Izrada web aplikacije putem Angulara 6 razvojnog okvira

Tuđan, Filip

Sveučilište
Sjever

Završni rad br. 597/MM/2018

Izrada web aplikacije putem Angulara 6 razvojnog okvira

Filip Tuđan, 0814/336

Varaždin, rujan 2018. godine

Sveučilište
Sjever

Odjel za Multimediju, oblikovanje i primjenu

Završni rad br. 597/MM/2018

Izrada web aplikacije putem Angulara 6 razvojnog okvira

Student

Filip Tuđan 0814/336

Mentor

mr.sc.Vladimir Stanisavljević

Varaždin, rujan 2018. godine

Sveučilište Sjever
Sveučilišni centar Varaždin
104. brigade 3, HR–42000 Varaždin

Prijava završnog rada

Definiranje teme završnog rada i povjerenstva

ODJEL: Odjel za multimediju, oblikovanje i primjenu

PRISTUPNIK: Filip Tuđan | MATIČNI BROJ: 0814/336

DATUM: 19.09.2018. | KOLEGIJ: Programski alati 2

NASLOV RADA: Izrada web aplikacije putem Angulara 6 razvojnog okvira

NASLOV RADA NA ENGL. JEZIKU: Development of a web aplication by Angular 6 framework

MENTOR: mr.sc. Vladimir Stanisavljević | ZVANJE: viši predavač

ČLANOVI POVJERENSTVA:
1. doc.dr.sc. Dejan Valdec - predsjednik
2. dr.sc. Andrija Bernik, pred. - član
3. mr.sc. Vladimir Stanisavljević, v.pred. - mentor
4. dr.sc. Robert Logožar, v. pred. - zamjenski član
5.

Zadatak završnog rada

BROJ: 597/MM/2018

OPIS:

Programski jezik JavaScript osnova je rada svih dinamičkih web stranica. Dugi niz godina standardiziran je kroz ECMA script specifikaciju. Razvojni okviri olakšavaju rad s JavaScriptom dok druga programska rješenja modificiraju sam jezik kako bi se kod čim lakše pisao. AngularJS već neko vrijeme spada u takve razvojne okvire odnosno programska rješenja, a u svojoj 6. inačici donosi donosi brojne izmjene i poboljšanja.

U ovom radu potrebno je:
* opisati osnove najnovijeg AngularJS-a razvojnog okvira
* opisati postupak instalacije i osnovnog korištenja iz odabranog razvojnog okruženja te postavljanje na javni poslužitelj
* detaljno obraditi mogućnosti i sintaksu AngularJS-a te na primjerima pokazati njihovo korištenje,
* osmisliti i oblikovati demo web stranicu na kojoj će biti prikazane glavne mogućnosti, načini korištenja AngularJS-a, a u radu objasnit primjere koda,
* ukratko usporediti AngularJS sa sličnim programskim rješenjima

Detaljno opisati sve korištene tehnologije i korake u radu potrebne da bi se ostvarilo traženo te detaljno opisati stečena iskustva i postignute rezultate.

REPUBLIKA HRVATSKA 5 SVEUČILIŠTE SJEVER

ZADATAK URUČEN: 21.09.2018 | POTPIS MENTORA: Vlado

Predgovor

Zahvaljujem svim profesorima i djelatnicima Sveučilišta Sjever u Varaždin na stečenom znanju, pomoći i savjetima neophodnima mojoj struci.
Također, posebno zahvaljujem svom mentoru prof.mr.sc. Vladimiru Stanisavljeviću na doprinosu nastanku ovoga rada, usmjeravanju, vodstvu i stručnoj pomoći.
Na kraju, zahvaljujem svim kolegicama i kolegama što su svojim prisustvom uljepšali moje studentsko razdoblje.

Sažetak

Rad se temelji na razvojnom okviru pod nazivom Angular, u verziji 6, u kojem je napravljena aplikacija koja prikazuje vremensku prognozu u Europskim gradovima, uglavnom glavnim gradovima. Aplikacija je napravljena uz pomoć Visual Studio Code-a, a cilj joj je da olakša praćenje vremenske prognoze u narednih šest dana za gradove koji su prikazani u aplikaciji sa minimalnim brojem klikova i sa vrlom lijepom preglednošću. U početnom dijelu rada su objašnjena neka općenita svojstva koja dovode do izrade aplikacije, bez kojih izrada aplikacije ne bi bila moguća. U srednjem dijelu rada su prikazani i objašnjeni alati bez kojih aplikacija ne bi postojala i ne bi mogla funkcionirati. U završnom dijelu rada je prikazan način izrade aplikacije, pomoću kojih programa, biblioteka itd. je izrađen, te konačan izgled aplikacije. Na kraju rada je zaključak, literatura, popis slika i kodova koji su korišteni u radu.

Ključne riječi: Razvojni okvir, aplikacija, Angular, CSS, Javascript, komponente, modul, servis, Bootstrap, Visual Studio Code, HTML

Abstract

The work is based on a development framework called Angular, in version 6, in which an application is presented that shows the weather forecast in European cities, mostly in major cities. The application was created with the help of the Visual Studio Code, and its goal is to facilitate the weather forecasts for the next six days for cities that are shown in the app with a minimum number of clicks and with great visibility. In the initial part of the paper, some general features that lead to the creation of applications are explained, without which application creation would not be possible. In the middle part of the work, there are explained and explained tools without which the application would not exist and could not function. In the final part of the article, you can see how the applications were created, by which programs, libraries, etc. were created, and the final appearance of the application. At the end of the paper is the conclusion, the literature, the list of images and codes used in the work.

Keywords: Framework, Application, Angular, CSS, Javascript, Component, Module, Service, Bootstrap, Visual Studio Code, HTML

Sadržaj

1. Uvod

U svijetu današnjice programiranje je sve više popularni pojam jer se tehnologija razvija iz dana u dan. U programiranju i korištenju raznih programskih jezika i razvojnih okvir-a se nalazi sve više ljudi koje sa oduševljenjem prihvaćaju takvo zanimanje. Svijet programiranja, kodiranja se sve više širi i svakim danom je sve opširniji pa je u to zanimanje potrebno uložiti dosta zanimanja i truda. Ljudi koji su u tom poslu moraju konstantno pratiti trendove, nove razvojne okvir-e i sl. Pojmovi programa i programiranja prisutni su danas na svakom koraku, pogotovo u masovnim medijima (Internet, TV, tisak…). Budući da smo u svakodnevnom životu okruženi sve složenijim elektroničkim uređajima (računalima, perilicama, smartphonima, mikrovalnim pećnicama i sl.) njihova upotreba i upute za rad sve više u sebi uključuju neki od postupaka programiranja, od jednostavnih do složenijih. Problem kod bilo kojeg programiranja je da iskustvo i način razmišljanja često nisu usklađeni s načinom programiranja uređaja i naprava, pa se s toga treba prilagoditi takvom načinu programiranja, jer danas ima puno načina programiranja kroz razne programe, razvojne okvir-e, alate i sl. Razni programski jezici su omogućili brzi razvoj tehnologije koji se proširio svijetom. Ima velik broj raznih programskih jezika od onih starijih (strojni jezici, simbolički jezici) do onih novijih (viši programskih jezici kao što je C,C++ i sl., ili jezici prilagođeni krajnjim korisnicima kao što je SQL. Kroz godine studiranja najviše sam naučio raditi u jezicima kao što su HTML, CSS te JavaScript. U današnje vrijeme ti jezici su omogućili sve više dinamičnosti te interaktivnih elemenata na stranicama u odnosu na prije gdje su sve stranice bile uglavnom statične. Omogućili su sve veću interakciju korisnika i stranice ili aplikacije. Ja ću se u ovom radu više usmjeriti na web programiranje aplikacije u kojem će mi biti potrebno znanje JavaScripta, CSS-a te framework tehnologija koja je neizbježna za web development. Razvojni okvir je dizajniran da podržava izgradnju web aplikacija, web servisa, resursa i sl. On omogućuje standardni način za izradu i pokretanje web aplikacija. Postoje mnoštvo kvalitetnih i zanimljivih razvojnih okvir-a za web development kao što su Laravel, React JS, Node JS, Ruby, Symfony, ASP.NET, Meteor JS i razni drugi.

Za web aplikaciju potrebno je naučiti raditi sa razvojnim okvirom pod nazivom Angular 6. Angular je trenutno najpopularniji web development razvojni okvir. Kreiran je od strane Googla, besplatan je i dizajniran je specifično za single-page web aplikacije. Angular se sastoji od tri stavke, od komponente, servisa i modula. Završni rad se sastoji od dva dijela, od onog teoretskog i praktičnog. U teoretskom dijelu su objašnjena pomagala (Angular, Javascript, CSS, Bootstrap, Visual Studio Code, HTML i sl.) u stvaranju web aplikacije. U tom teoretskom dijelu je objašnjena svaka stavka koja je potrebna da se predstavi određeno pomagalo, npr. na koji način

funkcionira, gdje se koristi , kako se koristi, koje su prednosti tog korištenja, što se sa njima sve može raditi i sl. U praktičnom dijelu je prikazan cilj dolaska do željene aplikacije. Prikazane su funkcije, programi, razvojni okvir, biblioteke koje su se koristile da bi se napravila aplikacija na kvalitetan način.

2. Kodiranje i programiranje

Programiranje[1] je pisanje uputa računalu što i kako učiniti, a izvodi se u nekom od programskih jezika. Programiranje[2] je umjetnost i umijeće u stvaranju programa za računala. Stvaranje programa sadrži u sebi pojedine elemente dizajna, umjetnosti, znanosti, matematike kao i inžinjeringa. Osoba koja stvara program zove se programer. Programi ili upute za računalo pišu se u programskom jeziku upotrebom određene sintakse i pravila koja vrijede za svaki programski jezik (ili tip), koji se potom prevodi u strojni jezik koje je osobito za određeno računalo te je ovisno o njegovoj arhitekturi. Prevođenje s višeg programskog jezika na strojni provodi se putem programa prevodioca (kompajler) ili se naredbe u višem jeziku izravno prevode preko takozvanog p_koda u strojni jezik.

Pri izradi svakog programa potrebno je proći kroz 4 osnovne faze ili koraka:

1. Analiza problema
2. Kreiranje algoritma (crtanje dijagrama toka)
3. Pisanje programskog koda
4. Unos programskog koda u računalo i pokretanje programa

Računalo razumije samo binarni jezik , jezik koji se sastoji od dva simbola, „0“ i „1“. Unutar računala znakovi su prikazani pomoću binarnih brojeva. Postupak pripisivanja simbola znakovima vanjskog svijeta naziva se kodiranje. Skup takvih simbola naziva se kod. Kodni sustav je dogovor o načinu kodiranja. Najrasprostranjeniji kodni sustav je ASCII kod. Pod kodiranjem razumijevamo prevođenje određenih podataka u simboličke oblike s ciljem njihove obrade putem računala. Prema tome, računalna obrada podataka pretpostavlja kodiranje podataka pri ulazu u središnju procesorsku jedinicu i nakon njihove obrade dekodiranjem radi distribucije korisnicima. To drugim riječima znači da su prihvaćanje, čuvanje i obrada podataka u računalu mogući samo ako su oni kodirani pomoću dualnih simbola 0 i 1 .

