ELEKTROTEHNIČKI FAKULTET UNIVERZITETA U BEOGRADU

INTERAKTIVNI KURSEVI ZA INTERNET PROGRAMIRANJE

– Diplomski rad –

Kandidat:

Bojan Anđelkoski 2013/0482

Mentor:

Doc. dr Zoran Čiča

Beograd, mart 2018.

SADRŽAJ

1. UVOD

U današnje doba gotovo ne postoji osoba koja nije koristila Internet. Pojednostavljenom definicijom, Internet se određuje kao svetska komunikaciona mreža, koja je takođe nazvana "mreža svih mreža" i sastoji se od velikog broja zasebnih računara uvezanih u mrežnu strukturu. Tehnički gledano, Internet predstavlja globalni informacioni sistem, logički povezan jedinstvenim sistemom adresiranja putem Internet protokola (TCP/IP - *Transmission Control Protocol/Internet Protocol*), ili drugih protokola kompatibilnih sa Internet protokolom, i koji obezbeđuje, koristi ili omogućava servise visokog nivoa za ličnu i poslovnu primenu.

WWW (*World Wide Web*) je sistem Internet servera koji podržava hipertekst za pristup određenom broju Internet protokola. Upotreba Interneta raširena je danas po čitavom svetu. Računari se raspoznaju po domenu (*domain name*), a listu domena vodi *Network Information Centre* (NIC). Zahvaljujući sposobnosti da se podrži rad sa multimedijom i naprednim programskim jezicima, WWW predstavlja komponentu Interneta koja se najbrže razvija.

Da bi se koristio WWW neophodno je da na uređaju postoji instaliran softver (*software*) za čitanje WWW prezentacija - tzv. pretraživač (*browser*). Trenutno najpoznatiji pretraživači su: Chrome, Firefox, Internet Explorer, Opera i dr. Cilj pretraživača je da čita HTML (*HyperText Markup Language*) stranice i prikaže ih kao veb stranice (*website*).

Veb stranica jeste mesto na Internetu koje sačinjava više različitih informacionih celina ili aplikacija. Svaka veb stranica može biti kombinacija teksta, slika, animacija, audio i video sadržaja, hiperlinkova (*hyperlinks*) i slično. Pomoću hiperlinkova povezuje se različiti sadržaj i različite veb stranice na Internetu.

Glavni cilj aplikacije, realizovane u okviru ove teze, je da se kroz interaktivne kurseve nauči Internet programiranje. Razlog zašto najčešće korisnici, koji žele da nauče Internet programiranje, odustaju ili smatraju da je veoma teško da nauče ove tehnologije jeste to što kursevi tipično ne nude praktičnu implementaciju teorije koje izučavaju. To znači da korisnik nakon završetka kursa ne poseduje praktičnu stranu koja je najbitniji deo prilikom izučavanja ovih tehnologija.

Zbog prethodno pomenutog problema, ideja ove teze je da se realizuje aplikacija kojom će kroz teoriju, koju je neophodno naučiti, korisnik morati da reši praktične probleme i direktno primeni naučenu teoriju. Velika prednost interaktivnih kurseva je što mogu biti veliki podsticaj korisniku da nastavi da izučava datu oblast nakon što vidi njenu praktičnu primenu, ali i mogućnosti koje pružaju alati koje izučava i može ih implementirati.

U okviru ovog rada, najpre će se definisati osnovni pojmovi vezani za Internet programiranje. Zatim će se ukratko prikazati razvojni alati korišćeni prilikom izrade aplikacije, primena ovih alata, kao i razloge zašto su upravo ti alati korišćeni. Nakon toga će biti prikazana struktura veb aplikacije kao i struktura baze podataka. Da bi korisnici znali kako se veb aplikacija koristi, detaljnije će biti predstavljene sve funkcije aplikacije, posebno za svaki korisnički tip korisnika. Nakon toga će biti prikazana jedna karakteristična implementacija koja je veoma korisna, zato što se koristi u gotovo svim modernim veb aplikacijama, a to je autentifikacija. Na kraju, biće izvedeni zaključci i date ideje i pravci za dalje unapređivanje aplikacije.

2. OSNOVE INTERNET PROGRAMIRANJA

Internet programiranje je trenutno jedno od najtraženijih i najplaćenijih zanimanja u svetu [4]. Svakodnevno se izradi više od 400000 veb stranica, što govori da je Internet programiranje veoma popularna tema u svetu programiranja [5].

Osnovni pojmovi koji se koriste u svetu Internet programiranja su:

- HTML (*HyperText Markup Language*) je opisni (*markup*) jezik koji omogućava definisanje strukture jedne veb stranice pomoću opisnih tagova (*markup tags*). Osim strukture veb stranice, pomoću opisnih tagova definiše se i sadržaj veb stranice (tekst, slike ...), podaci o autoru, naslov dokumenta i slično. Trenutni standard za HTML je HTML 5.2. HTML standarde definiše W3C (*World Wide Web Consortium*) organizacija.
- CSS (*Cascading Style Sheets*) je jezik koji se koristi da bi se definisao stil HTML dokumenta. Pomoću CSS-a može se lako da promeniti izgled (boje, pozadine, veličine, font, ...) veb stranice. Kao i HTML, standarde za CSS definiše W3C organizacija. Trenutni CSS standard je CSS3.
- Za razliku od HTML-a i CSS-a, Javascript je skriptni programski jezik koji je zasnovan na objektima, i pomoću koga se statični HTML sadržaj može dinamički menjati. Javascript omogućava da se vrši promena sadržaja i stil veb stranice na osnovu akcija koje je posetilac napravio (klik, skrol, ...), nakon određenog vremena, u nekom intervalu, nakon učitavanja stranice i slično. Standard za Javascript definiše *ECMA International* organizacija, a trenutni standard je ECMA-262.
- PHP (*Hypertext Preprocessor*) je programski jezik koji se pokreće na serveru (*server-based*). Pomoću PHP programskog jezika može se generisati sadržaj za veb stranice u zavisnosti od stranice koju korisnik želi da poseti. Ovaj programski jezik omogućava rad sa datotekama, zatim, matematičke operacije, povezivanje sa bazama podataka, a ima mogućnost za OOP (*Object Orinted Programming*)... PHP kod se ne prikazuje krajnjem korisniku, nego samo sadržaj koji je rezultat operacija koje se izvrše u PHP programskom jeziku. Standard koji definiše PHP je PSR (*PHP Standard Recommendation*), a zadnja verzija PHP-a je 7.2.
- MySQL (*My Structured Query Language*) je sistem za upravljanje bazama podataka. Konkretnije, MySQL je sistem za upravljanje relacionom bazom podataka, odnosno, RDBMS (*Relational Database Managment System*). Većina savremenih veb aplikacija za upravljanje podacima oslanja se na MySQL. Poslednja verzija MySQL-a je 8.0.

Tehnologija koja se koristi za izradu veb stranica i veb aplikacija svakodnevno se menja i unapređuje. Kreiraju se razvojni alati koji olakšavaju posao programera, ali ujedno, samim tim, i "kreiraju" lošije programere, zato što počinju da primenjuju te razvojne alate bez prethodnih osnovnih znanja o Internet programiranju.

3. RAZVOJNI ALATI KORIŠĆENI PRI IZRADI APLIKACIJE

Aplikacija iz ovog rada realizovana je u vidu SPA (*Single Page Application*). SPA su veb aplikacije koje imaju samo jednu HTML stranicu i dinamički ažuriraju sadržaj te HTML stranice u zavisnosti od interakcije korisnika. Trenutno su SPA principi najpopularnija i najnovija rešenja za kreiranje veb aplikacija. Stranice, kao što su: Facebook, Instagram, Google, Youtube, Amazon, Airbnb, počinju da menjaju (a neki od njih su i potpuno zamenili) starije tehnologije i svoje servise dizajniraju kao SPA.

Najpopularniji programski jezik za realizovanje SPA je Javascript. Pomoću raznih implementacija ovog programskog jezika dolazi se do rešenja koja omogućavaju kreiranje SPA.

Do pre nekoliko godina Javascript je bio smatran samo za korisnički (*client-side*) programski jezik koji nije imao mogućnost da bude korišćen da bi se izvršile akcije na serveru (kao što to, na primer, omogućava PHP). Kako je popularnost ovog programskog jezika drastično skočila u poslednjih par godina[6], razvila se mogućnost da se Javascript koristi i kao programski jezik za interakcije sa serverom (*server-side*) pomoću odgovarajućih razvojnih alata. Samim tim, PHP programski jezik polako gubi svoju slavu i sve se više programera odlučuje da izučava Javascript, pre nego PHP.

Za realizaciju ove aplikacije korišćena su dva razvojna alata:

1) Laravel - za interakciju sa bazom podataka.
2) Vue.js - za kreiranje SPA.

3.1. Laravel - razvojni alat na PHP osnovi

Laravel je besplatni razvojni alat, čiji je kod dostupan javnosti (*open source*) i baziran je na PHP programskom jeziku. Kreiran je od strane Taylor Outwell, i namenjen je programerima veb aplikacija baziranih na PHP programskom jeziku koje prate MVC (*Model View Controller*) arhitekturu. Trenutno je u svetu najpopularniji razvojni alat zasnovan na PHP, zato što omogućava brzi razvoj veb aplikacija.

MVC arhitektura je jako popularan način razvoja veb aplikacija[7]. Zasniva se se na tome da se veb aplikacija može podeliti na tri dela:

- *Model* – deo koji je zadužen za interakciju aplikacije sa bazom podataka
- *View* – deo koji je zadužen za prikaz sadržaja aplikacije korisniku
- *Controller* – deo koji povezuje *Model* i *View*. Služi za obradu podataka koji korisnik preko *View* dela želi da sačuva ili izmeni u bazi podataka, ili da se izvrši neka obrada sadržaja kada korisnik želi da dobije neke informacije iz baze podataka.

