Matematički fakultet

Univerzitet u Beogradu

„eŠkola veba“ – elektronska platforma za učenje veb programiranja

Master rad

Mentor: prof. dr Miroslav Marić

Kandidat: Nemanja Jurić

Beograd, avgust 2017.

Sadržaj

1 Uvod

Napredak tehnologije i sveopšta dostupnost računara i informacija doveli su do ekspanzije informaciono-komunikacionih tehnologija i njihove integracije u mnoge segmente savremenog društva. Svakodnevno se pojavljuju novi internet proizvodi i veb alatke. Razvoj tehnologije i Interneta je uslovio potrebu za ljudima koji znaju da programiraju. Danas svako može da počne da uči i usavršava se u ovoj oblasti čak iako nema uslova da pohađa kurs u tradicionalnoj obrazovnoj ustanovi (škole i fakulteti). Razlog tome je veliki broj dostupne literature i internet kurseva. Tradicionalne metode učenja su nezamenljive jer predavač može posvetiti određeno vreme učeniku i raditi sa njim individualno, ali su one vremenski ograničene na period koji učenik provodi u obrazovnoj ustanovi. Tada veliki značaj dobijaju alternativni oblici učenja kao što su edukativne elektronske platforme.

Platforma *eŠkola veba* je elektronska škola veb programiranja koja omogućava interaktivno učenje kroz pažljivo osmišljene kurseve. Inicijativa za kreiranje platforme je pokrenuta na Matematičkom fakultetu Univerziteta u Beogradu. Na njoj se trenutno nalaze kursevi u okviru kojih se obrađuju *HTML, CSS, JavaScript, jQuery, Bootstrap, AngularJS* i *TypeScript*, a u planu je njena nadogradnja novim kursevima u okviru kojih će se obrađivati *Angular2, PHP, Java* i druge tehnologije.

Za kreiranje platforme su korišćeni *HTML, CSS, JavaScript, jQuery, Bootstrap, Angular 2* i *Sass*, a za izradu tekstualnog editora je korišćena *JavaScript* biblioteka *CodeMirror*. Sve tehnologije korišćene za izradu platforme su besplatne, a kôd je postavljen na *GitHub* i dostupan je za preuzimanje. U radu će biti predstavljene tehnologije korišćene za izradu platforme, njena struktura, dizajn i kako se njenim korišćenjem može unaprediti nastava informatike kada je u pitanju programiranje za veb.

1.1 Elektronske platforme za učenje veb programiranja

Postoji veliki broj internet kurseva za učenje veb programiranja. Internet sajt *W3Schools* je jedan od poznatijih i sadrži kurseve na kojima se obrađuju osnovni aspekti programskih jezika koji se koriste na vebu. Osnovan je 1998. godine od strane norveške softverske kompanije *Refsnes Data*. Sajt obuhvata kurseve iz oblasti veb programiranja i kroz njih pokriva najznačajnije tehnologije i jezike koji su osnovna današnjeg veba. Obuhvaćene su tehnologije koje se koriste i na klijentskim i na serverskim stranama. Svaki kurs je podeljen na lekcije, a svaka lekcija sadrži veliki broj propratnih primera. Svaki primer se može menjati u editoru koji je sastavni deo svake lekcije. Kursevi su besplatni i nalaze se na adresi https://www.w3schools.com. Sajt *W3Schools* je uzet kao model za izradu platforme *eŠkola veba* kada je u pitanju organizacija kurseva i lekcija.

Pluralsight je još jedan u nizu edukativnih sajtova koji obrađuju istu tematiku. Za razliku od ostalih, kursevi na ovom sajtu su u obliku video materijala. Korišćenje multimedije pomaže pri usvajanju znanja, a lekcije su zanimljivije i dinamičnije. Kao nedostatak se može navesti manjak interakcije sa korisnikom u vidu primera i zadataka koji se mogu rešavati direktno na sajtu. Sajt se može besplatno koristiti deset dana, a više informacija se može naći na zvaničnoj strani https://www.pluralsight.com.

Treehouse predstavlja najcelovitije rešenje od do sada pomenutih edukativnih sajtova. Kursevi su dobro struktuirani i organizovani, svaku lekciju prati video materijal, a postoji i mogućnost rešavanja zadataka i isprobavanja primera direktno na platformi. Moguće je pratiti napredak i birati planove po kojima se prelaze kursevi. Posebna pažnja je usmerena ka dizajnu koji je savremen i intuitivan. Sajt se može besplatno koristiti sedam dana i nalazi se na adresi https://teamtreehouse.com.

Primetan je značajno manji broj internet sajtova na srpskom jeziku sa sličnom namenom i približnim kvalitetom. Otuda proističe ideja za stvaranje elektronske platforme za učenje veb programiranja na srpskom jeziku sa dobro struktuiranim kursevima i sa savremenim i intuitivnim korisničkim dizajnom.

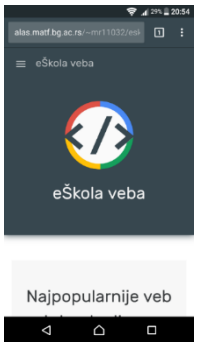
2 Postavljeni zahtevi

Platforma *eŠkola veba* svojom arhitekturom i dizajnom treba da omogući brzo i jednostavno učenje veb programiranja. Kako broj posetilaca interneta sa mobilnih uređaja raste iz godine u godinu, jedan od tehničkih zahteva koji je trebao biti ispunjen je svakako dizajn koji se prilagođava svim veličinama ekrana. Pored vizuelnog aspekta, posebna pažnja je posvećena strukturi i odabiru kurseva. Sadržaj kurseva obuhvata najbitnije tehnologije i programske jezike koji se koriste na vebu. Prisutni su kursevi koji obrađuju tehnologije koje čine osnovu veba, kao i one naprednije. Svaki kurs se sastoji od lekcija koje se logički nadovezuju jedna na drugu. Moderno gradivo objašnjeno kroz interaktivne lekcije je još jedan zahtev koji je trebalo ispuniti pri realizaciji ovog projekta. Po ugledu na neke najbolje edukativne platforme, *eŠkola veba* ima ugrađen editor u kome se mogu isprobavati primeri i rešavati postavljeni zadaci sa ispisom rezultata koda.

2.1 Korisnički dizajn aplikacije

Dobar korisnički dizajn treba da omogući lako i intuitivno korišćenje ne odvraćajući pažnju korisnika. Na polju dizajna su se u velikoj meri poštovale smernice *Google Material Design*-a[1], njegova paleta boja, raspored i izgled elemenata, ali i pojedine animacije. Svaka sekcija aplikacije se ističe zasebnom bojom. Korisnički interfejs je ravan i bez tekstura, perspektiva se postiže samo na pojedinim mestima korišćenjem senki.

Nagli razvoj prenosivih uređaja poput mobilnih telefona i tableta uslovio je razvoj veb aplikacija. Danas se kao preduslov često sreće upravo zahtev da aplikacija bude potpuno funkcionalna, intuitivna i laka za korišćenje na mobilnim uređajima. Pri izradi ove aplikacije je posebna pažnja posvećena upravo ovom zahtevu. Za ovu svrhu je korišćen radni okvir *Bootsrap 4* o kojem će biti više reči u daljem tekstu rada. Na mobilnim uređajima dizajn podseca na dizajn *Android* aplikacija što olakšava njeno korišćenje (*Slika 1*).



Slika 1 - Naslovna strana na mobilnim uređajima

[1] *Material Design* je dizajn razvijen od strane kompanije *Google* za izradu *Android* aplikacija i veb strana namenjenih mobilnim uređajima. Zvanično je predstavljen 2014. godine. Osnovu dizajna predstavlja digitalni papir (materijal) koji se može „pametno“ širiti i menjati oblik. Pored njega, *Material Design* definiše izgled brojnih vizuelnih elemenata kao što su polja za unos teksta, dugmići i sl.

2.2 Navigacija kroz aplikaciju i pregled strana

Naslovna strana omogućava laku navigaciju kroz aplikaciju. Ona sadrži sve osnovne elemente koje se mogu naći na naslovnim stranama većine sajtova. Na samom vrhu se nalazi navigacija sa poljem za brzu pretragu i najbitniji linkovi, zatim zaglavlje sa nazivom i logom aplikacije, spisak svih kurseva i alata, spisak korišćenih tehnologija i podnožje sa linkovima ka društvenim mrežama (*Slika 2*).