2.1. HTML

Osnovni jezik koji koristimo za izradu web stranica je HTML[3] (eng. HyperText Mark-up Language). On našim preglednicima (eng. browser) govori sve o sadržaju i izgledu neke web stranice, kako bi je preglednik mogao pravilno prikazati. To je jezik kojim stranice razgovaraju s našim web preglednicima, ali i jezik koji bi svaki webmaster, odnosno izrađivač web stranica, trebao znati. Postoje mnogi alati koji nam omogućavaju izradu web stranica i bez najosnovnijeg poznavanja HTML koda. Ti alati automatski generiraju HTML kod, prateći pritom upute koje korisnik zadaje preko manje ili više jednostavnog korisničkog sučelja. Riječ je o tzv. vizualnom uređivanju, gdje korisnik odabirom određenih alata stvara, odnosno uređuje web stranicu koja mu se u realnom vremenu prikazuje onako kako će se kasnije prikazivati u nekom od preglednika.

Za razliku od ručnog pisanja koda, koje zahtjeva tek program za uređivanje teksta (npr. Notepad), vizualno uređivanje ima nekoliko prednosti. Osim što korisnik odmah vidi sadržaj svoje web stranice, program sam generira potreban kod, pa je moguća pojava grešaka u pisanju koda svedena na minimum. Alati koji nude vizualno uređivanje web stranica najčešće nude i mogućnost ručnog pisanja koda. Među najpopularnijim alatima za vizualnu izradu web stranica su Microsoftov FrontPage i Macro-medijin Dreamweaver. Ručno pisanje koda u programu za obradu teksta prilično je neugodno, već zbog same činjenice da korisnik ne vidi što zapravo radi. Ono što zapravo radi jest – kodira naslijepo. Nakon dopisivanja nekoliko novih linija koda korisnik bi trebao prekinuti rad i svoj uradak pregledati u web pregledniku. Taj proces nije samo zamoran, već može biti i prilično frustrirajuć, jer su greške u ručnom pisanju koda gotovo neizbježne. S druge pak strane, poznavanje HTML-a neophodno je za svako imalo ozbiljnije izrađivanje web stranica. Čak i programi poput Dreamweavera i FrontPagea imaju svoje nedostatke. Tako ponekad dodaju nepotrebne linije koda, koje samo povećavaju vrijeme učitavanja naših stranica, a ponekad ono što želimo dodati jednostavno nije dostupno putem korisničkog sučelja. Svoje stranice vjerojatno nećete u cijelosti pisati ručno, ali svakako ćete doći do trenutka kada ćete nešto jednostavno morati sami dopisati ili ispraviti. Tada će vaše znanje HTML-a biti od velikog značaja.

HyperText Markup Language ili skraćeno HTML, jezik je za, kako mu i samo ime kaže, označavanje hipertekstualnih dokumenata. Hipertekstualne dokumente danas najčešće susrećemo na Internetu, prvenstveno na Webu. Oni se od običnih dokumenata razlikuju po tome što sadrže hiperveze (hiperlinkove) kojima su povezani s drugim (hipertekstualnim) dokumentima. Svaku

web stranicu koja sadrži barem jednu hipervezu ubrajamo u skupinu hipertekstualnih dokumenata. HTML nije programski jezik niti su ljudi koji ga koriste programeri. Njime ne možemo izvršiti nikakvu zadaću, pa čak ni najjednostavniju operaciju zbrajanja ili oduzimanja dvaju cijelih brojeva. On služi samo za opis naših hipertekstualnih dokumenata i za ništa više od toga.

HTML komande se pišu u vidu tzv. tag - ova. Jedan tag je u komanda koja govori našem browseru što i kako napraviti tj. na koji način prikazati sadržaj naših stranice. HTML tagovi su "case insensitive" tj. svejedno je da li ih pišemo malim ili velikim slovima. Tagovi se pišu unutar oznaka "<" i ">" (bez znakova navoda) npr: <html>. Ovaj tag se nalazi na početku svakog HTML dokumenta i on govori našem browseru da je datoteka koju je upravo počeo učitavati baš HTML dokument i da kao takvog treba i prikazati. Na kraj HTML dokumenta se stavlja završni HTML tag:</html>. Ovaj tag govori browseru da je to kraj našeg HTML dokumenta. Većina tagova ima i početni i završni tag. Završni tag se dobiva dodavanjem znaka "/" i označava mjesto na kojem prestaje djelovanje početnog taga.

Postoje i tagovi kod kojih ne moramo stavljati završni tag kao sto je recimo tag
 koji služi za prelazak u novi red (o ovom tagu ćemo govoriti kasnije u dijelu o formatiranju teksta).

Svaki HTML dokument se sastoji od dva dijela: zaglavlja (engl. head) i tijela (engl. body). Zaglavlje se odvaja tagovima <head> i </head>, a tijelo dokumenta tagovima: <body> i </body>. Sve ono što napišemo u zaglavlju dokumenta neće se prikazati u prozoru browsera već obično služi samo da pruži neke informacije o našoj stranici.

```
32  </div>
33
34  <ul class="menulist">
35    <li><a class="menuitems" href="index.html">Home</a></li>
36    <li><a class="menuitems" href="about_me.html">About me</a></li>
37    <li><a class="menuitems" href="my_works.html">My works</a></li>
38    <li><a class="menuitems" href="contact.html">Contact</a></li>
39  </ul>
40  </div>
```

Slika 2.1.Izbornik putem Html-a koji je povezan na stranicu

2.2. CSS

CSS (Cascading Style Sheets), odnosno kaskadni stilovi, jednostavan su mehanizam za dodavanje stilova: fontova, boja razmaka između paragrafa, uređivanje tablica. Svojim dolaskom CSS[4] je izazvao pravu revoluciju na internetu i to zahvaljujući nizu prednosti koje ima pred tabličnim layoutom (korištenje tablica za formiranje stranice). Korištenjem CSS-a postalo je moguće odvojiti prezentaciju podataka i dizajn od same strukture podataka. HTML kod postaje pregledniji i manji što znači da ga je puno lakše kontrolirati, a također je moguće jednostavnom primjenom parametara promijeniti izgled stranice. CSS je donio čitav niz načina za uređivanje prikaza podataka koji do tada nisu postojali u samom HTML-u, a web programeri su razvili korisne tehnike kojima možete uštediti dragocjeno vrijeme prilikom izrade internet stranica. Mogućnosti formatiranja HTML-a poprilično su ograničene. Kada dizajniramo izgled stranice u HTML-u, ograničeni smo na tablice, kontrola fontova, i nekoliko stilova teksta poput bold i italic.

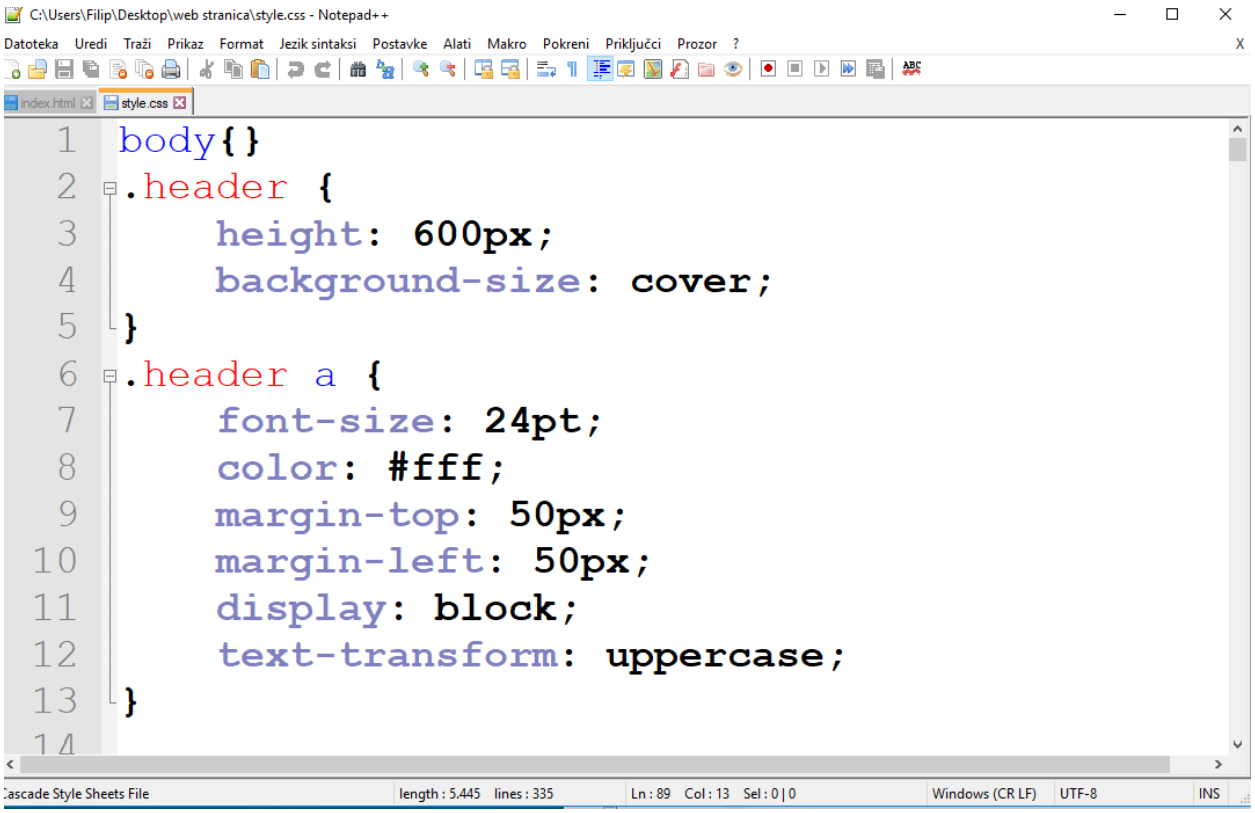
Sa stilskim obrascima možemo:

- Pažljivo kontrolirati svaki aspekt prikaza na stranici (odrediti razmak između linija, znakova, margine stranica, pozicije slika i drugo).
- Primijeniti promjene na cijelu internet stranicu
- Možemo osigurati dosljedan dizajn na cijelu internet stranici tako da isti stilski obrazac koristimo za svaku pojedinu stranicu.
- Dati internet pregledniku instrukcije za kontrolu izgleda stranice.
- Kreirati dinamične stranice. Pomoću CSS-a definiramo pravila u stilskom obrascu koji određuje kako želimo da sadržaj opisan određenim HTML kodom izgleda i povezujemo stilska pravila i HTML kod.

Korištenjem CSS obrasca možemo kontrolirati bilo koji dio segmenta na Internet stranici:

- Podešavanje pozadine (boja pozadine, slike na pozadini).
- Opcije okvira (kontrolira okvire povezane sa stranicom, liste, tablice, slike, blok elemente.
- Opcije klasifikacija (kontroliraju na koji se način elementi poput slika ponašaju na stranici u odnosu na ostale elemente.
- Uređivanje listi.
- Uređivanje margina.
- Kontrola padding-a (kontrola količine praznog prostora oko bilo kojeg blok elementa na stranici).

