. studenta, godina upisa:	D 747 R, 13.10.2014.
OIB studenta:	44304353897
Mentor:	Prof.dr.sc. Goran Martinović
Sumentor:	
Predsjednik Povjerenstva:	Doc.dr.sc. Alfonzo Baumgartner
Član Povjerenstva:	Doc.dr.sc. Zdravko Krpić
Naslov diplomskog rada:	Web aplikacija za prilagodljivo prikazivanje turističkih sadržaja
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak diplomskog rada:	U radu je potrebno proučiti modele mogućnosti i potrebe nuđenja turističkih sadržaja korisnicima, te osmisliti model prilagodljivog prikazivanja izabranih turističkih sadržaja u ovisnosti o uobičajenim i prognoziranim vremenskim prilikama. Web aplikaciju i bazu podataka treba programski ostvariti koristeći aktualne tehnologije kao što su HTML, CSS, jQuery/Javascript, te AngularJS, te sve prikladno testirati.
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3Postignuti rezultati u odnosu na složenost zadatka: 2Jasnoća pismenog izražavanja: 3Razina samostalnosti: 3
Datum prijedloga ocjene mentora:	07.09.2016.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

IZJAVA O ORIGINALNOSTI RADA

Osijek, 21.09.2016.

Ime i prezime studenta:	Goran Ivoš
Studij:	Diplomski sveučilišni studij Računarstvo, smjer Procesno računarstvo
Mat. br. studenta, godina upisa:	D 747 R, 13.10.2014.
Ephorus podudaranje [%]:	0

Ovom izjavom izjavljujem da je rad pod nazivom**: Web aplikacija za prilagodljivo prikazivanje turističkih sadržaja**

izrađen pod vodstvom mentora Prof.dr.sc. Goran Martinović

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

Sadržaj

1. UVOD

Ovaj rad bavi se problematikom dinamičkog prikazivanja sadržaja ovisno o vremenskoj prognozi i uvjetima za određeno područje, primjerice grada. Cilj je napraviti web aplikaciju koja će prikazivati trenutnu vremensku prognozu za grad Osijek i na temelju tih parametara dinamički ponuditi prijedloge aktivnosti koje bi pojedinci mogli uzeti u obzir prilikom oblikovanja svog slobodnog vremena. Web aplikacija mora povlačiti vremensku prognozu za grad Osijek i na svome sučelju prikazivati prijedloge aktivnosti koji će se povlačiti iz lokalne baze podataka na temelju obrade podataka pomoću algoritma odlučivanja. Algoritam ne ulazi u detalje predviđanja vremenske prognoze, već rukuje podacima koje dobiva iz vanjskog izvora.

Drugo poglavlje prikazuje ulogu računalnih tehnologija u turizmu i ukratko se bavi problematikom računalnih tehnologija koje se primjenjuju u području turizma. Na kraju drugog poglavlja predstavljena su slična rješenja koja se mogu pronaći u obliku web stranica i portala, te web aplikacija, s detaljnijom analizom aplikacije Foursquare. Treće poglavlje pruža uvid u idejno rješenje web aplikacije za prilagodljivo prikazivanje turističkih sadržaja pružajući opis ideje vlastitog rješenja. Također, definirani su korisnički zahtjevi, predstavljen je funkcionalni model aplikacije kao i prijedlog dizajna korisničkog sučelja te je opisan algoritam odlučivanja. Četvrto poglavlje pruža uvid u rješenje web aplikacije za prilagodljivo prikazivanje turističkih sadržaja. Pružen je opis korištenih tehnologija i alata kako za funkcionalni dio, tako i za korisničko sučelje. Opis implementacije je podijeljen na dva glavna dijela i pruža detaljan uvid ostvarenja funkcionalnog dijela aplikacije, a zatim i implementacije korisničkog sučelja. Peto poglavlje pruža upute za korištenje aplikacije s opisanim koracima i elementima sučelja. Također, obavljeno je testiranje aplikacije na slučaju korištenja kako bi se analizirala ispravnost algoritma odlučivanja i ostalih funkcija zaduženih za obradu i prikaz informacija.

2. RAČUNALNE TEHNOLOGIJE U TURIZMU

U današnje vrijeme kada je tehnologija sve više prisutna u svim sferama života, našla je svoje mjesto i u turizmu pružajući razne informacije turistima i svima koji se njima koriste. Postoji mnogo mobilnih i web aplikacija koje turistima pružaju informacije o vremenskoj prognozi, znamenitostima nekog područja ili ih usmjeravaju do najbližih hotela ili restorana [1].

2.1. Uloga računalnih tehnologija u turizmu

Kako razvojem tehnologije računala postaju sve manja i dostupnija sve većoj populaciji, razumljivo je kako će svoju primjenu pronaći u različitim područjima, pa tako i u turizmu. Postoje mnogobrojni internetski portali i stranice, pa tako i aplikacije, koje su usmjerene ka pružanju informacija i usluga vezanih za turističke sadržaje. Primjerice, danas je uz pomoć računala i interneta moguće rezervirati smještaj na drugom kraju svijeta u svega nekoliko jednostavnih koraka, bilo da je riječ o hotelu ili privatnom smještaju. Također, vrlo su cijenjeni internetski portali na kojima korisnici (turisti) mogu ostaviti kratku recenziju i osvrt na neku lokaciju ili područje koje su posjetili. Na taj način moguće je stvoriti popularne lokacije na nekom području i time olakšati odabir aktivnosti za turiste koji se tek spremaju posjetiti neku od tih lokacija. Nedvojbeno je kako računalna tehnologija ima sve veću ulogu na području turizma, a pojavom pametnih telefona, tableta i pametnih satova ona postaje sve izraženija.

2.2. Problematika računalnih tehnologija u turizmu

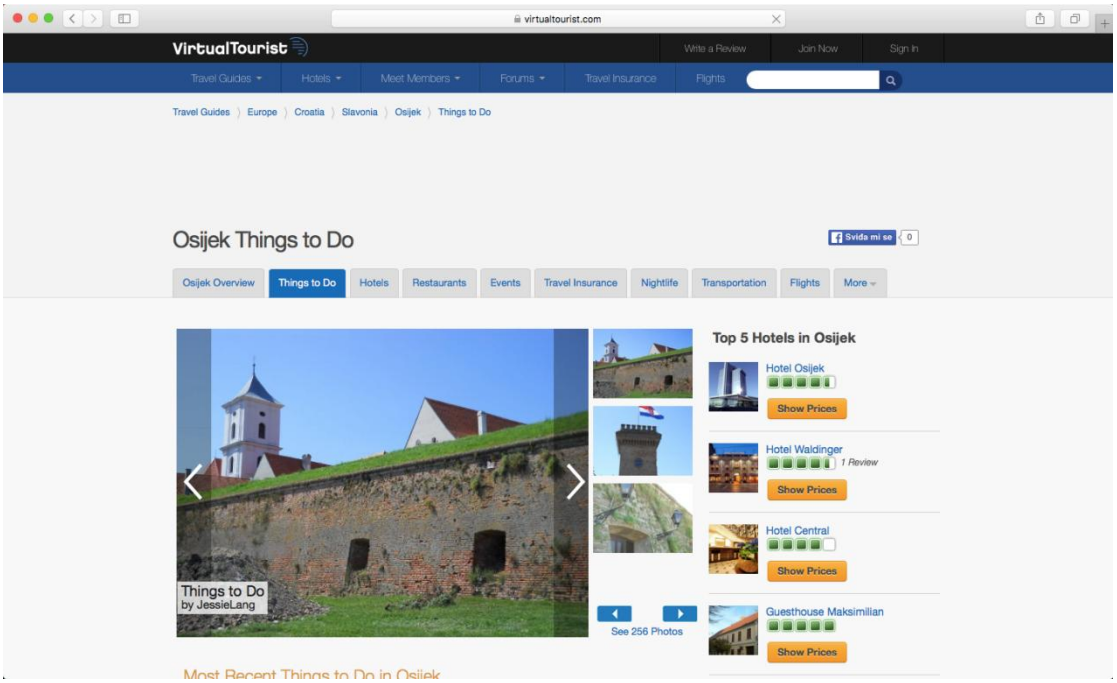
Pružiti pravu informaciju u pravo vrijeme čini glavnu problematiku za većinu sustava, pa tako i u području računalnih tehnologija koje spadaju u sferu turizma. U današnje vrijeme kada raspolažemo velikom količinom informacija, postavlja se zadatak kojemu je cilj razlučiti koje su informacije važne, a koje nisu u određenom trenutku pod određenim uvjetima. Izvori informacija na kojima se temelje modeli su raznoliki. Primjerice [2], kao izvor podataka se koristi poznata aplikacija „Flickr“ kako bi se izvukle informacije o popularnim lokacijama među turistima. Model temeljen na „mudrosti gomile“ pruža informacije koje maksimiziraju interes turista temeljen na njihovim preferencijama i predviđenom budžetu. Postoje i kompleksniji modeli koji se temelje na izgradnji baza znanja [3]. Baza znanja se gradi prikupljanjem informacije o korisnikovom kretanju kroz prostor u određenom vremenskom periodu. Ovakav sustav prikazuje informacije temeljene na trenutnoj korisnikovoj lokaciji i prijašnjim prikupljenim podacima iz baze znanja. Cilj je korisniku pružiti traženu informaciju, a vezano za turističke sadržaje može ih biti nekoliko. Upravo iz razloga što različiti korisnici traže različite informacije i podatke,

potrebno je donijeti odluku i bazirati se na određeno područje, kako korisnik ne bi bio pretrpan različitim informacijama koje su za njega manje bitne ili možda u potpunosti nevažne. U današnje vrijeme sam prikaz informacija je vrlo bitan, a prvenstveno se misli na korisničko sučelje koje bi trebalo biti razumljivo, intuitivno, pregledno te lako za korištenje. Brzina, odnosno odziv sustava na korisnikove upite također igra veliku ulogu, pogotovo kada je sustav razvijen na različitim web tehnologijama, uzimajući u obzir činjenicu da korisnici pristupaju sustavu s različitih vrsta uređaja.

2.3. Pregled sličnih rješenja

Pregled sličnih rješenja, koja pružaju turističke informacije te informacije o vremenskoj prognozi, bazirao se na aplikacijama i internet portalima koje pružaju te informacije za različita područja ili gradove u svijetu, pa tako i Osijek.

Slika 2.1 prikazuje sučelje internetskog portala VirtualTourist koji turistima pruža informacije o gradu ili području, kao što su hoteli, događanja, restorani, mogućnost leta zrakoplovom i tako dalje prema [4].

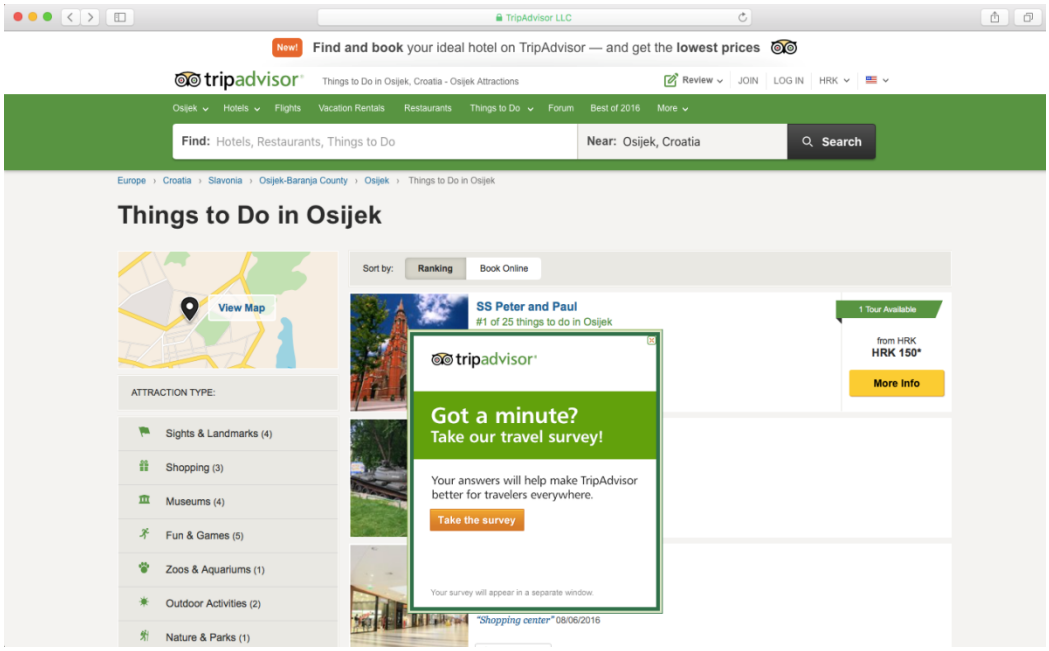


Sl. 2.1 Internetski portal VirtualTourist

Korisnicima portala, također, je omogućeno da unesu vlastite recenzije i osvrte o gradu i na taj način drugima sugeriraju izbor aktivnosti u gradu (engl. *Things to do*).

Slično tome, a vezano isključivo za grad Osijek, internetska stranica Turističke zajednice grada Osijeka pruža informacije o osječkim znamenitostima, muzejima, galerijama, atrakcijama, događanjima, smještaju te restoranima prema [5].

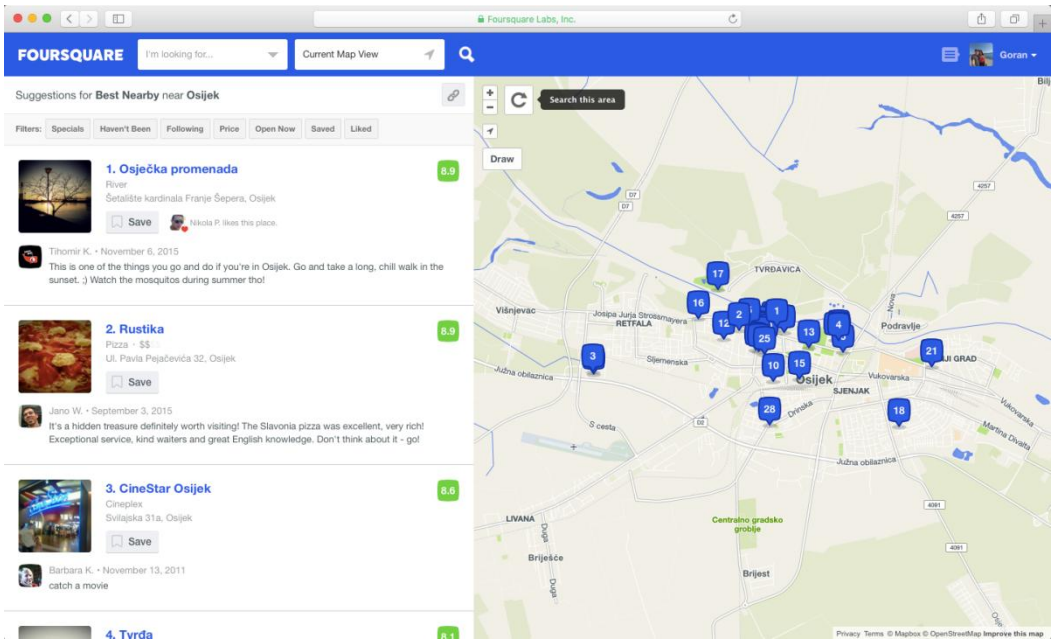
Slika 2.2 prikazuje sučelje internetskog portala TripAdvisor čiji je cilj olakšati korisnicima pronalaženje smještaja na svojima putovanjima, no uz to nudi još dodatne usluge.