Program koji se napiše u Laravelu ne zahteva prevođenje (tj. kompajliranje), nego se interpretira pri svakom izvršavanju. Laravel ne poseduje početnu (tj. glavnu) funkciju, već jednostavno sadrži skup naredbi, koje se izvršavaju jedna za drugom, od prve do poslednje, gde se poslednja naredba ujedno smatra i krajem programa.

Pomoću Laravel-a možemo lako i efikasno da kreiramo veb aplikacije. Sve funkcije i mogućnosti koje ovaj razvojni alat pruža su modularne. To znači da se može koristiti samo ono što je potrebno za razvoj željene veb aplikacije, a da se u aplikaciji ne dodaje dodatni programski kod koji ne bi potencijalno bio od bilo kakve koristi.

Laravel nas lišava pisanja koda uobičajenih zadataka koji se koriste u većini veb aplikacija, kao što su: provere identiteta, rutiranje, sesije i keširanje. Ima jedinstvenu arhitekturu, pa omogućava programerima da kreiraju sopstvenu infrastrukturu koja je specijalno dizajnirana za njihovu primenu. Laravel se ne koristi samo za veće veb aplikacije već je i odličan izbor i za male i mikro veb aplikacije.

Razni moduli, a i sam Laravel, se mogu instalirati pomoću *composer*-a. Composer je menadžer PHP komponenti. Koristi se da bi instalirali, ažurirali ili obrisali komponente zasnovane na PHP programskom jeziku sa veb aplikacije.

Laravel se takođe može koristiti kao RESTful API (*Representational State Transfer Application Program Interface*). Svrha ovog veb servisa je da omogući laku komunikaciju između korisnika i servera bez potrebe da se prethodno stanje sačuva (*stateless*). Ovaj veb servis nudi keširanje tako što omogućava korisniku brži pristup podacima. Postoje četiri metode koje se koriste kod REST servisa, a to su:

1) GET - metoda koja služi za dohvatanje resursa.
2) POST - metoda koja služi za kreiranje resursa.
3) PUT - metoda koja služi za ažuriranje resursa.
4) DELETE - metoda koja služi za brisanje resursa.

Artisan je ime komandne linije u Laravelu. Ona pruža mnogo korisnih komandi tokom razvoja aplikacije. Pomoću ove komandne linije mogu se izvršavati različite akcije kao, na primer:

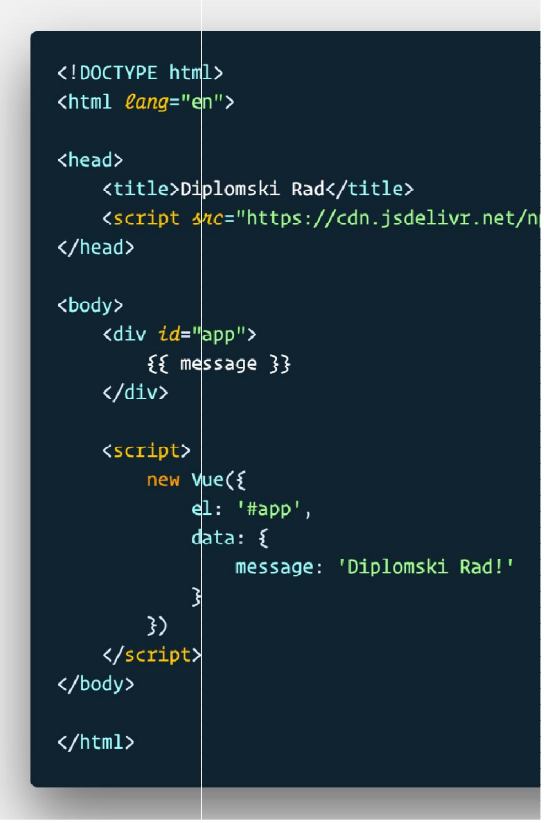
- Kreiranje Controller-a,
- Kreiranje View-a,
- Kreiranje Model-a,
- Kreiranje Migracije (koje se koriste da bi definisali strukturu neke tabele unutar struktura podataka),
- Kreiranje Seeder-a (koje se koriste da bi popunili bazu sa nekim sadržajem koji je pogodan za testiranje razvijane aplikacije),
- Kreiranje Middleware-a (koji se koriste da bi izvršili neku operaciju/proveru pre nego što korisnik može da pristupi nekoj ruti),
- ...

Jedna od najvažnijih osobina kod Laravel-a je to što ima odličnu dokumentaciju. U dokumentaciji su detaljno opisani svi moduli, njihov način rada i primeri kako se oni mogu koristiti.

3.2. Vue.js - Razvojni alat na osnovu Javascript-a

Vue.js je besplatni razvojni alat, čiji je kod dostupan javnosti (*open source*) i zasnovan je na Javascript programskom jeziku. Koristi se za kreiranje SPA, kao i za interaktivne korisničke interfejse. Kreirao ga je programer Evan You, a namenjen je programerima koji žele da lako kreiraju moderne i interaktivne veb aplikacije.

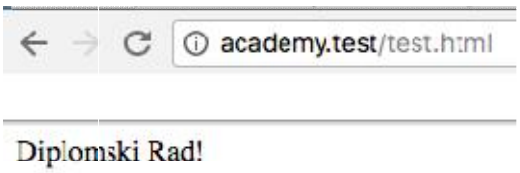
Veoma je bitno napomenuti da za korišćenje ovog razvojnog alata nije potrebno da se vrši kompajliranje tj. postoji mogućnost da se on direktno koristi u postojećim veb aplikacijama. To znači da radi kao neke starije Javascript biblioteke (kao, na primer, jQuery), ali uvodi moderniji pristup i novi nivo načina interakcije između korisnika i veb aplikacija. Ukoliko se želi koristiti Vue.js bez kompajliranja, potrebno je da se u HTML dokumentu doda Vue.js biblioteka, zatim, se kreira nova Vue instanca, a potom se izabere HTML element na kojem će se Vue instanca pokrenuti.



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

<head>
    <title>Diplomski Rad</title>
    <script src="https://cdn.jsdelivr.net/n
</head>

<body>
    <div id="app">
        {{ message }}
    </div>

    <script>
        new Vue({
            el: '#app',
            data: {
                message: 'Diplomski Rad!'
            }
        })
    </script>
</body>

</html>
```

Slika 3.2.1 - Korišćenje Vue.js direktno u HTML bez potrebe kompajliranja

Ako se otvori fajl *test.html* čiji je izvorni kod prikazan na slici 3.2.1, dobija se rezultat koji je prikazan na slici 3.2.2.



Slika 3.2.2 - Izgled veb stranice čiji je izvorni kod prikazan na slici 3.2.1

Najpopularnija karakteristika ovog razvojnog alata su Vue Komponente (*Vue Components*). Nakon definisanja jedne komponente ona se može koristiti više puta na različitim mestima aplikacije, što omogućava da se kod koji se ponavlja definiše jednom, a koristi više puta. Svaka Vue komponenta se sastoji od tri dela:

- *<template></template>* tag u kome se definiše struktura komponente pomoću HTML-a.
- *<script></script>* tag u kome se definiše način funkcionisanja komponente pomoću Javascript-a.
- *<style></style>* tag u kome se definiše stil i izgled komponente.

Nakon definisanja komponente, njene funkcije i izgleda, definiše se koji će tag ona koristiti. To znači da se ona u HTML dokumentu dodaje tako što se napiše tag koji je prethodno definisan za datu komponentu. Primera radi, ako je napravljena komponenta *TestComponent* za koju je rezervisan tag *<test-component>*, ta komponenta i sve njene funkcije se mogu koristiti u HTML dokumentu (ili u nekoj drugoj komponenti) tako što se samo napiše taj *<test-component></test-component>* tag. Kao što se vidi, ovo je vrlo jednostavno (slika 3.2.3).

Slika 3.2.3 - Struktura jedne Vue komponente

Druga bitna osobina Vue.js razvojnog alata je reaktivnost (*reactivity*). Reaktivnost omogućava da se, u zavisnosti od interakcije korisnika sa veb aplikacijom, izmeni neka promenljiva ili izvrši neka operacija bez potrebe da se promeni lokacija veb stranice. Da bi se bolje razumelo zašto je reaktivnost bitna, razmotriće se dva jednostavna primera.

1) Cilj je promeniti naslov na veb stranici u zavisnosti od teksta koji korisnik unosi u nekom elementu za unos. Pomoću reaktivnosti Vue.js-a može se direktno menjati naslov kako korisnik unosi tekst u elementu za unos, a da veb stranica ne mora da se ponovo učita.

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

<head>
    <title>Diplomski Rad</title>
    <script src="https://cdn.jsdelivr.net/npm/vue"></s
</head>

<body>
    <div id="app">
        <h1>{{ message }}</h1>

        <input type="text" v-model="message">
    </div>

    <script>
        new Vue({
            el: '#app',
            data: {
                message: 'Diplomski Rad!'
            }
        })
    </script>
</body>

</html>
```

Slika 3.2.4 - Izgled koda koji korišćen za ilustraciju jednostavnog primera reaktivnosti Vue.js-a.

Slika 3.2.5 - Prikaz koda koji je definisan na slici 3.2.4 u pretraživaču.

2) Cilj je ispisati listu postojećih korisnika u veb aplikaciji. Nakon učitavanja veb stranice, korisnike smeštamo u niz *korisnici*. Nakon ispisivanja svih korisika, pretpostavimo da je na neki način obrisan jedan korisnik sa liste. Ono što treba primetiti je da se veb stranica nije ponovo učitala, a korisnik se ne vidi u sadržaju veb stranice. Na slici 3.2.5 se može videti kod (HTML fajl) kojim se može testirati ovaj primer tako što se kopira dati kod u neku datoteku, snimi se datoteka kao test.html i otvori se taj fajl u pretraživaču.

Slika 3.2.5 - Kod koji se može koristiti za ilustraciju reaktivnosti Vue.js-a.