- Kontrola pozicioniranja elemenata.
- Kontrola veličine elemenata.
- Uređivanje tablica.
- Uređivanje teksta.



Slika 2. CSS kod

Na ovoj slici vidimo primjer uređivanja body elementa u HTML-u pomoću CSS-a. Tu smo odredili aspekte za naslov(header), koja je veličina fonta, koja boja slova, kako je pozicioniran tekst i sl.

2.3. JavaScript

Uz HTML i CSS, treći jezik kojim je važno znati rukovati je Javascript[5]. JavaScript je skriptni jezik kojim možemo kreirati dinamičke web stranice. Dinamičke web stranice omogućavaju međudjelovanje s korisnikom, upravljanje web preglednikom ili dinamičko kreiranje web stranice.

JavaScript se izvodi na klijentskom računalu i može se izvoditi u svim modernim web preglednicima. Izvorno ga je razvila tvrtka Netscape. JavaScript je primjena ECMAScript standarda.

JavaScript je objektno zasnovan skriptni jezik. Uključujemo ga u web stranicu da bi je učinili dinamičnijom. HTML (osnovni kod web stranica) se koristi samo za oblikovanje i uređivanje elemenata stranice (tekst, forme, linkove i tabele), ali nema šanse da diktiramo kako će se ti elementi ponašati. Mogućnost uključenja JavaScript skripte daje nam mnogo veću kontrolu kako se web stranica ponaša. Zajedno sa HTML-om i CSS-om JavaScript čini DHTML (Dinamic HTML).

Objektno je zasnovan jer programer ne definira samo tip podataka, nego i vrstu operacija (funkcija) koje se mogu primjeniti na strukture podataka. Na ovaj način, struktura podataka postaje objekat koji uključuje i podatke i funkcije. Pored toga, programeri mogu da kreiraju odnose između jednog i drugog objekta. Na primjer, objekti mogu da pridobiju karakteristike od drugih objekata.

Skriptni je jezik jer se sastoji od serije komandi koje se očitavaju u interpreteru (program prevodioc), a da se predhodno ne kompajlira sadržaj (compiler- program prevodioc). Odnosno ne prevodi se u strojni jezik (binarni kod- 1 i 0) iz koga nikada nećemo saznati originalni jezik, nego se komande direktno "čitaju" iz koda (source code ili bytecode). Zbog ove karakteristike JavaScript se izvršava na strani korisnika (client side), tj. na računalu na kojem je pokrenut sadržaj sa JavaScript-om. Sam po sebi, HTML dozvoljava posjetiocu da pošalje podatke k serveru na obradu. Nažalost ako ti podaci nisu validni cijeli proces se mora ponoviti sve dok se ne unesu validni podaci. Ovo je jedan od osnovnih razloga nastanka JavaScript-a koji provjerava vjerodostojnost podataka na klijentovom pregledniku (browser-u) i tako olakšava posao na web-u.

Rani skript jezici su se često nazivali batch jezici. Ovo su neki skript jezici : ASP, JSP, PHP, Perl, Tcl, Python itd.

JavaScript[6] je najpopularniji skriptni jezik na Internetu kojeg podržavaju svi poznatiji preglednici (Internet Explorer, Mozilla Firefox, Netscape, Opera). Evo nekoliko stvari koje je svaki prosječan surfer na internetu vidio: padajući meni, neobični pokazivač miša, iskakajući prozor, sat ... Međutim postoje ozbiljnije primjene JavaScript-a kao što su:

- Detekcija preglednika -Određuje se tip preglednika koji se koristi pri pregledu vaše web stranice. Zavisno od preglednika, može se prikazati drugačija stranica posebno dizajnirana za taj preglednik.
- Kontrola prozora -mogu se kontrolirati dimenzije, meniji, dugmadi, vrijeme prikaza itd.
- Web kolačići (cookies) -pohranjeni podatci na korisnikovom računaru, koji se automatski preuzimaju sljedeći put kada korisnik posjeti vašu web stranicu
- Provjera sadržaja -dobar primjer za ovo je provjeravanje da li je korisnik upisao znak @ u polje za upis email adrese

2.3.1. Prednosti i nedostatci JavaScripta

Neke od prednosti JavaScripta su:

- Manja potreba za komunikaciju sa serverom. Može se provjeriti ispravnost podatka prije slanja stranice na server. To smanjuje promet prema serveru.
- Trenutni odziv korisniku. Korisnik ne treba čekati da se stranica ponovno učita kako bi vidio je li zaboravio unijeti neki podatak
- Povećana interaktivnost. Moguće je kreirati sučelje koje reagira na korisnikove akcije tipkovnice ili miša.
- Bogatije sučelje. Možete koristiti JavaScript za uhvati i pusti komponente ili animirati bogatije sučelje prema korisniku.

Neki nedostatci JavaScripta su:

- JavaScript je klijentski programski jezik i nije mu dopušteno pisanje ili čitanje datoteka. Ova osobina je ugrađena radi sigurnosnih razloga.
- JavaScript ne može se koristiti za mrežne aplikacije jer za nju ne postoji podrška.
- JavaScript nema mogućnost višenitnog ili višeprocesorskog izvođenja.[6]

```
54          <script>
55
56      $('body').vegas({
57    overlay: true,
58    transition: 'fade',
59    align: 'top',
60    transitionDuration: 5000,
61    delay: 6000,
62    animation: 'random',
63    animationDuration: 2000,
64    slides: [
65                        { src: "img/img01.jpg" },
66                        { src: "img/img02.jpg" }
67
68
69    ]
70  });
71  </script>
```

Slika 3. Prikaz JavaScript programa

Ovo je primjer jednog JavaScript programa kojemu je funkcija da mijenja slike ovisno o vremenskom razmaku kojeg smo odredili.

2.4. Razvojni okvir

Web razvojni okviri[7] su transformirali svijet programiranja i postali su veoma važni u svakom razvojnom procesu. Razvojni okvir je softverski alat koji omogućuje način za izgradnju i pokretanje web aplikacija. Kao rezultat ne trebate pisati kod sami i gubiti vrijeme tražeći moguće greške. Postoje dvije glavne funkcije na koje se razvojni okvir dijeli: na rad na serverskoj strani (backend) ili na strani klijenta (frontend). Frontend razvojni okviri se većinom bave vanjskim dijelom web aplikacija, zapravo ono što korisnik vidi kada otvori aplikaciju. Backend se bavi unutarnjim dijelom web aplikacija, kako sve funkcionira na želje korisnika.

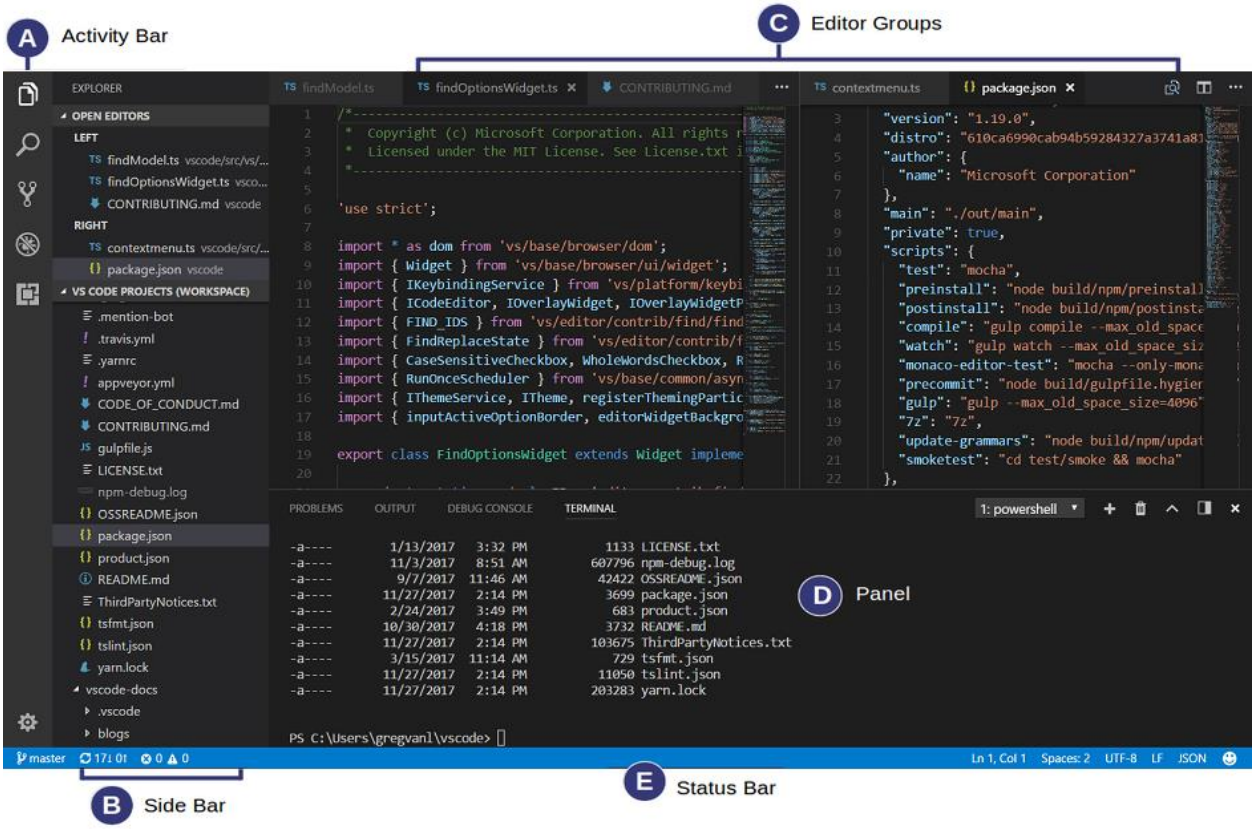
Serverska strana razvojnih okvira je ta da pravila i arhitektura tih razvojnih okvira vam ne dopuštaju gradnju web aplikacija sa dobro razvijenim sučeljem. Ti razvojni okviri imaju ograničenu funkcionalnost, ali svejedno možete kreirati jednostavne stranice, forme različitih tipova. Oni mogu formirati vanjski dio podataka i poboljšati sigurnost u slučaju nekakvih hakerskih napada. Serverska strana razvojnih okvira radi uglavnom na pojedinim ali važnim detaljima bez kojih aplikacija ne bi normalno funkcionirala. Ovo su jedni od najpopularnijih server razvojnih okvira: Django (Python), Zend (PHP), Express.js (Javascript), Ruby on Rails (Ruby).

Klijentska strana razvojnih okvira za razliku od serverske strane nema veze sa logikom tvrtke. Njegov posao je unutar pretraživača, on može poboljšati i ubacivati nova korisnička sučelja, može kreirati mnoge animacije sa frontend razvojnim okvirima kao i sa single-page aplikacijama. Ovo su jedni od najpopularnijih klijentskih razvojnih okvira: Angular, Ember.js, Vue.js, React.js. Svi koriste javascript kao kodni jezik.

