Izrada web aplikacija uz pomoć PHP okvira Laravel

Silvio, Đurić

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Silvio Đurić

IZRADA WEB APLIKACIJA UZ POMOĆ PHP OKVIRA LARAVEL

ZAVRŠNI RAD

Varaždin, 2019.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

VARAŽDIN

Silvio Đurić

Matični broj: 43452/14-R

Studij: Informacijski sustavi

IZRADA WEB APLIKACIJA UZ POMOĆ PHP OKVIRA LARAVEL

ZAVRŠNI RAD

Mentor:

Prof.dr.sc. Dragutin Kermek

Varaždin, rujan 2019.

Silvio Đurić

Izjava o izvornosti

Izjavljujem da je moj završni/diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

Struktura ovoga rada će kao uvertiru u PHP te PHP okvir Laravel, predstaviti i opisati najpoznatije programske jezike na koje se najčešće može naići kada se tema programiranja pokrene. No, prije nego sami spomen programskih jezika postane glavni fokus, dati će se nekoliko riječi o samome pojmu web aplikacija, njihovoj arhitekturi, razini izvođenja te sigurnosti. A onda se kao nastavak nadovezuju kratki opisi programskih jezika HTML, CSS, JavaScript, Java, Python, C++, C# i u konačnici PHP. PHP je iz razloga stavljen na zadnje mjesto, jer se tema rada dalje nastavlja na Laravel. Sami Laravel je zasebna cjelina za sebe, unutar te cjeline biti će opisana povijest nastanka i održavanja Laravela, arhitektura Laravela te analiza odluke koju programer donosi kada mu se kao opcija nudi Laravel. Nakon cjeline koja će predstaviti Laravel u više detalja će biti opisan proces baze podataka i komunikacije s bazom podataka kroz Laravel. A na posljetku rada prije zaključka i literature predstaviti će se praktični dio tj. izrada aplikacije korištenjem PHP okvira Laravel kroz koji će detaljno biti opisani zahtjevi aplikacije, implementacija i analiza konačnog rješenja.

Ključne riječi: PHP, Laravel, programski jezik, web aplikacija, programiranj

Sadržaj

Slike i tablice

1. Uvod

Predmet ovog završnog rada je konstruktivna razrada teme izrade web aplikacija uz pomoć PHP okvira Laravel, a u tu svrhu ovaj rad će definirati nekoliko ciljeva koji bi se trebali reflektirati na samom sadržaju odnosno razradi teme ovoga rada. Neki od tih ciljeva su:

- Upoznavanje s pojmom web aplikacija
- Predstavljanje najpoznatijih programskih jezika
- Detaljna analiza i opis PHP-a te PHP okvira
- Prikaz Laravela, njegove arhitekture te temeljnih vrijednosti
- Opis odnosa Laravela i baze podataka
- Analiza te opis programskog rješenja

Glavni cilj rada će biti obuhvatiti sve ove ciljeve te ih zaokružiti u jednu smislenu cjelinu koja je svoje naslove i podnaslove ukomponirala kako bi ostavila trag i dojam nakon čitanja.

Temeljne vrijednosti ovoga rada mogu se očitati iz transparentnosti te validnosti izvora koji su korišteni za pisanje rada. Svi izvori će biti detaljno navedeni te pobliže opisani kako bi se njihova vjerodostojnost mogla potvrditi u bilo kojem trenutku. No, kako bi sami izvori mogli dati svu svoju vrijednost koristiti će se istraživačka tehnika promatranja i analize sadržaja koja daje dovoljno vremena da se sve informacije iz izvora procesiraju te na kraju prikažu u radu.

Naposljetku ovoga uvoda bitno je istaknuti da je struktura rada planski napisana kako bi krenula od najjednostavnijih pojmova u prvome naslovu na koji se nadovezuju ostali – složeniji pojmovi. Bez ovakve strukture rad sam po sebi ne bi bio kompletan a ni smislen. Osim strukture glavnog sadržaja rada zaključak će imati također bitnu ulogu u zaokruživanju ove teme jer će se u zaključku rada napraviti kratki presjek i analiza svih postavljenih ciljeva iz ovoga uvoda kako bi se dobio bolji dojam koliko je rad zapravo opisao predmet i objekt promatranja.

2. Metode i tehnike rada

Za izradu pisanog dijela rada se koristila istraživačka tehnika: analize sadržaja koji dolazi iz izvora korištenih za pisanje ovoga rada koja je imala veliki doprinos u kvaliteti pisanja te razrade glavne teme rada koja se fokusira na PHP i njegov okvir Laravel. Osim toga ova tehnika je omogućila bolje upoznavanje s predmetom rada tako da se može bolje objasniti i razumjeti. Druga tehnika koja se koristila za izradu pisanog dijela rada je deskriptivna metoda koja je zapravo metoda opisivanja i koja promatrani i analizirani dio rada detaljno opisuje kako bi stvorila pisani sadržaj rada.

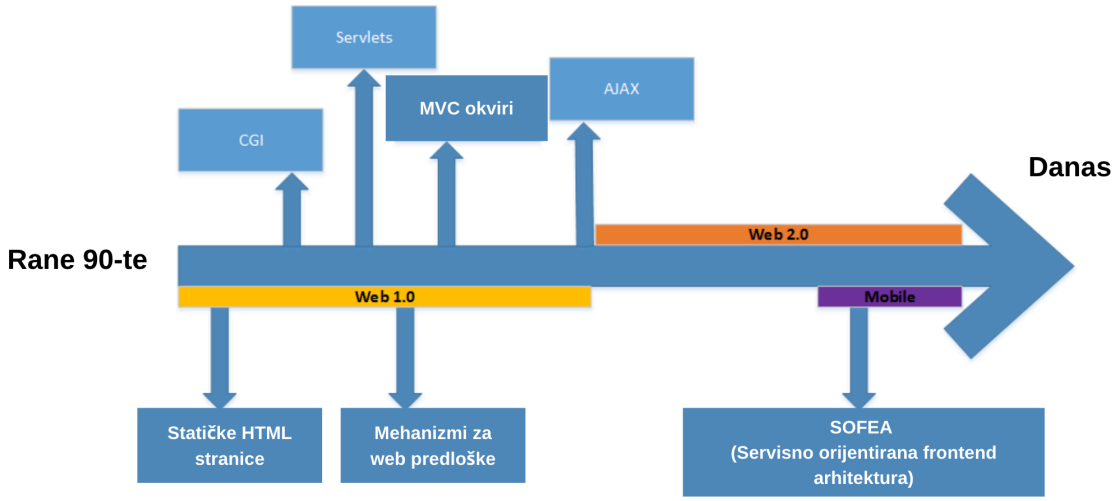
Za izradu pisanog dijela rada korišten je program Word, internetski pretraživači Chrome i Safari, alat za prevođenje teksta (Google prevoditelj), PDF preglednik Adobe, lightshot (alat za kreiranje slike desktopa). Osim internetskih izvora korištene su i knjige koje za svoju temu imaju neke od programskih jezika spomenutih u radu ili su usko povezane sa samom temom rada koja se odnosi na Laravel.