Vue.js može da se koristi u kombinaciji sa velikim brojem razvojnih alata. Veb aplikacije se najčešće sastoje od dva dela:

- *Frontend* - deo aplikacije koji gleda korisnik (*client-side*).
- *Backend* - deo aplikacije koji ne gleda korisnik i služi da se korisnik na neki način poveže sa serverom (*server-side*).

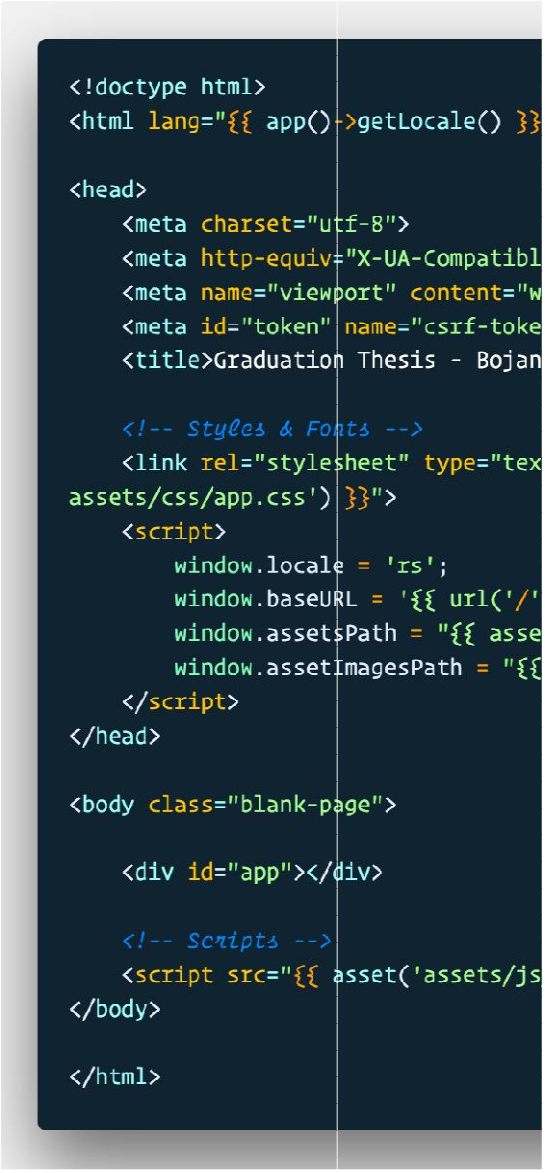
Kako je već napomenuto da se Vue.js koristi za interaktivne korisničke interfejse, to znači da se Vue.js koristi za izradu *Frontend* dela veb aplikacije. Drugi deo, tj. *Backend* može da se napravi pomoću različitih tehnologija i razvojnih alata (*Node.js*, *Express*, *Laravel*, *Rails*, i sl.).

Za potrebe aplikacije realizovane u ovoj tezi koristiće se Vue.js za *Frontend* deo, a Laravel kao RESTful API za *Backend* deo.

4. STRUKTURA APLIKACIJE

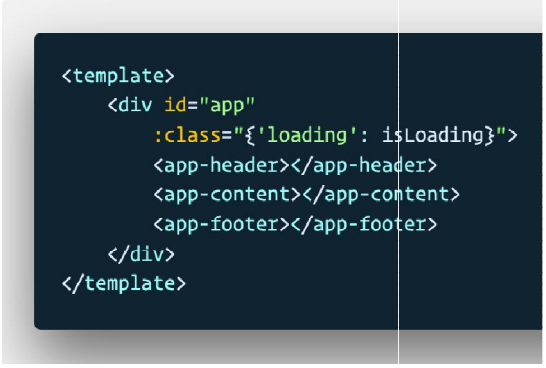
Kao što je već napomenuto, projekat je realizovan pomoću Laravel-a, koji služi kao RESTful API tj. *Backend* i Vue.js koji je zadužen za korisnički deo tj. *Frontend.*

Da bi se kreirao SPA, potrebno je svaku putanju, koju korisnik može otvoriti, prebaciti u HTML dokument gde će se pokrenuti Vue instanca. Nakon pokretanja Vue instance, svaka putanja koju korisnik želi da poseti biće obrađena od strane Vue.js-a.



```
<!doctype html>
<html lang="{{ app()->getLocale() }}

<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatibl
    <meta name="viewport" content="w
    <meta id="token" name="csrf-toke
    <title>Graduation Thesis - Bojan

    <!-- Styles & Fonts -->
    <link rel="stylesheet" type="tex
assets/css/app.css') }}">
    <script>
        window.locale = 'rs';
        window.baseURL = '{{ url('/'
        window.assetsPath = "{{ asse
        window.assetImagesPath = "{{
    </script>
</head>

<body class="blank-page">

    <div id="app"></div>

    <!-- Scripts -->
    <script src="{{ asset('assets/js
</body>

</html>
```

Slika 4.1 - Izgled HTML stranice gde se pokreće Vue.js instanca



Slika 4.2 - Glavna Vue komponenta realizovane aplikacije.

Struktura aplikacije se bazira na Vue komponenti. Za svaki deo aplikacije je kreirana nova Vue komponenta. Na slici 4.2 je prikazana glavna komponenta realizovane aplikacije. Ona se sastoji od tri Vue komponente:

1) <*app-header*> - komponenta zadužena za gornji deo sajta (*header*).
2) <*app-content*> - komponenta zadužena za glavni sadržaj sajta (*main*).
3) <*app-footer*> - komponenta zadužena za donji deo sajta (*footer*).

Da bi se prikazao sadržaj tj. stranica u zavisnosti od putanje na kojoj je trenutno posetilac, koristi se specijalni tag koji Vue.js omogućava, a to je <*router-view*> tag. Za svaku putanju definiše se koju komponentu treba zameniti na mestu <*router-view*> taga. Za realizovanu aplikaciju definisane su sledeće stranice (komponente) koje prikazuju odgovarajući sadržaj u zavisnosti od tipa korisnika:

1) Course - stranica na kojoj je prikazan jedan kurs,
2) Courses - stranica na kojoj su prikazazni svi kursevi,
3) CreateCourse - stranica na kojoj se kreira novi kurs,
4) CreateLesson - stranica na kojoj se kreira nova lekcija,
5) Dashboard - kontrolna tabla,
6) Home - početna stranica sajta,
7) Lesson - stranica na kojoj je prikazana jedna lekcija,
8) PageNotFound - stranica koja je prikazana kada korisnik poseti neku putanju koja nije definisana tj. ne postoji,
9) Playground - stranica na kojoj korisnik može da testira svoje znanje iz oblasti HTML-a, CSS-a, i Javascript-a i uživo proveri kako će njegov kod izgledati kad bude pregledan u nekom pretraživaču,
10) Questions - stranica na kojoj se nalazi spisak pitanja koja su korisnici postavili,
11) UpdateLesson - stranica na kojoj se nalazi forma za ažuriranje lekcije,
12) UpdateUser - stranica na kojoj se nalazi forma za ažuriranje korisnika,
13) User - stranica na kojoj su prikazani osnovni podaci o korisniku,
14) Users - stranica na kojoj su prikazani svi korisnici.

5. STRUKTURA BAZE PODATAKA

Baza podataka za realizovanu aplikaciju je definisana pomoću *Schema* klase koja je deo Laravel razvojnog alata. Nekoliko bitnih metoda za definisanje strukture tabela klase Schema, koje služe da se opiše kolona untar tabele u bazi podataka jesu:

- *increment()* - kreiranje kolone koja je unikatni identifikator i ima svojstvo *AUTO_INCREMENT*(za svaki novi zapis se identifikator povećava za 1),
- *integer()* - kreiranje *INTEGER* (celobrojne) kolone,
- *string()* - kreiranje *VARCHAR*(niz znakova) kolone,
- *default()* - podrazumevana vrednost za datu kolonu,
- *timestamp()* - kreiranje dve *TIMESTAMP* kolone (created_at i updated_at),
- *unique()* - dodavanje uslova da je kolona unikatna tj. da ne mogu postojati dve iste vrednosti u tabeli za ove kolone,
- *boolean()* - kreiranje kolone čije vrednosti mogu biti samo 0 i 1,
- *text()* - kreiranje *TEXT* (tekstualni sadržaj) kolone,
- *nullable()* - dodavanje uslova da sadržaj unutar ove kolone može biti prazan,
- *unsignedInteger()* - kreiranje *UNSIGNED INTEGER* (celi broj veći od nule) kolone,

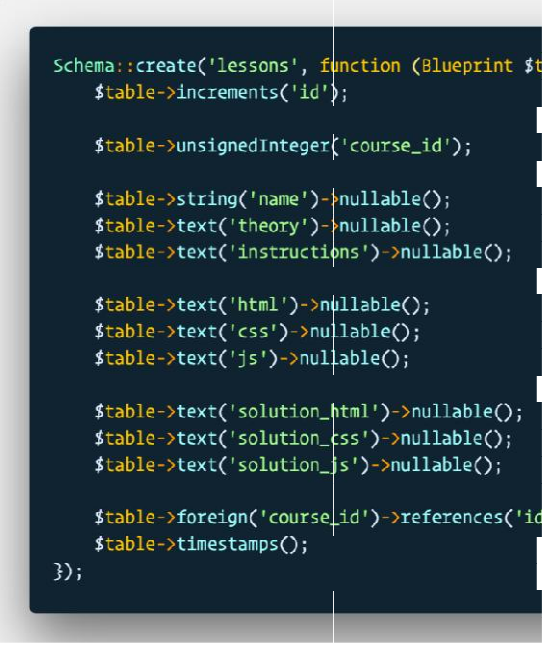
Realizovana baza podataka se sastoji od 6 tabela:

1) users - tabela u kojoj su smešteni podaci o korisnicima,
2) courses - tabela u kojoj su smešteni podaci o kursevima,
3) lessons - tabela u kojoj su smešteni podaci o lekcijama,
4) questions - tabela u kojoj su smešteni podaci o pitanjima.
5) leson_user - tabela u kojoj su smešteni podaci o rešenjima koje su korisnici ispratili za odgovarajuću lekciju,
6) roles - tabela u kojoj su smešteni tipovi korisnika.