Sl. 2.2 Internetski portal TripAdvisor

Uz glavnu funkciju, pronalaženje smještaja, TripAdvisor nudi funkcije pronalaženja restorana i znamenitosti u blizini željene lokacije. Također, korisnici se mogu registrirati i kreirati svoj profil pomoću kojeg mogu ostavljati recenzije i osvrte o lokacijama koje su posjetili prema [6].

Slika 2.3 prikazuje sučelje internetskog portala Foursquare, koje osim web aplikacije, pruža i mobilnu aplikaciju pomoću kojih je moguće pronaći zanimljiva mjesta i lokacije u okolici.



Sl. 2.2 Internetski portal Foursquare

Foursquare koristi lokacijske tehnologije kako bi korisnicima servirala različite sadržaje. Pomoću aplikacije moguće je pretraživati željene lokacije, ali također, aplikacija koristi korisnikovu trenutnu lokaciju prema [7]. Sadržaj koji aplikacija prikazuje temelji se na prijedlozima samih korisnika aplikacije. Korisniku je omogućeno da se prijavi na lokaciji (engl. *check-in*) na kojoj se trenutno nalazi, te napiše kratki komentar, dodijeli ocjenu i priloži fotografiju. Na taj način pojedine lokacije unutar grada, primjerice muzeji, kazališta, restorani i tako dalje, bivaju ocijenjeni od strane korisnika te rangirani prema popularnosti na listi unutar kategorije u koju spadaju. Dodatno, unutar aplikacije korisnik se može povezati sa svojim prijateljima sa različitih društvenih mreža poput Facebook-a ili Twitter-a. Na taj način, korisnik može vidjeti lokacije koje njegovi prijatelji preporučuju i time povećati šansu da i sam posjeti neke od tih lokacija. Sadržaj koji aplikacija preporučuje na temelju korisnikove lokacije ovisi o angažiranosti drugih korisnika koji se nalaze unutar te zajednice, pa time sadržaj može biti ograničen odnosno vrlo oskudan ili ga pak može biti puno ukoliko se radi o nekom velikom gradu, primjerice New York.

Navedeni primjeri sličnih rješenja temelje se na angažiranosti svojih korisnika kako bi se popunio sadržaj koji će se prikazivati. Algoritmi koji odlučuju o tome koji sadržaj se prikazuje uglavnom se temelje na principu ocjenjivanja, te rangiranja prema popularnosti među korisnicima, što podliježe subjektivnom sudu korisnika. Aplikacija Foursquare ide korak dalje

pružajući mogućnost korisniku da se poveže sa zajednicom ljudi na svojim društvenim mrežama, računajući na činjenicu kako korisnik ima veće povjerenje prema poznatoj zajednici.

Motiv za izradu web aplikacije za prilagodljivo prikazivanje turističkih sadržaja je implementacija algoritma koji neće zahtijevati angažiranje korisnika, već je cilj istražiti drugačiji model, te koristiti dodatni izvor informacija, odnosno vremensku prognozu, pri čemu će se model algoritma za odlučivanje temeljiti na podacima vremenske prognoze u stvarnome vremenu.

3. MODEL WEB APLIKACIJE ZA PRILAGODLJIVO PRIKAZIVANJE TURISTIČKIH SADRŽAJA

U ovom poglavlju prikazani su ideja, pristup i model algoritma za izradu web aplikacije.

3.1. Ideja vlastitog rješenja

Cilj ovog rada je izrada web aplikacije koja bi na temelju trenutne vremenske prognoze za grad Osijek pružala prijedloge aktivnosti koji mogu, ali i ne moraju biti vezane za turističku ponudu grada Osijeka. Aktivnosti bi se dinamički generirale ovisno o vremenskoj prognozi te bi služile isključivo kao prijedlog koji bi pojedinac mogao uzeti u obzir prilikom oblikovanja svog slobodnog vremena. Vremenska prognoza sastoji se od osnovnih vrijednosti informacija kao što su temperatura zraka, opis, brzina vjetra, vrijeme izlaska i zalaska Sunca.

Na temelju tih vrijednosti, zadaća aplikacije je ponuditi prijedlog aktivnosti koje se mogu obavljati. Primjerice, ukoliko je vani sunčano aplikacija bi ponudila aktivnosti poput: "Šetnja osječkom promenadom", "Šetnja osječkim parkovima", "Kupanje na vanjskim bazenima", no ukoliko su vremenski uvjeti nepovoljni za vanjske aktivnosti, tada bi aplikacija ponudila aktivnosti poput: "Posjet Muzeju Slavonije", "Posjet Muzeju likovnih umjetnosti", "Kupanje na zatvorenim bazenima" i slično. Aktivnosti bi bile kratkog opisnog tipa, popraćene fotografijom i dodatnim tekstom koji bi pružao zanimljive informacije te bi bile podijeljene u dvije osnovne skupine - vanjske i unutarnje aktivnosti. Također, aktivnosti bi bile smještene na karti, a korisniku bi se omogućio pregled pojedine aktivnosti pomoću Google Maps usluga.

3.2. Specifikacije zahtjeva

Aplikacija mora zadovoljiti osnovne standarde korisnikovog iskustva kao što su intuitivnost, preglednost i jednostavnost. Dizajn sučelja aplikacije mora biti prema trenutnim standardima, a sadržaj i komponente sučelja moraju biti prilagodljive prema rezoluciji uređaja sa kojeg se pristupa aplikaciji. S tehničke strane, aplikacija mora biti dostupna za korištenje putem svih popularnijih web preglednika.

Tablica 3.1 prikazuje popis korisničkih zahtjeva koje aplikacija treba zadovoljavati. Svi zahtjevi imaju veliki prioritet, a odziv sustava prilikom izvršavanja svake od pojedinih radnji mora biti zadovoljavajući.

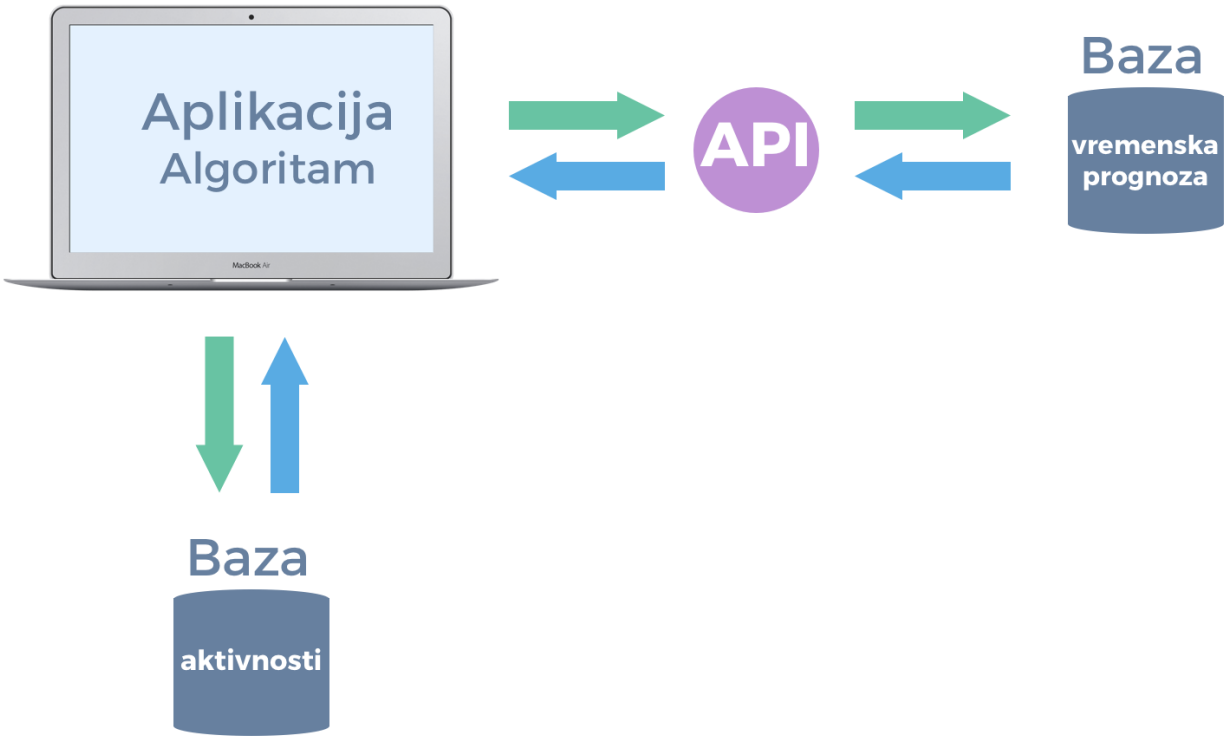
Tab. 3.1 Popis korisničkih zahtjeva aplikacije

Id	Prioritet	Opis
		Opći zahtjevi korisnika
1	1	Prilikom pokretanja aplikacija povlači vremensku prognozu za grad Osijek i prikazuje na sučelju aplikacije
2	1	Kada se dohvati prognoza, analizira se je li povoljna ili nepovoljna te se iz baze povlače aktivnosti sukladno tome
3	1	Sučelje aplikacije popunjava se aktivnostima iz baze podataka
4	1	Korisnik može pregledavati aktivnosti i odabrati detaljan prikaz aktivnosti
5	1	Korisnik odabire detaljan prikaz aktivnosti, otvara se skočni prozor sa fotografijom, nazivom te kratkim opisom kao i dugmetom za prikaz aktivnosti na karti
6	1	Korisnik odabire dugme za prikaz aktivnosti na karti, pri čemu se otvara nova kartica web preglednika i lokacija se prikazuje pomoću Google Maps usluga

Aplikacija mora izvršiti predradnje prije nego ju korisnik može koristiti. Potrebno je dohvatiti podatke o vremenskoj prognozi za grad Osijek koje zatim analizira algoritam. Nakon analize, iz baze podataka se povlače odgovarajuće aktivnosti te se popunjava sučelje aplikacije. Korisnik zatim koristi aplikaciju navigirajući kroz njeno sučelje te interakcijom sa elementima sučelja pregledava aktivnosti.

3.3. Funkcionalni model aplikacije

Funkcionalni model se može podijeliti na dva glavna dijela i to, prvi dio koji obavlja zadaću prikupljanja informacija o vremenskoj prognozi, a drugi dio je zadužen za komunikaciju s poslužiteljem odnosno lokalnom bazom podataka, dok algoritam za odlučivanje čini spoj navedenih dijelova. Prikupljanje podataka o trenutnoj vremenskoj prognozi vrši se iz vanjskog izvora pomoću OpenWeatherMap usluga [8]. Za komuniciranje s bazom podataka OpenWeatherMap koristi se njihov API koji služi za slanje upita te dobivanje odgovora iz baze podataka. Slika 3.1 prikazuje shemu aplikacije, odnosno komunikaciju sa dvjema bazama podataka u kojima se nalaze podaci.



Sl. 3.1 Shema web aplikacije

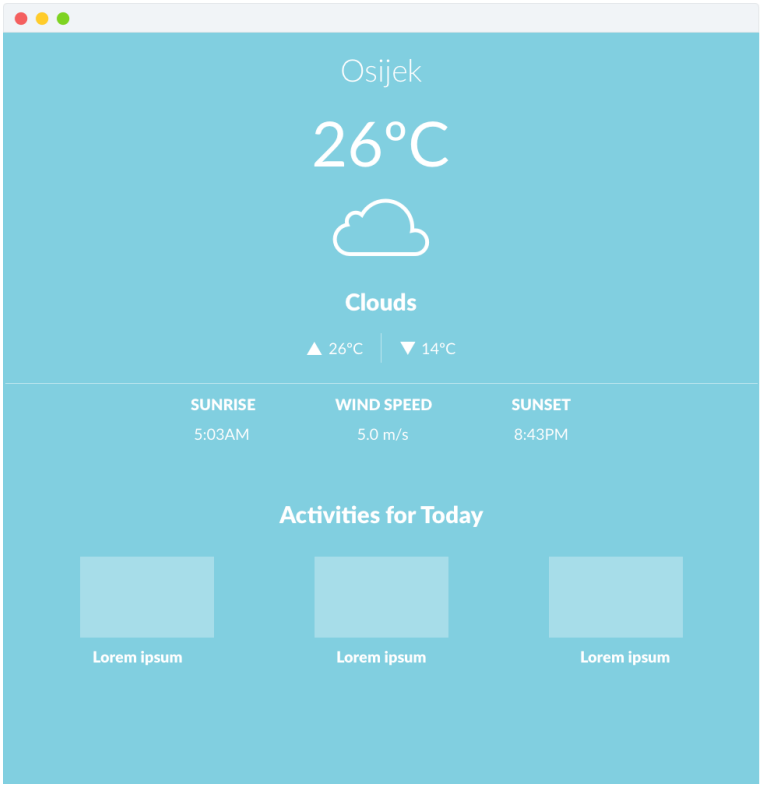
Nakon uspješnog dohvaćanja podataka vrši se njihova obrada putem algoritma za odlučivanje, zatim se sukladno odluci algoritma vrši komunikacija s lokalnim poslužiteljem i bazom podataka u kojoj su spremljene aktivnosti.