2.5. Visual Studio Code

Visual Studio Code[8] je besplatan progam u kojem se uređuje kod, napravljen je od strane Microsofta. On sadrži podršku za uklanjanje pogrešaka, ima ugrađenu Git-ovu kontrolu, ima dobar sistem označavanja sintaksa. Podržava velik broj programskih jezika i velik broj mogućnosti koje možda nisu dostupne za određeni programski jezik.

Visual Studio Code[9] je baziran na Elektornu, odnosno JavaScript aplikaciji napravljenoj uz pomoć Elektrona. U njemu se nalaze svakakve korisne stvari koje omogućuju lakši rad na aplikaciji. Command Palette je svojstvo koje dozvoljava da se izvrši određena radnja umjesto da se traže opcije po menijima unutar editora. U Visual Studio Code-u se veoma lako podesi mapa u kojoj su nam projekti, i to sve na način da nam bude što preglednije. Ima opcije da se mogu otvoriti više datoteka odjednom, te da se može editirati više linija odjednom.



Slika 4. Prikaz korisničkog sučelja u VS Code-u

2.6. Bootstrap

Boostrap[10] je kolekcija CSS klasa koju su napravili frontenderi Twittera. Bootstrap je najpopularniji frontend razvojni okvir koji obuhvaća HTML,CSS i JavaScript jezike. Bootstrap razvojni okvir dolazi sa dodatkom globalnih CSS stilova, stiliziranim HTML elementima koji su unaprijeđenim dodatnim klasama i naprednim sustavom mreže

Bootstrap sadrži mnogo ponovno iskoristivih komponenti koji omogućuju korištenje ikona, padajućih izbornika, navigacija, upozorenja, skočnih prozora i sl. Bootstrap također sadrži i jQuery pluginove koji se lako uključe u projekt. Također se vrlo lako mogu prilagoditi Bootstrapove komponente, LESS varijable i jQuery pluginovi kako bi dobili svoju osobnu verziju koja sadrži stvari koje su nama potrebne.

Bootstrap ima razvijen jako dobar sustav mreže. Mreža u grafičkom dizajnu je skup horizontalnih i/ili vertikalnih linija koje omogućuju strukturiranje sadržaja. Mreže se koriste kod dizajniranja rasporeda i strukture sadržaja kod print dizajna. U web dizajnu mreža je vrlo efektivna metoda za izradu konzistentnog rasporeda elemenata brzo i efikasno koristeći HTML i CSS. Bootstrap[11] uključuje sustav mreže koja je responzivna i prilagođena mobilnim uređajima. Bootstrapova mreža se proteže i do 12 stupaca ovisno o veličini ekrana uređaja. Također sadrži i predefinirane klase i mixine. Kako bi sadržaj stranice bio ispravno pozicioniran (margine, padding) potrebno ga je smjestiti u div sa klasom „.container“. Za izradu horizontalnih grupa stupaca potrebno je koristiti „.rows“ (redove). Sadržaj bi trebao biti smješten unutar stupaca, a samo stupci mogu biti neposredna djeca elementa roditelja. Za brzu izradu rasporeda elemenata unutar grida postoje predefinirane klase poput „.rov“ i .“col-xs-4“. Stupci stvaraju razmake između sadržaja stupaca koristeći padding.

3. Angular

JavaScript kao skriptni jezik omogućava nam mogućnost izvođenja programa na strani klijenta i može se koristiti u njegovom izvornom obliku, no to bi nepotrebno otežalo razvoj i preglednost aplikacije sa stajališta njezine izrade. Koristit će se neki od dostupnih zbirki funkcija ili razvojnih okvira koji će izradu aplikacije znatno olakšati. U ovom će se slučaju koristiti JavaScript razvojni okvir zvan AngularJS.

AngularJS je frontend razvojni okvir koji je razvio Google. To je strukturalni razvojni okvir napravljen na JavaScriptu i služi kako biste mogli napraviti dinamične web aplikacije. On dozvoljava da koristite standardni HMTL i CSS, ali isto tako proširuje mogućnosti HTML-a. Angular komponente su jako korisne, i relativno lako ih je koristiti.

Angular[12] je strukturni razvojni okvir za izradu dinamičnih web aplikacija. Pisan je pomoću JavaScript-a i kao takav potpuno se izvodi na klijentskoj strani. Dizajniran je s idejom CRUD (eng. Create Read Update and Delete), što bi predstavljalo jednostavne aplikacije dizajnirane za izradu, čitanje, ažuriranje i brisanje podataka, bez obrade podataka koja bi se odvijala na strani klijenta. Kako je HTML dizajniran za izradu statičnih stranica, pomoću njega je kreiranje dinamičkih aplikacija nemoguće. Koristeći AngularJS možemo proširiti funkcionalnosti HTML tagova na takav način da ih možemo dinamički obrađivati i dodavati nove mogućnosti koje inače nemamo koristeći HTML.

Neke od značajki AngularJS su tzv. povezivanje podataka, vlastiti sustav za kreiranje i upravljanje komponentama, podrška za upravljanje i validaciju formi, korištenje direktiva itd.

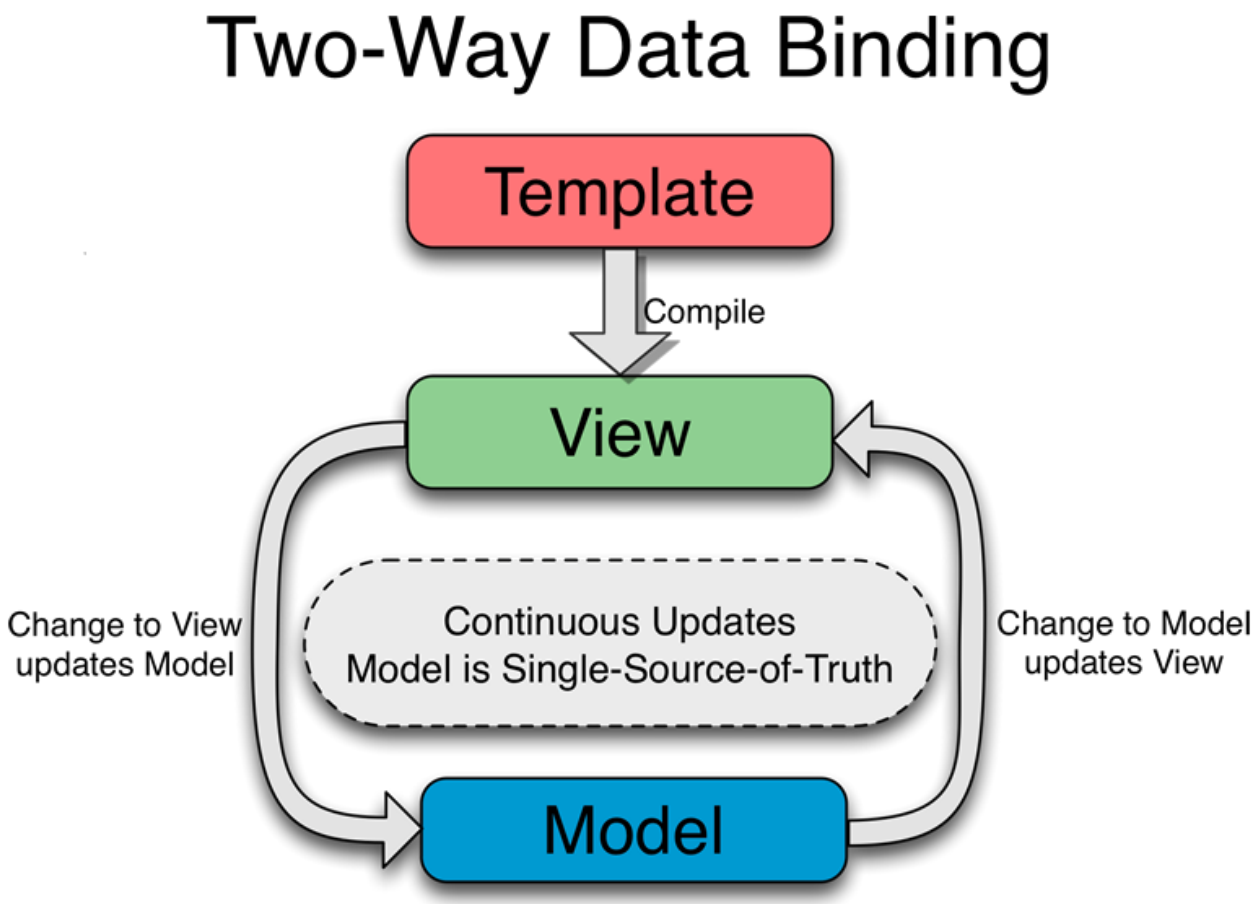
Angularov data binding i dependency injection eliminiraju to da vi morati pisati puno koda.. Angular dopušta da brzo i kvalitetno razvijete klijentski dio aplikacije, tj. “ono što korisnik vidi”. Da bi aplikacija bila potpuna, ona mora imati i svoj backend i bazu.

Angular to ne pokriva. Uvijek se u kombinaciji sa Angularom koristi još neka tehnologija. Za backend se može koristiti Javu, Ruby, C# , .NET-ov webapi i SQL server.

Angular je sve ono što bi HTML trebao biti, da je razvijen za razvoj aplikacija. HTML je odličan deklarativni jezik za statičke dokumente. S njim ne možete razvijati logiku i aplikacije kao takve i tu dolaze spomenuti razni razvojni okviri kao što su Angular, React …

Angular[13] pojednostavljuje razvoj aplikacija tako što daje jednu razinu apstrakcije programeru, ali to može utjecati na fleksibilnost aplikacije. Angular nije rješenje za svaku aplikaciju, ali za većinu CRUD aplikacija (Create, Read, Update, Delete) je i više nego dobar. Nije dobar za razvoj igara i GUI editora koji puno manipuliraju DOM elementima.

One su potpuno različite od klasičnih CRUD aplikacija i one koriste neka druga rješenja. No, većina poslovnih aplikacija spadaju u tu definiciju u kojoj je Angular dobar.



Slika 5. Angular povezan sa HTML-om pomoću javascript objektima.

3.1. Osnovna struktura Angulara

Angular aplikacija[14] sastoji se od mnoštva komponenti koje tvore stablo. Kako uočavamo da skup komponenti tvori logičku cjelinu, taj skup grupiramo u modul (eng. module). Svaka aplikacija stvorena Angular okvirom mora se sastojati od barem jednog modula, a taj modul treba sadržavati korijensku komponentu. Ostale komponente se ugnježđuju unutar korijenske komponente. Modul je TypeScript klasa koja grupira skup funkcionalnosti u jednu cjelinu. Funkcionalnosti koje Angular okvir pruža su: komponente, direktive, filteri, servisi i moduli. Kako bismo naznačili da se radi o modulu, prije njegove deklaracije smještamo dekorator @Module iz @angular/core biblioteke. Dekoratoru prosljeđujemo objekt koji sadrži informacije o tome koje su komponente, direktive i filteri deklarirani unutar modula, koji su servisi pruženi unutar modula, koji su dodatni moduli uveženi u modul, te, ukoliko se radi o korijenskom modulu, koja je korijenska komponenta modula. Nepisano je pravilo da se klasa koja predstavlja korijenski modul naziva AppModule. Taj korijenski modul uvozi BrowserModule modul koji deklarira sve ugrađene funkcionalnosti, poput direktiva i filtera, u svoj djelokrug. Tada se te funkcionalnosti, npr. NgIf direktiva ili async filter, mogu koristiti uz komponente deklarirane unutar modula. Naredbom platformBrowserDynamic().bootstrapModule(AppModule) vršimo postupak samopokretanja (eng. bootstrapping) modula i time naznačavamo da će se Angular aplikacija izvršavati unutar preglednika i to tako da uz aplikaciju dolazi i interni Angular prevoditelj koji vodi računa o generiranju dijelova aplikacije, odnosno on prevodi aplikaciju u trenutku kada je ona otvorena unutar preglednika. Zato se takav postupak prevođenja naziva just-in-time. Uz just-in-time, postoji i ahead-of-time prevođenje gdje se aplikacija unaprijed generira bez posredstva Angular prevoditelja unutar preglednika.