Slika 5.1 - Definicija tabele roles

Slika 5.2 - Definicija tabele courses

Slika 5.3 - Definicija tabele lesson_users



Slika 5.4 - Definicija tabele lessons

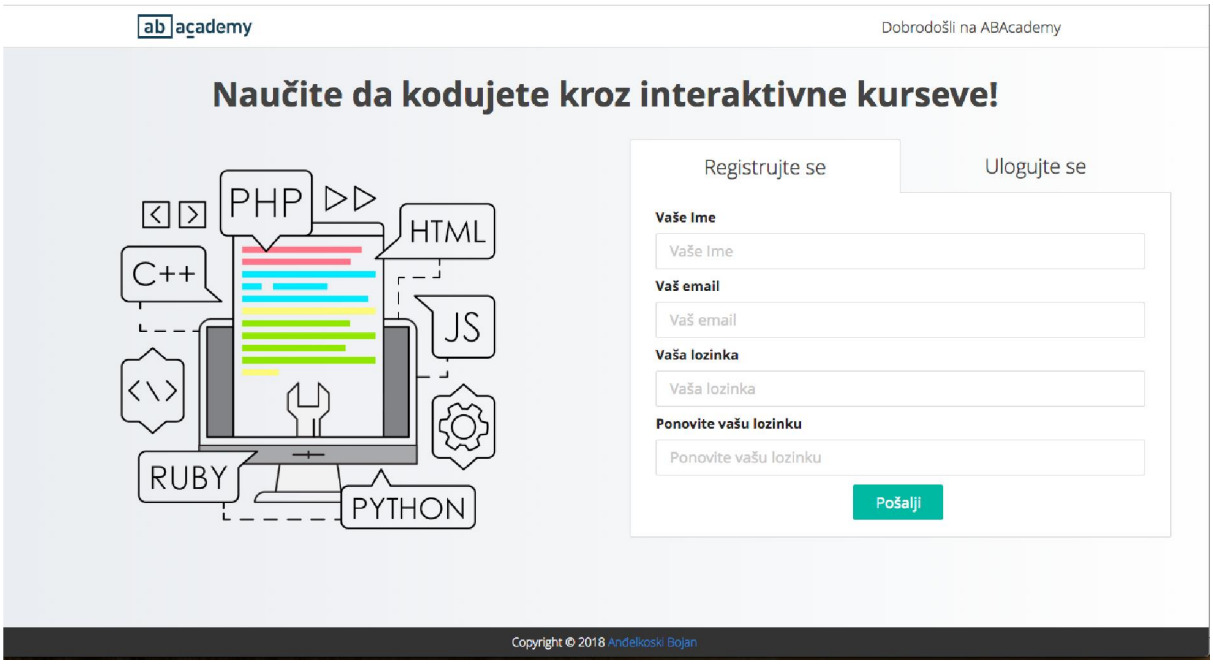
Slika 5.5 - Definicija tabele questions

Slika 5.6 - Definicija tabele users

6. UPUTSTVO ZA KORIŠĆENJE APLIKACIJE

U ovom odeljku će biti detaljnije objašnjene sve funkcije koje su implementirane u realizovanoj aplikaciji. Objasniće se na koji način oba tipa korisnika (administrator i običan korisnik) mogu da koriste sve funkcije koje im stoje na raspolaganju u realizovanoj aplikaciji. Uložen je napor da se učini upotreba aplikacije što jednostavnijom i intuitivnijom za sve korisnike.

Kada se otvori aplikacija u nekom pretraživaču, otvara se početna strana aplikacije. Na početnoj strani aplikacije postoje samo forma za registraciju i forma za logovanje. Registracija je besplatna, i otvorena za sve posetioce. Nakon registracije korisnik je automatski ulogovan i može da koristi sve funkcije aplikacije.



Slika 6.1 - Izgled početne strane aplikacije.

Nakon što se korisnik registruje ili uloguje, u zavisnosti od tipa korisnika prikazuje se odgovarajući sadržaj.

6.1. Administrator

Administrator je vlasnik aplikacije i on ima potpunu kontrolu nad svim sadržajem koji se nalazi u aplikaciji.

U gornjem delu aplikacije (*header*) administrator ima sledeće linkove:

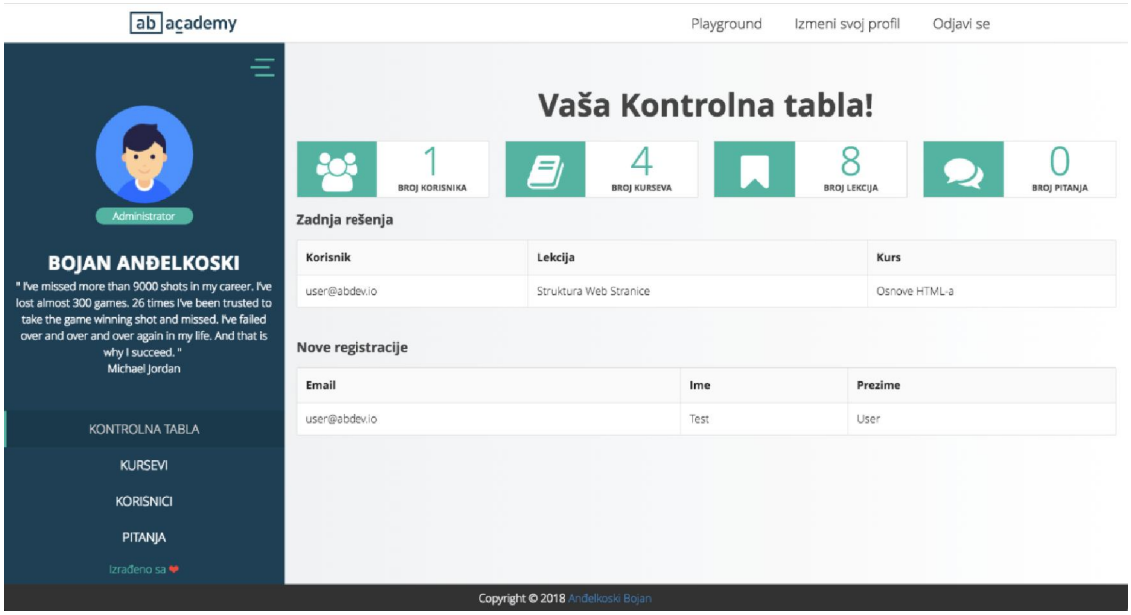
1) *Playground* - stranica na kojoj korisnici mogu da u realnom vremenu testiraju svoj HTML, CSS i Javascript kod,
2) *Izmeni svoj profil* - stranica na kojoj korisnici mogu menjati svoje informacije,
3) *Odjavi se* - link koji omogućava korisniku da završi korišćenje aplikacije i odjavi se.

U levom delu aplikacije (*sidebar*) administrator ima sledeće linkove:

1) *Kontrolna Tabla,*
2) *Kursevi* - stranica preko koje se mogu dodavati/ažurirati/brisati kursevi i lekcije,
3) *Korisnici* - stranica preko koje se mogu ažurirati i brisati korisnici,
4) *Pitanja* - stranica preko koje se može odgovarati na pitanja koja su korisnici postavili.

6.1.1. Kontrolna Tabla

Kada se administrator uloguje otvara se njegova kontrolna tabla. Na kontrolnoj tabli administrator može da vidi neke osnovne statistike, poslednja rešenja koje su korisnici ispratili i najnovije registracije.



Slika 6.1.1 - Izgled kontrolne table administratora

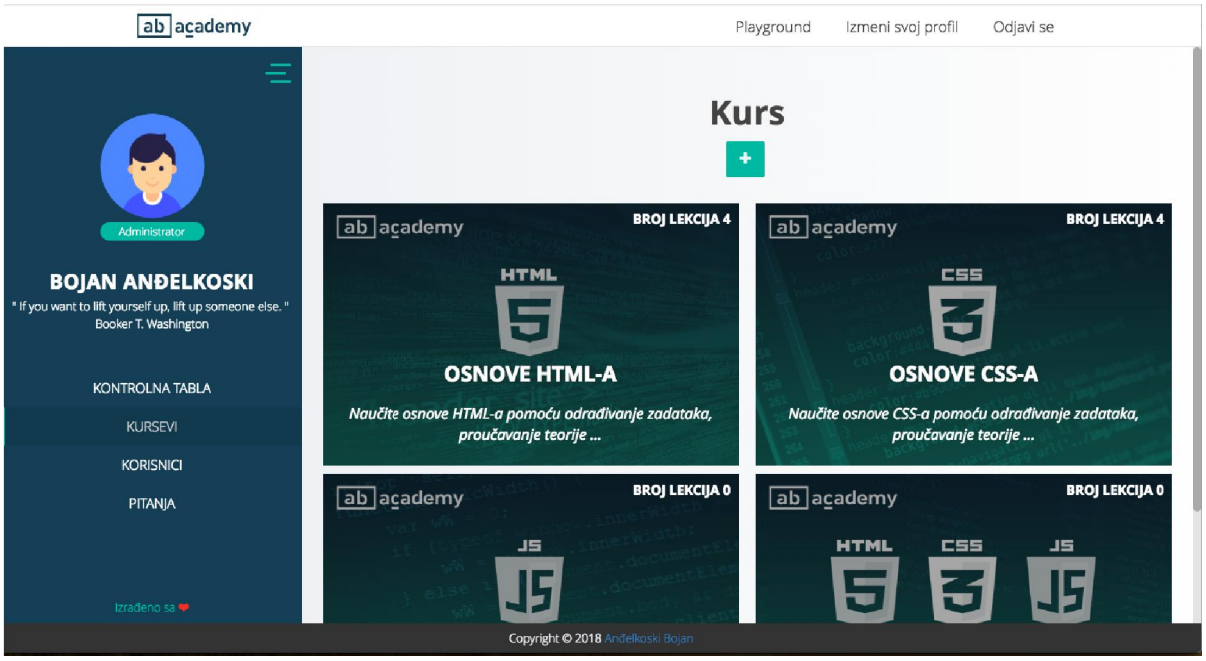
6.1.2. Kursevi

Administrator može da dodaje, ažurira i briše kurseve. Kada se uđe na stranicu „kursevi“, dobija se prikaz svih kurseva u aplikaciji. Takođe sa ove stranice se može dodatio novi kurs klikom na + ikonice.