Za ostvarenje funkcionalnog dijela web aplikacije potrebno je koristiti neki od API-ja (engl. *Application programming interface*) koji pruža mogućnost dohvaćanja informacija o vremenskoj prognozi, kao što je to primjerice OpenWeatherMap te AngularJS za programiranje funkcionalnosti web aplikacije, kao što je dohvaćanje podataka, parsiranje podataka, prikazivanje podataka na sučelju i tako dalje. Aktivnosti će biti pohranjene u lokalnu bazu podataka te je potrebno koristiti MySql odnosno SQLite i PHP koji se koristi na poslužiteljskoj strani [9].

3.4. Dizajn aplikacije

Za ostvarenje vizualnog izgleda aplikacije potrebno je koristiti HTML (engl. *HyperText Markup Language*) pri definiranju samih elemenata, CSS (engl. *Cascading Style Sheets*) za vizualno stiliziranje samih elemenata, pri čemu će se dodatno koristiti *Bootstrap* okvir za definiranje

elemenata i strukture web sučelja. Sučelje svojim vizualnim pristupom zadovoljava sve moderne standarde, a korisnikovo iskustvo korištenja aplikacije je na prvome mjestu. Slika 3.2 prikazuje inicijalni prijedlog dizajna sučelja web aplikacije.



Sl. 3.2 Dizajn sučelja i raspored elemenata web aplikacije

Napravljen je raspored svih elemenata aplikacije na samome sučelju u skladu s prethodno navedenim opisima. U gornjem dijelu se pružaju informacije o vremenskim uvjetima, dok donji dio služi za prikaz ponuđenih aktivnosti.

3.5. Algoritam odlučivanja

Algoritam odlučivanja zadužen je za donošenje odluke koja će utjecati na sam sadržaj koji će se prikazivati na sučelju, a time i na sadržaj informacija koje korisnik dobiva korištenjem aplikacije. Algoritam mora raspoznati dvije glavne skupine vremenskih uvjeta koji su bitni za implementaciju aplikacije, a to su povoljni, odnosno nepovoljni vremenski uvjeti. Algoritam ne ulazi u detalje predviđanja vremenske prognoze, već manipulira podacima koje dobiva iz vanjskog izvora, u ovome slučaju to je OpenWeatherMap baza podataka. Uz napomenu kako je potrebno uzeti u obzir točnost podataka (što ponekad može biti subjektivno), te ažurnost baze podataka usluge OpenWeatherMap.

Cilj algoritma nije ponuditi najtočniju vremensku prognozu u danome trenutku, već djelotvorno obraditi podatke s kojima raspolaže i na temelju njih donijeti odluku koja će za rezultat imati prilagodljivo prikazivanje sadržaja na sučelju aplikacije. Aktivnosti koje se prikazuju su podijeljene na unutarnje i vanjske, a njihova podjela ovisno o vremenskim uvjetima vidljiva je na slici 3.3.

Sl. 3.3 Podjela aktivnosti na vanjske i unutarnje ovisno o vremenskim uvjetima

S obzirom da su aktivnosti za prikaz podijeljene na vanjske i unutarnje, algoritam odlučivanja na temelju trenutnih vremenskih uvjeta mora donijeti odluku o tome koje aktivnosti se povlače iz baze podataka i prikazuju na sučelju aplikacije. Ukoliko vremenski uvjeti ukazuju na to da je vani sunčano, mjestimice oblačno ili oblačno tada je vrijeme pogodno za vanjske aktivnosti, no ukoliko ukazuju na to da je vani kiša, pljusak, grmljavinsko nevrijeme ili ukoliko sniježi tada je potrebno prikazati aktivnosti za unutarnje prostore. Algoritam odlučivanja raspolaže podacima dobivenim iz baze podataka OpenWeatherMap koji su samodostatni za donošenje odluka i izvršavanje daljnjih funkcija.

4. PROGRAMSKO RJEŠENJE WEB APLIKACIJE ZA PRILAGODLJIVO PRIKAZIVANJE TURISTIČKIH SADRŽAJA

U ovom poglavlju bit će predstavljeno rješenje aplikacije za prilagodljivo prikazivanje turističkih sadržaja kao i korištene tehnologije i alati koji su omogućili implementaciju rješenja. Aplikacija je podijeljena na osnovne dijelove, funkcionalni dio koji je zadužen za prikazivanje vremenskih uvjeta i aktivnosti koje su pohranjene u bazi te grafičko sučelje koje služi za prikaz informacija i interakciju s korisnikom.

4.1. Korištene tehnologije i alati

Za ostvarenje rješenja i izradu aplikacije, korišteno je nekoliko tehnologija i alata, koje su podijeljene u skupine ovisno o tome radi li se o funkcionalnom dijelu aplikacije ili korisničkom sučelju.

4.1.1. Tehnologije za implementaciju funkcionalnog dijela aplikacije

Za ostvarenje funkcionalnog dijela aplikacije koje uključuje dohvaćanje podataka o trenutnoj vremenskoj prognozi, komuniciranje s lokalnim poslužiteljem i bazom podataka te implementaciju algoritma odlučivanja, korištene su sljedeće tehnologije:

- AngularJS
- PHP
- Baza podataka - MySql
- HTTP i JSON
- MAMP

AngularJS je klijentski i poslužiteljski okvir temeljen na JavaScript-u, a služi za izradu web aplikacija. Razvijen je od strane Google-a, a prva inačica je bila javno dostupna 2010. godine. Svrha mu je pojednostaviti razvoj aplikacija pružajući okvir za klijentsku MVC (engl. *Model View Controller*) i MVVM (engl. *Model View Viewmodel*) arhitekturu. AngularJS prihvaća i proširuje tradicionalni HTML kako bi prikazao dinamički sadržaj pomoću dvosmjerne podatkovne veze (engl. two-way data-binding) koja omogućuje automatsku sinkronizaciju između modela [10].

PHP je poslužiteljski skriptni jezik koji se koristi u razvoju web aplikacija, ali se također koristi kao programski jezik opće namjene. Prvotno je razvijen od strane Rasmusa Lerdorfa 1994.

godine, a trenutno ga održava The PHP Group. Kratica PHP stoji za PHP:Hypertext Preprocessor. PHP kod može biti ugrađen unutar HTML-a ili može biti korišten u kombinaciji s web okvirnim sustavima ili sustavima za upravljanje sadržajem (engl. *CMS - Content Management System*), kao što je primjerice Wordpress. Standardni PHP prevoditelj (engl. *interpreter*) je pogonjen Zend Engine-om, koji je skriptni *engine* otvorenog koda. PHP može biti postavljen većini web poslužitelja te na gotovo svakom operacijskom sustavu i platformi [11].

MySql je besplatan sustav otvorenog koda za upravljanje bazama podataka. MySql baze i tablice su relacijskog tipa, što je pogodno za skladištenje i pretraživanje velikih količina podataka te predstavljaju osnovu svakog informacijskog sustava.

HTTP (engl. *HyperText Transfer Protocol*) predstavlja najčešću metodu prijenosa podataka i informacija na internetu. Radi se o protokolu zahtjev/odgovor (engl. *request/response*) za komunikaciju između poslužitelja (engl. *server*) i klijenta (engl. *client*). HTTP klijent najčešće inicira prijenos podataka nakon što uspostavi vezu sa poslužiteljem. Poslužitelj konstantno osluškuje zahtjeve na određenom komunikacijskom pristupu (najčešće port 80), čekajući da klijent pošalje niz znakova kao što je "GET / HTTP/1.1" kojim će zahtijevati uspostavljanje komunikacije.

Format JSON (engl. *JavaScript Object Notation*) je format podataka otvorenog standarda koji služi za slanje podataka u obliku teksta koje je čovjeku lako čitljiv. JSON je najčešći oblik podataka koji se koristi kod asinkrone komunikacije između poslužitelja i klijenta, a već je uvelike zamijenio XML format podataka.

MAMP je programski alat koje na računalo postavlja okruženje za lokalni poslužitelj. Pokretanjem poslužitelja pokreću se Apache i MySql poslužitelji koji su potrebni za bazu podataka i izvršavanje php skripti [12].

4.1.2. Tehnologije za implementaciju sučelja aplikacije

Za izradu sučelja aplikacije koje podrazumijeva implementaciju predloženog dizajna sučelja i raspored elemenata korištene su sljedeće tehnologije:

- HTML (engl. *Hyper Text Markup Language*)
- CSS (engl. *Cascading Style Sheets*)
- Bootstrap okvir
- Meteocons ikone za grafički prikaz vremenskih uvjeta
- FontAwesome font za ikone na sučelju

HTML je prezentacijski jezik za izradu web stranica, a prikaz hipertekst dokumenata je omogućen pomoću web preglednika. Važno je napomenuti kako HTML nije programski jezik, već služi samo za opis izgleda internetske stranice. Osnovni građevni elementi su oznake (engl. *tags*), pomoću kojih se opisuje kako će se nešto prikazati u web pregledniku te se definira struktura web stranice.

CSS je stilski jezik koji služi za opisivanje i definiranje izgleda dokumenta koji je napisan u prezentacijskom jeziku. Najčešće se koristi za definiranje vizualnog izgleda web stranica koje su napisane u HTML-u ili XHTML-u (engl. *Extensible Hypertext Markup Language*), no također se može koristit za definiranje izgleda bilo kojeg XML (engl. *Extensible Markup Language*) dokumenta, uključujući SVG (engl. *Scalable Vector Graphics*) i XUL (engl. *XML User Interface Language*).

Bootstrap okvir [13] je jedan od najpopularnijih HTML, CSS i JavaScript okvira koji služi za razvoj i dizajn prilagodljivih web stranica. Bootstrap se sastoji od gotovih predložaka, između ostalog, predložaka za forme, gumbe, navigaciju te ostalih komponenti sučelja. Nastao je 2011. godine u tvrtci Twitter, a glavni ljudi koji su radili na njegovom razvoju su Mark Otto i Jacob Thornton. U 2012. godini izašla je Bootstrap 2 inačica, čiji izgled se temeljio na mreži od 12 stupaca i komponenti za prilagodljivi dizajn. Prilagodljivi dizajn podrazumijeva da se izgled i raspored komponenti na sučelju prilagođava dinamički, uzimajući u obzir karakteristike uređaja na kojem se pregledava (mobilni uređaj, tablet računalo ili stolno računalu), odnosno prilagodba se vrši na temelju rezolucije ekrana na kojoj se sadržaj gleda. Važno je napomenuti kako je Bootstrap kompatibilan sa zadnjim inačicama svih popularnijih web preglednika (Google Chrome, Mozilla Firefox, Safari, Opera i tako dalje).

Za upotpunjavanje sučelja korištene su Meteocons [14] ikone koje su implementirane u obliku fonta, a služe za grafički prikaz trenutnog vremenskog stanja, kao i FontAwesome font [15] koji nudi razne ikone i znakove, a dio su sučelja i olakšavaju interakcijske radnje korisnika i sučelja.

4.2. Implementacija funkcionalnog dijela aplikacije

Funkcionalni dio aplikacije podijeljen je na dva glavna dijela, prvi dio koji obavlja zadaću prikupljanja informacija o vremenskoj prognozi, a drugi dio je zadužen za komunikaciju s poslužiteljem odnosno lokalnom bazom podataka, dok između njih stoji algoritam odlučivanja.

Programski kôd i sve funkcije koje su zadužene za funkcionalni dio aplikacije nalaze se unutar *app.js* dokumenta. Struktura HTML dokumenta je standardna, uz dodane AngularJS direktive.

Element *body* sadrži *ng-app* direktivu koja označava da je taj element "vlasnik" AngularJS aplikacije, kako je navedeno u programskom kôdu 4.1.

```
<body ng-app="WeatherApp">
```

Programski kôd 4.1 Deklaracija *ng-app* direktive

Direktiva *ng-app* ima vrijednost "*WeatherApp*", čime se kreira modul (engl. *module*) pod istim tim nazivom. Modul definira aplikaciju te sadrži različite dijelove aplikacije, a također sadrži i kontrolere.

Nakon toga, definira se kontroler (engl. *controller*) koji kontrolira podatke aplikacije, a zapravo se radi o JavaScript objektu koji se kreira sa standardnim JavaScript objektnim konstruktorom prema programskom kôdu 4.2.

```
<div ng-controller="MainController as main" ng-init="main.initCity()">
```

Programski kôd 4.2 Deklaracija *ng-controller* direktive

Kontroler je naziva MainController i postavljen je kao glavni kontroler (main), također unutar ovog div elementa je i *ng-init* direktiva koja pokreće funkciju *initCity()* koja je definirana u *app.js* dokumentu.

Ove dvije direktive čine glavni dio aplikacije, uz napomenu kako je potrebno unutar HTML dokumenta uključiti biblioteke koje su nužne za funkcioniranje aplikacije AngularJS.

Nakon što su definirane ove dvije direktive, u *app.js* dokumentu je potrebno definirati modul i kontroler koji zapravo pripada modelu. Jedan od parametara kontrolera je i *scope*, koji predstavlja poveznicu između HTML dijela, koji zapravo predstavlja *view* te JavaScript dijela, odnosno kontrolera. *Scope* je objekt sa svojim svojstvima i metodama, a dostupan je kako *view*-u, tako i kontroleru, što omogućuje da na lak način pristupamo vrijednostima iz kontrolera i prikazujemo ih u HTML dijelu, odnosno na sučelju.

Unutar *app.js* dokumenta potrebno je definirati modul i kontroler unutar kojeg se pišu sve funkcije koje će se izvršavati prilikom pokretanja aplikacije. Slika 4.1 prikazuje sve funkcije koje se nalaze unutar glavnog kontrolera i koje su zadužene za funkcionalni dio aplikacije.

```
vm.initCity = function () {};

vm.search = function(query) {};

vm.weather = {
  current: function() {}
};

vm.weatherAlgoritham = function(){};

vm.queryGoodWeather = function(){};

vm.queryBadWeather = function(){};

vm.showIcon = function() {};

vm.changeColour = function (color) {};

var round = function(num) {};

vm.kelvinToCelcius = function(kelvin) {};
```

Sl 4.1 Lista funkcija unutar glavnog kontrolera

Detaljan opis pojedine funkcije bit će predstavljen u nastavku.

4.2.1. Prikupljanje podataka o vremenskim uvjetima

Kako je već prije spomenuto, informacije o vremenskoj prognozi pruža OpenWeatherMap API. Kako bi se API mogao koristiti, potrebno je napraviti korisnički račun na njihovoj web stranici. Nakon registracije, potrebno je putem sučelja zatražiti jedinstveni API ključ (engl. API Key), kojeg je potrebno koristiti prilikom same implementacije aplikacije, a služi za slanje zahtijeva na njihov poslužitelj te dohvaćanje vremenske prognoze. Funkcije koje su zadužene za prikupljanje podataka o vremenskoj prognozi i njihovu obradu, odnosno, komuniciranje sa OpenWeatherMap bazom podataka su *initCity()*, *search()* i *weather.current()*.