Slika 2 - Naslovna strana

Na naslovnoj strani se ističe zaglavlje u okviru koga se nalazi logo aplikacije. Logo je kreiran u programskom paketu *Adobe Photoshop*[2] i ima oblik kruga u čijem centru se nalaze tri najkorišćenija karaktera u *HTML* dokumentima koja se koriste za označavanje etiketa (*Slika 3*).

[2] *Adobe Photoshop* je programski paket kompanije *Adobe Systems*. Trenutno je lider među profesionalnim programima na polju izrade i obrade digitalnih fotografija. Prvo izdanje ovog softvera je objavljeno 1990. godine za *Mac OS*, a 1992. godine za *Windows*.

Slika 3 – Logo platforme eŠkola veba

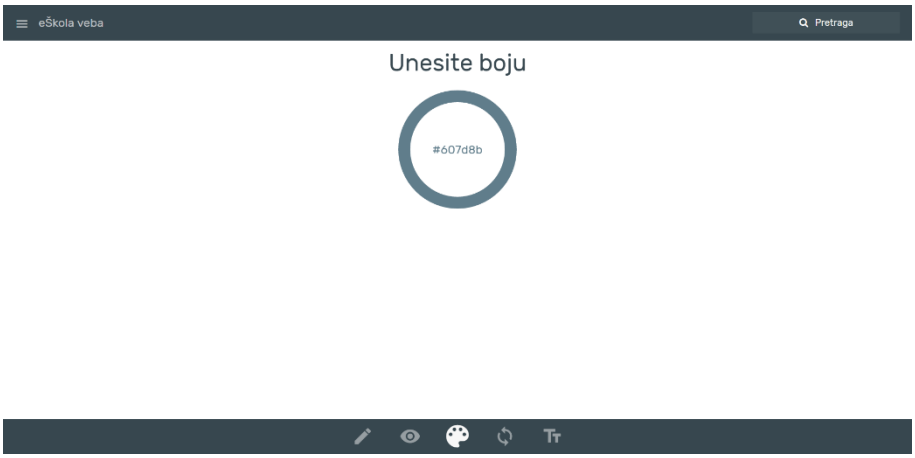
Ispod zaglavlja se nalazi spisak svih kurseva koji se nalaze na platformi, svaki predstavljen u obliku logoa koji simbolički predstavljaju taj kurs. Klikom na svaki kurs se otvara strana na kojoj se nalaze detalji o kursu kao što je broj lekcija koje sadrži i kratak prikaz obrađene tehnologije. Klikom na dugme *Startuj kurs* pokreće se prva lekcija kursa (*Slika 4*).

Slika 4 - Detalji o kursu

Ispod spiska svih kurseva nalaze se korisni alati. Prvi i najveći među njima je *Veb centar* koji prestavlja editor za veb programiranje sa mogućnošću pregleda napisanog koda, alatkom za ispis boja u različitim formatima, konverterom različitih jedinica i transliteratorom [3]ćirilica – latinica (*Slika 5*). Za alatku za ispis boja u različitim formatima je korišćena biblioteka *Angular 2 Color Picker (Slika 6).*

[3] Transliteracija ili preslovljavanje je prebacivanje znakova iz jednog pisma u drugo pri čemu se ne vodi računa o izgovoru već se prebacuje znak za znak.

Slika 5 - Veb centar - editor

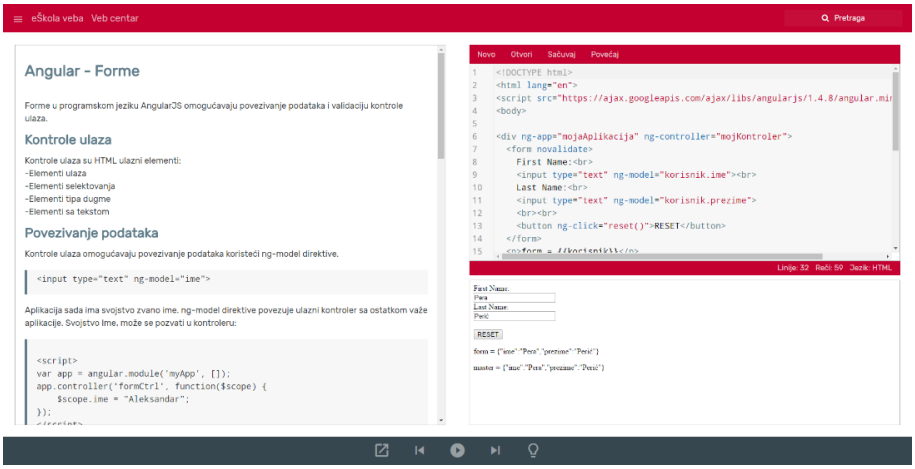


Slika 6 - Veb centar – alatka za ispis boja

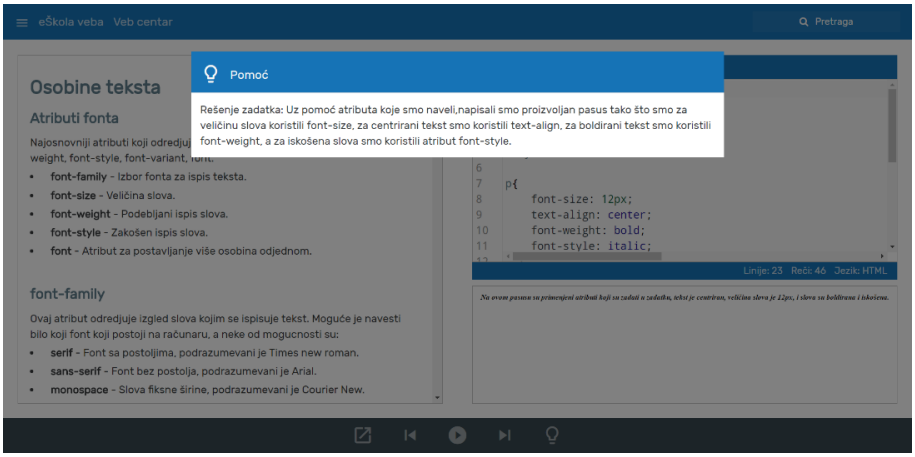
Na platformi se nalaze još dva korisna alata za prevođenje *TypeScript* koda u *JavaScript* i *CSS-a* u *Sass*. Ove dve alatke imaju identičan izgled koga karakterišu dva editora od kojih se u prvom unosi kôd, a u drugom vidi rezultat i traka sa akcijama na dnu prozora (*Slika 7*).

Slika 7 - TypeScript prevodilac

Stranica na kojoj su prikazane lekcije se sastoji iz nekoliko delova. Sa leve strane se nalazi tekst lekcije, sa gornje desne strane se nalazi editor u kojem se mogu isprobavati primeri i pisati kôd, a ispod njega je umanjeni prikaz prozora pretraživača sa prikazom unesenog primera. Na samom dnu strane se nalazi traka sa akcijama kao što je prelazak na prethodnu ili sledeću lekciju, pokretanje koda, uvećani prikaz prozora pretraživača i pomoć u vezi zadataka koji su sastavni deo većine lekcija (*Slika 8*). Ideja za ovakvom strukturom strane je proistekla iz ideje da se učenje novih sadržaja najlakše postiže kroz primere i primene naučenog. Vodeći se tom idejom, većina lekcija sadrži zadatke koji se mogu rešavati na licu mesta, a kao provera može služiti pomoć koja se nalazi uz svaki zadatak (*Slika 9*). Na platformi nije implementirana provera tačnosti urađenih zadataka zato što zadaci nemaju jedinstveno ispravno rešenje. U planu je implementacija testova koji će pratiti gradivo i koji će podržati proveru tačnosti rešenja zadataka.