Praktični dio rada izrađen je korištenjem alata za pisanje programskog koda Visual Studio Code, sqlite baze podataka, Visual Paradigm alata te iTerm-a (terminal).

3. Web aplikacije

3.1 Povijest web aplikacija

Web aplikacije se mogu označiti kao bitan faktor za razvoj samog interneta jer poboljšavaju kvalitetu rada pretraživača te funkcionalnosti cjelokupnog interneta. Sama povijest web aplikacija započinje početkom 90-tih tj. erom statičkih HTML web stranica gdje su web stranice prvotno bile tekst dokumenti u koje se tek kasnije dodala mogućnost grafičkog stiliziranja, slika, audio te video materijala (Uryutin, 2018). No, stranice su i dalje bile statične sve do 1995. godine kada je Brendan Eich kreirao JavaScript. Samu inspiraciju za kreiranjem JavaScript-a Eich je vukao iz Java-e, Scheme i Self-a. Ime JavaScript je nastalo radi bolje prihvatljivosti samog jezika na tržištu iako JavaScript nema nikakve veze s Java-om (Aston, 2015).



Slika 1. Grafički prikaz povijesti razvoja web-a te web aplikacija prema Tomcy J. 2015. (Izvor: Java Codebook, https://www.javacodebook.com/2015/03/31/history-of-web-application/, posjećeno 05. srpnja 2019.)

1995. godine Java Applet je kreiran u smislu male aplikacije napisane u Java programskom jeziku a korišten je za kompilaciju Java bajt koda, dok 1996. godina označava još jednu važnu godinu za web aplikacije iz razloga što je tvrtka Macromedia na tržište uvela Macromedia Flash koji je dao veliki doprinos web aplikacijama obogativši ih animacijama, što je ujedno omogućilo i rast interaktivnih video igara (Hoffmann, 2017).

Tri godine kasnije, točnije 1999. godine koncept web aplikacije se pojavljuje i u Java programskom jeziku. Sljedeći period koji je napravio veliku prekretnicu je označen 2005. godinom koja se također pamti kao i godina kraja Web-a 1.0 te godina početka Web-a 2.0. Te godine započela je transformacija iz statičkih web stranica u dinamičke web stranice, predstavljanjem Ajax-a koji omogućuje korištenje prilagodljivo (eng. responsive) dizajniranih web stranica te omogućuje korisniku da radi na webu puno brže i bolje (Uryutin, 2018).

Sljedeće godine su donijele razne promjene a jedna od najznačajnijih dogodila se 2014. godine kada kreiran je HTML5 te je unaprijeđen postojeći HTML standard. HTML5 podržava nove tipove multimedije, omogućava kreiranje web aplikacija koje su samostalne te ne zavise o pretraživaču ili određenoj platformi.

3.2 Osnovni principi rada web aplikacija

Kada se govori o osnovnim principima rada web aplikacija onda je bitno spomenuti nekoliko ključnih faktora koji će detaljnije biti objašnjeni u sljedećim podnaslovima. Jedan od tih faktora je arhitektura web aplikacije odnosno kvaliteta arhitekture web aplikacija, a pod time se misli na različite čimbenike kao što su produktivnost programera, iskoristivost, sigurnost, skalabilnost, transparentnost itd. U neke od osnovnih principa također je bitno izdvojiti razinu izvođenja web aplikacija te na koje se sve načine razina izvođenja može poboljšati u današnjem vremenu. Osim razine izvođenja posebno će se istaknuti sigurnost kao jedan od faktora koji danas igraju jako bitnu ulogu kada se radi o povjerenju u web aplikacije.

3.2.1 Arhitektura web aplikacija

Svaka web aplikacija ima dvije strane koje izvršavaju svoje programske kodove te funkcioniraju odnosno izvršavaju se jedna uz drugu, te strane se nazivaju klijentska i serverska strana web aplikacije. Klijentska strana je strana koja se izvršava u pretraživaču i odgovara na neke korisničke akcije. Druga strana odnosno serverska strana je strana koja se nalazi na serveru i odgovara na HTTP zahtjeve.

Programer ili tim programera koji razvijaju web aplikaciju odlučuju koji će se kod izvršavati na server strani, pa je tako moguće napraviti odluku u smjeru C#, Java-e, JavaScript-a, Python-a, PHP-a, Ruby-a itd. Bilo koji kod koji ima mogućnost odgovoriti na HTTP zahtjev, ima i mogućnost da se izvršava na server strani web aplikacije. Sama server strana je odgovorna za kreiranje stranice koju je korisnik zahtijevao kao i za spremanje različitih tipova podataka uključujući korisničke profile te unose korisnika. Serverska strana koda nikada nije vidljiva korisniku web aplikacije.

S druge strane kombinacija CSS-a, HTML-a i JavaScript-a koristi se za pisanje koda koji će se nalaziti na klijentskoj strani. Ovaj kod se izvršava unutar pretraživača i za razliku od koda sa server strane vidljiv je korisniku web aplikacije te ga taj krajnji korisnik može modificirati uz pomoć programerskih alata (eng. dev tools) koje nudi većina današnjih preglednika. Klijentska strana koda komunicira samo putem HTTP zahtjeva te nema mogućnost čitati datoteke sa servera direktno (Hackr.io, 2019).