Funkcija *initCity()* je funkcija koja se pokreće pomoću *ng-init* direktive, odnosno, to je glavna funkcija koja se inicijalizira prilikom pokretanja aplikacije. U toj funkciji se definira lokacija, odnosno grad za koji se želi prikazivati vremenska prognoza, u ovom slučaju to je grad Osijek. Također, unutar funkcije *initCity()* se poziva funkcija *search()* kojoj se kao parametar predaje željena lokacija, odnosno grad.
Funkcija *search()* kao parametar drži lokaciju (grad - query) koji je predan od strane *initCity()* funkcije. Formira se URL (engl. *Uniform Resource Locator*) kojim će se izvesti zahtjev za dohvaćanjem vremenske prognoze. URL je definiran kao varijabla, a oblik je naveden u programskom kôdu 4.3.

```
_apiUrlCurrent= 'http://api.openweathermap.org/data/2.5/weather?q=' + query + '&APPID='
+ _apiKey + ‘’;
```

Programski kôd 4.3 Definiranje URL-a pomoću varijable

Početni dio je definiran dokumentacijom OpenWeatherMap API-ja, te je on fiksan, a zatim mu se dodaje parametar *query* koji predstavlja lokaciju, a proslijeđen je od *initCity()* funkcije, nakon toga slijedi APPID parametar koji predstavlja *API Key*, čiji je postupak generiranja prethodno opisan. *API Key* je definiran kao varijabla u programskom kôdu 4.4.

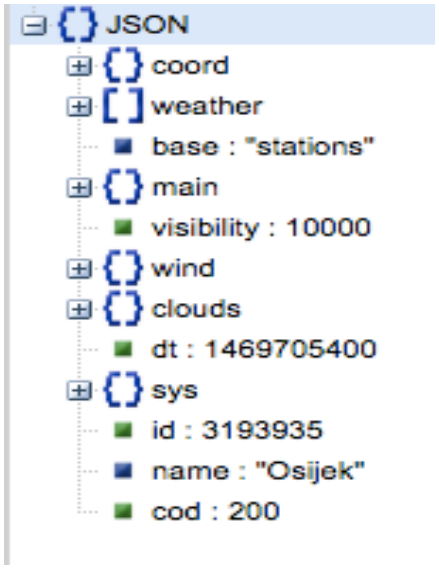
```
var _apiKey = ‘91d0e2598a2910e37b5ab91479d10a38';
```

Programski kôd 4.4 Definiranje *API Key-a* pomoću varijable

Nakon postavljanja URL-a, unutar funkcije *search()* poziva se funkcija *weather.current().*

Funkcija *weather.current()* šalje HTTP (engl. *HyperText Transfer Protocol*) zahtjev s parametrom URL-a koji je definiran u prethodnoj funkciji. U slučaju da primi podatke u JSON formatu, sprema ih u polje *dataCurrent*. U protivnom slučaju, ukoliko dođe do greške, u konzolu ispisuje do koje je HTTP greške došlo. Kada funkcija uspješno zaprimi podatke, pozivaju se funkcije *weatherAlgoritham()* i *showIcon().* Podaci koji su zaprimljeni su u JSON formatu, a slika 4.2 prikazuje primjer strukture podataka.


```
{
  "coord": {
    "lon": 18.69,
    "lat": 45.55
  },
  "weather": [
    {
      "id": 802,
      "main": "Clouds",
      "description": "scattered clouds",
      "icon": "03d"
    }
  ],
  "base": "stations",
  "main": {
    "temp": 297.3,
    "pressure": 1016,
    "humidity": 54,
    "temp_min": 295.15,
    "temp_max": 300.15
  },
  "visibility": 10000,
  "wind": {
    "speed": 4.6,
    "deg": 340
  },
  "clouds": {
    "all": 40
  },
  "dt": 1469705400,
  "sys": {
    "type": 1,
    "id": 5455,
    "message": 0.0316,
    "country": "HR",
    "sunrise": 1469676315,
    "sunset": 1469729848
  },
  "id": 3193935,
  "name": "Osijek",
  "cod": 200
}
```

Sl 4.2 Struktura podataka u JSON formatu koji je dobiven kao odgovor od OpenWeatherMap poslužitelja

Vidljivo je kako se u odgovoru OpenWeatherMap poslužitelja nalazi veća količina podataka. Neki od podataka se prikazuju na sučelju aplikacije, kao što su temperatura zraka, brzina vjetra, vrijeme izlaska i zalaska Sunca, dok se neki podaci prosljeđuju algoritmu za odlučivanje koji ih zatim obrađuje.

S obzirom da funkcionalnost aplikacije ovisi o uspješnom dohvaćanju podataka o vremenskoj prognozi, potrebno je testirati ispravnost funkcije *weather.current().* S obzirom da se koristi metoda slanja HTTP zahtijeva, unutar same funkcije je implementiran ispis stanja u konzolu. Ukoliko su podaci uspješno zaprimljeni, spremaju se u polje *dataCurrent*, ali se također ispisuju u konzolu. Ukoliko dođe do pogreške, nju je također potrebno ispisati u konzolu, kako bi se znalo o kojoj pogrešci je riječ, te ju eventualno ispraviti. Programski kôd 4.5 prikazuje ispis podataka u konzolu koji se nalazi unutar funkcije *weather.current()* ukoliko su podaci uspješno zaprimljeni.

```
$log.debug("Current weather data: %O", data);
```

Programski kôd 4.5 Ispis podataka u konzolu

No, ukoliko podaci nisu mogli biti zaprimljeni i došlo je do pogreške, tu pogrešku je potrebno obraditi i ispisati u konzolu, kako je prikazano programskim kôdom 4.6.

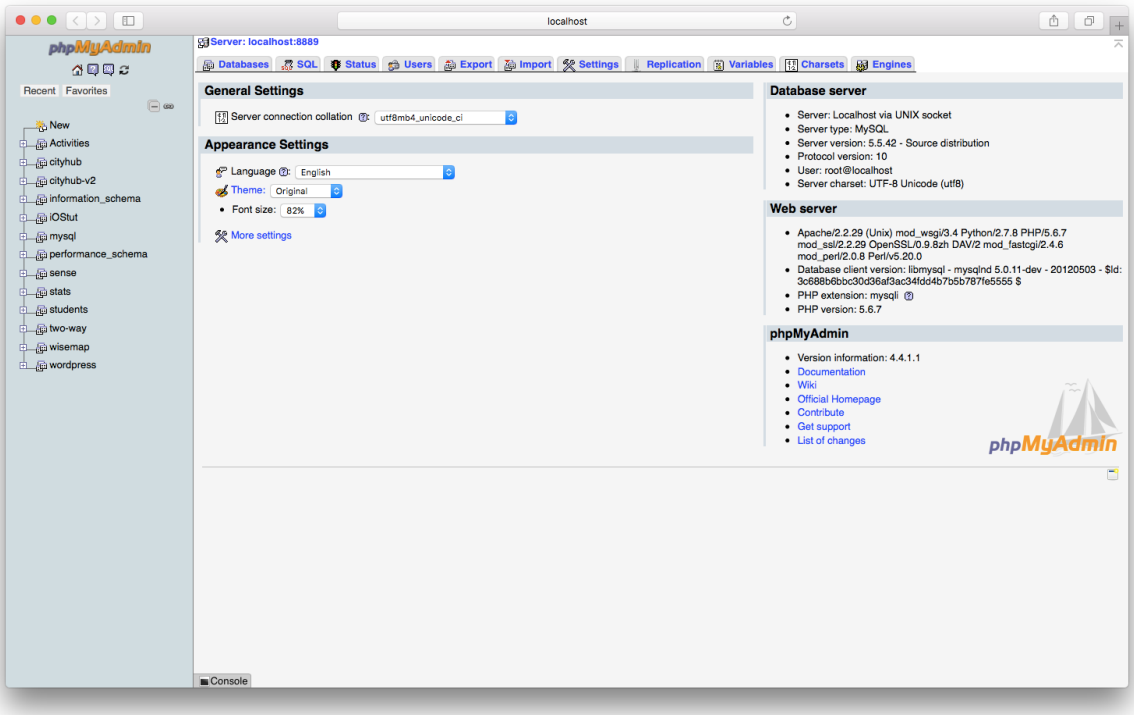
```
$log.error("HTTP error: ", err);
```

Programski kôd 4.6 Ispis pogreške u konzolu

Rukovanje pogreškama na ovaj način omogućuje uvid i rješavanje grešaka ukoliko je do njih došlo.

4.2.2. Model lokalne baze podataka

Baza podataka se nalazi na lokalnom poslužitelju koji se konfigurira pomoću MAMP programskog alata. MAMP stvara poslužiteljsko okruženje i time pokreće Apache poslužitelj i MySQL poslužitelj. Bazi podataka se pristupa pomoću web sučelja MAMP programskog alata otvarajući Tools > phpMyAdmin. Unutar phpMyAdmin sučelja moguće je vidjeti već kreirane baze podataka s pripadajućim tablicama te ih je također moguće uređivati i/ili kreirati nove baze podataka. Sučelje phpMyAdmin alata je vidljivo na slici 4.3.



Sl. 4.3 Sučelje alata phpMyAdmin

Lijevo na slici 4.3 se mogu uočiti sve baze podataka koje se nalaze na lokalnom poslužitelju. Aktivnosti su spremljene u *Activities* bazu podataka unutar koje se nalazi *activity* tablica. Struktura *activity* tablice je vidljiva sa slike 4.4.

	#	Name	Type	Collation	Attributes	Null	Default	Extra	Action
☐	1	**Name**	varchar(256)	utf8_unicode_ci		No	*None*		Change Drop Primary Unique Index Spatial Fulltext Distinct values
☐	2	**Info**	varchar(256)	utf8_unicode_ci		No	*None*		Change Drop Primary Unique Index Spatial Fulltext Distinct values
☐	3	**Latitude**	varchar(256)	utf8_unicode_ci		No	*None*		Change Drop Primary Unique Index Spatial Fulltext Distinct values
☐	4	**Longitude**	varchar(256)	utf8_unicode_ci		No	*None*		Change Drop Primary Unique Index Spatial Fulltext Distinct values
☐	5	**Weather**	int(11)			No	*None*		Change Drop Primary Unique Index Spatial Fulltext Distinct values
☐	6	**Id**	int(11)			No	*None*		Change Drop Primary Unique Index Spatial Fulltext Distinct values
☐	7	**Image**	varchar(256)	utf8_unicode_ci		No	*None*		Change Drop Primary Unique Index Spatial Fulltext Distinct values

Sl. 4.4 Struktura tablice *activity* unutar *Activities* baze podataka

Unutar *activity* tablice nalaze se slijedeći redci:

Name - ime aktivnosti

Info - dodatni tekst uz aktivnost

Latitude - zemljopisna širina lokacije na kojoj se aktivnost nalazi

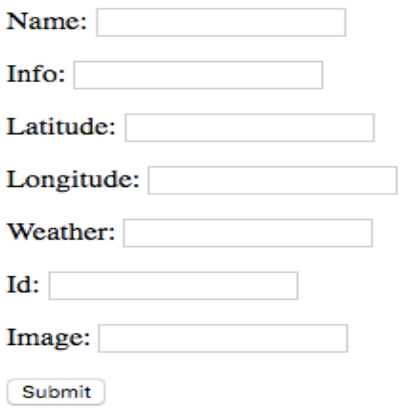
Longitude - zemljopisna dužina lokacije na kojoj se aktivnost nalazi

Weather - ključ po kojem se iz baze povlače aktivnosti (100 - označava aktivnosti za povoljne vremenske uvjete, 200 - označava aktivnosti za nepovoljne vremenske uvjete)

Id - identifikator aktivnosti

Image - putanja do fotografije koja je spremljena lokalno

Unos podataka u bazu vrši se pomoću jednostavnog web sučelja koje je definirano kao forma unutar *insert.html* dokumenta. Forma sadrži polja za unos podataka koja odgovaraju redovima tablice *activity*, a vidljiva je na slici 4.5.



Sl. 4.5 Forma za unos aktivnosti u bazu podataka

Pritiskom na "*Submit*" dugme, poziva se skripta *store.php* u kojoj su definirani podaci za spajanje na bazu podataka te unos podataka unesenih u formi. Važno je napomenuti kako forma za unos podataka nije namijenjena krajnjem korisniku aplikacije već samo služi kako bi se baza podataka dodatno proširivala s novim aktivnostima koje može vršiti ovlaštena osoba.

Algoritam odlučivanja nalazi se unutar funkcije *weatherAlgoritham(),* te će detaljan opis biti u narednom odjeljku, no kako je algoritam povezan s lokalnom bazom podataka, funkcije koje se pozivaju unutar njega navedene su u narednom tekstu. Ovisno o odluci algoritma, pozivaju se funkcije *queryGoodWeather()* ukoliko je riječ o povoljnim vremenskim uvjetima, odnosno *queryBadWeather()* ukoliko je riječ o nepovoljnim vremenskim uvjetima.

Funkcija *queryGoodWeather()* šalje HTTP zahtjev pozivajući *getGood* php skriptu unutar koje se nalaze podaci i funkcija za spajanje na bazu podataka i dohvaćanje podataka. Ukoliko se uspješno spoji na bazu, šalje upit u bazu kako je navedeno u programskom kôdu 4.7.

```
$query = "SELECT * FROM activity WHERE Weather='100'";
```

Programski kôd 4.7 Upit u bazu podataka po definiranom ključu

Rezultat upita se sprema unutar polja *$query*. Funkcija *queryGoodWeather()* zatim te podatke šalje na sučelje aplikacije.

Funkcija *queryBadWeather()* radi na isti način, a jedina razlika je što šalje drugačiji upit u bazu, kako je vidljivo u programskom kôdu 4.8.

```
$query = "SELECT * FROM activity WHERE Weather='200'";
```

Programski kôd 4.8 Upit u bazu podataka po definiranom ključu

Kako je ranije spomenuto, redak *Weather* je ključ po kojem se razlikuju aktivnosti koje su pogodne za povoljne vremenske uvjete, odnosno za nepovoljne vremenske uvjete, a može imati vrijednost 100 ili 200.