3.2. Izgradnja aplikacije Angulara

Za izgradnju aplikacije uobičajno je koristiti Webpack[15] - alat koji mnoštvo manjih datoteka veže u nekolicinu većih paketa, a između ostaloga, vodi računa i o tome da se TypeScript kod prevede u JavaScript kod pomoću TypeScript prevoditelja. Također, Webpack vodi računa o unošenju "3rd party" biblioteka o kojima ovisi aplikacija, ali i o tome da smanji veličinu koda uz postupke umanjivanja (eng. minify) i poružnjivanja (eng. uglify). AngularCLI interno koristi Webpack, pa koristeći naredbu ng build uz dodatne zastavice gradimo aplikaciju ukoliko je ona inicijalizirana pomoću tog alata. Nakon što je kod preveden i paketi izgrađeni, oni se uvoze unutar glavnog HTML dokumenta i time tvore web aplikaciju. Bitno je naglasiti da se unutar oznake body HTML dokumenta mora nalaziti oznaka koja odgovara selektoru korijenske komponente korijenskog modula kako bi se sav prikaz kojeg predstavlja aplikacija mogao pričvrstiti na tu oznaku.

3.3. Komponente Angulara

Osnovna gradivna jedinica Angular aplikacije je komponenta.[16] Koncept komponente generalno se pojavljuje u softverskom inženjerstvu kao nezavisna cjelina koda koja samostalno obavlja neku funkcionalnost nudeći i zahtjevajući pritom različita sučelja. Konkretno, u slučaju Angular aplikacije, sučelje neke komponente sastoji se od ulaznih svojstava (eng. input properties) i izlaznih događaja (eng. output events). Komponente se koriste zbog nekoliko praktičnih i očiglednih razloga. Naime, sama se komponenta unutar Angular aplikacije pojavljuje u obliku TypeScript klase. Unutar te klase ubacimo aplikacijsku logiku naše komponente tako da uočimo koncept koji ta komponenta treba predstavljati te deklariramo i definiramo članove koji se trebaju nalaziti unutar komponente. Budući da je komponenta definirana klasom, više instanca te komponente može se pojaviti u različitim dijelovima aplikacije, a time se postiže ponovna upotrebljivost komponente. Unutar komponente moguće je ukomponirati i druge komponente. Komponente sastavljene od drugih komponenata time tvore stablo, vjerno oponašajući samu strukturu HTML DOM-a (Document Object Model) - svaka komponenta tvori jedan čvor stabla i prikazuje se unutar preglednika.

Komponentu moramo deklarirati unutar modula u kojem se nalazi, stoga prije ključne riječi class koristimo ključnu riječ export kako bismo mogli uvesti komponentu u druge dijelove aplikacije. Iz @angular/core biblioteke uvozimo @Component dekorator kojeg postavljamo prije deklaracije klase i prosljeđujemo mu objekt s odgovarajućim svojstvima. Objekt koji prosljeđujemo govori na koji će se način vršiti prikaz HTML koda vezanog uz komponentu. Prije svega, atribut selector sadrži string kojim se određuje kojom će oznakom komponenta biti pozvana unutar neke druge komponente. Sve se komponente uvijek pozivaju iz drugih komponenata, osim ukoliko se radi o korijenskoj komponenti nekog modula. U ovom slučaju, komponentu pozivamo oznakom . Drugim riječima, komponente se ugnježđuju.

Vizualni dio komponente naziva se prikaz (eng. view), a njega definira predložak (eng. template). Predložak se sastoji od HTML koda i koda specifičnog za Angular aplikaciju. Npr., korištenje dvostrukih vitičastih zagrada koje u domeni Angular aplikacije, ali i drugih okvira, omogućavaju interpolaciju stringa (eng. string interpolation). Umjesto atributa template, možemo koristiti atribut templateUrl čija je vrijednost string relativnog puta do odgovarajućeg koda predloška, a time se postiže bolja preglednost budući da, koristeći atribut template, nije jednostavno pisati HTML i Angular kod unutar JavaScript stringa. Kako koristimo HTML, to je moguće dodati i CSS kod kojim uljepšavamo vizualni prikaz i definiramo stil izgleda, a jednako tako, i ovdje koristimo ili atribut styles čija je vrijednost JavaScript polje koje se sastoji od stringova koji predstavljaju CSS kod ili atribut stylesUrl čija je vrijednost polje stringova relativnih puteva do odgovarajućih CSS datoteka. Kao i u slučaju predloška, ukoliko “izbacimo” CSS stil izvan komponente u drugu CSS datoteku, dobivamo bolju preglednost koda. Prilikom prikazivanja komponente, posebno se treba voditi računa o tome na koji će način komponentna biti prikazana, odnosno enkapsulirana unutar vizualnog izgleda cijele aplikacije. Taj se princip u Angular okviru naziva enkapsulacija prikaza (eng. view encapsulation). Kako smo već spomenuli, prikazu komponente pridružujemo odgovarajući CSS stil kojeg na prijašnje opisani način uvodimo unutar komponente.

3.4. Angular moduli

Aplikacije u Angular-u su modularne i Angular ima svoj sustav modula (engl. modularity system) koji se naziva NgModules. Svaka aplikacija sadrzi barem jednu klasu modul (engl. module class) koji nazivamo korijenski modul (engl. the root module), u kodu obicno nazvan AppModule. On je dovoljan samo za jako male aplikacije. Veće aplikacije s više mogućnosti

sadrže veći broj modula, a svaki taj modul predstavlja blok koda posvećen jednoj značajki aplikacije. Modul je klasa koja ima @NgModule dekorator. NgModule je dekoratorska funkcija koja uzima jedan metapodatak objekt (engl. metadata object) čija svojstva opisuju modul.
Neke od najvažnijih svojstva kojima opisujemo module su:

- declarations - klase pogleda (engl. view classes) koje pripadaju modulu. Angular ima tri tipa takvih klasa pogleda, a to su komponente (engl. components), direktive (engl. directives) i pipes. • exports - podskup deklaracija koje bi trebale biti vidljive i upotrebljive u predlošcima komponenata (engl. component templates) iz drugih modula.
- imports - moduli čije su izvozne klase (engl. exported classes) potrebne predlošcima komponenata deklariranih u ovom modulu.
- providers - popis usluga (engl. services) koje ovaj modul pruža. Te usluge postaju dostupne svim dijelovima aplikacije.
- bootstrap - glavni pogled aplikacije, koji se naziva korijenska komponenta (engl. the root component). On sadrži sve ostale poglede aplikacije. Samo glavni modul, tj. AppModule smije postaviti ovo svojstvo.

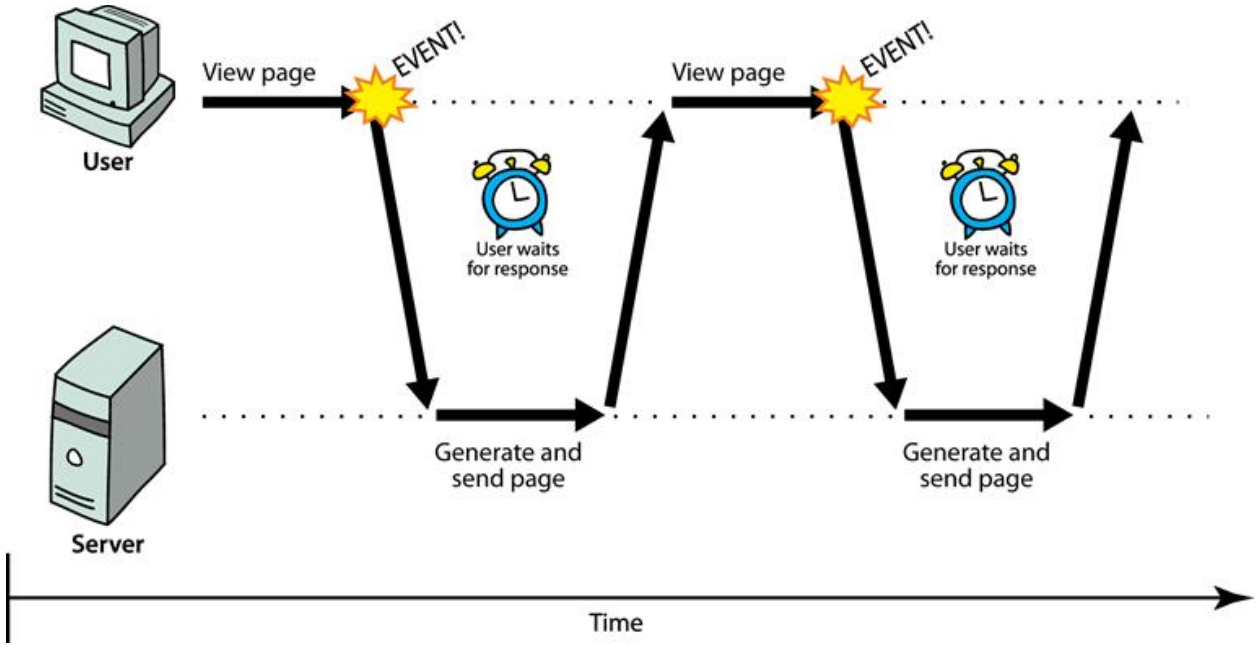
```
1 i m p o rt { NgModule } from ' @angular c o r e ' ;
2 i m p o rt { B rowse rModule } from ' @angular pl at f o r m −b r ow se r ' ;
3 @NgModule ( {
4       i m p o r t s          [ B rowse rModule ] ,
5       p r o v i d e r s      [ L o g ge r ] ,
6       d e c l a r a t i o n s  [ AppComponent ] ,
7       e x p o r t s          [ AppComponent ] ,
8       b o o t s t r a p      [ AppComponent ]
9 } )
10 e x p o r t  c l a s s  AppModule { }
```

Kod 1. App.module

Aplikacija se pokreće podizanjem modula AppModule.