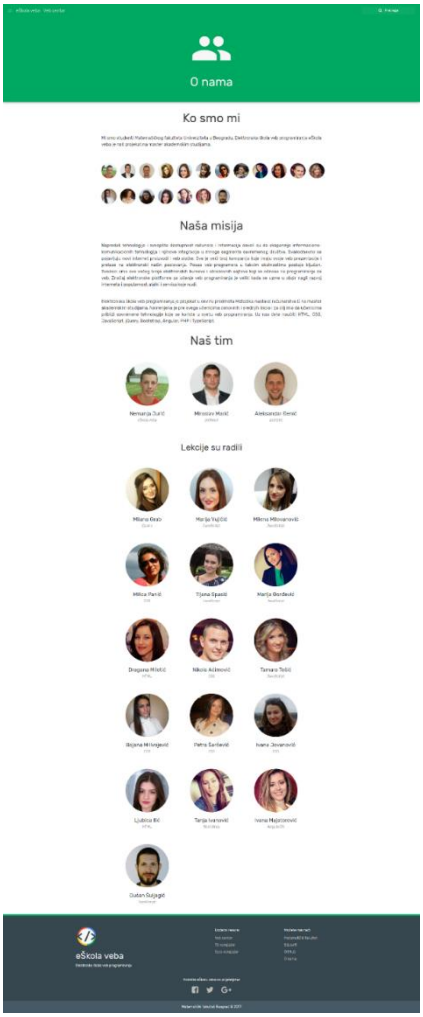


Slika 8 - Prikaz lekcija



Slika 9 - Pomoć u lekcijama

Na platformi se nalazi i strana koja je posvećena autorima sajta i onima koji su učestvovali u pisanju lekcija (*Slika 10*). Autor ovog rada je jedan od idejnih tvoraca projekta *eŠkola veba* i zaslužan je za implementaciju platforme.



Slika 10 – Tim zaslužan za razvoj platforme

3 Tehnologije korišćene za izradu platforme

Platforma *eŠkola veba* je aplikacija napisana u programskom jeziku *TypeScript*. Za njen razvoj je korišćen radni okvir *Angular 2*. Pored pomenutih tehnologija, za izradu aplikacije su korišćeni još i:

- jezik za opis veb stranica – *HTML 5*,
- jezici za definisanje izgleda elemenata na veb stranama – *CSS 3* i *Sass*,
- programski jezik *JavaScript*,
- *JavaScript* biblioteka *jQuery* i
- radni okvir za razvoj modernih veb strana *Bootstrap 4*.

Radi lakšeg razvoja u radnom okviru *Angular 2*, korišćen je alat *Angular CLI*. Sve pomenute tehnologije će biti naknadno objašnjene. Editor koji je korišćen za rad je *Visual Studio Code* kompanije *Microsoft* koji je besplatan i dostupan za preuzimanje sa zvaničnog sajta.

3.1 HTML 5

HTML (*engl. Hyper Text Markup Language*) je jezik namenjen opisu strukture veb stranica. Pomoću njega se lako može formatirati izgled stranice i opisati njeni elementi kao što su naslovi, paragrafi, linkovi, tabele, citati i sl. Pored navedenih elemenata, u okviru jezika *HTML* je moguće dodati elemente koji detaljnije opisuju sam dokument, kao što su kratak opis dokumenta, podaci o autoru, jezik na kojem je dokument napisan itd. Ovi podaci su poznati kao meta podaci i jasno su odvojeni od ostatka dokumenta meta etiketama. Primer prostog *HTML* dokumenta koji sadrži naslov, podnaslov i paragraf se može videti na *slici 11*.

```
<!DOCTYPE html>
<html lang="en">

<head>
    <title>Naslov</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link href="css/style.css" rel="stylesheet">
</head>

<body>
    <h1>Naslov</h1>
    <h3>Podnaslov</h3>
    <p>
        Ovo je paragraf.
    </p>
</body>

</html>
```

Slika 11 – Primer HTML dokumenta

Istorija *HTML*-a počinje 1980. godine kada je Tim Berners-Li[4] za istraživače u organizaciji za nuklearno istraživanje *CERN*[5] predložio sistem pomoću koga bi mogli da koriste i dele dokumente. Projekat se bazirao na hipertekstu[6], a njegova prezentacija i prvi prototip je bio završen već krajem iste godine. Početkom 90-ih godina je specificirao *HTML* i time opisao 20 elemenata koji su činili tadašnji dizajn jezika *HTML*. Projekat međutim nije formalno usvojen od strane ove organizacije. 1994. godine osnovao je *W3C* na Masačusetskom tehnološkom institutu *(eng. Massachusetts Institute of Technology)* koji je usvojen kao standard prema kojem će sve veb stranice u budućnosti raditi na isti način. Godine 1995, organizacija *IETF (engl. Internet Engineering Task Force)* je izdala specifikaciju jezika *HTML* pod nazivom *HTML 2.0* sa namerom da se na njoj temelje sva dalja usavršavanja i nadogradnje ovog jezika. Godine 2000, *HTML* postaje internacionalni standard, a 2004. godine počinje sa radom *HTML 5* u koji su uvedeni novi elementi i atributi. Uvedena je podrška za audio i video datoteke, kao i grafički elementi.

3.2 CSS3

CSS (eng. Cascading Style Sheets) je jezik pomoću koga se može definisati izgled elemenata veb stranica. Prvobitno je njegovu ulogu imao *HTML*, kojim se definisala struktura, sadržaj i izgled veb stranica. *HTML* je vremenom je postajao komplikovaniji i imao sve više mogućnosti za definiciju izgleda elemenata, ali time i težak za čitanje i održavanje. Različiti pretraživači su prikazivali iste elemente na različite načine pa se stvorila potreba za doslednom tehnikom prikazivanja elemenata koju bi koristili svi pretraživači. Da bi se ovo postiglo, na *W3C* forumu je predloženo devet različitih metoda od kojih su izabrane dve metode na čijim temeljima je kasnije nastao *CSS*. Za razliku od dotadašnjih rešenja, u jeziku *CSS* je dozvoljeno da više opisa utiče na izgled elementa, tj. definicija stilova je mogla da nasleđuje osobine od druge. U decembru 1996. godine objavljena je prva verzija jezika *CSS* pod nazivom *CSS1*. 1997. godine objavljena je verzija *CSS2* koji je sadržavao ispravke iz prethodne verzije, a 1998. godine je objavljen *CSS3* čiji razvoj traje još uvek.

Sintaksa jezika *CSS* se sastoji od opisa izgleda elementa u dokumentu. Više opisa se može odnositi na jedan element, ali isto tako se i jedan opis ili skup opisa može odnositi na više elemenata. Na taj način se opisi slažu jedan preko drugog da bi definisali konačan izgled određenog elementa, a na kraju i celog dokumenta. Primer koda napisanog u jeziku *CSS* u kojem su elementu tipa paragraf dodata svojstva koja opisuju font, veličinu i boju teksta i boju pozadine se može videti na *slici 12*. Svaki opis se sastoji iz tri elementa:

- definicije elemenata na koji se opis odnosi,
- svojstva i
- vrednosti.

[4] Tim Berners-Li *(eng. Tim Berners-Lee)* je kreator veba i čelnik *World Wide Web* konzorcijuma

[5] Evropska organizacija za nuklearno istraživanje *(eng. European Organization for Nuclear Research)* poznatija kao CERN, smeštena u okolini Ženeve, je najveći centar na svetu za istraživanje elementarnih čestica

[6] Hipertekst (*eng. hypertext*) je modularna struktura koja se sastoji od velikog broja međusobno povezanih veb stranica

```
p{
    font-family: Arial, Helvetica, sans-serif;
    font-size: 16px;
    text-align: center;
    background-color: dodgerblue;
    color: white;
}
```

Slika 12 - Primer CSS koda

Prvo se definišu ciljni elementi na koje će da se opis odnosi, a zatim se nizom parova svojstvo-vrednost definiše izgled ciljnih elemenata. U jeziku *CSS* su podržani i komentari koji se navode između znakova /* i */ kao u programskom jeziku *C*. Svojstva elemenata na koja želimo da utičemo predstavljaju niz ključnih reči koje su definisane *W3C* standardom i mogu se kategorisati u grupe kao što su: definicija pozadine, ivica, okvir, prikaz, dimenzije, font, margine, pozicija itd. Vrednosti su ključne reči ili brojevi koji mogu, ali i ne moraju da imaju jedinicu veličine. *CSS* se može dodati u dokument na tri načina – direktno u elementu koristeći atribut *style*, u zaglavlju dokumenta ili u zasebnoj datoteci koja se uključuje u dokument etiketom *link*. Ograničenja jezika *CSS* se ogledaju u sledećem:

- Nedoslednost pretraživača – različiti pretraživači različito interpretiraju pojedina svojstva. Ova nedoslednost se može zaobići korišćenjem tzv. *Reset CSS* dokumentima koji većini elemenata dodaju neke predefinisane vrednosti koje svi pretraživači interpretiraju na isti način.
- Opisi elemenata se ne mogu bazirati na drugim objektima, tj. u jeziku *CSS* nije podržano nasleđivanje.
- Nedostatak aritmetičkih operacija – donekle prevaziđeno uvođenjem izraza *calc()*, ali nije podržano od strane svih pretraživača.
- Neslaganje pojedinih opisa – kombinovanje svojstava *position, float* i *display* često dovodi do neočekivanih rezultata.
- Ne postoje striktna pravila o redosledu opisa.
- Nepostojanje promenljivih.
- Složenost – veliki sajtovi imaju problema sa organizacijom *CSS* dokumenata jer oni postaju veoma složeni i teški za održavanje.

Navedena ograničenja, ali i mnoga druga su rešena *CSS* pretprocesorima kao što je npr. *Sass*. Više o jeziku *CSS* se moće naći na [2].

3.3 JavaScript

JavaScript je skriptni programski jezik koji je prvenstveno namenjen za programiranje na vebu. Najkorišćeniji je za programiranje aplikacija koje se izvršavaju na klijentskim stranama, ali u poslednje vreme postaje sve popularnija njegova primena i na serverskim stranama pojavom radnog okvira *Node.js*. On je programski jezik visokog nivoa[7] koji zapravo predstavlja implementaciju *ECMAScript* standarda. *ECMAScript* standard predstavlja specifikaciju programskog jezika *JavaScript* uvedenu od strane organizacije *Ecma International*. Pored jezika *HTML* i *CSS*, *JavaScript* predstavlja jednu od tri ključne tehnologije na vebu. Ima podršku u svim modernim pretraživačima. Prvobitna ideja je bila da *JavaScript* liči na programski jezik *Java*, ali danas on ne sadrži ni jedan njegov ključni element, osim onoga što su oba jezika nasledila od programskog jezika *C*. Prvobitno se zvao *LiveScript*, a današnje ime je dobio zbog marketinškog dogovora između kompanija *Netscape* koja ga je stvorila i *Sun*. Danas je zaštitni znak kompanije *Oracle Corporation*.

Sintaksa jezika *JavaScript* je slična sintaksi programskog jezika *C*, imaju jako sličnu sintaksu naredbi granjanja i petlji, a i postoje razlike između izraza i naredbi. *JavaScript* je dinamički tipiziran jezik, promenljivoj kojoj je dodeljena vrednost tipa *number* kasnije može biti dodeljena vrednost tipa *string*. Postoji i podrška za objektno-orijentisane koncepte, ali za razliku od ostalih objektno-orijentisanih jezika, *JavaScript* koristi prototipove umesto klasa, interfejsa i nasleđivanja. To je međutim promenjeno izlaskom *ECMAScript* 6 standarda koji u *JavaScript* uvodi ove pojmove. *ECMAScript* 6 standard još uvek nije potpuno podržan od strane svih pregledača. Više o jeziku *JavaScript* se može naći na [3].

3.4 TypeScript

TypeScript je programski jezik razvijen od strane kompanije *Microsoft*. On je strog nadskup programskog jezika *JavaScript* što znači da je svaki *JavaScript* kôd ujedno i validan *TypeScript* kôd. *TypeScript* u *JavaScript* uvodi statičku tipiziranost i koncepte objektno orijentisanih jezika. Na njemu su radili inženjeri koji su radili na razvoju programskih jezika *C#*, *Delphi* i *Turbo Pascal*. Može se koristiti za razvoj *JavaScript* aplikacija za izvršavanje na veb klijentu ili serveru i dizajniran je za razvoj velikih aplikacija što je bio jedan od ključnih nedostataka jezika *JavaScript*. *TypeScript* se kompajlira u *JavaScript*. Na *slici 13* je data funkcija napisana u jeziku *TypeScript* i identična funkcija napisana u jeziku *JavaScript*. Može se primetiti da se u jeziku *TypeScript* mogu navoditi tipovi promenljivih kao i povratnih vrednosti funkcija.

[7] Programski jezik visokog nivoa je programski jezik sa jakom apstrakcijom u odnosu na bazično korišćenje računara, može sadržati elemente iz prirodnog jezika i time učiniti programiranje lakšim.

```
//TypeScript
function pozdrav(osoba: string): string {
    return 'Zdravo ' + osoba;
}
//JavaScript
function pozdrav(osoba) {
    return 'Zdravo ' + osoba;
}
```

Slika 13 - Primer TypeScript koda koji sadrži tipove promenljivih i iskompajliranog JavaScript koda

Na *slici 14* je dat primer klase u jeziku TypeScript i iskompajliranog *JavaScript* koda. Sintaksa pisanje klasa i pravljenja novih instanci klase u jeziku *TypeScript* je slična onoj kakva se viđa u objektno orijentisanim jezicima kao što su *Java* i *C#*.

```
//TypeScript
class Osoba {
    ime: string;
    constructor(ime: string) {
        this.ime = ime;
    }
    pozdrav(): string {
        return "Zdravo " + this.ime;
    }
}

let osoba = new Osoba("Nemanja");

//JavaScript
var Osoba = (function () {
    function Osoba(ime) {
        this.ime = ime;
    }
    Osoba.prototype.pozdrav = function () {
        return "Zdravo " + this.ime;
    };
    return Osoba;
}());
var osoba = new Osoba("Nemanja");
```

Slika 14 – Primer klase u jeziku TypeScript i iskompajlirani JavaScript kôd

TypeScript je prvi put uključen u *Microsoft Visual Studio 2013 (Update 2)* uz *C#* i ostale programske jezike kompanije *Microsoft*. Podržan je od strane najpopularnijih tekstualnih editora kao što su *Visual Studio Code, Sublime Text, Atom, WebStorm* i drugi. Više detalja o jeziku *TypeScript* se može naći na njegovom zvaničnom sajtu [7].

3.5 jQuery

JQuery je *JavaScript* biblioteka dizajnirana da pojednostavi neke često korišćene funkcije jezika *JavaScript* kao što je navigacija ka dokumentu, selektovanje *DOM*[8] elemenata, pravljenje animacija, obrađivanje događaja i razvoj *AJAX*[9] aplikacija. Kompanija *Microsoft* ga je uključila u *Visual Studio* za upotrebu u *ASP.NET MVC* aplikacijama. *JQuery* pojednostavljuje sintaksu za pronalaženje, selektovanje i manipulaciju *DOM* elemenata koristeći *CSS* selektore u tu svrhu. Pored toga, omogućava menjanje njihovih svojstava i atributa, kao i vezivanje određenih događaja za selektovani element. Arhitektura biblioteke *jQuery* je takva da dozvoljava pisanje dodataka kako bi se proširila njena funkcionalnost. Postoji na hiljade dodataka napravljenih u tu svrhu koji pokrivaju širok spektar funkcija. *JQuery* je objavljen u januaru 2006. godine, a od 2015. godine postaje najzastupljenija *JavaScript* biblioteka na vebu. *JQuery* je besplatna biblioteka otvorenog koda.