Nakon što funkcije *queryGoodWeather()* i *queryBadWeather()* pomoću HTTP poziva pozovu odgovarajuće php skripte, podatke koje one proslijede ispisuju se u konzolu radi njihove provjere. Na slici 4.6 vidi se primjer ispisa podataka u konzolu iz funkcije *queryGoodWeather()*.

Sl. 4.6 Ispis podataka u konzolu iz funkcije *queryGoodWeather()*

Na lijevoj strani slike 4.6 vidi se da je funkciji proslijeđeno polje unutar kojeg se nalazi 10 objekata, pri čemu svaki objekt predstavlja jednu aktivnost s atributima iz baze podataka. Na desnoj strani se vidi detaljniji pregled pojedinog objekta unutar polja, odnosno, vide se svi atributi jedne aktivnosti.

Unutar php skripti koje služe za dohvaćanje podataka (*getGood.php* i *getBad.php*), također postoje metode rukovanja pogreškama. Prvo je potrebno provjeriti je li se moguće spojiti na poslužitelj, a zatim na samu bazu podataka. U oba slučaja, ako dođe do pogreške, nju je potrebno ispisati u konzolu kako bi se dalje rukovalo s njom. Ukoliko je spajanje uspješno i podaci su

dohvaćeni, kako bi se uvjerili u njihovu ispravnost, podaci se ispisuju u JSON formatu kako je navedeno u programskom kôdu 4.9.

```
echo json_encode($records);
```

Programski kôd 4.9 Ispis podataka u JSON formatu

Navedene metode olakšavaju rješavanje eventualnih pogrešaka, ali također olakšavaju rukovanje podacima, jer se ispisom u konzolu dobiva uvid o kakvim podacima se radi te u kojem formatu se nalaze, odnosno, jesu li ispravno zaprimljeni. Na taj način se osigurava da korisnik ima ispravne podatke prikazane na sučelju.

4.2.3. Implementacija algoritma za odlučivanje

Pregledom dokumentacije OpenWeatherMap API-ja i pregledavanjem službenih primjera koji su ponuđeni, viđeno je da odgovor koji nudi API sadrži detaljne podatke, te su pojedini vremenski uvjeti podijeljeni na temelju id-a (identifikatora). Ta činjenica će uvelike olakšati razvoj algoritma koji će odlučivati kada su vremenski uvjeti povoljni za vanjske aktivnosti, odnosno kada su uvjeti nepovoljni i kada je potrebno na sučelju aplikacije prikazati unutarnje aktivnosti. Vremenski uvjeti su podijeljeni u glavne kategorije unutar kojih postoji detaljnija raspodjela.

Algoritam se temelji na identifikatoru vremenskih uvjeta (primjerice, identifikator koji označuje da je trenutno sunčano ili da je grmljavinsko nevrijeme u tijeku), a dodatno se koristi i trenutna temperatura zraka kako bi se izbjegle vanjske aktivnosti ako je vani hladno. S obzirom da aplikacija pruža prijedloge aktivnosti, algoritam također ispisuje poruku na sučelje u određenim okolnostima, primjerice ako je vani sunčano, a temperatura je visoka, postoji opasnost od toplinskog udara.

Funkcija *weatherAlgoritham()* je zapravo zadužena za odlučivanje o kakvim se vremenskim uvjetima radi, odnosno jesu li oni povoljni ili nepovoljni. Unutar funkcije definirane su dvije varijable. Prva varijabla je trenutna temperatura, a druga varijabla predstavlja trenutno vremensko stanje, odnosno njezin identifikator (i*d*), kako je navedeno u programskom kôdu 4.10.

```
var temp = vm.kelvinToCelcius(vm.dataCurrent.main.temp);
var condition = vm.dataCurrent.weather[0].id;
```

Programski kôd 4.10 Definiranje varijabli za temperaturu i vremensko stanje

Kako bi se dohvatila trenutna temperaturu, potrebno je pristupiti polju *dataCurrent* u kojemu je spremljen odgovor od poslužitelja, odnosno, spremljeni su podaci u JSON formatu. Kako se vidi na slici 4.2, polje se sastoji od dodatnih polja, a temperatura se nalazi unutar *main*, stoga unutar *dataCurrent* sa operatorom "." prvo se pristupa *main*, a zatim *temp* kako bi se dohvatila temperatura. Može se uočiti da je temperatura izražena u Kelvinima, stoga se predaje funkciji *kelvinToCelzius()* koja će napraviti konverziju u Celzijeve stupnjeve. Na isti način se pristupa i identifikatoru trenutnih vremenskih uvjeta (*id*).

Kako bi se znalo značenje pojedinog identifikatora (*id*-a) potrebno se referencirati u dokumentaciju OpenWeatherMap API-ja gdje je detaljno izlistano te opisano značenje svakog identifikatora koji se može pojaviti, a jedan je od primjera opisan tablicom 4.1. Tablica prikazuje listu identifikatora (*id*) za grmljavinsko nevrijeme te opis i značenje pojedinog identifikatora. Preostale tablice s drugim vremenskim uvjetima i njihovim identifikatorima se mogu pronaći u službenoj dokumentaciji prema [16]. Vremenske prilike su podijeljene u osnovne kategorije kao što su kiša, snijeg, oblačno, a zatim se kategorije granaju na detaljnije opise vremenskih prilika.

Tab 4.1 Tablica s identifikatorima za grmljavinsko nevrijeme i opis pojedinog identifikatora

Id	Meaning
200	thunderstorm with light rain
201	thunderstorm with rain
202	thunderstorm with heavy rain
210	light thunderstorm
211	thunderstorm
212	heavy thunderstorm
221	ragged thunderstorm
230	thunderstorm with light drizzle
231	thunderstorm with drizzle
232	thunderstorm with heavy drizzle

Prilikom odlučivanja jesu li vremenske prilike povoljne ili nepovoljne, dovoljno je uzeti u obzir samo glavne kategorije vremenskih uvjeta koje su predstavljene u tablici 4.2.

Tab 4.2 Glavne kategorije vremenskih uvjeta OpenWeatherMap baze podataka

ID	Category
2xx	Thunderstorm
3xx	Drizzle
5xx	Rain
6xx	Snow
800	Clear
80x	Clouds

Tablica 4.2 prikazuje popis glavnih kategorija vremenskih uvjeta i njihove identifikatore pri čemu valja naglasiti da identifikator svake kategorije započinje drugim brojem, što olakšava filtriranje i odlučivanje o kojoj kategoriji je riječ, odnosno o kakvim vremenskim uvjetima se radi. Budući da kategorije olakšavaju odlučivanje, algoritam je implementiran pomoću *if/else* petlje, pri čemu se provjerava varijabla *condition*, odnosno o kojoj kategoriji je riječ te se u nekim slučajevima dodatno provjerava i varijabla *temp*, odnosno temperatura.

Na sučelju aplikacije se prikazuje obavijest, odnosno upozorenje, za što je potrebno provjeravati trenutnu temperaturu. Primjerice, ukoliko je vani sunčano bez oblaka, a ujedno je i temperatura veća od 26°C, postoji opasnost od vrućine i izlaganja Suncu, tako da će se na sučelju ispisati adekvatna poruka: *„Sky is clear, but temperature is high, so take care of Sun”.*

Ako je vani sunčano i bez oblaka, a temperatura je između 15°C i 26°C, tada je vrijeme idealno za vanjske aktivnosti, no ako je temperatura manja od 15°C, tada postoji mogućnost da je vani hladno, što je popraćeno porukom: *„Sky is clear, but it can be chilly outside”*.

Funkcioniranje algoritma temelji se na provjeri vrijednosti *condition* varijable koja sadržava vrijednost identifikatora trenutnog vremenskog stanja te provjere trenutne temperature zraka koja je sadržana unutar varijable *temp.* Referencirajući se na tablicu 4.2, algoritam odlučuje radi li se o povoljnim vremenskim uvjetima ili nepovoljnim vremenskim uvjetima. Povoljni vremenski uvjeti se smatraju oni s identifikatorima koji počinju brojevima 800(Clear - sunčano) te 80x(Clouds - oblaci). Nepovoljni vremenski uvjeti se smatraju oni s identifikatorima koji počinju brojevima 2xx(Thunderstorm – grmljavinsko nevrijeme), 3xx(Drizzle - kiša), 5xx(Rain – kiša) i 6xx(Snow - snijeg). Provjera temperature isključivo se vrši za točnost poruke koja se ispisuje na sučelju aplikacije.

U tablici 4.3 bit će ukratko prikazano kako algoritam funkcionira navodeći vrijednosti *condition* varijable koja označava identifikator vremenskih uvjeta, na koje se može referencirati u tablici 4.2 te vrijednosti *temp* varijable, koja označava trenutnu temperaturu, u slučaju da ju je potrebno provjeravati. Unutar *weatherAlgoritham()* funkcije pozivaju se *queryGoodWeather()* i *queryBadWeather()* funkcije.

Tab 4.3 Algoritam (*weatherAlgoritham()* funkcija) koji odlučuje jesu li vremenski uvjeti povoljni ili nepovoljni

condition	**temp**	**Poziv funkcije**	**Poruka**
== 800	>= 26	queryGoodWeather()	Sky is clear, but temperature is high, so

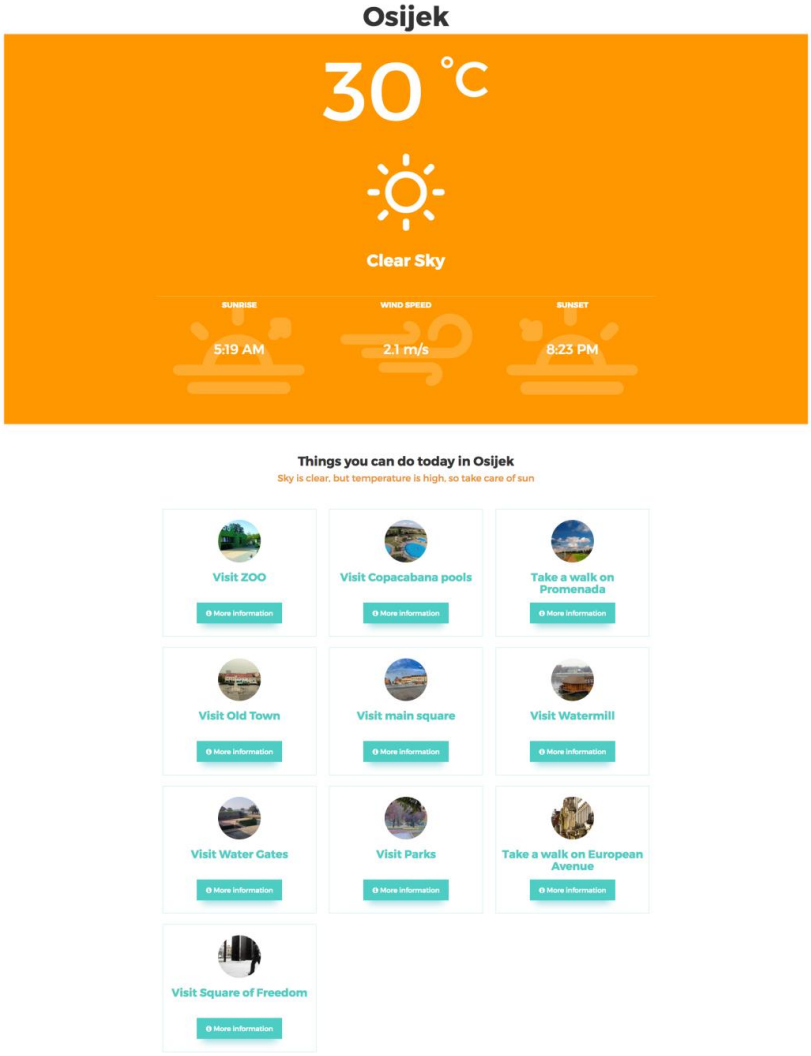
			take care of Sun
== 800	<=25 && >=15	queryGoodWeather()	Sky is clear, and temperature is ideal, enjoy
== 800	<= 14	queryGoodWeather()	Sky is clear, but it can be chilly outside
>= 801 && <= 804	>= 26	queryGoodWeather()	There are few clouds, but temperature is high, so take care of Sun
>= 801 && <= 804	<=25 && >=15	queryGoodWeather()	There are few clouds and temperature is ideal, enjoy
>= 801 && <= 804	<=14	queryGoodWeather()	There are few clouds and it can be chilly outside
>= 300 && <= 531	-	queryBadWeather()	Its raining, so enjoy in indoor activities
>= 600 && <= 622	-	queryBadWeather()	Its snowing, so enjoy in indoor activities
>= 200 && <= 232	-	queryBadWeather()	Its thunderstorm outside, so consider not going out

U tablici 4.3, unutar stupca "Poziv funkcije" moguće je vidjeti kada su vremenski uvjeti povoljni, odnosno kada su nepovoljni te sukladno tome, pozivaju se funkcije *queryGoodWeather()* i *queryBadWeather().*

Funkcije *queryGoodWeather()* i *queryBadWeather()* služe za dohvaćanje podataka iz baze podataka u kojoj su spremljene aktivnosti, a objašnjene su u odjeljku 4.2.2. Na ovaj način programski je implementiran algoritam odlučivanja kako je navedeno u modelu. Algoritam rukuje s podacima dobivenim od OpenWeatherMap poslužitelja i na temelju njih odlučuje o povoljnim i nepovoljnim vremenskim uvjetima i sukladno tome prikazuju se podaci na sučelju aplikacije.

4.3. Implementacija sučelja aplikacije

Sučelje aplikacije podijeljeno je na dva glavna dijela - dio koji pruža informacije o vremenskim uvjetima te dio koji izlistava prijedlog aktivnosti, pri čemu je inicijalni prijedlog dizajna sučelja u manjoj mjeri izmijenjen. Slika 4.6 prikazuje sučelje aplikacije za prilagodljivi prikaz turističkih sadržaja.



Sl. 4.6 Sučelje web aplikacije za prilagodljivo prikazivanje turističkih sadržaja

Gornji dio sučelja prikazuje informacije o trenutnim vremenskim uvjetima, a donji dio sučelja prikazuje listu aktivnosti koje se predlažu na temelju vremenskih uvjeta. U narednom tekstu će biti opisana ta dva dijela sučelja i njihove komponente.