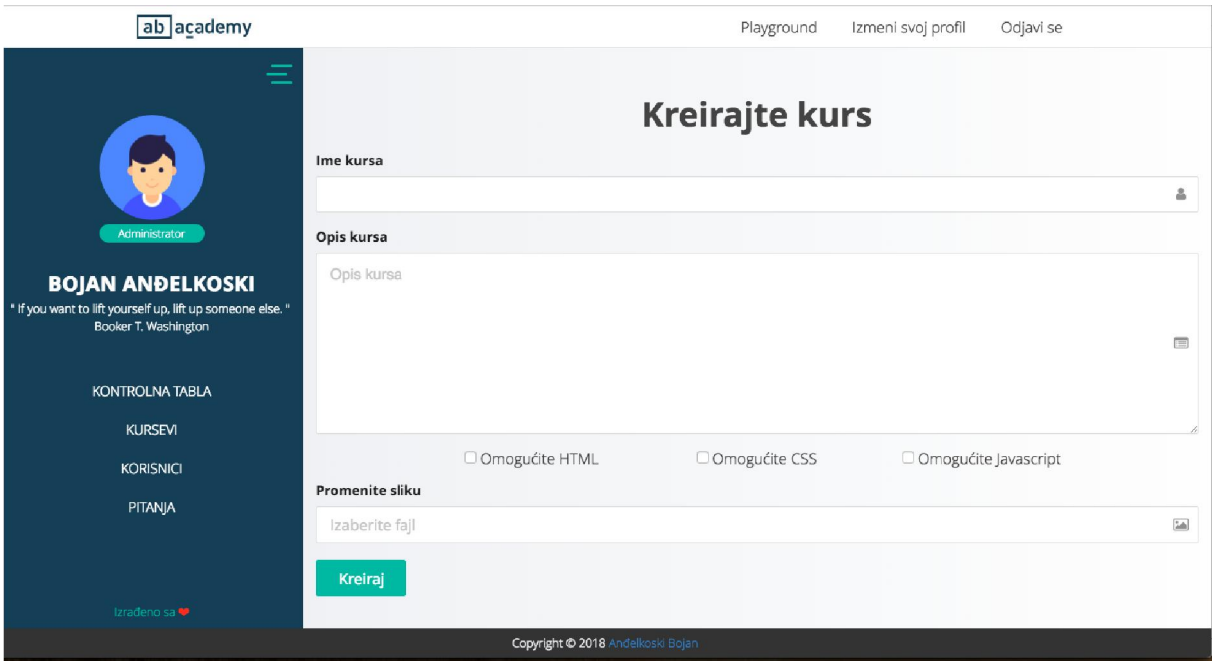
Klikom na neki kurs otvara se stranica tog kursa u okviru koga se mogu menjati podaci tog kursa (forma se nalazi ispod lekcije tog kursa). Briše se kurs tako što se klikne na crveno dugme, a omogućeno je da se vide sve lekcije tog kursa kao i da se dodaju/ažuriraju/brišu lekcije tog kursa.

Pri kreiranju i ažuriranju kurseva, za svaki kurs treba popuniti ime i opis kursa, izabrati jezike koji su omogućeni za dati kurs (HTML, CSS, Javascript), i izabrati sliku tog kursa. Sliku koja se izabere za kurs takođe će biti iskorišćena za sve lekcije tog kursa.

Pri kreiranju i ažuriranju kurseva, podrazumeva se da su sva polja popunjena, i da je izabrana validna slika. Ukoliko nisu ispunjeni uslovi validacije sadržaja pojaviće se greška.



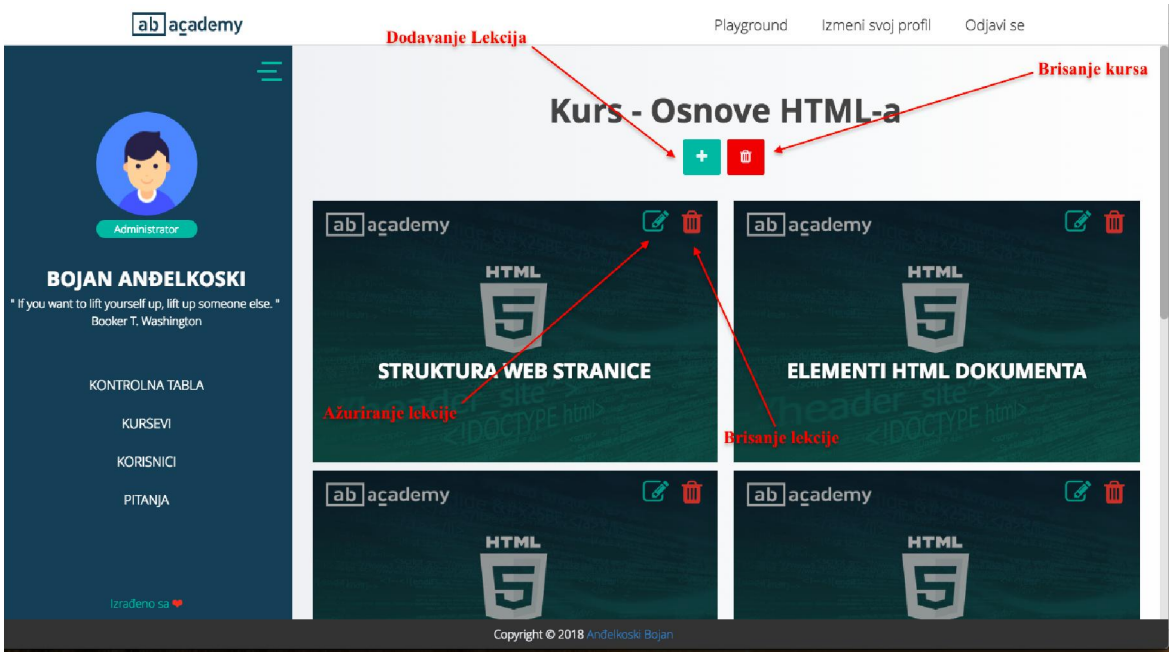
Slika 6.1.2.1 - Izgled stranice kursevi (administrator)



Slika 6.1.2.2 - Izgled forme za kreiranje kursa

6.1.3. Lekcije

Nakon što se uđe na stranicu nekog kursa, pojaviće se spisak svih lekcija koje taj kurs ima. Klikom na zeleno + dugme može se dodati lekcija tom kursu. Klikom na zeleno dugme sa olovkom unutar lekcije otvoriće se forma za ažuriranje lekcije. Klikom na crveno dugme unutar lekcije vrši se brisanje lekcije.

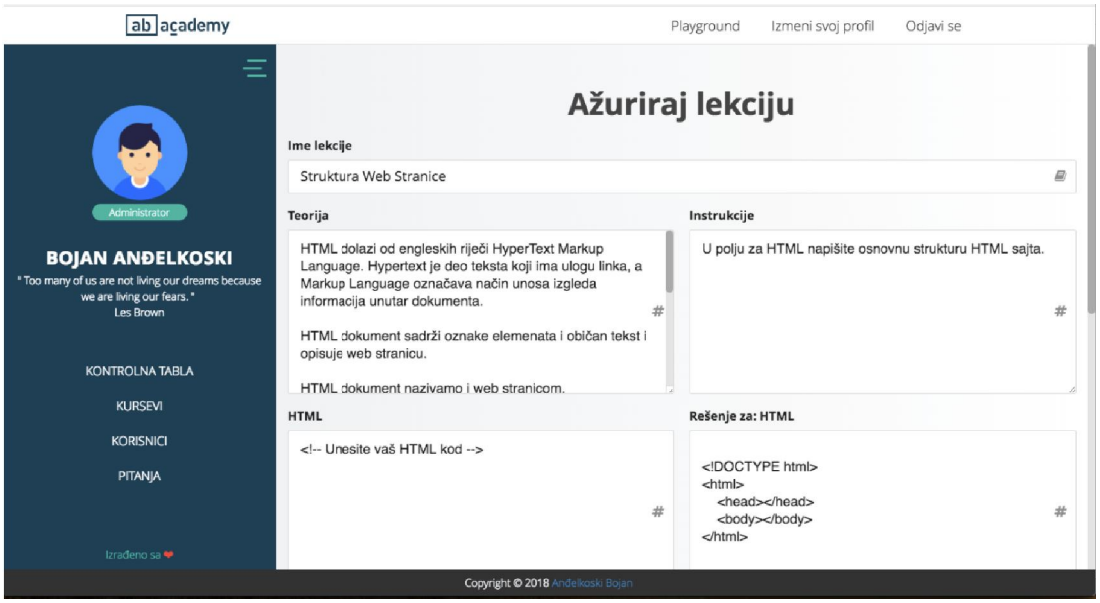


Slika 6.1.3.1 - Izgled stranice kursa (administrator)

Forma za ažuriranje/kreiranje nove lekcije je malo komplikovanija u odnosu na onu za kurseve. Forma se sastoji od sledećih polja:

- *Ime lekcije*,
- *Teorija* - teorija za datu lekciju.
- *Instrukcije* - instrukcije za praktični primer koji korisnik treba da reši za datu lekciju.
- *HTML* - U ovom polju definiše se koji se HTML prikazuje i učitava kada korisnik isprati svoje rešenje. Veoma je zgodno koristiti ovo polje kada se, na primer, želi testirati korišćenje klasa i identifikatora u CSS-u. Ako je za odgovarajući kurs isključen HTML, onda korisnik ne može da menja HTML polje kada pokušava da reši praktični primer odgovarajuće lekcije, te je zgodno da se definiše neki HTML gde će korisnik testirati svoja znanja iz CSS-a i videti svoje rešenje kako bi izgledalo u pretraživaču. Da bi ovo bilo jasnije, dat je jedan primer. Recimo da je kurs vezan samo za CSS, i cilj je naučiti korisnika da promeni boju nekog naslova unutar H1 taga. Ako korisnik uspešno reši zadatak, on će moći da vidi rezultat svog rešenja jedino ako je u ovom HTML polju bio definisan taj H1 tag.
- Rešenje za HTML/CSS/Javascript - rešenja koja se očekuju da korisnik napiše za odgovarajući jezik (HTML/CSS/Javascript).
- CSS/Javascript - Kao i kod HTML-a, ova polja su veoma zgodna ukoliko se želi definisati neki CSS/Javascript koji će biti unapred postavljen u polju za CSS/Javascript kada korisnik otvori odgovarajuću lekciju.