3.6 Angular

Angular je radni okvir programskog jezika *JavaScript* razvijen od strane kompanije *Google* i zajednice pojedinaca i drugih kompanija. Omogućava razvoj dobro struktuiranih jednostraničnih veb aplikacija (*eng. Single Page Application - SPA*[10]*)*. Prva verzija ovog radnog okvira, zvanično nazvana *AngularJS*, je razvijena od strane Miška Heverija (*eng. Miško Hevery*) 2009. godine. Poslednja stabilna verzija *AngularJS* okvira u vreme pisanja ovog rada nosi oznaku 1.6.0. Sledeća verzija pod nazivom *Angular 2* je najavljena na *ng-Europe* konferenciji u septembru 2014. godine kao verzija koja će uvesti neke drastične promene i nove koncepte u *AngularJS*. *Angular 2* je zvanično izašao dve godine kasnije, 14. septembra 2016. godine. *Angular 2* nije predstavljao samo osveženu verziju svog prethodnika, već su njime uvedene neke konceptualne promene. Neke od promena u *AngularJS* koje su uvedene izlaskom *Angular 2* verzije su bolja modularnost, uprošćeno rutiranje, sasvim drugačije korišćenje komponenti i direktiva itd. Dosta pažnje je bilo usmereno ka performansama gde testovi pokazuju da je *Angular 2* i do 5 puta brži od svog prethodnika u prikazivanju strana sa velikim brojem elemenata. To je postignuto tako što je smanjen broj ciklusa u kojima *Angular* posmatra promene neke promenljive. Jedan takav ciklus je poznatiji pod nazivom *digest cycle*. *Angular* je kompatibilan sa svim modernim veb pregledačima, a od verzije 1.3 je podržan u *Internet Explorer-u 9* i novijim verzijama. Više o radnom okviru *Angular* se može videti na zvaničnom sajtu [6].

[8] *DOM (Document Object Model)* predstavlja strukturnu prezentaciju *HTML* ili *XML* dokumenata čiji su elementi organizovani u stablo u kome svaki čvor predstavlja jedan element.

[9] *AJAX* je skup tehnika koje se koriste na klijentu za kreiranje asinhronih zahteva serveru. Asinhroni zahtevi se izvršavaju u pozadini i ne utiču na prikaz i ponašanje trenutne stranice

[10] SPA (*eng. Single Page Application*) je veb aplikacija ili veb sajt čiji se ceo sadržaj nalazi na jednoj strani i time pruža korisničko iskustvo slično onom koje imaju *desktop* aplikacije

3.7 Bootstrap 4

Bootstrap predstavlja besplatan skup alata za kreiranje modernih veb stranica i aplikacija čiji je cilj olakšavanje programiranja za veb. *Bootstrap* se sastoji od šablona za tipografiju, kreiranje formulara, dugmadi, navigacije, iskačućih prozora i ostalih komponenti. *Bootstrap*, prvobitno nazvan *Twitter Blueprint*, su razvili Mark Oto (*eng. Mark Otto*) i Džejkob Tornton (*eng. Jacob Thornton*) iz kompanije *Twitter*. Pre ovog radnog okvira je korišćen veliki broj biblioteka za razvoj korisničkog interfejsa što je dovelo do nedoslednosti i teškog održavanja. *Bootstrap* je projekat koji je na jednom mestu okupio većinu funkcionalnosti pomenutih biblioteka i na taj način olakšao razvoj. Prva zvanična verzija je objavljena u avgustu 2011. godine, u drugoj verziji se pojavljuje prilagodljiv dizajn, a tek od treće verzije se uvodi podrška za mobilne uređaje. *Bootstrap* 4 je izašao kao velika nadogradnja svog prethodnika jer je modularan i potpuno prepisan u *Sass*. *Bootstrap* se sastoji od niza *Sass* fajlova koji definišu različite komponente. Dobra strana ovakve modularnosti se ogleda u tome sto programeri mogu birati komponente koje žele da koriste u svom projektu, a da pri tome ne učitavaju sve funkcionalnosti. Posebna pogodnost ovog alata je ta što se elementi veb strane mogu smeštati u mrežu koja sadrži dvanaest kolona čija se širina prilagođava širini prozora pretraživača. Kôd radnog okvira *Bootstrap* je besplatan i javno dostupan [4].

3.8 Sass

Sass (eng. Syntactically Awesome Stylesheets) je skriptni jezik koji se interpretira u *CSS*. *Sass* sadrži dve sintakse. Originalna sintaksa, zvana „*uvučena sintaksa*“ koristi uvlačenje koda za odvajanje blokova. Nova sintaksa *Scss* koristi zakrivljene zagrade za odvajanje blokova kao i *CSS*. *Sass* proširuje *CSS* tako što mu dodaje mehanizme koji su dostupni u tradicionalnim programskim jezicima. *Sass* interpreter prevodi *Sass* u *CSS*. Postoji mogućnost da interpreter posmatra *Sass* datoteke i prevede ih u *CSS* datoteke kada god je *Sass* datoteka promenjena.

Prevodilac koji prevodi jezik *Sass* u *CSS* je softver otvorenog koda koji je napisan u programskom jeziku *Ruby*, međutim, pored ove zvanične, postoje implementacije i u drugim programskim jezicima, kao što su *PHP*, *C* i *JavaScript*. *Sass* pruža mogućnosti za korišćenje promenljivih, ugnježdavanja, nasleđivanje selektora funkcijom *@extend*, itd. Svaka validna *CSS* datoteka je i validna *Sass* datoteka [5]. Primer *Sass* koda i prevedenog *CSS koda* je dat na *slici 15*.

```
$boja: dodgerblue;
$velicina-slova: 20px;

p{
    color: $boja;
    font-size: 16px;
    &:hover{
        font-size: $velicina-slova;
    }
}

p{
    color: dodgerblue;
    font-size: 16px;
}

p:hover{
    font-size: 20px;
}
```

Slika 15 - Primer Sass koda i prevedenog CSS koda

4 Aplikacije u radnom okviru *Angular 2*

4.1 Jednostranične aplikacije

Platforma eŠkola veba je po svojoj strukturi jednostranična aplikacija za čiju izradu je korišćen radni okvir *Angular 2* i alat pod nazivom *Angular CLI*. Jednostranična aplikacija je veb aplikacija ili veb sajt čiji se ceo sadržaj nalazi na jednoj strani i time pruža korisničko iskustvo slično onom koje imaju standardne aplikacije koje se sreću na računarima. Kod ovakve vrste aplikacija svi neophodni resursi za njeno izvršavanje se učitavaju pri prvom učitavanju aplikacije, a u toku rada se dinamički učitavaju po potrebi i to najčešće kao posledica akcije korisnika. Ovakva dinamičnost podrazumeva čestu komunikaciju aplikacije sa veb serverom na kojem se nalazi. Jednostranična aplikacija premešta izvršavanje poslovne logike sa servera na klijentski računar. Upravo ovo premeštanje cele aplikacije na klijentski računar je uslovilo razvoj *JavaScript* alata za razvoj aplikacija koji usvajaju ovakvu njihovu strukturu i ubrzavaju razvoj. Najpoznatiji među njima su *Angular, Ember, Meteor, React* i drugi. Postoji nekoliko tehnika koje omogućavaju komunikaciju klijentskog računara i servera. Najkorišćeniji je svakako *AJAX*, a tehnologije koje se pri toj komunikaciji najčešće koriste su *JSON*[11] i *XML*[12]. Dinamičnost jednostraničnih aplikacija se ogleda u tome što ona ne učitava kompletnu *HTML* stranu pri promeni prikaza, već upućuje *AJAX* zahtev serveru koji joj kao odgovor vrati podatke potrebne za prikaz nove strane u *JSON* formatu. Time je unapređeno

[11] *JSON* (*JavaScript Object Notation*) je najkorišćeniji standard koji se koristi za prenos podataka tokom *AJAX* zahteva

[12] *XML* je jezik za obeležavanje koji definiše skup pravila za strukturni opis dokumenata

korisničko iskustvo tako što je povećana brzina učitavanja aplikacije i smanjeno je vreme čekanja pri prelasku sa jedne stranice na drugu. Iako se pri promeni stranice ne učitava kompletna stranica već samo neki njen deo, dugme za povratak na prethodnu stranu u pretraživaču se i dalje može koristiti za navigaciju kroz aplikaciju.

4.2 Arhitektura aplikacija

Aplikacije za čiji je razvoj korišćen radni okvir *Angular 2* se sastoje iz kombinacije komponenti koje se dalje sastoje iz dinamičkih *HTML* šablona (*eng. templates*) kontrolisanih direktivama i klasa koje kontrolišu rad aplikacija. Pored komponenti, za izradu aplikacija se koriste i servisi koji najčešće služe za komuniciranje sa serverom, ali mogu služiti i za čuvanje i razmenjivanje podataka između komponenti. Komponente i servisi se mogu logički grupisati u module. Prilikom pokretanja prvo se pokreće glavni modul (*root module*) aplikacije [1].