4.3.1. Prikazivanje elemenata korisničkog sučelja

Nakon što su u funkcionalnom dijelu prikupljeni i obrađeni svi podaci, potrebno ih je prikazati na sučelju aplikacije. Podatke je potrebno poslati iz funkcionalnog dijela na sučelje aplikacije, što se vrši pomoću parametra *scope*, kako je već spomenuto u potpoglavlju 4.2. *Scope* je svojevrsna poveznica između HTML dijela, koji zapravo predstavlja *view* te JavaScript dijela, odnosno kontrolera u kojoj se nalaze sve funkcije i podaci potrebni za prikaz. *Scope* je objekt sa svojstvima i metodama, a dostupan je kako *view*-u (HTML dio), tako i kontroleru, što omogućuje da

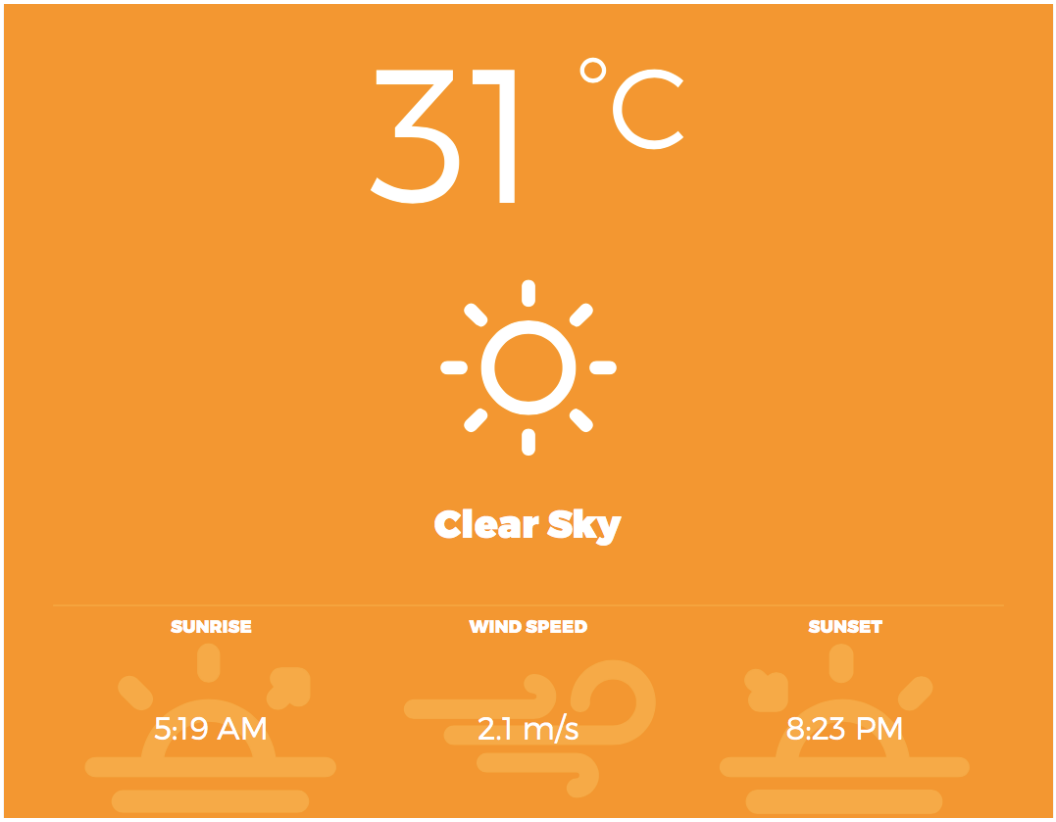
se na jednostavan način pristupa vrijednostima iz kontrolera i prikazuje ih u HTML dijelu, odnosno na sučelju, a jedan od primjera prikazan je programskim kôdom 4.11.

```
<p class="weather-now">{{ main.dataCurrent.weather[0].description }}</p>
```

Programski kôd 4.11 Ispis podataka iz kontrolera na sučelju aplikacije

Prethodno je naveden primjer prikazivanja vrijednosti iz kontrolera na sučelju aplikacije. Ovdje se prikazuje opis vremenskog stanja, kako je od prije poznato, podaci su spremljeni u *dataCurrent* polje, a pojedinom podatku pristupa se na isti način kao i u funkcionalnom dijelu, odnosno kontroleru. Navedeni podatak se prikazuje unutar „*p*“ HTML elementa, odnosno paragrafa, kojemu je moguće dodijeliti klasu te ga stilizirati pomoću CSS-a.

Slika 4.7 pruža detaljniji prikaz gornjeg dijela sučelja gdje se ispisuju trenutni vremenski uvjeti. Na sučelju je prikazana informacija o temperaturi zraka, kratki opis stanja popraćen sa ikonom te informacije o brzini vjetra, vremenu izlaska i zalaska Sunca. Za ovaj dio sučelja vezane su funkcije *showIcon()* koja je zadužena za prikaz prigodne ikone, te *changeColor()* funkcija koja služi za promjenu boje pozadine ovog dijela sučelja kako bi se dodatno dočarali vremenski uvjeti.



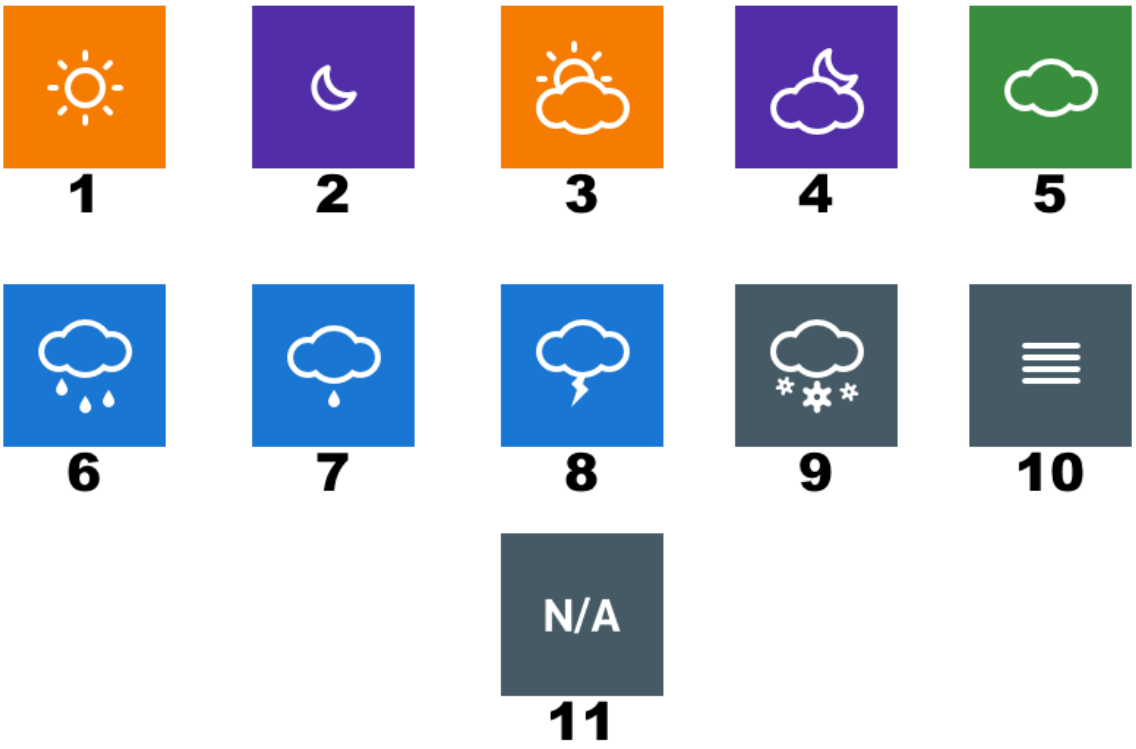
Sl. 4.7 Dio sučelja web aplikacije za prikaz trenutnih vremenskih uvjeta

Dio sučelja prikazan na slici 4.7 zadužen je za pružanje informacija o vremenskim uvjetima, a njegova glavna funkcionalnost se temelji na OpenWeatherMap API-ju sa kojega dolazi vremenska prognoza. Kako bi se vizualna informacija upotpunila, osim prigodnih ikona koje predstavljaju trenutno vremensko stanje, dodatno je implementirana dinamička izmjena pozadine ovisno o vremenu.

Funkcija *showIcon()* služi za prikaz ikone na sučelju koja opisuje vremensko stanje. OpenWeatherMap API pruža svoje ikone, no u ovome slučaju one su zamijenjene sa Meteocons fontom. Unutar odgovora koji je dobiven sa OpenWeatherMap poslužitelja nalazi se naziv ikone koja odgovora vremenskom stanju (pogledati sliku 4.2). Kako je odgovor spremljen unutar *dataCurrent* polja, pristupanje ikoni vrši se slično kao pristupanje identifikatoru ili temperaturi. Pomoću *switch/case* izjava, prolazi se kroz sve ikone koje se mogu pojaviti ovisno o vremenu, te se one zamjenjuju adekvatnom ikonom iz Meteocons fonta. Važno je napomenuti kako Metecons font ima definirana odgovarajuća slova za pojedine ikone, prema [11].

Funkcija *changeColor()* služi za promjenu boje pozadine sučelja, a poziva se unutar funkcije *showIcon()* ovisno o kakvim vremenskim uvjetima se radi. Ova funkcija prima jedan parametar, a to je boja. Ovaj parametar se šalje na sučelje, a zatim se postavlja kao css klasa odjeljka u kojem je definiran izgled i komponente sučelja. Boje su definirane unutar *theme.css* dokumenta. Ova funkcionalnost je implementirana kako bi se poboljšalo korisnikovo iskustvo (engl. *UX - User Experience*) prilikom korištenja aplikacije, te na vizualan način pokušalo dodatno dočarati vremenske prilike.

Slika 4.8 prikazuje različite boje pozadine sučelja te različite ikone koje dočaravaju vremenske prilike.



Sl. 4.8 Pozadina sučelja i ikone za različite vremenske prilike

U narednom tekstu bit će opisana pojedina stanja koja odgovaraju pojedinom broju ispod sličice:

1 - Sunčano, bez oblaka, dan

2 - Bez oblaka, noć

3 - Na nebu ima malo oblaka, dan

4 - Na nebu ima malo oblaka, noć

5 - Nebo je oblačno

6 - Pljusak

7 - Kiša

8 - Grmljavinsko nevrijeme

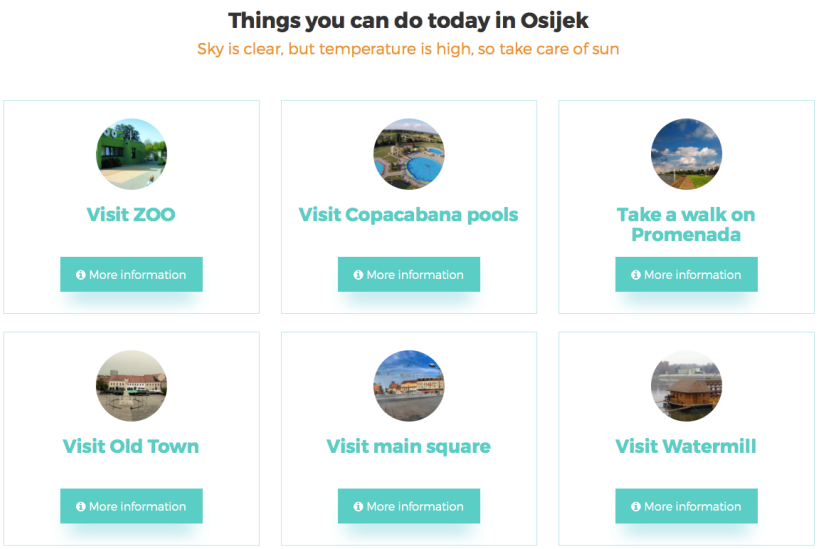
9 - Snijeg

10 - Magla

11 - Nedefinirano vrijeme

Pozadina sučelja dinamički se mijenja ovisno o trenutnom vremenskom stanju, s obzirom da je na temelju ikone koja se nalazi unutar podataka dobivenih s OpenWeatherMap poslužitelja moguće definirati o kojem vremenskom stanju je riječ.

Nakon dohvaćanja aktivnosti iz lokalne baze podataka, isti se šalju na sučelje u JSON formatu, kako bi se prikazali u vizualnom obliku na sučelju kako je vidljivo na slici 4.9.