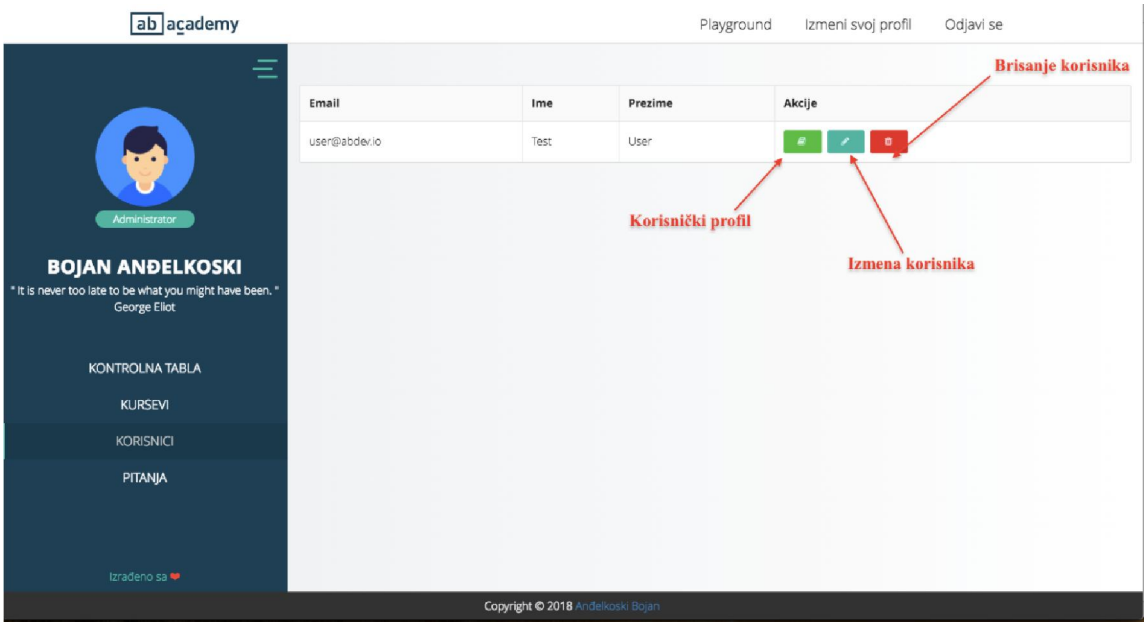
Važna napomena za lekcije: **Integrisani pretraživač na stranice Playground uvek kombinuje HTML, CSS, Javascript bez obzira da li su oni omogućeni ili nisu za odgovarajući kurs te lekcije.**



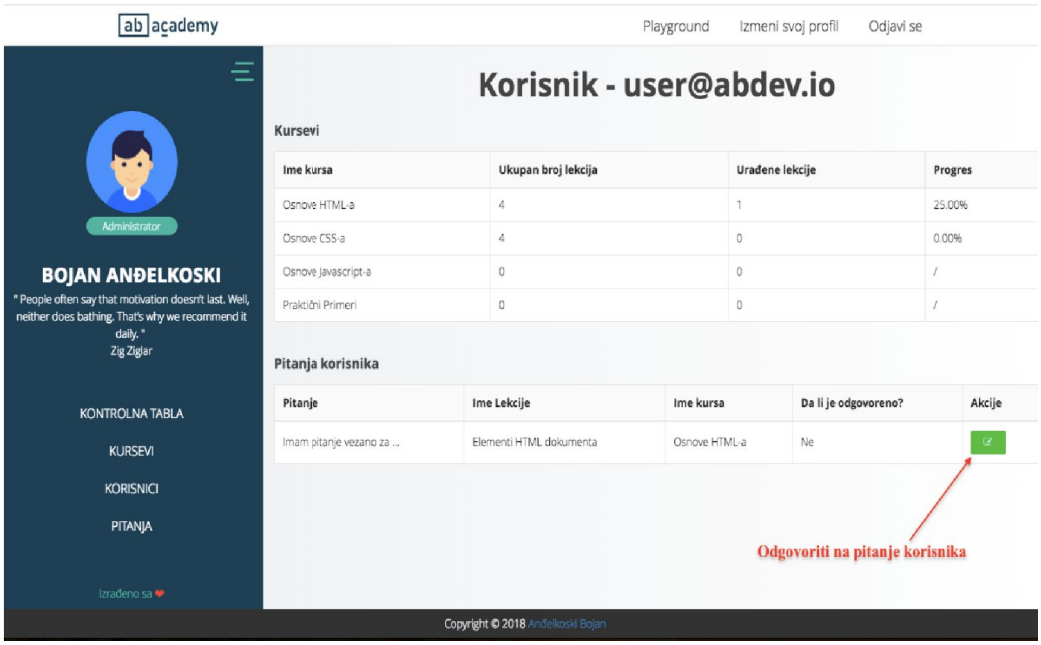
Slika 6.1.3.2 - Izgled forme za kreiranje/ažuriranje lekcije

6.1.4. Korisnici / Ažuriranje profila

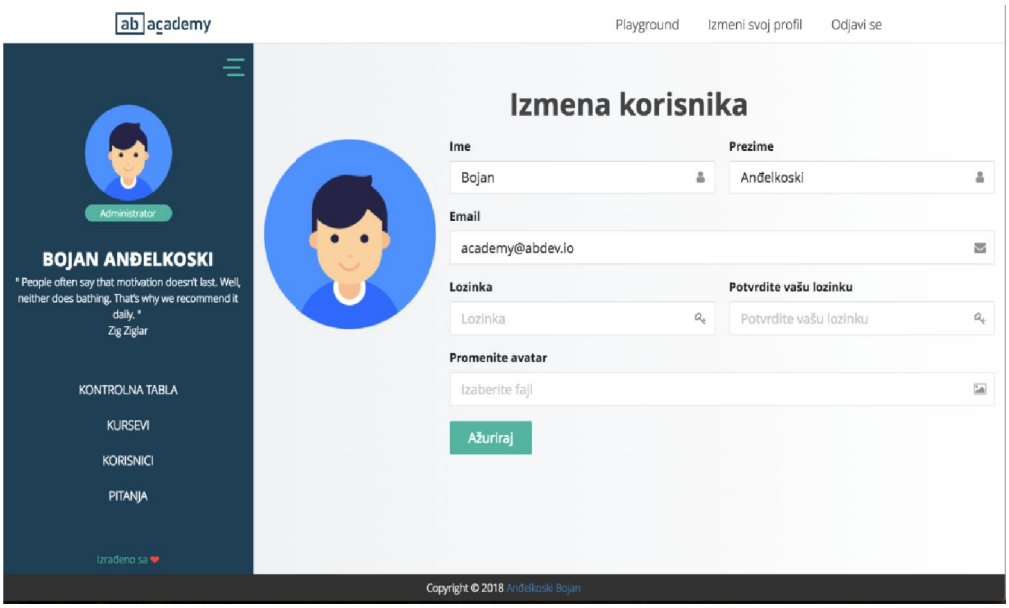
Na stranici „korisnici“ mogu se videti svi korisnici i njihovi podaci. Svakog korisnika administrator sajta može da ažurira, obriše i otvori korisnički profil u kome su prikazana poslednja /poslednje data rešenja tog korisnika, koliko tačnih rešenja je taj korisnik poslao za svaki kurs ponaosob. Takođe su na toj stranici prikazana sva pitanja koja korisnik ima, pa administrator može lako da pruži odgovor na sva pitanja koja je korisnik postavio tj. poslao administratoru.