Slika 2. Grafički prikaz arhitekture web aplikacija prema Hackr.io 2019. (Izvor: Hackr.io, https://hackr.io/blog/web-application-architecture-definition-models-types-and-more, posjećeno 07. srpnja 2019.)

Na slici 2 moguće je vidjeti jednostavni prikaz rada web aplikacije koja ima svoju klijentsku i serversku stranu. Klijentska strana je ono što korisnik web aplikacije vidi te je moguće vidjeti da klijentska strana prikazuje HTML koji mu serverska strana šalje a ujedno i radi Ajax upite (eng. request) na web server koji će mu odgovoriti (eng. response) s JSON podacima koji se iskorištavaju na klijentskoj strani za prikaz podataka ili manipulaciju tih podataka kako bi se izvršile određene akcije.

3.2.2 Razina izvođenja web aplikacija

Razina izvođenja web aplikacija se najčešće poistovjećuje sa brzinom web aplikacija, te dok je to dijelom točno, brzina web stranice nije nužno jedina „metrika“ kojom se može izmjeriti kvaliteta izvođenja web aplikacije. Prilikom izrade web aplikacija uvijek je nužno paziti na količinu ali i na „težinu“ koda. Na što se misli kad se kaže „težina“ koda? Težinom koda se može opisati programski kod koji zahtijeva duže vrijeme učitavanja ili više pozivanja koja možda nisu potrebna, jedan od najboljih primjera je JavaScript koji se često može dodatno unaprijediti pisanjem „čišćeg“ te jednostavnijeg koda npr. Vanilla JavaScript-a umjesto jQuery-a koji ima puno veću težinu nego Vanilla JavaScript.

Slika 3. Grafički prikaz ubrzanja stranice korištenjem drugog tipa slike. Izrada autora.

Osim težine koda razne druge stvari utječu na razinu izvođenja web aplikacija, pa tako je uvijek potrebno obratiti pažnju na veličinu slika koje se koriste unutar aplikacije ili sličnih resursa koji mogu uzimati dosta vremena za učitavanje same stranice.

DNS problemi te mrežna povezanost također imaju veliki utjecaj na izvođenje web aplikacija jer menadžment web prometa koji se kreće između servera i klijenta je potrebno nadzirati tj. raditi konstantni monitoring kako bi se moglo točno znati što uzrokuje probleme (Julien, 2018). Na kraju se još mogu spomenuti unapređenja koja mogu poboljšati izvođenje web aplikacija kao što su dodavanje specifičnih HTML elemenata u zaglavlje (eng. head) stranice (npr. eng. prefetch, preconnect, dns prefetch, preload). Ovi HTML elementi omogućuju web aplikaciji da ovisno o korisnikovim kretnjama po stranici unaprijed učitava resurse koji se nalaze na određenoj poveznici koju korisnik web aplikacije želi odabrati (Roberts, 2019).

Slika 4. Grafički prikaz najčešćih problema koji utječu na izvođenje web aplikacija/stranica prema Keycdn 2019. (Izvor: Keycdn, https://www.keycdn.com/support/pinpoint-website-performance-issues, posjećeno 09. srpnja 2019.)1

3.2.3 Sigurnost web aplikacija

Što je to sigurnost web aplikacija? Pa ovako, sigurnost web aplikacija se može opisati kao proces zaštite web stranica te online servisa protiv različitih sigurnosnih prijetnji koje otvaraju ranjivosti u programskom kodu aplikacije. Neki od razloga koji motivira treću stranu da „napada“ web aplikacije/stranice su: izvlačenje podataka koji imaju visoku vrijednost na tržištu, uništavanje sustava ili podataka u svrhu kreiranja štete itd.

Stuttard i Pinto su napravili testiranje između 2007. i 2011. godine testirajući preko stotinu web stranica te su dobili sljedeće rezultate:

- Pokvarena ovjera (eng. authentication) (62%) – ovaj problem napadačima omogućava da pogode „slabe“ lozinke, koriste silu pristupa (eng. brute force) ili jednostavno preskoče cijelu prijavu
- Pokvarena kontrola pristupa (71%) – kontrola pristupa je velika ranjivost te napadač u ovome slučaju može pristupiti podacima korisnika sa serverske strane

[1] Time to first byte (TTFB) je vrijednost u mikrosekundama izmjerena od trenutka slanja upita sve dok korisnik ne primi prvi bajt podataka koje šalje poslužitelj.
CDN u doslovnom prijevodu znači – distribucija podataka, obično statičnog sadržaja kao što su fotografije, CSS, JavaScript programa i drugih strukturalnih komponenti koje za svoj rad zahtijevaju puno vremena i samim time usporavaju rad web stranica.

- SQL ubrizgavanje (32%) – ovakva vrsta napada funkcionira tako da napadač napravi unos koji poremeti komunikaciju između web aplikacije i baze podataka što napadaču omogućava da dobavi podatke koje mu pretraživač vrati, a inače ne bi vratio (Stuttard, Pinto 2011).

Ovi napadi mogu biti jako štetni za bilo koju tvrtku ili osobu koja se svakodnevno koristi s web aplikacijama koje imaju osjetljive podatke, bilo osobne ili korisničke. Pa se tako u svrhu osiguravanja od ovih različitih napada kreiraju različiti sigurnosni standardi koji bi trebali osigurati osnovnu zaštitu od gubitka podataka kroz ranjivosti web aplikacije.

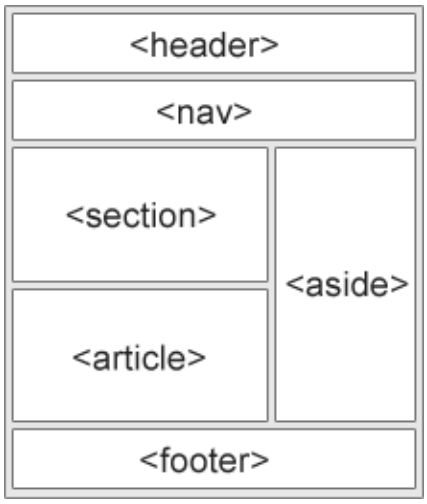
Jedan od standarda je PCI DSS standard koji se označava forsiranom sigurnosnom procedurom koja osigurava web aplikacije od prijetnji.