Može se izdvojiti osam glavnih elemenata koji se koriste u razvoju aplikacija u radnom okviru *Angular 2*:

- moduli,
- komponente,
- šabloni (*templates*),
- meta podaci (*metadata*),
- povezivanje podataka (*data binding*),
- direktive,
- servisi i
- ubrizgavanje zavisnosti (*dependency injection*).

Angular aplikacije se sastoje iz modula. Svaka aplikacija sadrži makar jedan modul – *root module* koji se po konvenciji naziva *AppModule*. Svaki modul je klasa sa *@NgModule* dekoratorom[13] koji sadrži jedan objekat čija svojstva opisuju modul.

Komponente kontrolišu deo aplikacije koji se naziva *view* i koji predstavlja *HTML* prikaz komponente. Unutar klase komponente se nalazi kôd kojim je implementirana njena funkcionalnost. *View* neke komponente se nalazi u šablonu (*template*) koji se sastoji od *HTML*-a i *Angular template* sintakse koja proširuje *HTML*.

Metapodacima se definiše ponašanje klasa. Klasa može biti modul, komponenta, servis, direktiva ili *pipe* (filter) dok se to eksplicitno ne definiše metapodacima. Metapodaci se dodeljuju klasama putem dekoratora. Na primer, *@Component* je dekorator kojim se definiše da će klasa koja sledi biti komponenta.

Povezivanje podataka je mehanizam povezivanja podataka sa *HTML*-om ili između komponenti. Angular podržava više vidova povezivanja podataka:

- Interpolacija (*eng. interpolation*) – prikazivanje podataka na *HTML* strani korišćenjem dvostrukih vitičastih zagrada.

[13] *Dekoratori* su funkcije koje menjaju *JavaScript* klase i dodaju im metapodatke koji opisuju šta ta klasa predstavlja i kako da se ponaša

- Povezivanje svojstava (*eng. property binding*) – prenos podataka od roditeljske komponente ka komponenti koja je njeno dete.
- Povezivanje izazvano događajem (*eng. event binding*) – poziv metode neke komponente kada se desi neki događaj nad tom komponentom, npr. klik miša.
- Dvosmerno povezivanje (*eng. two-way data binding*) – kombinuje prethodna dva u jednu notaciju korišćenjem *ngModel* direktive.

Direktive su komponente bez šablona. Drugim rečima, one nemaju svoj *HTML* prikaz. Direktiva se definiše *@Directive* dekoratorom. Direktive mogu da menjaju strukturu prikaza ili da menjaju izgled elemenata. Direktive koje menjaju strukturu prikaza mogu da dodaju ili brišu elemente i primeri takvih direktiva su **ngFor* i **ngIf*. Direktive koje menjaju izgled i ponašanje elemenata mogu da im dodaju *CSS* klase i stilove ili upisuju neki tekst u elemente. Primeri takvih direktiva su *ngModel* i *ngClass*. Takođe je moguće i dozvoljeno pisati svoje direktive.

Servisi u radnom okviru *Angular* mogu imati najrazličitije funkcije, ali najčešće služe za izdvajanje logike komuniciranja sa serverom iz komponenti i za čuvanje i razmenjivanje podataka između njih. Oni često predstavljaju dodatni sloj između komponente i mesta pohranjivanja podataka sa kojima ona radi. Komponente rade sa podacima koje dobijaju putem servisa i ne brinu se o tome na koji način je servis došao do njih. Ova tehnika je u programiranju poznata kao *Repository Pattern*. Sa tehničke strane servis je klasa. Za razliku od komponenti, servisi ne zahtevaju registraciju u modulu kojem pripadaju. Iako imaju veoma prostu strukturu, servisi su jako bitni u strukturi aplikacije zbog svoje široke funkcije. Servisi moraju da na neki način budu dostupni da bi komponente mogle komunicirati sa njima. Njihova dostupnost se omogućuje putem tehnike pod nazivom *Dependecy Injection* koja je često korišćena u razvoju softvera.

Dependecy Injection je ustaljeni obrazac (*Design Pattern*) kreiranja nove instance neke klase sa kreiranjem instanci klasa od kojih ona zavisi. *Angular* koristi ovu tehniku da kreira instancu komponente sa servisom koji ona koristi i od koje zavisi. To se postiže tako što se u konstruktoru komponente kao parametar prosledi promenljiva koja predstavlja novu instancu servisa koji komponenta koristi, a servis prethodno registruje u nizu *Providers* koji je atribut *@Component* dekoratora. *Dependecy Injection* je jedna od ključnih stvari u kreiranju aplikacije jer se prožima kroz celu njenu strukturu.

Observer Pattern je još jedna ustaljeni obrazac koji je našao široku upotrebu u aplikacijama napisanim u radnom okviru *Angular 2*. Za ovu tehniku je karakterističan objekat pod nazivom *Subject* koji u sebi sadrži listu posmatrača, pod nazivom *Observers*, koje obaveštava o nekim promenama u modelu podataka. *Observer Pattern* se u *Angular* aplikacijama najčešće koristi pri transferu podataka iz nekog servisa kada se na njih mora čekati neko vreme.

4.3 Alat za razvoj Angular aplikacija - Angular CLI

Angular CLI (*Command Line Interface*) je alat koji omogućava kreiranje novog *Angular 2* projekta, dodavanje njegovih elemenata i izvršavanje različitih zadataka tokom razvoja kao što su pokretanje, testiranje i pripremanje projekta za produkciju. Kreiranje projekta se zadaje komandom *ng new* [naziv projekta] koja kreira novi direktorijum pod zadatim nazivom i u njemu kostur aplikacije sa osnovnim podešavanjima (može se koristiti i komanda *ng init* koja ima istu funkciju samo što ne kreira novi direktorijum). Dodavanje elemenata se zadaje komandom *ng g component* | *directive* | *service* | *pipe* | *module* [naziv]. Izvršavanjem ove komande se kreira zadati element i registruje u modulu ako je potrebno.

Najveća prednost ovog alata leži u tome što je jako pogodan za brzu i laku pripremu projekta za produkciju. Komandom *ng build* kreira se direktorijum *dist* i u njemu datoteke aplikacije spremne za produkciju. Izvršavanjem ove komande se u direktorijum *dist* kopira sadržaj direktorijuma *assets* u kojem se najčešće smeštaju stilovi (*css* i *sass* datoteke), slike, zvukovi i fontovi. Pored njega, moguće je dodati i još neke direktorijume koji će se kopirati u *dist* direktorijum. Sva dodatna podešavanja je moguće napisati u datoteci *angular-cli.json*. Ovom komandom se *TypeScript* datoteke prevode u *JavaScript* datoteke, minifikuju i smeštaju u nekoliko *JavaScript* datoteka, a od stilova se takođe kreiraju *JavaScript* datoteke. Finalni izlaz celog procesa je minifikovan kôd aplikacije i njegovo smeštanje u svega nekoliko datoteka:

- *inline.bundle.js,*
- *main.bundle.js,*
- *polyfills.bundle.js,*
- *scripts.budle.js,*
- *styles.budle.js,*
- *vendor.bundle.js.*

Može se primetiti da se u nazivu svake od novonastalih datoteka pojavljuje reč *bundle*. *Bundle* je snop ili paket koji nastaje spajanjem više datoteka. U cilju bržeg učitavanja datoteka često se vrši i njihovo minifikovanje koje predstavlja proces uklanjanja nepotrebnih karaktera iz koda, najčešće belina.