Slika 6.1.4.1 - Izgled stranice korisnici



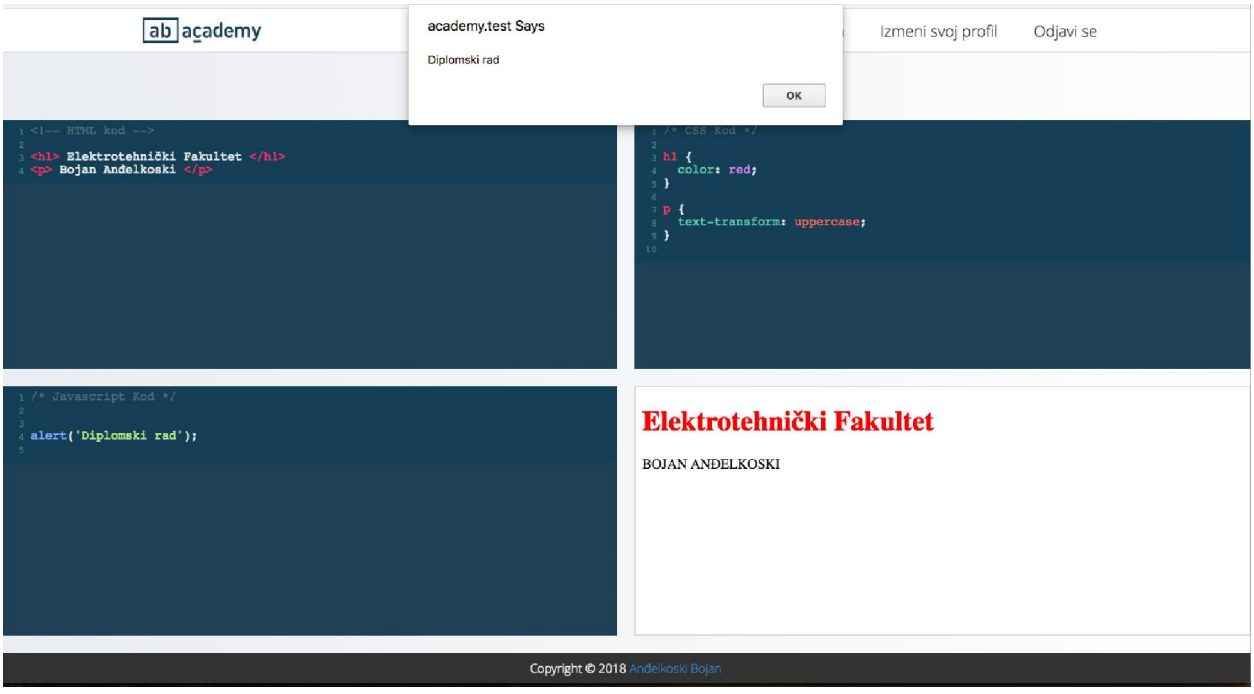
Slika 6.1.4.2 - Izgled stranice jednog korisnika



Slika 6.1.4.3 - Izgled forme za ažuriranje korisnika

6.1.5. Playground

Administrator ima mogućnost da koristi *Playground*. Pomoću *Playground*-a, korisnici mogu direktno da pogledaju kako će njihov HTML/CSS/Javascript kod izgledati u nekom pretraživaću.



Slika 6.1.5.1 - Izgled Playground-a

6.2. Korisnici

Korisnici mogu da prate kurseve, rešavaju praktične primere, postavljaju pitanja, ažuriraju svoje podatke i slično.

U gornjem delu aplikacije (*header*) korisnik ima sledeće linkove:

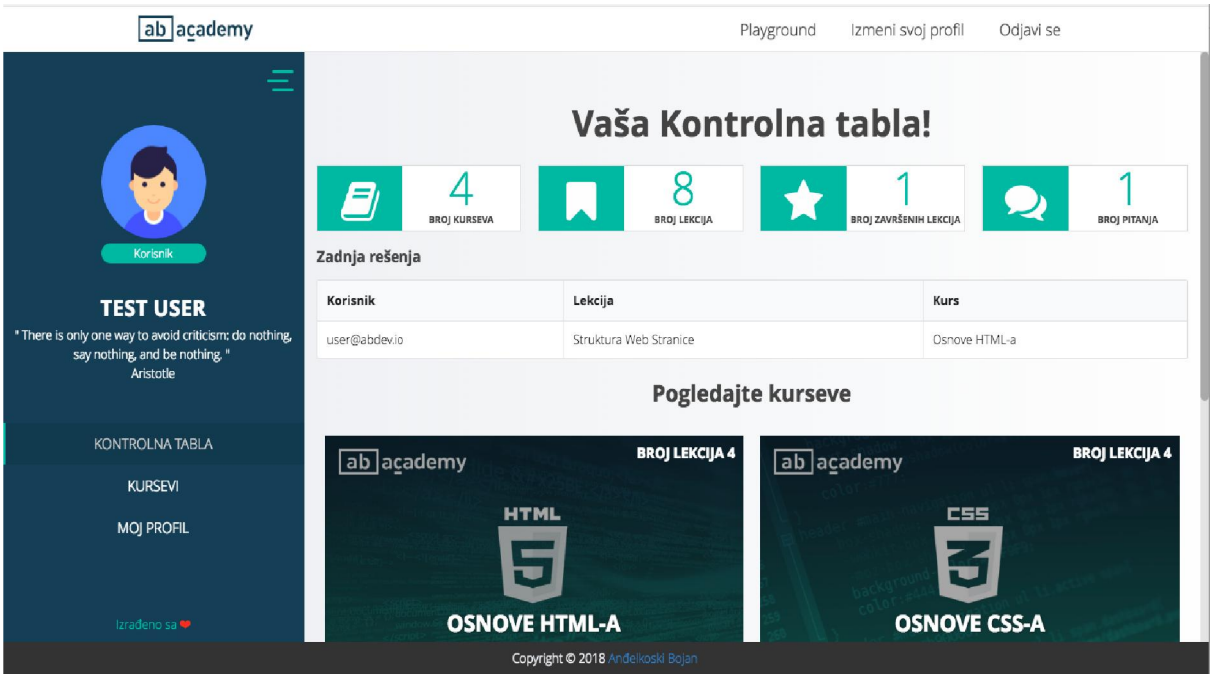
1) *Playground* - stranica na kojoj (u okviru koje) korisnici mogu da testiraju svoj HTML, CSS i Javascript kod u realnom vremenu,
2) *Izmeni svoj profil* - stranica na kojoj korisnici mogu menjati svoje informacije,
3) *Odjavi se* - link koji omogućava korisniku da završi korišćenje aplikacije i odjavi se.

U levom delu aplikacije (*sidebar*) korisnik ima sledeće linkove:

1) *Kontrolna Tabla,*
2) *Kursevi* - stranica preko koje korisnik može da pristupi kursevima,
3) *Moj Profil* - stranica preko koje korisnici mogu videti napredak za svaki kurs ponaosob kao i informaciju da li je administrator odgovorio na njihova pitanja.

6.2.1. Kontrolna Tabla

Kada se korisnik uloguje otvori mu se korisnička kontrolna tabla na kojoj se nalaze osnovne statistike, poslednja rešenja (poslednje uneta rešenja) koja je korisnik poslao, i izdvojeni kursevi koje korisnik može da prati.



Slika 6.2.1.1 - Izgled korisničke kontrolne table

6.2.2. Kursevi

Na ovoj stranici korisnik može da vidi sve kurseve koje je administrator postavio. Klikom na kurs otvara se spisak lekcija koje taj kurs ima.

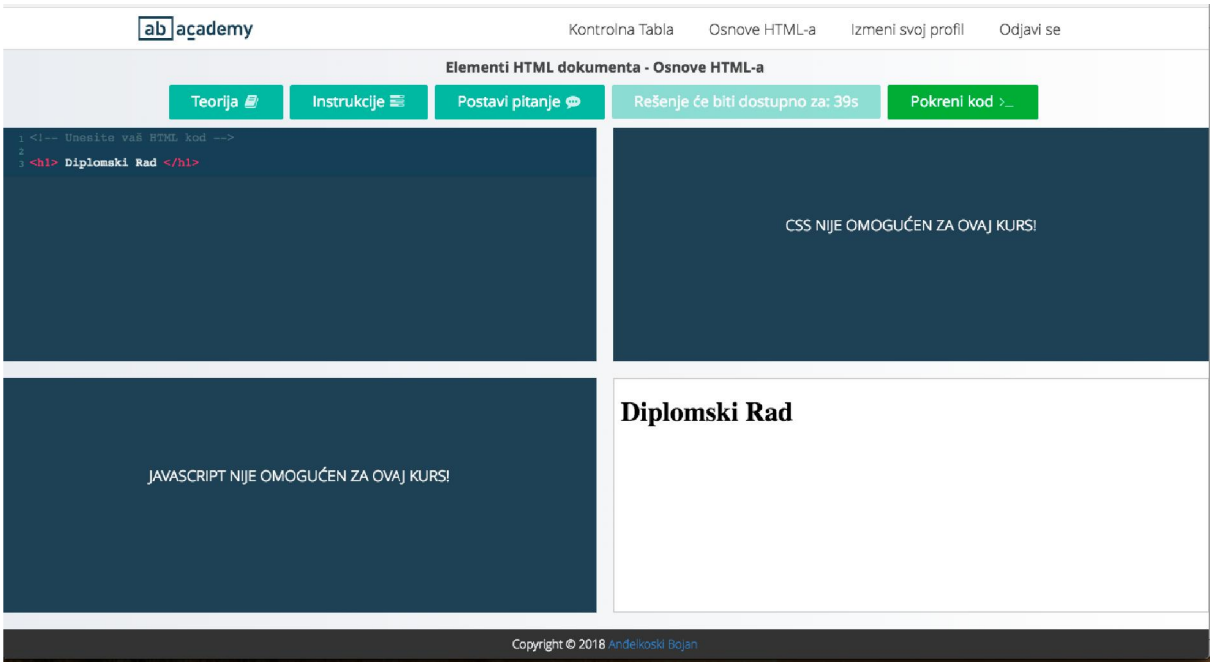
6.2.3. Lekcije

Na ovoj stranici korisnik može da vidi sve lekcije za kurs koji je otvorio. Ukoliko je uspešno završio lekciju, onda se žuta zvezdica pojavljuje unutar odgovarajuće lekcije. Klikom na lekciju se otvara *Playground* za tu lekciju.

6.2.4. Playground za lekcije

Kada korisnik otvori neku lekciju, pojavi se Playground, ali sa dodatnim opcijama. Za odgovarajuću lekciju korisnik može da:

1) Pročita teoriju - Klikom na dugme *Teorija* otvara se dijalog u kome (u okviru koga) je ispisana sva teorija za odgovarajuću lekciju,
2) Pročita instrukcije za zadatak koji ta lekcija ima - Klikom na dugme *Instrukcije* otvara se dijalog u kome su ispisane instrukcije za primer koji korisnik treba da reši,
3) Postavi pitanje za lekciju - Klikom na dugme *Postavi pitanje* otvara se dijalog u okviru koga se prikazuje forma u kojoj korisnik može da postavi svoje pitanje vezano za odgovarajuću lekciju,
4) Pogleda rešenje za lekciju - Klikom na dugme *Rešenje,* koje je dostupno 60 sekundi nakon otvaranja lekcije, korisnik može da vidi rešenje lekcije,
5) Proveri rešenje i pokrene kod u integrisanom pretraživaču - Klikom na dugme *Pokreni kod* korisnik proverava rešenje za odgovarajuću lekciju. Ako je uspešno rešio zadatak, korisnik može da se vrati nazad na kurs ili da ide na sledeću lekciju.