Web aplikacijski vatrozid (eng. firewall) su hardverska i softverska rješenja koja se koriste za kreiranje zaštite od prijetnji prema web aplikacijama. Ta rješenja su dizajnirana kako bi ispitala dolazeći promet te blokirala pokušaje napada što na kraju osigurava bilo koji oblik sanacije koji bi bio potreban ako bi se prijetnja provela u djelo.

4. Razvoj web aplikacija u programskim jezicima

4.1 HTML i CSS

U ovome dijelu rada opisati će se HTML te njemu stilska poveznica CSS odnosno SCSS. No, treba krenuti nekim redoslijedom pa zato prvo pitanje na koje će se odgovoriti će biti: što je to zapravo HTML (eng. Hypertext Markup Language)? Za HTML se može reći da je temeljna tehnologija koju jedan programer pristupnog dijela (eng. frontend) mora imati u svojoj bazi znanja. Tu je moguće napraviti distinkciju da HTML nije programski jezik nego jezik za označavanje (eng. markup), što znači da je on zapravo sustav za identificiranje i opisivanje različitih komponenti kao što su zaglavlja (eng. header), odjeljak (eng. paragraph), lista itd. HTML je jezik koji se koristi za kreiranje dokumenata web stranice, a do danas postoji nekoliko verzija HTML koje se koriste, no trenutno najznačajnija verzija HTML-a je HTML5. (Robbins, 2012). Riječ hipertekst (eng. hypertext) označava poveznica koja povezuje web stranice s jedne na drugu. Kao što je rečeno HTML koristi označavanje kako bi prikazao tekst, slike ili bilo koji drugi sadržaj namijenjen za prikaz na web stranici. HTML nudi različite elemente s kojima se sadržaj može prikazivati, a u današnje vrijeme jako je bitno na umu imati semantički HTML koji je bitna okosnica za bilo kojeg programera u procesu njegovog razvoja. Pa tako za sljedeće elemente možemo reći da su semantički: <article>, <aside>, <details>, <figcaption>, <figure>, <footer>, <header>, <main>, <mark>, <nav>, <section>, <summary>, <time> (Goodman, 2002). Ako se napravi pregled izvornog koda bilo koje web stranice moguće je naići na elemente koji sadrže atribute kao „id“ ili „class“ npr.   kako bi označili navigaciju, zaglavlje (eng. header) ili podnožje (eng. footer) što ni u kojem slučaju nije pravilno jer osim što se gubi semantika HTML-a, otežava se pristupačnost gluhim, slijepim te ostalim korisnicima s poteškoćama. Stoga je bitno koristiti elemente koji postoje i kreirani su za tu svrhu npr. <nav> za navigaciju, <header> za zaglavlje ili <footer> za podnožje.



Slika 5. Grafički prikaz semantičke strukture HTML stranice. Izrada autora.

U nastavku se može vidjeti prikaz osnovne strukture semantički napisanog HTML-a. Iznad zaglavlja (eng. header) još fali <html> element te element glave (eng. head). Samo zaglavlje, navigacija, sekcije, članci te podnožje (eng. footer) se nalaze unutar <body> elementa.

```
<!DOCTYPE html>
<html>
<head>
<title>Naslov</title>
</head>
<body>

<header></header>

<h1>Poglavlje stranice</h1>
<p>Odsječak stranice</p>

<footer></footer>

</body>
</html>
```

Na isječku koda koji se nalazi iznad moguće je vidjeti pravilnu semantičku strukturu HTML dokumenta.

Odmah uz rame HTML-a ide i CSS (eng. Cascading Style Sheet) te dok se HTML koristi za označavanje sadržaja web stranice, CSS opisuje kako bi se taj sadržaj trebao prikazivati na stranici. Sami CSS opisuje kako se elementi trebaju prikazivati na ekranu, papiru, glasovno ili kroz neki drugi medij pa se iz tog razloga često naziva prezentacijom. CSS je standardizirani oblik stiliziranja HTML koda koji se koristi na webu te je razvijen kroz nekoliko verzija a to su: CSS1 – koji više nije značajan te se ne koristi, CSS2.1 koji se preporučuje te CSS3 koji je podijeljen na manje module te napreduje prema statusu standardne verzije koja se koristi na webu.

```
<style>
body {
      background-color: lightblue;
}
h1 {
      color: white;
      text-align: center;
}
p {
      font-family: verdana;
      font-size: 20px;
}
</style>
```

Sami CSS se može pisati unutar HTML-a točnije unutar <style> elementa koji se nalazi u HTML-u. Na primjeru koji se nalazi na prethodnom isječku koda moguće je vidjeti CSS koji je napisan upravo na taj način. Osim te varijante CSS je moguće pisati u odvojenim datotekama koje se onda uključuju unutar HTML putem posebnih HTML oznaka npr.

- `<link rel="stylesheet" href="styles.css">`

No CSS ima i svoju verziju koja se naziva pred procesor (eng. preprocessor) odnosno SCSS. Ovakvo pisanje CSS-a omogućava programeru da koristi značajke koje još nisu dio šireg CSS standarda te omogućava bolji protok pisanja jer se piše u strukturalnom obliku kao HTML. Na webu je moguće lako doći do CSS-a te vidjeti što točno koja stranica stilizira na koji način. Sve što treba učiniti je istražiti (eng. inspect) web stranicu uz pomoć web alata koji postoje za gotovo se preglednike (Meyer, 2011).

```
.klasa {
      background-color: lightblue;
}
.klasa h1 {
      color: white;
      text-align: center;
}
.klasa p {
      font-family: verdana;
      font-size: 20px;
}
```

U isječku koda u nastavku rada moguće vidjeti da se SCSS piše slično kao i HTML (strukturalno) što omogućava pregledniji kod te lakše pisanje samog koda. Za pisanje SCSS-a najčešće se koristi SASS koji je jezik dizajniran na ideji SCSS-a od strane Hamptona Catlina.