4. NodeJS

NodeJS[17] je serverska JavaScript platforma koja sadrži velik dio biblioteke JavaScript-a. Radi na V8 JavaScript engine-u, a to znači da je brz u izvršavanju procesa. Omogućava korištenje jedinstvenog jezika između front-end-a i back-end-a, to znači da je veoma lako realizirati aplikaciju pomoću jednog programskog jezika. Njegova arhitektura je idealna za rad na real-time web aplikaciji. Programiranje u NodeJS-u se obavlja sinkronizirano, to znači da se linija koda izvršava, sistem čeka rezultat, rezultat se procesuira i zatim se izvršavanje programa nastavlja.
Glavni cilj Node.js platforme je pružiti siguran i lak način za izgradnju mrežnih aplikacija visoke performanse u JavaScriptu.JavaScript se tradicionalno izvršava u web pregledniku, odnosno zadaci pisani u tom programskom jeziku se osvježavaju u korisničkim pretraživačima. Stoga se još prije dva desetljeća javila potreba za pokretanjem na poslužiteljskoj razini te se Node.js pokazao kao najpraktičnije rješenje za to.



Slika 6. Proces u NodeJS-u

NodeJS sadrži podršku za upravljanje paketima pomoću NPM alata koji dolazi sa svakom NodeJS instalacijom. Neki od najpopularnijih NPM modula su : express, connect, socket.io, Jade, mongo, redis, forever i sl.

4.1. Prednosti NodeJS-a

Prednost prije svega je ta da je Node.js izutetno popularna platforma, koja predstavlja drugi najgledaniji projekt na GitHub-u, ima velik broj pratioca u svojoj Google grupi i IRC kanalu i više od 15.000 modula zajednice objavljenih u menadžerskom paketu NPM.

Najosnovnija namjena Node.js-a je da služi za kreiranje servera za web aplikacije. S obzirom da Node.JS koristi JavaScript na serveru postoje i druge prednosti. JavaScript je jezik koji se koristi u raznim NoSQL bazama podataka tako da je povezivanje sa njima znatno usklađeno. Programeri mogu pisati web aplikacije na jednom jeziku, koji omogućava dijeljenje koda između klijenta i servera, putem „context“ prekidača. Node.JS podaci se čuvaju u JSON obliku, što omogućava da se cjelokupna poslovna logika, kao i pristup bazi, bazira na jednom programskom jeziku, jer je uklonjena i potreba za konverzijom formata podataka.

Node.JS je izuzetno brza programska platforma.[18] Zajedno sa Googl-ovim V8 JavaScript engine-om ostvaruje nevjerojatnu brzinu. Nekoliko puta je brži od skriptnih jezika kao što su Ruby i Python. Neblokirajuća arhitektura Node.JS-a je idealna za pravljenje real-time web aplikacija. JavaScript je izabran kao jezik zbog svog malog core API-ja i nonblocking callbacks-a. To znači da je moguće izgraditi kompletan Node.JS ekosistem oko neblokirajućih paketa. Node.js omogućava korištenje jedinstvenog jezika između front-end i back-end aplikacije, što znači da se kompletna aplikacije može realizirati pomoću jednog programskog jezika. To je od posebnog značaja jer za konstrukciju web aplikacije je potrebno znati dosta programskih jezika. Razlog i za porast popularnosti Node.JS-a je uvođenje NPM (Node Package Manager), koji pruža programerima jednostavan način instalacije i upravljanje modula zajednice.

Node.js namijenjen je za rad na namjenskom HTTP poslužitelju. Node.js aplikacije su događaji pokrenuti asinkrono. Programski kod napisan na Node.js platformi ne slijedi tradicionalni model primanja, obrade, slanja i čekanja podataka. Node.js obrađuje dolazne zahtjeve i stalne događaje šaljući zahtjeve jedan za drugim, bez čekanja odgovora koristeći događaje. Platforma je bazirana na Google-ovom V8 engine-u, što mu je istovremeno i prednost i mana. Prednost je iz razloga što je brz i aplikacije se na obje strane (na poslužiteljskoj i korisničkoj) pišu u jednom jeziku.

4.2. NPM

Kada se priča o Node.js-u, svakako se mora napomenuti korištenje Node Package Managera (NPM). To je alat koji dolazi zajedno sa svakom Node.js instalacijom. Ideja NPM nadogradnje je javna dostupnost dodatnih komponenti i resursa koji su dostupni kroz jednostavnu instalaciju. NPM služi kao razvojni okvir (hrv. programerski okvir) za Node.js development (hrv. razvoj projekta) s velikom količinom raznih dodataka za razne Node.js aplikacije. NPM funkcionira na način da stvori jednostavan web poslužitelj iz datoteke koja se nalazi u projektu.

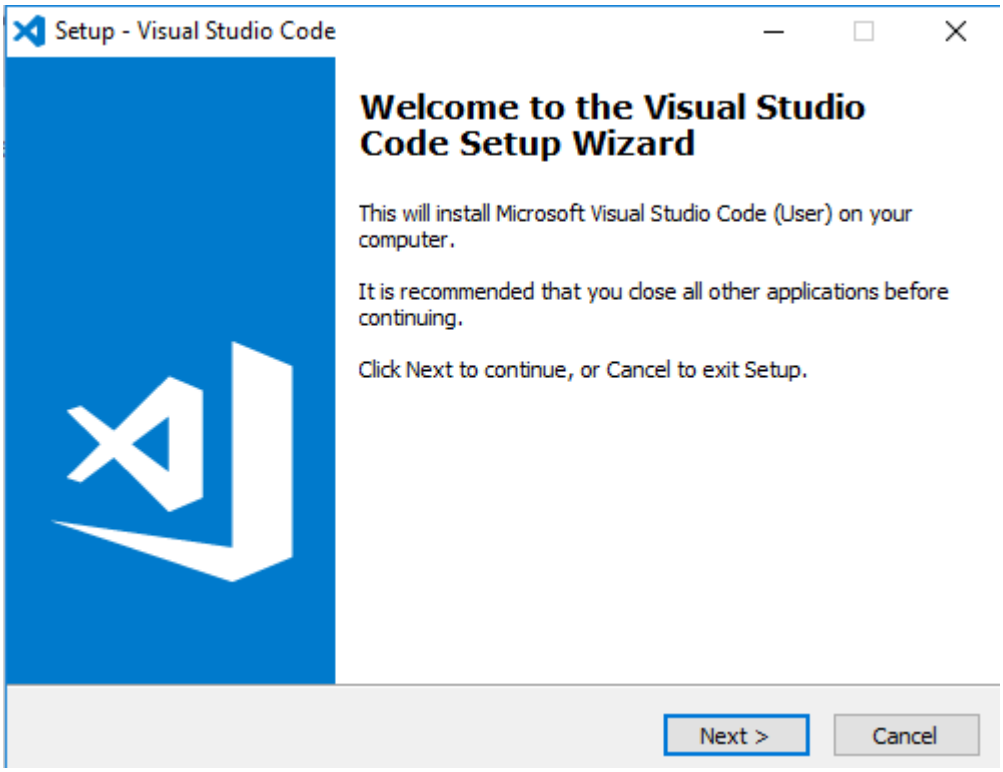
NPM se može koristiti u nizu primjera: poslužuje GUI za preglednike (dostupno na http://localhost:1337/), modificira package.json, pokreće naredbe s klijentske strane (npr. „npm install angular –save“), prebacuje konzolu preko websocket-a do klijentske strane i tako dalje. Neki od najpopularnijih NPM package-a (paketa odnosno biblioteka) su: Lodash, Request, Async, Underscore i Express.

5. Praktični rad

U ovome poglavlju će biti pobliže objašnjena izrada aplikacije za prikaz vremenske prognoze diljem Europe putem Angulara 6. Prvi korak je bila instalacija Visual Studio Code-a, NodeJS-a, AngularCLI. Sljedeći korak je bila izrada novog projekta sa nazivima mapa, datoteka i sl. Nakon toga je prikazano kako se izradila aplikacija, kroz koje stavke i na koji način. Programski kod koji je vidljiv prikazuje kako se aplikacija razvija. Kao posljednji korak su objašnjena sredstva koja su bila koristna u kreiranju aplikacije.

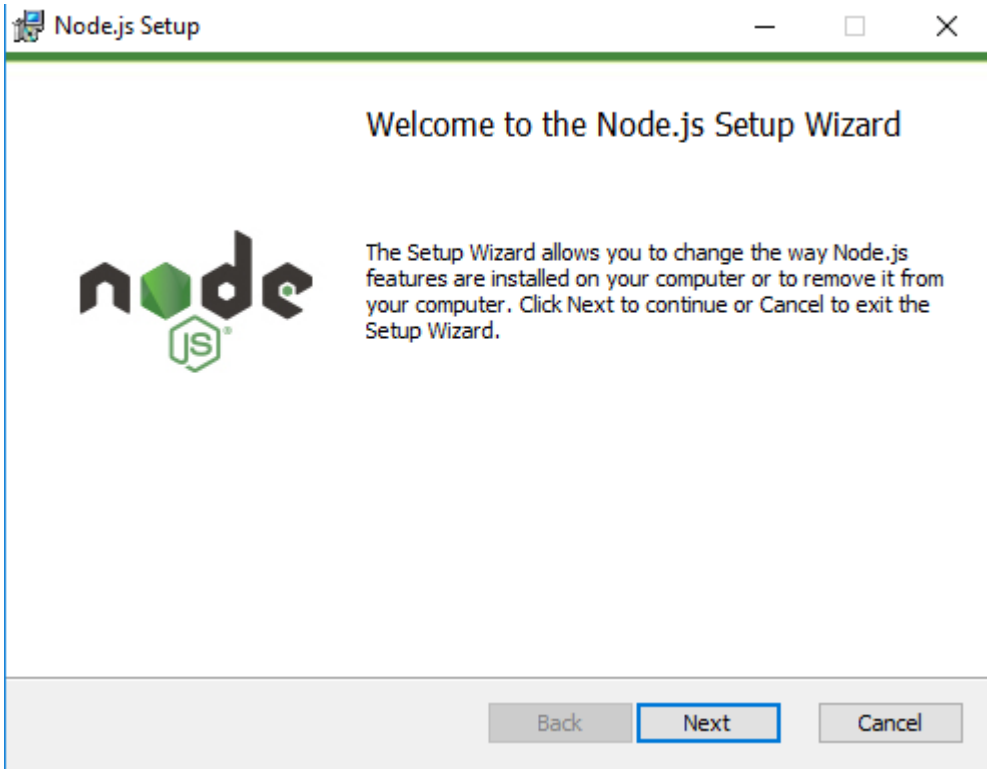
5.1. Instalacija alata

Prvi korak kod instalacije alata je instalacija Visual Studio Code-a koji će poslužiti za pisanje Typescript,Javascript i CSS koda za željenu aplikaciju.



Slika 7. Instalacija Visual Studio Code

Sljedeći korak je instalacija NodeJS-a koji će poslužiti kao brza javascript platforma koja omogućava korištenje jezika između back-end-a i front-end-a.



Slika 8. Instalacija NodeJS-a

Nakon toga je slijedila instalacija Angular CLI, pomoću kojeg se u command promptu napravio projekt, dodale nove datoteke i pokretala aplikacija.
Za to su se koristile tri funkcije (naredbe):

- ng new- pomoću kojeg se kreirao novi projekt
- ng generate- pomoću kojeg su se dodale nove datoteke potrebne za aplikaciju
- ng serve- pomoću kojeg se pokretala aplikacija, te se prikazivala na http://localhost:4200/ adresi

Na kraju, za potrebe lakšeg kreiranja aplikacije je bila potrebna instalacija Bootstrapa kroz command prompt putem NodeJS-a.