Slika 6.2.1.1 - Izgled Playground-a za lekciju

7. ANALIZA AUTENTIFIKACIJE

Autentifikacija korisnika je jedan od najvažnijih problema u svim modernim veb aplikacijama. Podaci korisnika moraju biti zaštićeni i bezbedni kao i sam proces autentifikacije i registracije korisnika.

Proces autentifikacije u realizovanoj aplikaciji je razvijen na sledeći način:

1) Korisnik unosi svoje podatke (e-mail/elektronsku poštu i lozinku),
2) Server proverava ispravnost unetih podataka,
3) Ukoliko su podaci ispravni, server vraća JWT (*JSON Web Token*), koji se čuva u lokalnoj memoriji (*localstorage*) pretraživača,
4) Korisnik je ulogovan.

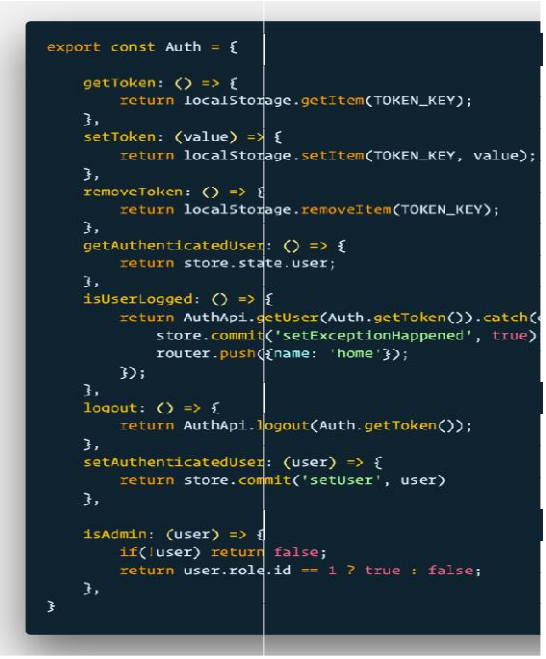
Pre nego što posetilac pristupi nekoj putanji, vrše se provere da li posetilac ispunjava uslove da poseti tu putanju. Ovo je korisno ukoliko se želi izvršiti restrikcija nekih stranica za određene korisnike. Pa tako, običan korisnik neće moći da vidi forme za izmenu kurseva, a posetilac koji se nije registrovao i ulogovao neće moći da vidi kurseve i da rešava lekcije i slično.

Posetilac/Korisnik može da pristupi putanji ako i samo ako je ispunjen bar jedan od sledećih uslova:

1) Putanja zahteva autentifikaciju i u lokalnoj memoriji postoje informacije o korisniku,
2) Putanja zahteva autentifikaciju i ukoliko ne postoje informacije o korisniku u lokalnoj memoriji, proverava se da li postoji token koji je sačuvan u lokalnoj memoriji i da li je taj token identičan tokenu koji je server zadao tom korisniku,
3) Pokušava da poseti stranicu za koju nije potrebna autentifikacija (putanja ne zahteva autentifikaciju).

Posetilac ne može da pristupi putanji i biće redirektovan ka odgovarajućoj putanji ukoliko se desi bar jedan od sledećih slučajeva:

1) Putanja zahteva autentifikaciju i ne postoje podaci o korisniku i ne postoji token u lokalnoj memoriji pretraživača.
2) Putanja ne zahteva autentifikaciju, a postoje podaci o korisniku u lokalnoj memoriji.
3) Putanja je namenjena određenom tipu korisnika koja nije ista kao tip korisnika koji pokušava da pristupi toj putanji.



```
export const Auth = {

    getToken: () => {
        return localStorage.getItem(TOKEN_KEY);
    },
    setToken: (value) => {
        return localStorage.setItem(TOKEN_KEY, value);
    },
    removeToken: () => {
        return localStorage.removeItem(TOKEN_KEY);
    },
    getAuthenticatedUser: () => {
        return store.state.user;
    },
    isUserLogged: () => {
        return AuthApi.getUser(Auth.getToken()).catch(
            store.commit('setExceptionHappened', true)
            router.push({name: 'home'});
        });
    },
    logout: () => {
        return AuthApi.logout(Auth.getToken());
    },
    setAuthenticatedUser: (user) => {
        return store.commit('setUser', user)
    },

    isAdmin: (user) => {
        if(!user) return false;
        return user.role.id == 1 ? true : false;
    },
}
```

Slika 7.1 - Javascript objekat koji izvršava razne operacije vezane za autentifikaciju

```
router.beforeEach((to, from, next) => {


/** Turning on the Loading Animation, and doing some ad
    store.dispatch('setIsLoading', true);
    store.commit('setIsPlayground', false);
    store.commit('setCourse', null);

    let requiresAuth = to.matched.some(record => record

    /** Authenticated Users only - User is Authenticate
    if (requiresAuth && Auth.getAuthenticatedUser()) re

    /** Authenticated Users only - User is not authenti
    if (requiresAuth && Auth.getToken()) {


/** If token exists, check wether that token is valid,
         * and redirect him to the dashboard, if not th
         **/
        return Auth.isUserLogged().then(res => {
            Auth.setAuthenticatedUser(res.data.user);
            return next({name: to.name});
        }).catch(err => {
            /* The token is invalid, return the user to
            store.commit('setExceptionHappened', true);
            return next({name: 'home'});
        });
    }

    /** The route is protected, the user is not logged
    if(requiresAuth && (!Auth.getAuthenticatedUser() ||
home'});

    /** Return to the intended route, and handle some e
    return next();
});
```

Slika 7.2 - Provera uslova pre nego što korisnik pristupi željenoj putanji

8. ZAKLJUČAK

U ovom dipolomskom radu je realizovana veb aplikacija koja ima za cilj da korisnike nauči Internet programiranju putem interaktivnih kurseva. Detaljno su opisani razvojni alati koji su korišćeni za izradu aplikacije. Objašnjeno je zašto su baš ovi alati pogodni za korišćenje prilikom razvoja veb aplikacija, zatim su navedene njihove prednosti, kao i način na koji se mogu koristiti. Prikazom mogućnosti ovih alata, što je bio i jedan od ciljeva ove teze, ukratko je objašnjeno u kom se pravcu kreće Internet programiranje i na šta treba obratiti posebnu pažnju ukoliko želite da se time bavite.

U radu su detaljno prikazane različite mogućnosti upotrebe aplikacije. Dato je uputstvo kako se aplikacija koristi, i to za oba tipa korisnika. Vodilo se računa da aplikacija bude jednostavna, pregledna, ali istovremeno i laka za korišćenje. Time je postavljena osnova za dalje unapređenje aplikacije.

Datom analizom autentifikacije je objašnjen jedan koncept, odnosno, način kako da se u aplikacijama reši autentifikacija korisnika. Jedan od najbitnijih delova veb aplikacije uvek treba biti bezbednost podataka korisnika, kao i prevencija pristupa putanjama koje nisu namenjene za određeni tip korisnika.

Detaljno je objašnjena struktura aplikacije i sve stranice koje postoje. Prikazan je način definisanja strukture baze podataka pomoću razvojnog alata Laravel, i koncizno je definisana struktura baze podataka.

Aplikacija je razvijena u skladu sa najnovijim standardima i principima. Gde god je to bilo potrebno, objašnjen je kod sa ciljem da se omogući ostalim programerima, koji žele da učestvuju u razvoju aplikacije, laka adaptacija principima rada i kodu aplikacije. Svakako, prostor za dalji razvoj aplikacije uvek postoji. Implementirana je osnova da se aplikacija može koristiti na više jezika, a može se završiti time što bi se dodao način promene jezika na *Frontend*-u.

Još nekoliko ideja koje bi mogle biti implementirane su:

- Dodavanje mogućnost za PHP i MySQL kurseve. Problem bi bila realizacija Playground-a, međutim, postoje implementacije za pokretanje PHP koda bez potrebe da se on prethodno kompajlira.
- Dodavanje mogućnosti da administrator može da šalje obaveštenja korisnicima,
- Dodavanje mogućnosti da korisnici mogu pisati beleške u sklopu neke lekcije,
- Dodavanje mogućnosti da administrator može da menja slike, boje i tekst na sajtu,
- Dodavanje mogućnosti da administrator može da kreira događaje za korisnike,
- Dodavanje mogućnosti da korisnik može da prati svoj kalendar za predstojeće događaje.

LITERATURA

[1] Alberto, L. & Indra, W. (2000) -*Communications and Networking: Fundamental Concepts and Key Architecture.*

[2] John R. Levine, Margaret Levine Young (2015) - *The Internet For Dummies 14th Edition.*

[3] Hege Refsnes, Ståle Refsnes (2010) - *Learn HTML and CSS with w3schools*

[4] *PYPL PopularitY of Programming Language[Online]: http://pypl.github.io/PYPL.html*

[5] *Forbes - Highest paying tech jobs [Online]: https://goo.gl/GHZC9X*

[6] *TIOBE - Software Quality Company [Online]: https://www.tiobe.com/tiobe-index/*

[7] *MVC architecture [Online]: https://goo.gl/6gMi1Z*

[8] *Laravel Framework Documentation [Online]:* https://laravel.com/docs/5.6

[9] *Vue.js Framework Documentation*[*Online*]: https://vuejs.org/v2/guide/