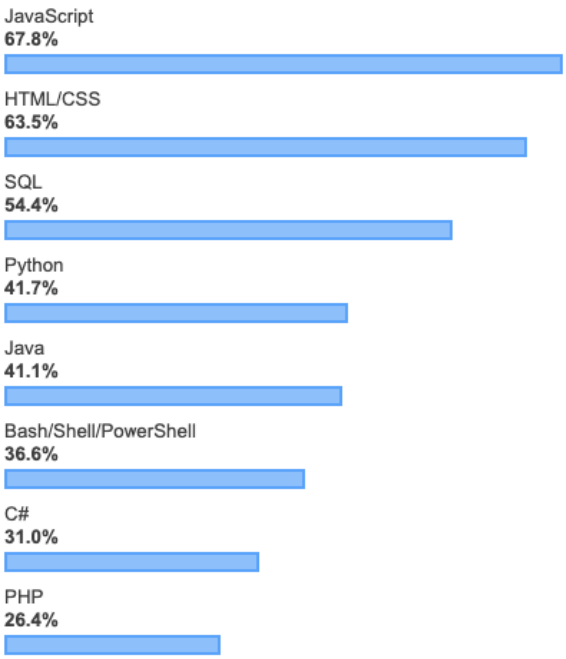
```
.klasa {
      background-color: lightblue;
      h1 {
            color: white;
      }
      p {
            font-family: verdana;
            font-size: 20px;
      }
}
```

4.2 JavaScript

JavaScript je interpretirani (eng. interpreted) programski jezik sa objektno orijentiranim (OO) mogućnostima. Sintaktički, jezgra (eng. core) JavaScript programskog jezika jako nalikuje C-u, C++-u i Java-i sa programskim konstruktorima kao „if“, „while“ petlja i „&&“ operator.

JavaScript je najčešće korišten u web preglednicima što je u tom kontekstu generalna svrha jezgre povećana s objektima koji omogućavaju skriptama kako bi vršile interakciju s korisnicima, kontroliraju web preglednike te izmjenjuju sadržaj dokumenta koji se pojavljuje unutar web preglednika (Flanagan, 2006).

U 2019. godini JavaScript se svrstava u top tri programska jezika kojima se udio potražnje na tržištu povećao (prva dva su Python i Java) (Saeed, 2019). Za JavaScript se može reći da je vitalan alat za svakog programera pristupnog dijela (eng. frontend) ali u isto vrijeme s velikom relevantnošću kod programera podrške (eng. backend). „Stackoverflow“ koji je najveća zajednica gdje programeri mogu učiti, razvijati te dijeliti svoja znanja s drugim programerima svrstava JavaScript na prvo mjesto ljestvice najpopularnijih jezika oko kojih se vode razne rasprave (StackOverflow, 2019).



Slika 6. Poredak programskih jezika prema popularnosti u najvećoj programerskoj zajednici „Stackoverflow“ prema StackOverflow, 2019. (Izvor, Stackoverflow, https://insights.stackoverflow.com/survey/2019, posjećeno 24. kolovoza 2019.)

JavaScript programski jezik je kreiran 1995. godine od strane Brendana Eicha koji je u to vrijeme radio u Netscape Communications tvrtki, a inspiriran je Java, Scheme te Self programskim jezicima. JavaScript se originalno zvao „Netscape Mocha“ jer je Eich jezik razvio te ga s tvrtkom Netscape tržišno pozicionirao odmah uz Java-u kao što su to napravili i Visual Basic te C++. Nakon toga JavaScript je imao mnogo uspona i padova u popularnosti osobito zato jer tehnologija koja je postojala u to vrijeme nije mogla stići sve mogućnosti JavaScript-a.

Za JavaScript je uglavnom moguće reći da ima bogatu i fascinantnu povijest što ga je dovelo do toga da je postao standardni programski jezik weba ali ne samo iz razloga što je bio jedan od jezika koji su tu bili odmah na početku nego zato što je otvoren, standardiziran ali u isto vrijeme ima jako dobru dinamiku i integraciju s DOM-om[2] (Aston, 2015).

Bitno je napomenuti da je službeno ime JavaScript-a zapravo ECMAscript prema ECMA-262 standardu ali se koristi samo kad se jezik referencira prema standardu. Nakon dugo vremena stabilnosti za JavaScript nekoliko znakova je slutilo na promjenu. Firefox 1.5 web preglednik uključuje novi JavaScript interpreter pod verzijom 1.6. Ta nova verzija uključivala je novu (ne standardiziranu) metodu manipulacije sa poljima. Nakon te verzije započelo je aktivnije objavljivanje verzija JavaScript programskog jezika. Trenutno aktivna verzija je ES6 – ECMSscript 6 (Flanagan, 2006).

4.2.1 Vanilla JS i ES6

Unutar ovoga podnaslova biti će objašnjena dva pojma koja su sve popularnija u zadnje vrijeme a to su Vanilla JavaScript te ES6. Što je to zapravo ES6? ES6 je samo skraćenica za ECMAscript što je već spomenuto u uvodu u ovaj naslov. S obzirom da je JavaScript zaostajao za nekim ostalim programskim jezicima kada se usporedi različite sintaktičke forme kao što je deklariranje konstantnih varijabli, deklariranje područnih (eng. scoped) varijabli, raspakiravanje podataka iz polja, kraća sintaksa za deklariranje funkcija itd.

[2] DOM – Document object model je definirani standard za pristup dokumentima te se unutar W3C-a opisuje kao platforma i jezično neutralno sučelje koje omogućuje programima i skriptama da dinamično pristupaju i ažuriraju sadržaj, strukturu i stil dokumenta.

ECMAscript 6 dodaje mnogo novih značajki koje će nadoknaditi sve prethodno nabrojane stvari koje su do sada nedostajale u JavaScript-u, što će u konačnici programerima omogućiti da pišu manje a urade više (Prusty, 2015).