5 Arhitektura aplikacije eŠkola veba

Arhitektura aplikacija razvijenih u radnom okviru *Angular* je modularna. Modularnost arhitekture se ogleda u tome da se aplikacije sastoje iz modula, a najmanje jednog. Smatra se dobrom praksom da svaki modul bude logička celina. To se može postići na više načina zavisno od same aplikacije, najčešće jedan modul obavlja jednu funkciju ili ako se aplikacija može podeliti na zasebne celine, onda svaka od njih treba biti jedan modul. Modul može sadržati komponente, direktive, servise i filtere. *eŠkola veba* se sastoji iz jednog glavnog modula - *AppModule* i devetnaest komponenti koje mu pripadaju:

- *AppComponent*,
- *AboutComponent*,
- *BookComponent*,
- *BottomNavigationComponent*,
- *CourseComponent*,
- *CourseDetailsComponent*,
- *EditorComponent*,
- *FooterComponent*,
- *HeaderComponent*,
- *HomeComponent*,
- *LessonsInputComponent*,
- *MenuComponent*,
- *ModalComponent*,
- *NavigationComponent*,
- *PageNotFoundComponent*,
- *SassCompilerComponent*,
- *SearchComponent*,
- *TsCompilerComponent*,
- *WebCenterComponent*.

Pored komponenti, u aplikaciji se nalaze i dva servisa:

- *SchoolService*,
- *RouteService*.

Filter koji dozvoljava ispis *HTML* dokumenata unutar *DOM* elemenata je napisan korišćenjem elementa *Pipe* – *SanitizerPipe*.

Glavni (*root*) direktorijum projekta sadrži dva poddirektorijuma sa nazivima *src* i *node_modules*. Direktorijum *src* sadrži datoteke same aplikacije, dok su biblioteke standardno smeštene u direktorijumu *node_modules*. Praksa je da se svaki modul aplikacije nalazi u zasebnom direktorijumu, pa se tako *AppModule* nalazi na adresi *src/app*. Pored *app* direktorijuma, u *src* direktorijumu se nalaze još i *assets* u kojem se nalaze *.css* i *.sass* datoteke, slike i fontovi. Ručno dodate biblioteke se nalaze u *libs* direktorijumu, a kursevi u direktorijumu *courses*. Pored dva pomenuta direktorijuma, u *root* direktorijumu projekta se nalaze i konfiguracione datoteke kao što su *package.json* u kojem su pobrojane sve korišćene biblioteke i *angular_cli.json* koja sadrži konfiguraciju *Angular CLI* alata (*Slika 11*).

Slika 16 – Hijerarhija direktorijuma

5.1 Moduli

AppModule je glavni i jedini modul aplikacije i zavisi od nekoliko modula koji su standardni za svaku *Angular 2* aplikaciju:

- *BrowserModule,,*
- *FormsModule,*
- *HttpModule,*
- *RouterModule.*

Pored njih su dodata još dva modula:

- *CodemirrorModule* i
- *ColorPickerModule*

koji su preuzeti sa sajta *GitHub* sa adresa *https://github.com/chymz/ng2-codemirror* i *https://github.com/Alberplz/angular2-color-picker*.

Svaka *Angular 2* aplikacija mora sadržati *BrowserModule* koji je odgovoran za pokretanje aplikacije i sadrži neke osnovne i najčešće korišćene direktive. *RouterModule* je

odgovoran za rutiranje aplikacije što je jedna od osnovnih osobina jednostraničnih aplikacija i doprinosi da korisnik ima isto iskustvo kao da koristi klasičnu aplikaciju. Njegova metoda *forRoot* kao argument prima niz objekata koji predstavljaju rute koje se koriste u aplikaciji. Na osnovu *URL* parametra komponenta zna da prikaže detalje kursa. Za čitanje parametara i njihovo prosleđivanje komponentama odgovoran je *RouteService*. Pored svega navedenog, u glavnom modulu je definisano i koja komponenta će se prikazivati prva pri pokretanju aplikacije, tzv. *bootstraping*, kao i niz *providers* kome su prosleđena oba servisa koja se koriste u aplikaciji.

Dobra je praksa da svaka komponenta kao i direktiva obavlja tačno jednu funkciju i da kao takva čini celinu. Time se postiže da se data komponenta može iznova i iznova koristiti na više mesta uz eventualne promene njene konfiguracije.

5.2 Komponente

Prva komponenta koja je roditeljska komponenta svih ostalih je komponenta pod nazivom *AppComponent.* U njenom *HTML* prikazu se nalaze pozivi komponenti *NavigationComponent, HeaderComponent, FooterComponent* i direktiva *router-outlet.* Pomenute komponente služe za iscrtavanje navigacije, zaglavlja i podnožja, dok direktiva *router-outlet* služi za iscrtavanje komponenti koje kontroliše modul *RouterModule* i koje je definisano u nizu koji se prosleđuje kao parametar metodi *forRoot* u glavnom modulu aplikacije. Kada se *URL* adresa poklopi sa nekom od definisanih ruta, iscrtava se njena komponenta. Komponenta *NavigationComponent* služi za prikazivanje navigacije i u okviru sebe poziva još jednu komponentu *MenuComponent* koja služi za prikaz menija koji u sebi sadrži linkove ka svim lekcijama platforme i važnim delovima sajta. *NavigationComponent* koristi resurse servisa *RouteService* kako bi menjala boju pozadine navigacije.

Najveća i funkcionalno najbitnija komponenta u celoj aplikaciji je *CourseComponent*. U njoj je implementirana funkcionalnost učitavanja lekcija, prelaska sa jedne na druge lekciju, prikaza primera i pomoći i pokretanja koda unetog u editor. Najzanimljivija od njih je svakako funkcionalnost vezana za pokretanje koda i prikaza u donjem desnom uglu ekrana. Ona se pokreće pri svakom novom učitavanju lekcije kada ona ima svoj primer koji treba odmah prikazati kao i na svaki klik dugmeta u traci sa akcijama pri dnu strane. Ona se nalazi u funkciji *previewCode()* koja kao parametre prima kôd koji treba pokrenuti, naziv *iframe* elementa u kojem će se prikazati i u kom režimu – normalnom ili uvećanom. Svaki put kada se pokrene, prvi korak je da se proveri da li u editoru ima unetog koda, ako ga nema izlazi se iz funkcije. Ako u editoru postoji neki kôd vrši se upit kojim se utvrđuje koji kurs je trenutno aktivan. Ukoliko aktivni kurs nije ni *TypeScript* ni *PHP*, tada se rade sledeće akcije:

1. Poziva se funkcija *removeIframe()* koja iz *DOM-a* izbacuje iframe u kojem se pokreće kôd.
2. Poziva se funkcija *makeIframe()* koja pravi i u *DOM* ubacuje *iframe*.
3. Vrši se upit da li iframe treba da se nalazi u *fullscreen* režimu ili ne i u zavisnosti od toga otvara novi prozor u *iframe* elementu ili u novoj kartici pretraživača.
4. U *document* objektu novog prozora upisuje uneti kôd.

Ova funkcionalnost se znatno razlikuje ukoliko je kurs na koji se primenjuje neki od gore pomenuta dva. Ukoliko je u pitanju *PHP*, uneti kôd se šalje preko *PHP Fiddle*[14] *API*-ja i čeka se njegov odgovor koji u sebi sadrži rezultat izvršavanja *PHP* koda. Nakon uspešno primljenog odgovora, rezultat se upisuje u iframe. Ukoliko je trenutni kurs *TypeScript*, implementacija funkcionalnosti je nešto komplikovanija, a svodi se na to da se u *DOM-u* pravi novi *script* element u koji se upisuje uneti kôd, zatim se taj kôd preko *TypeScript* prevodioca prevodi u *JavaScript* kôd, a onda on upisuje u *iframe*. U celom ovom procesu se koristi i *localStorage*[15]. Implementacija funkcionalnosti pokretanja koda je data na slici (*Slika 12*).

```
makeIframe() {
    var iframe = document.createElement('iframe');
    iframe.name = 'frame';
    iframe.style.backgroundColor = 'white';
    document.getElementsByClassName('lesson-preview')[0].appendChild(iframe);
  }

  removeIframe() {
    if (document.getElementsByName('frame').length > 0) {
      document.getElementsByClassName('lesson-
preview')[0].removeChild(document.getElementsByName('frame')[0]);
    }
  }

  previewCode(code: string, iframeName: string, fullscreen?: boolean) {
    if (code.length === 0) {
      return;
    }
    this.removeIframe();
    this.makeIframe();
    if (fullscreen === undefined || fullscreen === false) {
      var preview = window.open("", iframeName);
    } else if (fullscreen === true) {
      var preview = window.open("");
    }
    if (this.course === 'php') {
      $.ajax({
        url: "http://phpfiddle.org/api/run/code/json",
        type: "post",
        data: "code=" + encodeURIComponent(code),
        cache: false,
        dataType: "json",
        success: function (data) {
          preview.window.document.write(data.result);
        }
      })
    } else if (this.course === 'ts') {
      this.setInputStorage();
      this.compile();
      var out = localStorage.getItem('output');
      if (preview.window.document.body.hasChildNodes) {
```

[14] *PHP Fiddle* je internet stranica koja omogućava izvršavanje *PHP* koda direktno u pretraživaču. Ista funkcionalnost se može postići kroz *API* poziv *PHP Fiddle* servisu.