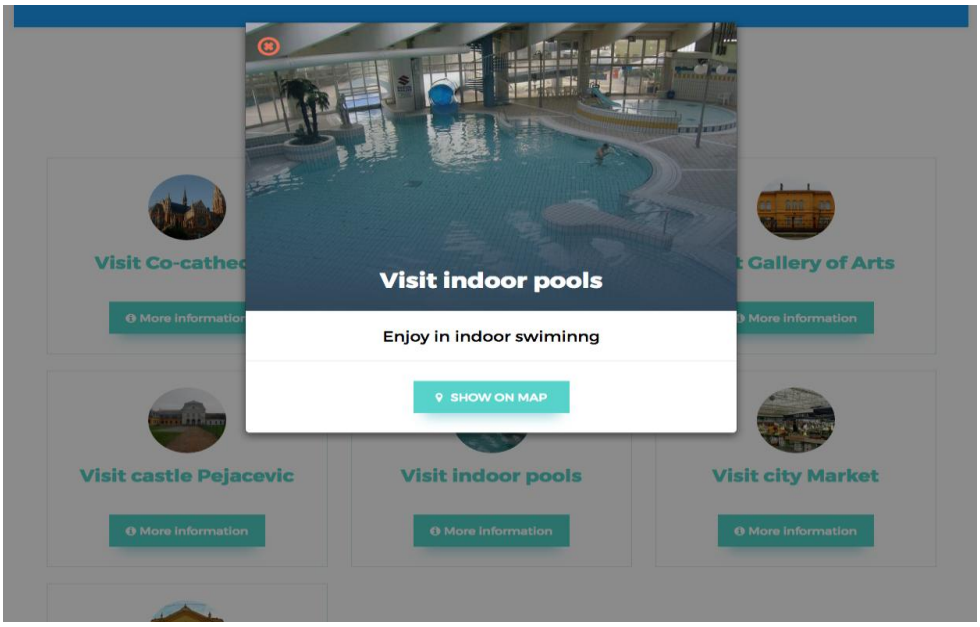
Sl. 4.9 Vizualni prikaz predloženih aktivnosti na sučelju aplikacije

Slika 4.9 prikazuje vizualni prikaz aktivnosti na sučelju aplikacije, odnosno, predstavlja donji dio sučelja aplikacije gdje se izlistavaju sve aktivnosti koje su predložene ovisno o vremenskim uvjetima. Također, moguće je uočiti poruka ispod naslova u narančastoj boji koja se na sučelje šalje iz funkcije *weatherAlgoritham()*. Aktivnosti su predstavljene unutar kvadratnih elemenata koje sadrže sliku aktivnosti, naslov aktivnosti te dugme koje omogućuje detaljniji uvid u pojedinu aktivnost. Funkcionalnost ovog dijela sučelja je povezana sa *weatherAlgoritham()* funkcijom koja poziva funkcije *queryGoodWeather()* i *queryBadWeather()*, koje zatim iz baze povlače podatke te ih šalju na sučelje u JSON formatu. Prikaz podataka na sučelju, odnosno parsiranje JSON formata vrši se pomoću AngularJS direktive *ng-repeat* koja ponavlja HTML strukturu sve dok ne dođe do kraja polja podataka. Ova direktiva omogućuje lagano izvlačenje podataka iz polja, a sintaksa je prikazana programskim kôdom 4.12.

```
ng-repeat = “(ključ, vrijednost) in myArray”
```

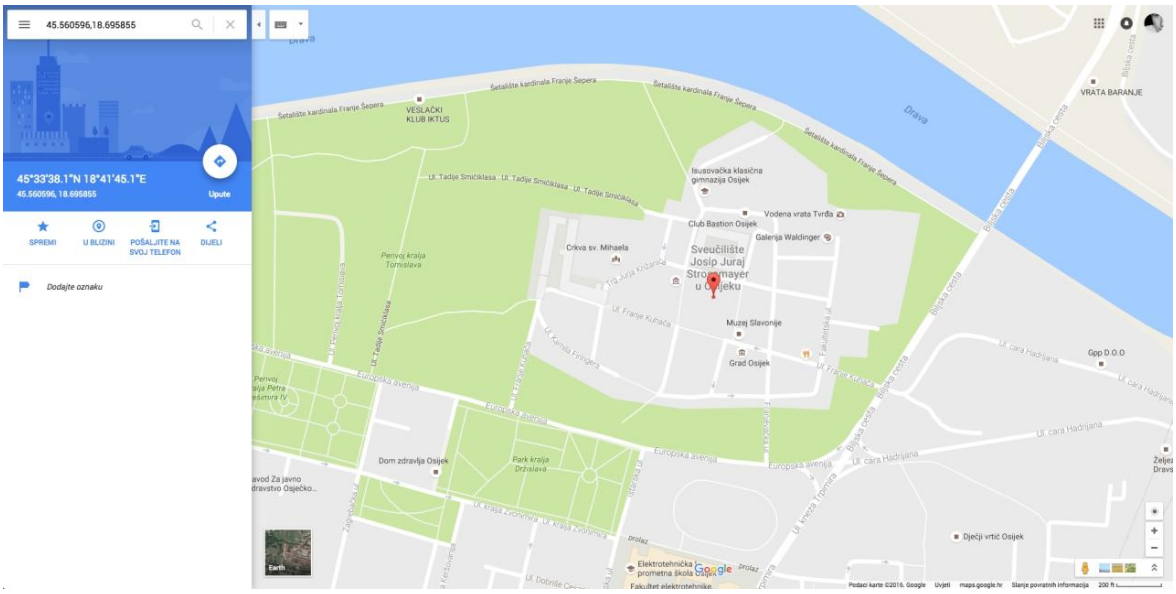
Programski kôd 4.12 Ispisivanje pojedinih podataka iz polja podataka

Vrijednost se iz polja izvlači po zadanome ključu, a pristup pojedinoj vrijednosti vrši se sa operatorom “.”. Detalji pojedine aktivnosti nalaze se unutar skočnog prozora kako je prikazano slikom 4.10.



Sl. 4.10 Skočni prozor koji prikazuje detalje pojedine aktivnosti

Skočni prozor sadrži fotografiju aktivnosti (parametar *Image* u lokalnoj bazi podataka), naziv aktivnosti (parametar *Name* u lokalnoj bazi podataka), dodatne informacije (paramterar *Info* u lokalnoj bazi podataka) te gumb "*Show on map*" koji omogućuje prikaz aktivnosti na karti Google Maps usluge. Lokacija se prikazuje tako da se gumbu predaju zemljopisna širina (*Latitude*) i zemljopisna dužina (*Logitude*) koje se nalaze u bazi, a označavaju lokaciju pojedine aktivnosti. Primjer prikaza pojedine lokacije pomoću Google Maps usluge prikazan je na slici 4.11.



Sl. 4.11 Prikaz lokacije aktivnosti pomoću Google Maps usluge

Korisnik zatim može koristiti usluge Google Maps-a, kao što je na primjer, traženje upute za navigaciju od njegove trenutne lokacije do lokacije aktivnosti, spremiti lokaciju na svoj Google profil, pogledati što je u blizini lokacije i tako dalje.

Prikaz prethodno navedenih elemenata, osim usluge Google Maps i prikaza aktivnosti na karti, implementiran je pomoću HTML elemenata pri čemu je korišten Bootstrap okvir koji pruža mogućnost raspodjele elemenata unutar mreže koja se sastoji od 12 stupaca.

Sučelje aplikacije te njezini elementi dinamički se prilagođavaju razlučivosti uređaja na kojem se aplikacija koristi. To podrazumijeva promjenu veličine pojedinih elemenata, poput fontova ili kvadratnih elemenata u kojima se nalaze aktivnosti, pa sve do razmještaja pojedinih elemenata. Primjerice, na većim razlučivostima, poput zaslona prijenosnog računala, postoje tri aktivnosti (kvadratna elementa) u jednome redu, dok na manjim razlučivostima, poput mobilnog uređaja, postoji jedna aktivnost u retku. Dinamička prilagodba elemenata sučelja omogućena je pomoću Bootstrap okvira, odnosno, korištenjem njegovih preddefiniranih elemenata i css klasa.

Dizajn sučelja aplikacije napravljen je minimalistički prema trenutnim trendovima u dizajnu web aplikacija i web stranica. Pozadina gornjeg dijela sučelja na kojem se prikazuje trenutno vremensko stanje je dinamička kako bi korisnik dobio dodatne vizualne informacije. Donji dio sučelja, gdje su izlistane predložene aktivnosti je bijele pozadine s istaknutim bitnim elementima i samo nužnim informacijama s obzirom da postoji mogućnost dodavanja novih aktivnosti i proširivanja sustava, pa da iz tog razloga korisniku bude lako pretraživati aktivnosti.

Prilikom implementacije sučelja, aplikacija je testirana u nekoliko web preglednika te je podržana od strane svih novijih inačica web preglednika za sve platforme, a također je testirana na web preglednicima na mobilnom uređaju kako je prikazano slikom 4.12.

Sl. 4.12 Web preglednici i uređaji na kojima je izvršeno testiranje sučelja

Na prijenosnom računalu testiranje je izvršeno u Google Chrome, Safari i Firefox web preglednicima, dok je na mobilnom uređaju testiranje izvršeno na web preglednicima Google Chrome i Safari. Testiranje je uključivalo kontrolu ispravnog prikazivanja svih elemenata sučelja u svim navedenim web preglednicima, kao i provjeru podržanosti pojedinih css svojstava.

5. UPUTE ZA UPOTREBU I TESTIRANJE APLIKACIJE

U tekstu koji slijedi opisane su upute za korištenje aplikacije te je izvršeno testiranje na temelju slučaja korištenja.

5.1. Upute za upotrebu aplikacije

Prilikom korisnikovog pristupanja aplikaciji u pozadini se odrađuju sve potrebne predradnje, odnosno, dohvaćaju se podaci o vremenskoj prognozi koji se zatim analiziraju, a pred korisnika se stavljaju sve definirane informacije kao i prijedlozi aktivnosti. Korisnik pretražuje informacije i popis predloženih aktivnosti koje su izlistane u donjem dijelu korisničkog sučelja. Slika 5.1 prikazuje korake korištenja aplikacije.

Sl. 5.1 Koraci korištenja web aplikacije

U prvom koraku korisniku je ponuđeno sučelje i pregled svih informacija koje su mu ponuđene. Aktivnosti su predstavljene unutar kvadratnih elemenata koje sadrže sliku aktivnosti, naslov aktivnosti te dugme koje omogućuje detaljniji uvid u pojedinu aktivnost. Pritiskom na dugme "*More information*", otvara se skočni prozor sa dodatnim informacijama prikazanim u drugom koraku. Skočni prozor se otvara preko postojećeg sučelja koje je u pozadini te postaje zatamnjeno. Kako bi se skočni prozor zatvorio, potrebno je pritisnuti kućicu "X" u gornjem lijevom uglu prozora ili pritisnuti pokazivačem miša na područje koje se nalazi izvan skočnog prozora, a zatim korisnik može nastaviti pregledavati ostale informacije te na isti način otvoriti neku od drugih ponuđenih aktivnosti. Skočni prozor sadrži fotografiju aktivnosti (*Image*), naziv aktivnosti (*Name*), dodatne informacije (*Info*) te gumb "*Show on map*". Pritiskom na gumb "*Show on map*" otvara se nova kartica web preglednika te se lokacija aktivnosti prikazuje na

Google Maps karti kako je prikazano u trećem koraku. Korisnik tada može pogledati lokaciju aktivnosti na karti, te koristiti ostale usluge koje pruža Google Maps, kao što su upute za navigaciju do lokacije aktivnosti od korisnikove trenutne lokacije, spremanje lokacije aktivnosti na svoj Google profil i tako dalje. Korisnik se može vratiti u karticu gdje se nalazi aplikacija, te nastaviti sa pregledom ostalih informacija.

5.2. Testiranje aplikacije

U sklopu statičnog testiranja izvršen je pregled koda unutar *app.js* dokumenta gdje se nalaze funkcije koje obavljaju funkcionalni dio aplikacije. Uklonjene su varijable koje nikada nisu korištene, referencirane su samo one varijable koje imaju dodijeljene vrijednosti, što je također vezano za već spomenuto rukovanje greškama, s obzirom da se radi o manipulaciji podataka koji se dohvaćaju sa poslužitelja te iz baze podataka. Također, uklonjen je sav kod koji se ne bi nikada izvršio ili zbog postavljanja uvjeta ne bi došlo do njegovog izvršavanja te su uklonjene sve sintaksne pogreške.

Testiranje funkcionalnosti aplikacije izvršeno je na primjeru korištenja. U prethodnom poglavlju objašnjena je upotreba ispisivanja podataka u konzolu kako bi se provjerila uspješnost njihovog zaprimanja te omogućila usporedba između zaprimljenih podataka i onih koji su ispisani na sučelju. Slika 5.2 prikazuje ispis u konzoli gdje primjećujemo da su podaci uspješno zaprimljeni sa OpenWeatherMap poslužitelja.

Sl. 5.2 Ispis konzole - podaci s OpenWeatherMap API-ja

Također se može primijetiti da je identifikator (*id*) vrijednosti 800, a opis trenutnog stanja "*clear sky*" što će biti vidljivo na sučelju. Ukoliko se referencira na tablicu 4.3 iz prethodnog poglavlja, algoritam bi na temelju identifikatora trebao pozvati funkciju *queryGoodWeather()* koja bi iz baze podataka trebala povući aktivnosti koje su namijenjene za povoljne vremenske uvjete (u bazi podataka su označene sa *Weather = '100'*).

Daljnjim pregledom ispisa konzole, može se zaključiti da su podaci uspješno dohvaćeni iz baze podataka te da su inicijalizirani od strane *queryGoodWeather()* funkcije, što znači da je algoritam uspješno odradio svoju zadaću. Slika 5.3 prikazuje postupak dohvaćanja podataka iz baze te kako su pozvani iz funkcije *queryGoodWeather().*

Sl. 5.3 Ispis konzole - podaci iz lokalne baze podataka

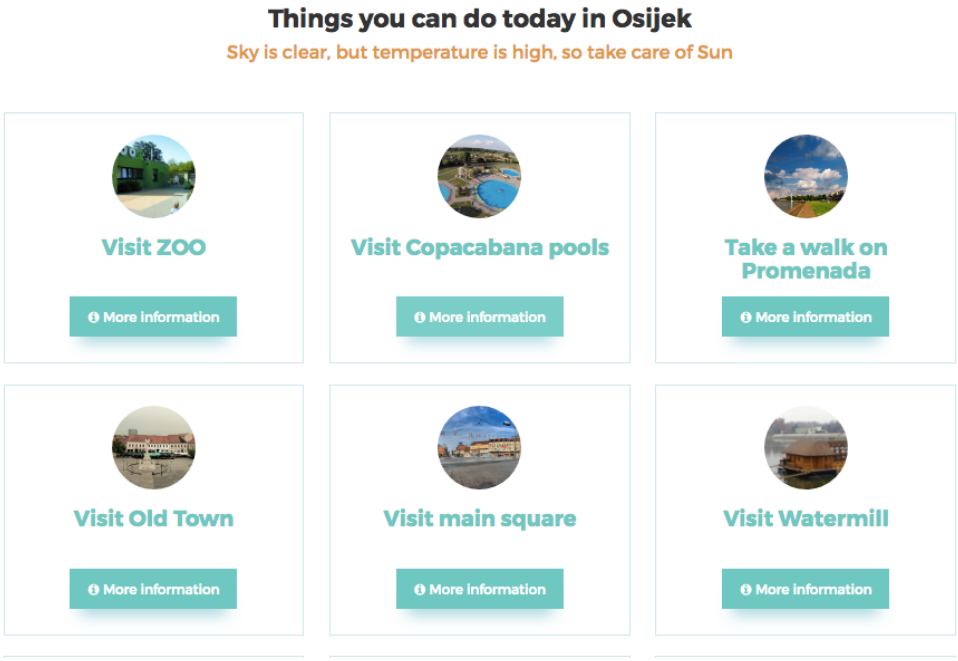
Moguće je zaključiti kako je algoritam odradio zadanu funkciju i donio ispravnu odluku na temelju implementiranog algoritma odlučivanja i raspoloživih podataka. Podaci vezani za temperaturu, opis vremenskog stanja, brzina vjetra te vrijeme izlaska i zalaska Sunca trebaju se prikazati na gornjem dijelu sučelja. Donji dio sučelja koji korisniku prikazuje prijedlog aktivnosti trebao bi prikazati aktivnosti vezane za povoljne vremenske uvjete.