Vanilla JavaScript označava JavaScript bez ikakvih dodatnih biblioteka kao što su npr. jQuery (Darveau, 2018). Kada se Vanilla JavaScript gleda kroz povijest onda je moguće reći da su programeri izbjegavali taj oblik za rješavanje mnogo problema jer je njihov kod morao biti korišten kroz sve web pretraživače (Chrome, FireFox, Safari, IE9, IE 6-7 itd.), stoga su se često okretali bibliotekama kao što je to jQuery. Ali razvojem ECMAscript 6 to više nije potrebno raditi jer Vanilla JavaScript može riješiti brojne probleme s kojima se programeri danas susreću (Darveau, 2018). U nastavku je moguće vidjeti primjer zašto se nije uvijek potrebno okretati rješenju koje koristi biblioteku.

```
//jQuery
$('.moj #novi .selektor');
//Vanilla JS
document.querySelectorAll('.moj #novi .selektor');
```

4.2.2 Vue.js

Vue.js je JavaScript programski okvir koji je kreirao Evan You, programer koji je radio u Google-u, a razlog zašto ga je kreirao je bio zato što je učestalo pišući JavaScript kod koji se ponavlja primijetio da to oduzima jako puno vremena i resursa. U potrazi za nekim alatom koji bi riješio taj problem nije bilo previše uspjeha jer nije pronašao nikakav programski okvir, biblioteku ili alat. U to vrijeme Angular se koristio već jako puno, React.js je tek bio u svojim počecima a programski okviri kao Backbone.js su korišteni za jako velike aplikacije. Upravo to je dalo ideju Evan-u da kreira svoj programski okvir i nazove ga Vue.js, nakon toga Vue.js je samo rastao jer je odmah dobio popularnost te je zajednica programera odgovorila velikim doprinosima u razvoju Vue.js-a (Filipova, 2016).

Vue.js se spominje iz razloga što će u praktičnom dijelu rada biti korišten kao jedan od JavaScript okvira za razvoj pristupnog (eng. frontend) dijela aplikacije. Vue je jedan od najrazvijenijih JavaScript programskih okvira, a osobito je postao popularan u posljednje dvije godine (2018, 2019) jer ga mnogi smatraju savršenim utjelovljenjem između React.js-a te Angulara. Vue se na svojoj službenoj stranici definira kao jedan od progresivnih programskih okvira te se smatra inkrementalno prilagodljivim okvirom (Vuejs, 2019).

Instalacija Vue-a je prilično jednostavna, a svodi se na uključivanje tog programskog okvira putem <script> elementa, gdje Vue nudi dvije verzije a to su razvojna i produkcijska verzija.

```
<!– razvojna verzija, uključuje korisna konzolna upozorenja –->
<script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js">
</script>

<!– produkcijska verzija, optimizirana za veličinu i brzinu –->
<script src="https://cdn.jsdelivr.net/npm/vue"></script>
```

Na sljedećem isječku koda moguće je vidjeti HTML te JavaScript pisan u Vue.js-u koji rezultiraju ispisom jednostavne pozdravne poruke u DOM. Unutar HTML koda moguće je vidjeti da se poruka označava prema logici predloška (eng. template) a sami ispis poruke postiže se na osnovi „id-a". Taj „id" se označava kao element unutar Vue.js koda te se putem podataka (eng. data) šalje odnosno dodjeljuje poruka koja se želi ispisati.

```
<div id="app">
<h1>{{ poruka }}</h1>
</div>

var mojObjekt = new Vue({
   el: '#app',
   data: {poruka: 'Hello Vue!'}
});
```

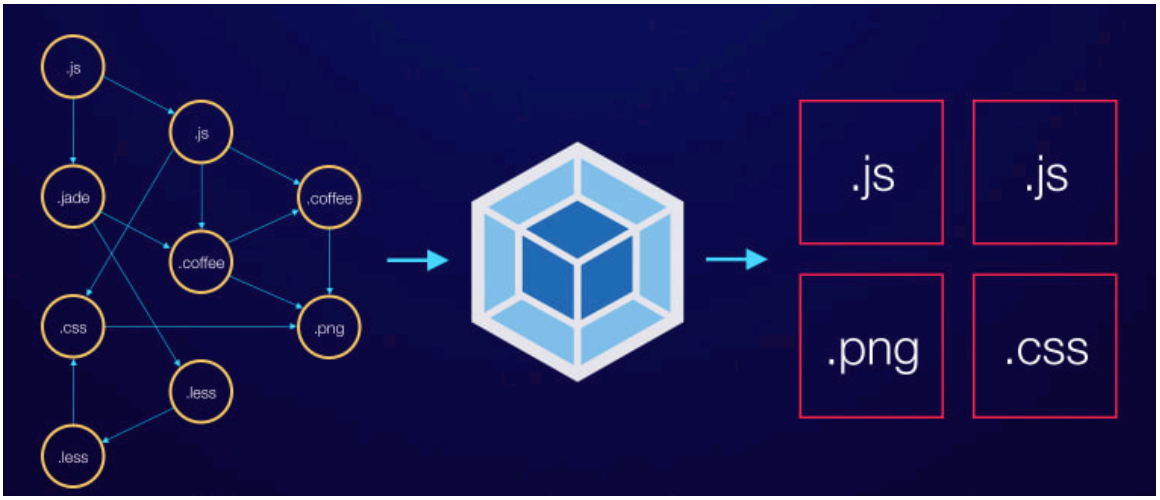
4.2.3 GRUNT, GULP i Webpack

Prije nego je JavaScript postao toliko popularan programski jezik programeri nisu koristili alate kao GRUNT, GULP ili Webpack. No, danas većina tvrtki bazira svoj razvoj odnosno automatizaciju na GRUNT-u, GULP-u ili Webpack-u. Tu se može postaviti pitanje: zašto koristiti alate za automatizaciju? A na to pitanje se može dati nekoliko odgovora:

- CSS automatsko dodavanje prefiksa (eng. autoprefixing) i pred procesiranje
- Smanjivanje veličine datoteka (eng. minification) i čišćenje koda
- Brisanje komandi iz konzole
- Pokretanje jediničnih testova

- Optimizacija slika (Kuprenko, 2018)

Ovo su samo neki od najistaknutijih razloga zašto je dobro koristiti ove alate iako postoji još mnogo drugih pozitivnih razloga koji se mogu istaknuti. Osobito je bitno istaknuti Webpack, čija je popularnost jako porasla posljednje dvije do tri godine (2017, 2018, 2019) jer je postao jako praktičan i brz za korištenje. Webpack će također biti korišten unutar ovoga projekta za izradu praktičnog dijela rada, pa će zato rad u nastavku prikazati jednostavnu ilustraciju kako Webpack zapravo funkcionira.



Slika 7. Prikaz jednostavnog objašnjenja funkcioniranja webpack-a, prema Scotch. (Izvor: Scotch, https://scotch.io/tutorials/setting-up-webpack-for-any-project, posjećeno 25. kolovoza 2019.)