[15] *LocalStorage* je funkcionalnost *HTML5* jezika koja omogućava čuvanje podataka između sesija pretraživača. Za razliku od kolačića omogućava veći kapacitet za čuvanje podataka (do 5 MB).

```
        var body = preview.window.document.getElementsByTagName('body')[0];
        while (body.firstChild) {
          body.removeChild(body.firstChild);
        }
      }
      var script = preview.window.document.createElement("script");
      script.textContent = out;
      preview.window.document.body.appendChild(script);
      this.clearStorage();
    } else {
      preview.window.document.write(code);
    }
    preview.window.document.close();
  }

  setInputStorage() {
    if (this.code != null && this.code.length > 0) {
      localStorage.setItem('input', this.code);
    }
  }

  clearStorage() {
    localStorage.removeItem('input');
    localStorage.removeItem('output');
  }

  compile() {
    var script = document.createElement("script");
    script.type = "text/javascript";
    script.id = 'javaScript';
    var scriptString = `
            tsc();
        `;
    script.innerHTML = scriptString;
    document.body.appendChild(script);
    setTimeout(() => document.body.removeChild(document.getElementById('javaScript')),
200);
  }
```

Slika 17 – Implementacija funkcionalnosti pokretanja koda

Pored ove komponente postoji još jedna komponenta sa sličnom ulogom pod nazivom *BookComponent*. Ona praktično izvršava istu funkciju kao u *CourseComponent* sa razlikom što ne sadrži funkcionalnost vezanu za pokretanje koda. Ona se koristi za kurseve koji nemaju mogućnost pokretanja koda.

Komponenta *BottomNavigationComponent* je komponenta odgovorna za iscrtavanje trake sa akcijama na dnu ekrana. Na klik svakog dugmeta emituje se događaj koji u sebi sadrži informacije o tome koje dugme je kliknuto i koja funkcija treba da se izvrši. Taj događaj zatim biva uhvaćen od strane komponenti koje izvršavaju funkciju. Događaj koji se emituje je objekat tipa *EventEmitter* i predstavlja jedan od načina na koji mogu komponente međusobno komunicirati.

5.3 Servisi

Aplikacija sadrži dva servisa *SchoolService* i *RouteService* koji su odgovorni za dobijanje podataka koji se koriste u komponentama i za upravljanje navigacijom, odnosno rutiranjem kroz aplikaciju. Prvi od njih, *SchoolService*, implementira funkcije odgovorne za dobijanje podataka kao što su tekstovi lekcija, primera i pomoći, podaci o članovima tima koji su radili na aplikaciji, kursevima, spisku lekcija u svakom kursu i akcijama koje su aktivne na trenutnoj strani. Servis sadrži i logiku vezanu za dobijanje podataka o prethodnoj i sledećoj lekciji u odnosu na tekuću. Servis koristi *Observer Pattern* za implementaciju ovih funkcija i *Http* servis, a podatke iščitavaju iz *Json* fajlova koji se nalaze u *Data* direktorijumu. Nazivi ovih datoteka su:

- *Courses.json*,
- *Buttons.jsoni*,
- *Persons.json* i
- *Technologies.json*.

Ovakva arhitektura servisa ima mnogo pogodnosti. Logika prikupljanja podataka je potpuno izdvojena iz komponenti što predstavlja oblik *Design Pattern-a* pod nazivom *Repository Pattern*. Pogodnost se ogleda u tome što se lokacija podataka može brzo i lako promeniti ili čak tehnologija u kojoj su sačuvani, a da se aplikacija ne menja mnogo, samo ovaj servis. Dodavanje novih kurseva i lekcija je maksimalno uprošćeno i ne zahteva nikakvo znanje programiranja, a vrši se prostim unosom podataka u *Json* datoteke. Primer jednog kursa u *JSON* formatu je dat na slici (*Slika 13*).

```
{
        "name": "Veb",
        "id": "veb",
        "logo": "W",
        "type": "book",
        "lang": "",
        "displayLang": "",
        "detailsText": "Internet je svetski sistem umreženih računarskih mreža koji
povezuje veliki broj različitih mreža i računara širom cele planete na jedan
nehijerarhijski način. S obzirom na to da je veoma kompleksan, teško je definisati ga
jednom rečenicom. Sa jedne strane, internet se definiše preko hardverskih, komunikacionih i
softverskih komponenti koje ga sačinjavaju: kablovi, bežična veza, sateliti, ruteri,
internet dobavljači...",
        "lessons": [
            {
                "name": "Kako radi veb",
                "url": "veb_uvod",
                "type": "lesson",
                "parentCourse": "veb",
                "sublessons": null
            }
        ]
    }
```

Slika 18 – Primer konfiguracija kursa u JSON formatu

6 Zaključak

Veb programiranje je postala česta delatnost u savremenom društvu. U ovakvim okolnostima veb programer postaje veoma tražena profesija, a školovanje stručnih kadrova potreba. U tu svrhu je pokrenut projekat *eŠkola veba* – platforma čiji je zadatak da pomogne u savladavanju tehnologija koje se koriste na vebu. Namenjena je svima koji žele da uđu u svet veb programiranja, kao i onima koji žele da usavrše svoje znanje i nauče nešto novo. Osmišljena je da lako i intuitivno omogući učenje programiranja za veb kroz kurseve u kojima se obrađuju popularne tehnologije. Ideja za ovaj projekat je proistekla iz činjenice da postoji mnoštvo sličnih platformi na engleskom jeziku koje neretko nisu besplatne. *eŠkola veba* je besplatna, a lekcije koje su napisane na srpskom jeziku je izdvajaju od mnoštva pomenutih sličnih sajtova. Na njoj su okupljene lekcije iz različitih tehnologija koje se koriste ne vebu i kao takva čini jednu zaokruženu celinu. Njena interaktivnost i mogućnost pokretanja koda direktno iz lekcija treba da pomogne korisnicima da lakše savladaju tehnologije, a zadaci da provere naučeno.

Konačna verzija platforme nije napisana u tehnologijama u kojima je bila napisana na samom početku projekta. Na početku se koristio programski jezik *PHP* za implementaciju pojedinih delova i *JavaScript* biblioteka *jQuery* za učitavanje lekcija. Prva verzija platforme nije bila jednostranična aplikacija u potpunosti i nije koristila prednosti radnog okvira *Angular*, ali je intezivno korišćenje *AJAX*-a prikrivalo postojanje velikog broja statičkih stranica. Korisnički dizajn platforme se takođe menjao dok nije zadovoljio standarde koje je autor postavio pred sebe i poprimio svoj sadašnji oblik. Rad na projektu *eŠkola veba* je veoma doprinela na usavršavanju autora rada u pogledu veb programiranja kroz konstantno istraživanje novih tehnologija i produbljivanje postojećeg znanja.

Platforma je javno dostupna na adresi http://edusoft.matf.bg.ac.rs/eskola_veba.

Literatura

[1] Lerner Ari, Coury Felipe, Murray Nate, Taborda Carlos, *ng-book 2*, 2016

[2] Sajt *World Wide Web* konzorcijuma: https://www.w3 .org/

[3] Sajt *Mozilla Developer Network*: https://developer.mozilla.org/en-US

[4] Zvaničan sajt radnog okvira *Bootstrap 4*: https://v4-alpha.getbootstrap.com/

[5] Zvaničan sajt jezika *Sass*: http://sass-lang.com/

[6] Zvaničan sajt radnog okvira *Angular*: https://angular.io/

[7] Zvaničan sajt jezika *TypeScript*: http://www.typescriptlang.org/docs/tutorial.html