Podaci vezani za ispis konzole sa slike 5.2 vidljivi su na gornjem dijelu sučelja koje je prikazano slikom 5.4.

Sl. 5.4 Testiranje primjerom korištenja – ispis podataka u gornjem dijelu sučelja

Uočava se kako su svi podaci ispisani na sučelju sukladno zaprimljenim podacima s poslužitelja OpenWeatherMap.

Slika 5.5 prikazuje donji dio sučelja na kojemu se trebaju prikazati aktivnosti, na temelju odluke algoritma, u ovome slučaju to su aktivnosti vezane za povoljne vremenske uvjete.



Sl. 5.5 Testiranje primjerom korištenja – ispis podataka u donjem dijelu sučelja

Na slici 5.5 vidljiv je prikaz upozorenja i aktivnosti na sučelju iz baze podataka. Ukoliko se ponovno referencira na tablicu 4.3 iz prethodnog poglavlja u kojemu je opisan algoritam, vidljivo je da bi se na sučelju trebala ispisati poruka "*Sky is clear, but temperature is high, so*

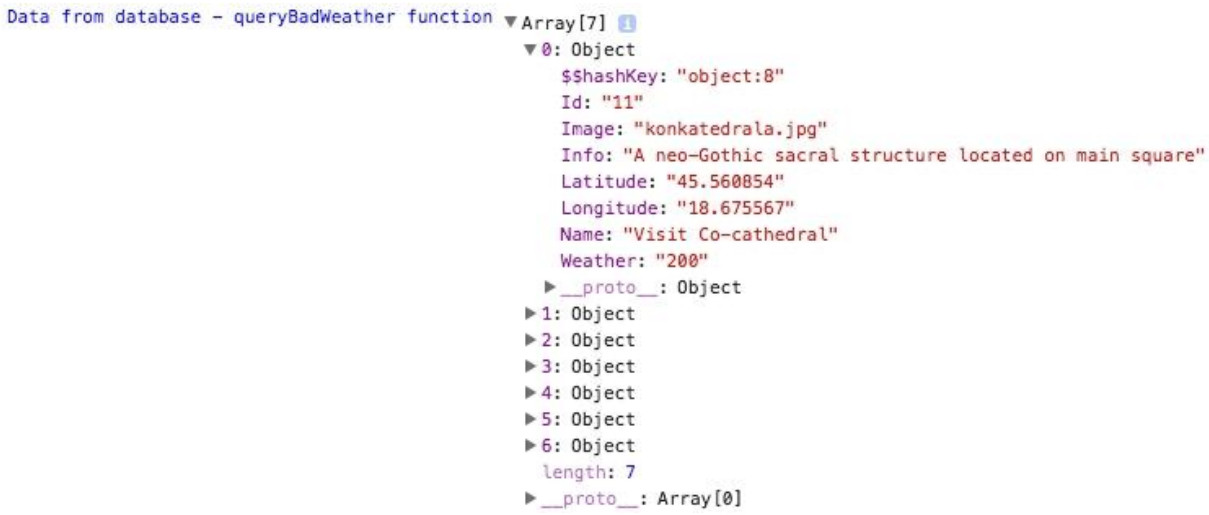
take care of Sun", ukoliko identifikator ima vrijednost 800, a temperatura je veća ili jednaka 26°C, što još jednom potvrđuje ispravnost algoritma.

Prvi test obavljen je tijekom sunčanog vremena, a drugi tijekom kišnog vremena. Sukladno tome, očekivani rezultati u drugom testu morali bi biti u potpunoj suprotnosti rezultatima prvog testa. Na slici 5.6 vidljiv je ispis konzole na kojemu su prikazani podaci sa OpenWeatherMap poslužitelja tijekom izvođenja drugoga testa.

Sl. 5.6 Ispis konzole - podaci sa OpenWeatherMap API-a

Sa slike 5.6 može se vidjeti da je identifikator (*id*) vrijednosti 501, a opis trenutnog stanja "*moderate rain*" što će biti vidljivo na sučelju. Ukoliko se referencira na tablicu 4.3 iz prethodnog poglavlja, algoritam bi na temelju identifikatora trebao pozvati funkciju *queryBadWeather()* koja bi iz baze podataka trebala povući aktivnosti koje su namijenjene za nepovoljne vremenske uvjete (u bazi podataka označene su sa *Weather = '200'*).

Sa slike 5.7 vidljivo je da su podaci uspješno dohvaćeni iz baze podataka te da su inicijalizirani od strane *queryBadWeather()* funkcije, što znači da je algoritam uspješno odradio svoju zadaću.

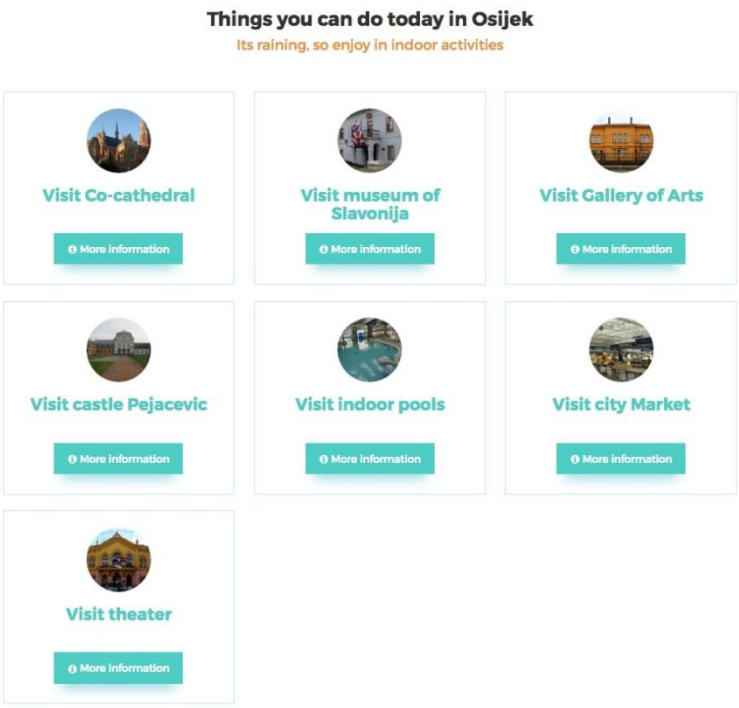


Sl. 5.7 Ispis konzole - podaci iz lokalne baze podataka

Ispis podataka na sučelju uspješno je obavljen. Podaci vezani za sliku 5.6 ispisani su u gornjem dijelu sučelja kako je vidljivo na slici 5.8.

Sl. 5.8 Testiranje primjerom korištenja – ispis podataka u gornjem dijelu sučelja

Aktivnosti iz lokalne baze podataka također su prikazane na sučelju kako je vidljivo na slici 5.9.



Sl. 5.9 Testiranje primjerom korištenja – ispis podataka u donjem dijelu sučelja

Nakon što su provedena testiranja u različitim vremenskim uvjetima moguće je zaključiti kako algoritam odlučivanja uspješno funkcionira. Na temelju ulaznih podataka koji dolaze sa OpenWeatherMap poslužitelja u obliku vremenske prognoze, te nakon njihove obrade, sustav vraća očekivani rezultat, a sve potrebne i definirane informacije na korisničkom sučelju su točne.

6. ZAKLJUČAK

U radu je obrađena problematika prilagodljivog prikazivanja turističkih sadržaja, u ovom slučaju, ovisno o vremenskoj prognozi. Cilj je bio koristiti nekoliko izvora podataka i na temelju njih servirati prilagođene informacije, kako vizualno tako i u kontekstu sadržaja tih informacija. Na temelju toga, postavljen je zadatak programiranja web aplikacije koja će ovisno o vremenskoj prognozi prikazivati prilagođeni sadržaj. Algoritam odlučivanja se nije bavio proračunima vremenske prognoze, nego je koristio raspoložive podatke i na temelju njih donosio odluke.

Kao izvor informacija o vremenskoj prognozi odabran je OpenWeatherMap API kao besplatna usluga. Sadržaj koji se prikazuje nalazi se u bazi podataka i služi kao drugi izvor informacija na kojeg je moguće utjecati, kako količinski tako i sadržajno. Aplikacija je izrađena sa svrhom da korisniku daje prijedlog o aktivnostima, pri čemu je pridana dodatna pažnja dizajniranju sučelja, kako bi korisnik dobivao samo potrebne informacije i kako bi što lakše koristio aplikaciju. Prilikom testiranja aplikacije i algoritma, pokazalo se da algoritam radi ispravno i ispunjava sve zadane uvjete, a korisnički zahtjevi koji su definirani su ostvareni.

Postoji mogućnost dorade same aplikacije, što bi podrazumijevalo da se za vremensku prognozu koristi neka od plaćenih usluga koja pruža točnije i ažurnije informacije. Također, postoji mogućnost dodavanja informacija odnosno podataka koji će se korisniku servirati.

LITERATURA

[1] I. Brilhante, J.A. Macedo, F.M. Nardini, R. Perago, C. Renso, Where Shall We Go Today? Planning Touristic Tours with TripBuilder, Conference on Information and Knowledge Management, San Francisco, CA, SAD, listopad – studeni 2013, br.13, str. 757–762.

[2] R. Baraglia, C. Frattari, C.I. Muntean, F.M. Nardini, F. Silvestri, RecTour: A Recommender System for Tourists, International Conferences on Web Intelligence and Intelligent Agent Technology, Washington, DC, SAD, prosinac 2012, str. 92-96.

[3] M. Kenteris, D. Gavalas, D. Economou, An innovative mobile electronic tourist guide application, Personal and Ubiquitous Computing, br.13, sv. 2, str.103-118, veljača 2009.

[4] Anonymous (2016) VirtualTourist. Dostupno na: https://www.virtualtourist.com/ (25.lipanj.2016.)

[5] Anonymous (2016) Turistička zajednica Osijek. Dostupno na: https://www.tzosijek.hr/ (25.lipanj.2016.)

[6] Anonymous (2016) TripAdvisor. Dostupno na: https://www.tripadvisor.com/ (22.kolovoz.2016.)

[7] Anonymous (2016) FourSquare(About). Dostupno na: https://foursquare.com/about (21.kolovoz.2016.)

[8] Anonymous (2016) OpenWeatherMap. Dostupno na: http://openweathermap.org/api (22.lipanj.2016.)

[9] Anonymous (2016) w3schools. Dostupno na: http://www.w3schools.com/ (22.lipanj.2016.)

[10] S. Seshadri, B. Green, AngularJS: Up and Running, Enhanced Productivity with Structured Web Apps, O'Reilly Media, Sebastopol, CA, SAD, 2014.

[11] D. Sklar, Learning PHP, A Gentle Introduction to the Web's Most Popular Language, O'Reilly Media, Sebastopol, CA, SAD, 2016.

[12] Anonymous (2016) MAMP. Dostupno na: https://www.mamp.info/en/ (23.lipanj.2016.)

[13] Anonymous (2016) Bootstrap. Dostupno na: http://getbootstrap.com/ (23.lipanj.2016.)

[14] Anonymous (2016) Meteocons. Dostupno na: http://www.alessioatzeni.com/meteocons/ (23.lipanj.2016.)

[15] Anonymous (2016) FontAwesome. Dostupno na: http://fontawesome.io/ (23.lipanj.2016.)

[16] Anonymous (2016) OpenWeatherMap (Weather Conditions). Dostupno na: http://openweathermap.org/weather-conditions (23.lipanj.2016.)

SAŽETAK

U sklopu ovog diplomskog rada osmišljena je i razvijena web aplikacija za prilagodljivo prikazivanje turističkih sadržaja za grad Osijek. Cilj rada bio je napraviti web aplikaciju koja će prikazivati trenutnu vremensku prognozu za grad Osijek i na temelju tih parametara dinamički ponuditi prijedloge aktivnosti koje bi pojedinci mogli uzeti u obzir prilikom oblikovanja svog slobodnog vremena. Funkcionalni dio aplikacije ostvaren je pomoću AngularJS-a, korišten je OpenWeatherMap API i lokalna baza podataka, a za sučelje, između ostalog, korišten je HTML, CSS i Bootstrap okvir. Nakon testiranja, utvrđeno je da implementirani algoritam odlučivanja radi ispravno.

Ključne riječi: AngularJS, turistički sadržaji, turizam, vremenska prognoza, web aplikacija

ABSTRACT

WEB APPLICATION FOR DYNAMIC DISPLAY OF TOURIST ATTRACTIONS

As part of this master thesis the web application for dynamic display of tourist attractions in the city of Osijek has been designed and developed. The aim of the work was to make a web application that will display the current weather for the city of Osijek, and based on these parameters it will dynamically offer suggestions of activities that individuals might want to consider when designing their free time. The functional part of the application has been developed with AngularJS, OpenWeatherMap API and the local database were used, and for the interface, among other things, HTML, CSS and Bootstrap framework were used. After testing, it was determined that the implemented algorithm is working properly.

Keywords: AngularJS, tourism, tourist attractions, weather forecast, web application

ŽIVOTOPIS

Goran Ivoš rođen je 4. lipnja 1991. godine u Virovitici. Pohađao je Osnovnu školu Ivana Gorana Kovačića u Gornjem Bazju u razdoblju od 1998. do 2006., gdje je sudjelovao na natjecanjima iz informatike u programskom jeziku Pascal. Nakon toga, pohađao je Strukovnu školu u Virovitici, smjer ekonomist u razdoblju od 2006. do 2010. Nakon završene srednje škole, 2010. godine upisuje stručni studij na Elektrotehničkom fakultetu u Osijeku, smjer Informatika, a 2013. stječe zvanje bacc.ing.el., smjer Informatika. U razdoblju od 2013. do 2014. pohađa program razlikovnih obaveza kako bi stekao uvjete za upis na diplomski studij. U razdoblju od 2014. do 2016. godine pohađa diplomski studij na Elektrotehničkom fakultetu u Osijeku, smjer Procesno računarstvo. Tijekom diplomskog studija počinje raditi na radnom mjestu *front-end developera* u tvrtki „Betaware“ iz Osijeka.

PRILOZI (na CD-u)

PRILOG 1. Web aplikacija za prilagodljivo prikazivanje turističkih sadržaja.docx

PRILOG 2. Web aplikacija za prilagodljivo prikazivanje turističkih sadržaja.pdf

PRILOG 3. Programski kod web aplikacije za prilagodljivo prikazivanje turističkih sadržaja