S lijeve strane na slici 7 je moguće vidjeti sve module s ovisnostima koje Webpack pretvara u statične dokumente odnosno sredstva koja se koriste za izgradnju projektnog rješenja.

4.3 Java

Nakon JavaScript-a potrebno je opisati i programski jezik Java s obzirom da je sami JavaScript potekao od ideje Java-e, te iako JavaScript i Java nisu ni po čemu slični i danas se često dovode u korelaciju. Što je onda zapravo Java? Java je programski jezik koji se koristi za izradu softvera za različite platforme. Programer koji piše Java aplikaciju svoj kod prevodi (eng. compiling) na operacijskom sustavu (Windows, Linux i Mac OS). Java svoju sintaksu „vuče“ iz programskih jezika C i C++, a razvijena je sredinom 1990-it od strane James A. Gosling-a (bivšeg računalnog znanstvenika za Sun Microsystems) (Technopedia, 2019).

Java se danas sreće u raznim domenama pa se tako aplikacije koje su izrađene uz pomoć Java-e moguće sresti u: bankarstvu, maloprodaji, informacijskoj tehnologiji, Android razvoju, financijskim uslugama, tržištu dionica, velikim podacima, znanstvenim i istraživačkim zajednicama itd. (Johari, 2019).

Iz ovoga je moguće zaključiti da Java nudi veliki broj prilika za razvijanje različitih aplikacijskih rješenja u različitim područjima poslovanja. Za Java-u se kaže da je objektno orijentiran jezik sličan C++-u ali s naprednijim i pojednostavljenim značajkama. Java je besplatna te se može pokretati na svim platformama. Kao što je već spomenuto Java-u krase razne značajke kao što su: jednostavnost koja se ogleda u tome što je Java uklonila kompleksne stvari kao što su pokazivači, operatori preopterećenja koje je moguće pronaći u C++-u. Java je prijenosan jezik što znači da je bilo koju aplikaciju napisanu u Java programskom jeziku moguće prenijeti na bilo koju drugu platformu.

Java programski jezik je objektno orijentiran što znači da se sve smatra objektom što posjeduje neko stanje ili ponašanje, a sve operacije se izvode korištenjem upravo tih objekata. Java je također i prilično siguran programski jezik jer se sav kod koji je pisan u njemu pretvara u bajt kod (eng. bytecode) koji nakon prevođenja nije čitljiv ljudima, Java omogućava razvoj sustava odnosno aplikacija bez virusa i mogućnosti kvarenja (Johari, 2019).

4.4 Python

Razvoj programskih aplikacija je također moguć uz pomoć jednog od najpopularnijih programskih jezika – Python. Naime, Python je interpretirani, objektno orijentirani programski jezik visoke razine sa dinamičnom semantikom. Visoka razina Pythona se također ocrtava i u podatkovnoj strukturi koja se kombinira s dinamičnim pisanjem i dinamičnim povezivanjem što ga čini jako pogodnim za brzi aplikacijski razvoj te povezivanje postojećih komponenti (Python, 2019).

Python podržava module i pakete što programere ohrabruje na modularno programiranje i iskorištavanje koda. Neke od mogućnosti Python programskog jezika su:

- Može se koristiti na serveru za kreiranje web aplikacija
- Koristi se uz programe za kreiranje tokova rada
- Python se može spojiti sa sustavom baze podataka. Također može čitati i modificirati datoteke

- Python se može koristiti za manipuliranje velikim podacima te izvođenjem kompleksnih matematičkih radnji
- Koristi se za ubrzano kreiranje prototipova ili za produkcijski spreman razvoj softvera (W3Schools, 2019).

Posljednja verzija Pythona koja se koristi je Python 3 ali unatoč tome Python 2 je i dalje jako popularan te se koristi u visokoj mjeri bez obzira što je Python 3 unaprijeđen s mnogim sigurnosnim mjerama. Samo pisanje programskog jezika Python je dizajnirano radi čitljivosti te ima mnogo sličnosti s engleskim jezikom te kao što je već spomenuto, matematikom. Sintaksa Pythona koristi novu liniju kako bi završila neku komandu suprotno drugim programskim jezicima koji često koriste točka-zarez ili zagrade. Osim toga Python se najviše oslanja na uvlačenje redaka koristeći prazni prostor u kodu kako bi definirao djelokrug npr. djelokrug petlji, funkcija i klasa. Ostali programski jezici često koriste „vitičaste“ zagrade za izvršavanje tog zadatka (W3schools, 2019).

Programeri se često odlučuju za korištenje Pythona jer povećava produktivnost koju pruža. S obzirom da nema kompilacijskog koraka uređivanje i pronalaženje pogrešaka je jako brz ciklus. Sami softver za praćenje izvršenja programa (eng. debugger) koji Python koristi je napisan u Pythonu.

Kod Pythona je bitno naglasiti razliku između interpretera i prevoditelja. Interpreter obrađuje program dio po dio te naizmjence čita redove koda i izvršava naredbe, a prevoditelj prevodi izvorni kod u objektni kod koji se izvršava na hardveru računala (Downey, 2014).

4.5 C++

Prema IEEE Spectrum tablici popularnosti jezika C++ se ističe kao drugi najpopularniji programski jezik prema podacima iz 2018. godine (Cass, 2018). Za C++ se može reći da je statički tip jezika, slobodne forme koje se najčešće prevodi. Možda i najbitnija značajka koja se treba istaknuti kada se govori o programskom jeziku kao C++ je to da je on objektno orijentiran programski jezik ali ne u potpunosti o čemu će više biti govora u nastavku teksta. Sami jezik je razvijen od strane Bjarne Stroustrup-a te se često opisuje kao produženim jezikom C-a (Technopedia, 2019). C++ se smatra naprednim programskim jezikom jer uključuje najniže ali i najviše razine programiranja.