5.2. Izrada novog projekta u Angularu

Kreiranje glavne datoteke u Angulara se radi uz pomoć Angular CLI-a koji kroz terminal(cmd) naredbom `ng new` „naziv datoteke“ kreira željenu datoteku. Angular CLI nam stvori četiri datoteke: HTML, CSS ,Typescript i JavaScript.

```
PS C:\Users\Filip> ng new vremenska-prognoza1
CREATE vremenska-prognoza1/angular.json (3665 bytes)
CREATE vremenska-prognoza1/package.json (1324 bytes)
CREATE vremenska-prognoza1/README.md (1035 bytes)
CREATE vremenska-prognoza1/tsconfig.json (408 bytes)
CREATE vremenska-prognoza1/tslint.json (2805 bytes)
CREATE vremenska-prognoza1/.editorconfig (245 bytes)
CREATE vremenska-prognoza1/.gitignore (503 bytes)
CREATE vremenska-prognoza1/src/favicon.ico (5430 bytes)
CREATE vremenska-prognoza1/src/index.html (305 bytes)
CREATE vremenska-prognoza1/src/main.ts (370 bytes)
CREATE vremenska-prognoza1/src/polyfills.ts (3194 bytes)
CREATE vremenska-prognoza1/src/test.ts (642 bytes)
CREATE vremenska-prognoza1/src/styles.css (80 bytes)
CREATE vremenska-prognoza1/src/browserslist (375 bytes)
CREATE vremenska-prognoza1/src/karma.conf.js (964 bytes)
CREATE vremenska-prognoza1/src/tsconfig.app.json (166 bytes)
CREATE vremenska-prognoza1/src/tsconfig.spec.json (256 bytes)
CREATE vremenska-prognoza1/src/tslint.json (314 bytes)
CREATE vremenska-prognoza1/src/assets/.gitkeep (0 bytes)
CREATE vremenska-prognoza1/src/environments/environment.prod.ts (51 bytes)
CREATE vremenska-prognoza1/src/environments/environment.ts (642 bytes)
CREATE vremenska-prognoza1/src/app/app.module.ts (314 bytes)
CREATE vremenska-prognoza1/src/app/app.component.html (1141 bytes)
CREATE vremenska-prognoza1/src/app/app.component.spec.ts (1034 bytes)
CREATE vremenska-prognoza1/src/app/app.component.ts (223 bytes)
CREATE vremenska-prognoza1/src/app/app.component.css (0 bytes)
CREATE vremenska-prognoza1/e2e/protractor.conf.js (752 bytes)
CREATE vremenska-prognoza1/e2e/tsconfig.e2e.json (213 bytes)
CREATE vremenska-prognoza1/e2e/src/app.e2e-spec.ts (315 bytes)
CREATE vremenska-prognoza1/e2e/src/app.po.ts (208 bytes)
```

Slika 9. Kreiranje datoteke u Angularu

Za pokretanje aplikacije nam služi naredba `ng serve` koja na http://localhost:4200/ adresi pokazuje aplikaciju koja je napravljena.

```
PS C:\Users\Filip\Desktop\vremenska prognoza-zavrsni-rad\vremenska-prognoza> ng serve
** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **

Date: 2018-08-31T17:27:17.121Z
Hash: c82d1e3f50f2a2fc070a
Time: 9730ms
chunk {main} main.js, main.js.map (main) 30.8 kB [initial] [rendered]
chunk {polyfills} polyfills.js, polyfills.js.map (polyfills) 227 kB [initial] [rendered]
chunk {runtime} runtime.js, runtime.js.map (runtime) 5.22 kB [entry] [rendered]
chunk {styles} styles.js, styles.js.map (styles) 15.6 kB [initial] [rendered]
chunk {vendor} vendor.js, vendor.js.map (vendor) 3.59 MB [initial] [rendered]
```

Slika 10. Pokretanje aplikacije u Angularu

5.3. Angular komponente

Nakon stvaranja projekta nalazi se komponenta pod nazivom app. Ta komponenta se sastoji od četiri datoteke :

- app.component.html – u ovoj se datoteci nalazi kod u HTML jeziku i služi kao predložak komponente
- app.component.css- u ovoj se datoteci nalazi kod u CSS jeziku i služi kao dizajn za tu komponentu
- app.component.ts- u ovoj se datoteci nalazi pozadinska logika iza komponente
- app.component.spec.ts- ovo je konfiguracijska datoteka

Komponentu prepoznajemo po dekoratoru Component koji se nalazi ispred klase od app.component. U dekoratoru se definira selektor pomoću kojeg se pronalazi komponenta. Definira se predložak i stilovi za komponentu.

```
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
```

Kod 2. Dekorator

U toj komponenti se nalazi navigacija i direktiva router-outlet. Router-outlet se nalazi na mjestu u kojem se prikazuje glavni dio stranice.

```
<ul class="navbar-nav ">
      <li class="nav-item dropdown">
        <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown4"
role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
          Jugoistočna Europa
        </a>
        <div class="dropdown-menu" aria-labelledby="navbarDropdown4">
          <a class="dropdown-item" href="/prognoza/523920">Varšava</a>
          <a class="dropdown-item" href="/prognoza/804365">Budimpešta</a>
          <a class="dropdown-item" href="/prognoza/851128">Zagreb</a>
        </div>
      </li>

    </ul>
```

Kod 3. Prikaz padajućeg izbornika iz navigacije

Nakon toga je dodana nova komponenta pod nazivom main-view. Kreirana je s naredbom `ng generate component main-view`. Naredba automatski generira četiri datoteke. U toj komponenti se nalazi prikaz vremenske prognoze za odabrani grad iz navigacije.

```
<div class="container">
<div *ngFor="let item of lista" class="card">
    <div class="card-body">
      <h5 class="card-title">{{item.datum}}</h5>
      <img class="card-img" src="{{item.poveznica}}" style="width: 50%; height:
50%" alt="Card image">
      <h3 class="card-text">{{item.temp | number : '1.0-0'}} °C</h3>
    </div>
    <ul class="list-group list-group-flush">
      <li class="list-group-item">Min. temperatura: {{item.min_t| number : '1.0-
0'}} °C</li>
      <li class="list-group-item">Max. temperatura: {{item.max_t| number : '1.0-
0'}} °C</li>
      <li class="list-group-item">Brzina vjetra: {{item.brzina_vjetra| number :
'1.0-0'}} km/h</li>
      <li class="list-group-item">Vlažnost: {{item.vlaznost| number : '1.0-0'}}
%</li>
      <li class="list-group-item">Tlak: {{item.tlak_zraka| number : '1.0-0'}} hPA
</li>
    </ul>
  </div>
</div>
```

Kod 4. Predložak za main-view komponentu

U ovom predlošku su smještene kartice. Svaka kartica označava vremensku prognozu za jedan dan, ukupno ih je šest, počevši od danas. U kartici se nalaze datumi za tih šest dana, slikovni prikaz vremena, minimalna temperatura, maksimalna temperatura, brzina vjetra, vlažnost i tlak zraka. Podaci prikazani u kartici se dohvaćaju iz liste definirane u typescript datoteci te komponete.

U datoteci sa typescript kodom je definirana lista, te se poziva servis.

```
export class MainViewComponent implements OnInit {

  id: number;
  lista: VremenskaPrognoza[];
  grad: string;

  constructor(
    private route: ActivatedRoute,
    private service: ApiService
  ) { }
```

```
  async ngOnInit() {
    this.id = +this.route.snapshot.paramMap.get('id');
    this.lista = await this.service.getPrognoza(this.id);
  }

}
```

Kod 5. Main-view.component.ts

U datoteci s CSS kodom je definiran dizajn kartica u aplikaciji, u ovom kodu se radi centriranje kartica i postavljanje margina na karticama.

```
.container {
    display: flex;
    justify-content: center;
    align-items: center;
}
.card {
    margin-top: 20px;
    margin-right: 10px;
}
```

Kod 6. Main-view.component.css

5.4. Angular servisi

Servisi u Angularu služe za međusobnu interakciju komponenata, te komunikaciju aplikaciji sa udaljenim REST servisima. Naredbom ng generate service api je stvoren novi servis pod nazivom api. Ovaj servis služi za spajanje s udaljenim servisom i dohvatom informacija o vremenskoj prognozi. Podaci su preuzeti sa sljedećeg URL-a: https://www.metaweather.com/api/ Podaci su razmijenjeni u JSON formatu.

```
import { Injectable } from '@angular/core';
import { HttpClient, HttpHeaders } from '@angular/common/http';

import { Observable, of } from 'rxjs';
import { catchError, map, tap } from 'rxjs/operators';
import { VremenskaPrognoza } from '../models/vremenska-prognoza';
```

Kod 7.Uvoz modula

Na početku servisa se uvoze potrebni moduli i biblioteke. Injectable iz modula angular/core služi za kreiranje servisa i omogućuje njegovo ubacivanje u klase koje zahtjevaju taj servis. HttpClient i HttpHeaders iz angular/common/http modula, služe za komunikaciju s udaljenim servisom.

HttpClient ima ugrađene metode get,post,put,delete i druge. VremenskaPrognoza je model koji prezentira vremensku prognozu za jedan dan.

```
@Injectable({
  providedIn: 'root'
})
```

Kod 8. Dekorator za servis

```
export class ApiService {
  private heroesUrl = 'https://www.metaweather.com/api/location/';
  constructor(
    private http: HttpClient
  ) { }

  getPrognoza(id: number) {
    const URL = `${this.heroesUrl}${id}/`;
    const lista: VremenskaPrognoza[] = [];
    this.http.get(URL).subscribe(response => {
      const prognoze: any[] = response['consolidated_weather'];
      prognoze.forEach(_ => {
        let vp: VremenskaPrognoza = new VremenskaPrognoza();
        vp.grad = response['title'];
        vp.datum = _['applicable_date'];
        vp.max_t = _['max_temp'];
        vp.min_t = _['min_temp'];
        vp.temp = _['the_temp'];
        vp.tlak_zraka = _['air_pressure'];
        vp.vlaznost = _['humidity'];
        vp.brzina_vjetra = _['wind_speed'];
```

…

```
    lista.push(vp);
      });
    });
    return lista;
```

Kod 9. Metoda za dohvat podataka

Prvo je definiran URL na kojem se nalaze podaci za vremensku prognozu. U kontruktoru je pomoću dependency injection-a stvoren http objekt. U metodi getPrognoza je prihvaćen parametar koji predstavlja oznaku grada. Nad objektom http je pozvana metoda pod nazivom get, a odgovor je obrađen i stvorena je lista vremenskih prognoza za zahtjevani grad.

5.5. Angular moduli

Kada je napravljen novi projekt, u sklopu tog projekta je generiran app.module koji je glavni i korijenski modul cijele aplikacije. Moduli su datoteke u kojima su definirani svi dijelovi programa.

```
import { BrowserModule } from '@angular/platform-browser';
import { HttpClientModule } from '@angular/common/http';
import { NgModule } from '@angular/core';

import { AppComponent } from './app.component';
import { MainViewComponent } from './components/main-view/main-view.component';

import { AppRoutingModule } from './/app-routing.module';
import { ApiService } from './services/api.service';

@NgModule({
  declarations: [
    AppComponent,
    MainViewComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    HttpClientModule
  ],
  providers: [ApiService],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Kod 10. Sadržaj app.module

Pri vrhu datoteke su uvezeni već ugrađeni moduli koji će biti potrebni za daljni razvoj aplikacije. U datoteci su još uvezeni i komponente i servisi. U tu datoteku je također uvezen routingmodule koji je zadužen za koordiniranje navigacijom. Module prepoznajemo po dekoratoru NgModule. Dekorator se sastoji od nekoliko polja (array). U polju declarations se nalaze komponente, u polju imports ostali moduli, a u polju providers svi servisi.

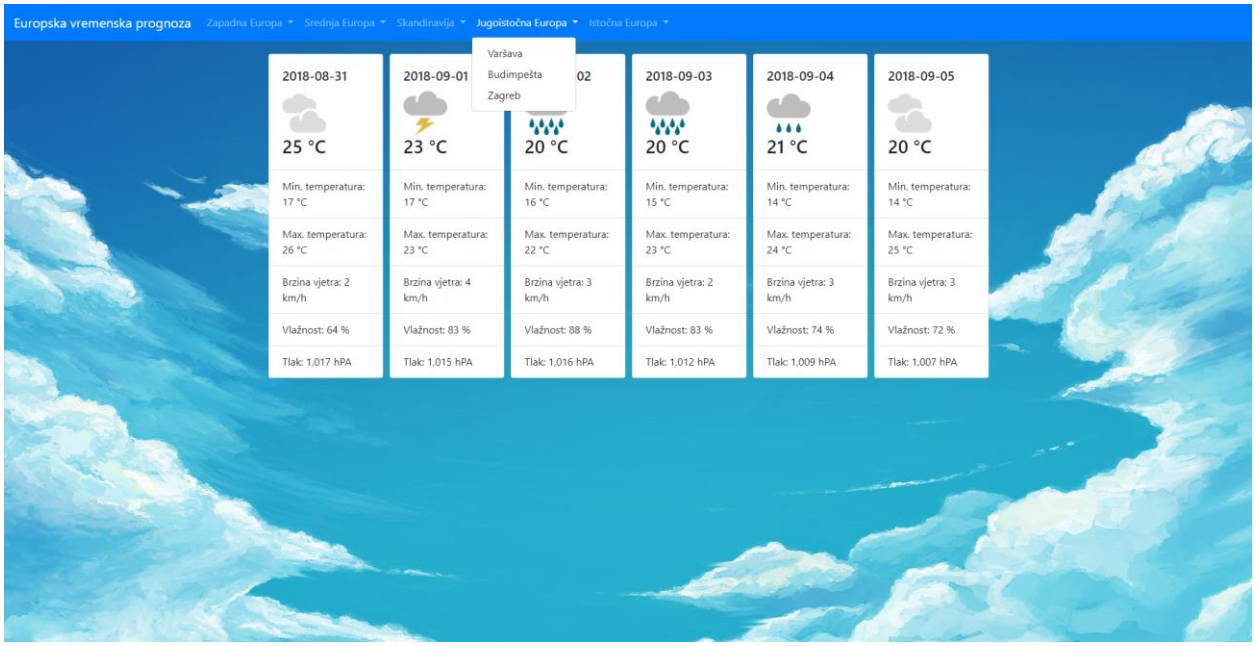
```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';

import { CommonModule } from '@angular/common';
```

```
import { MainViewComponent } from './components/main-view/main-view.component';

const routes: Routes = [
  { path: '', redirectTo: 'prognoza/851128', pathMatch: 'full' },
  { path: 'prognoza/:id', component: MainViewComponent }
];

@NgModule({
  imports: [
    CommonModule,
    RouterModule.forRoot(routes)
  ],
  exports: [ RouterModule ]
})
export class AppRoutingModule { }
```

Kod 11. Sadržaj AppRoutingModule

Potrebno je izdvojiti iz uvezenih modula RouterModule i Routes iz modula pod nazivom angular/router. U polju pod nazivom Routes su definirane rute za aplikaciju. Kada se aplikacija pokrene router usmjeruje na vremensku prognozu za grad Zagreb. Objekt Routes se sastoji od svojstava path koji predstavlja putanju i component koji označava komponentu za prikaz.

5.6. Konačan izgled aplikacije



Slika 11. Konačan izgled aplikacije

Aplikaciju možete koristiti na ovome URL-u :
https://vremenska-prognoza-36301.firebaseapp.com/prognoza/796597
Ali vam je potrebna google chrome proširenje, bez kojega ne biste mogli dohvaćati podatke vremenske prognoze diljem svijeta, pa je i za to postavljen URL:
https://chrome.google.com/webstore/detail/allow-control-allow-origi/nlfbmbojpeacfghkpbjhddihlkkiljbi?utm_source=gmail

6. Zaključak

Angular je definitivno jedan od najpopularnijih razvojnih okvir-a današnjice. Njime se koriste sve veće tvrtke u svijetu pa čak i u Hrvatskoj. Jednostavan je za primjenu i za učenje. Najčešće se koristi za izradu single-web aplikacija. Angular je jedan od prvih, tzv. deklarativnih razvojnih okvira koji donosi mnoštvo preinaka u razvoj web aplikacija, na način koji je ljudima razumljiviji, te koje isto tako poboljšavaju kvalitetu koda i čine ga lakšim za održavanje.

Angular je vrlo fleksibilan što se tiče serverske komunikacije, pa stoga dopušta rad sa bilo kojom server-side tehnologijom. Na kraju se postavlja pitanje zašto Angular? Zato jer je open-source i na njemu konstantno radi Google tim, koji taj razvojni okvir održava i usavršava. Sveobuhvatan je, veoma je brzo rješenje za brz razvoj front-end dijela developmenta. Nisu mu potrebni drugi plugin-ovi ili razvojni okviri. Veoma je pregledan i dobro grafički osmišljen, ima velik broj opcija koje pojednostavljuju korištenje, lako se počinje.

Sa svim tim saznanjima bitno je reći da Angular gradi budućnost ljudi koji se bave nekakvim oblikom web developmenta.

U Varaždinu 28.09.2018. Tvrtko

Datum Potpis

33

Sveučilište Sjever

IZJAVA O AUTORSTVU
I
SUGLASNOST ZA JAVNU OBJAVU

Završni/diplomski rad isključivo je autorsko djelo studenta koji je isti izradio te student odgovara za istinitost, izvornost i ispravnost teksta rada. U radu se ne smiju koristiti dijelovi tuđih radova (knjiga, članaka, doktorskih disertacija, magistarskih radova, izvora s interneta, i drugih izvora) bez navođenja izvora i autora navedenih radova. Svi dijelovi tuđih radova moraju biti pravilno navedeni i citirani. Dijelovi tuđih radova koji nisu pravilno citirani, smatraju se plagijatom, odnosno nezakonitim prisvajanjem tuđeg znanstvenog ili stručnoga rada. Sukladno navedenom studenti su dužni potpisati izjavu o autorstvu rada.

Ja, FILIP TUĐAN *(ime i prezime)* **pod punom moralnom, materijalnom i kaznenom odgovornošću, izjavljujem da sam isključivi autor/ica završnog/diplomskog (*obrisati nepotrebno*) rada pod naslovom** IZRADA WEB APLIKACIJE PUTEM ANGULAR 6 RAZVOJNOG OKVIRA **(*upisati naslov*) te da u navedenom radu nisu na nedozvoljeni način (bez pravilnog citiranja) korišteni dijelovi tuđih radova.**

IZRADA WEB

Student/ica:
(upisati ime i prezime)
Filip Tuđan
(vlastoručni potpis)

Sukladno Zakonu o znanstvenoj djelatnost i visokom obrazovanju završne/diplomske radove sveučilišta su dužna trajno objaviti na javnoj internetskoj bazi sveučilišne knjižnice u sastavu sveučilišta te kopirati u javnu internetsku bazu završnih/diplomskih radova Nacionalne i sveučilišne knjižnice. Završni radovi istovrsnih umjetničkih studija koji se realiziraju kroz umjetnička ostvarenja objavljuju se na odgovarajući način.

Ja, FILIP TUĐAN *(ime i prezime)* **neopozivo izjavljujem da sam suglasan/na s javnom objavom završnog/diplomskog (*obrisati nepotrebno*) rada pod naslovom** IZRADA WEB APLIKACIJE PUTEM ANGULAR 6 RAZVOJNOG OKVIRA **(*upisati naslov*) čiji sam autor/ica.**

Student/ica:
(upisati ime i prezime)
Filip Tuđan
(vlastoručni potpis)

Literatura

[1] https://books.google.hr/books/about/AngularJS.html?id=EjAhSZpynsYC&redir_esc=y, dostupno 20.08.2018.

[2] http://www.angularjsbook.com/, dostupno 20.08.2018.

[3] http://www.portalalfa.com/1/Html/uvod.htm, dostupno 20.08.2018.

[4] https://www.srce.unizg.hr/files/srce/docs/edu/osnovni-tecajevi/c220_polaznik.pdf, dostupno 20.08.2018.

[5] https://www.mojwebdizajn.net/web-programiranje/vodic/javascript/uvod-u-javascript.aspx,dostupno 20.08.2018.

[6] http://www.znanje.org/knjige/computer/JavaScript/2010/index.htm, dostupno 20.08.2018.

[7] https://djangostars.com/blog/what-is-a-web-Razvojni okvir/ ,dostupno 21.08.2018.

[8] https://en.wikipedia.org/wiki/Visual_Studio_Code, dostupno 1.9.2018.

[9] https://pcchip.hr/softver/korisni/10-savjeta-za-vecu-produktivnost-kod-koristenja-visual-studio-codea/, dostupno 1.9.2018.

[10] https://www.slideshare.net/DamjanPavlica/vodi-za-rad-sa-bootstrapom-69948458, dostupno 1.9.2018.

[11] http://m.aplitap.hr/blog/5-razloga-za-koristenje-bootstrap-Razvojni okvira, dostupno 1.9.2018.

[12] https://www.popwebdesign.net/popart_blog/2015/05/angularjs-Razvojni okvir/,dostupno 21.08.2018.

[13] https://pcchip.hr/helpdesk/angular-programiranje-1-dio-uvod-u-angular-4/,dostupno 21.08.2018.

[14] https://docs.angularjs.org/guide/introduction, dostupno 22.08.2018.

[15] https://angular.io/api/core, dostupno 22.08.2018.

[16] https://angular.io/api/core/Component, dostupno 22.08.2018.

[17] https://www.popwebdesign.net/popart_blog/2015/06/sta-je-node-js/, dostupno 1.9.2018.

[18] http://www.academia.edu/23674866/Izrada_web_aplikacije_za_prodaju_putem_interneta_na_osnovi_Node.js_tehnologije, dostupno 3.9.2018.

Popis slika

Popis primjera kodova