Sada se postavlja pitanje zašto se C++ ne naziva čistim objektno orijentiranim programskim jezikom? Pa upravo zato što C++ sadrži funkcionalnosti C programskog jezika te ga je stoga najispravnije zvati hibridom. To u isto vrijeme znači da C++ ima sve funkcionalnosti koje su dostupne u C jeziku kao:

- Univerzalno korišteni modularni programi
- Učinkovitost
- Prenosivi programi za različite platforme

Nadalje kada se govori o objektno orijentiranom programiranju bitno je istaknuti da se fokus osobito stavlja na objekte odnosno na aspekt na koji se problem odnosi. Objektno orijentirano programiranje kombinira podatke i funkcije. Klase definiraju određene objektne tipove definirajući podatke i funkcije objekta tog tipa (Kirch-Prinz, Prinz, 2001). Prednosti objektno orijentiranog programiranja su:

- smanjena osjetljivost na pogreške - objekt kontrolira pristup svojim podacima
- laka ponovna iskoristivost – objekti održavaju sami sebe te u tom slučaju mogu biti korišteni kao građevni blokovi za druge programe
- mali zahtjevi za održavanjem – objektni ti može modificirati svoje osobne unutrašnje podatke bez zahtijevanja promjena u aplikaciji (Kirch-Prinz, Prinz, 2001).

Nadalje za programski jezik C++ je moguće reći da ima mogućnost izrade raznovrsnih programa, i to počevši od platforma 16-bitnih aplikacija pa do onih 32-bitnih, statičkih, dinamičkih DLL-datoteka, Windows Help i dr.

Način na koji se izvršavaju programi specifičan je po tome što programski jezik C++ većinom rabi Windows resurse u izvršavanju svojih zadataka. Ako se fokus skrene na objektni dio programiranja to se svakako može potvrditi jer na taj način radi većina programskih jezika (Kovačević, 2004).

4.6 C#

C# je moderan programski jezik koji ima generalnu namjenu, objektno je orijentiran te se smatra programskim jezikom visoke razine. Sintaksa ovog programskog jezika je slična sintaksi C-a i C++-a ali mnoge značajke tih jezika nisu podržane u C# iz razloga kako bi se sami jezik pojednostavio. C# programi se sastoje od jedne ili više datoteka koje imaju „.cs“ ekstenziju te koji sadrže definicije klasa i drugih tipova. Ti tipovi datoteka se prevode u C# prevoditelju koji se naziva „csc“ nakon čega se kao rezultat dobiju prevedene datoteke koje se mogu izvršiti te prikazati značajke koda napisanog u njima (Nakov, Kolev, 2013).

C# se najčešće koristi za mrežne aplikacije te razvoj web aplikacija. Sami razvoj jezika započeo je u siječnju 1999. godine u Nizozemskoj kada je programer po imenu Anders Hejsberg formirao tim znanstvenika kako bi razvili C# programski jezik u svrhu dodavanja još jednog važnog dijela Microsoft NET programskom okviru. C# je inicijalno kreiran prema sličnim karakterizacijama objektno orijentiranog C jezika, a ime je dobio specifično iz razloga kako bi se Microsoft osigurao od tužbe drugih. U siječnju 2000. godine .Net je objavio C# (Technopedia, 2019).

Kako se ne bi ponovno nabrajale iste prednosti objektno orijentiranog programiranja mogu se spomenuti neke od prednosti samog C# programskog jezika, pa tako postoje:

- Učahureni (eng. encapsulated) potpisi metoda koji se nazivaju delegatima – omogućuje obavijesti o događaju koje su osigurane po tipu
- Atributi koji pružaju deklarativne podatke o tipu za vrijeme izvršavanja programa
- XML dokumentacija uredu
- Jezično integrirani upiti (LINQ koji pružaju ugrađene mogućnosti upita kroz razne izvore podataka (Microsoft, 2015)

Proces pokretanja C# aplikacije je mnogo jednostavniji kada se to usporedi s programskim jezicima C i C++ te je mnogo fleksibilniji od Java-e. Ne postoje odvojene datoteke koje se koriste kao zaglavlje te nema zahtjeva da metode i tipovi budu deklarirani u određenom redoslijedu. C# izvorna datoteka može definirati bilo koji broj klasa, struktura, sučelja i događaja (Microsoft, 2015).

C# ima ujedinjeni sustav tipova. Svi C# tipovi uključujući primitivne tipove kao što su to „int“ i „double“, nasljeđuju iz jednog korijenskog objektnog tipa. S obzirom na to, svi tipovi dijele isti set zajedničkih operacija te vrijednosti bilo kojeg tipa mogu biti spremljene, prebačene te nad njima se mogu izvršavati razne operacije u konzistentnom ponašanju (Hejlsberg, Wiltamuth, Golde 2006).

4.7 PHP

4.7.1 Povijest PHP jezika

PHP kao programski jezik je kreiran 1994. godine od strane Rasmus-a Lerdorfa. Prva verzija PHP-a je bio jednostavni set CGI (eng. Common Gateway Interface) binarnih naredbi napisanih u programskom jeziku C. Prva uporaba PHP-a je bila kada je Rasmus upravo PHP koristio kako bi pratio posjećenost njegovom online životopisu te je iz tog razloga PHP nazvan PHP alati osobne početne stranice (eng. Personal Home Page Tools). No, kako je vrijeme prolazilo PHP se proširivao raznim novim funkcionalnostima, pa je tako Rasmus ponovno prepisao cijeli PHP koji sadrži mnogo bogatiju i veću implementaciju. Novi model PHP jezika bio je mnogo sposobniji prilikom komuniciranja s bazom podataka. U lipnju 1995. godine Rasmus se odlučio objaviti izvorni kod (eng. source code) javnosti što je omogućilo programerima da koriste PHP programski jezik kako to njima odgovara. Osim toga ovaj potez je omogućio korisnicima PHP-a da popravljaju greške u kodu te generalno sami unaprijede kod (Tatroe, MacIntyre, Lerdorf, 2015).

U rujnu 1995. godine Rasmus je odlučio dodatno proširiti PHP ali u isto vrijeme i promijeniti mu ime pa ga je tako nazvao FI – interpreter formi (eng. forms interpreter). Nova implementacija je uključivala neke od temeljnih funkcionalnosti koje se koriste u PHP i danas. Imala je slične varijable kao i Perl programski jezik, automatsku interpretaciju varijabli za formu te ugrađenu HTML sintaksu. No, kako bi programeri mogli uključiti te koristiti HTML sintaksu unutar PHP dokumenata bilo je potrebno koristiti HTML komentare. Iako sama metoda nije bila baš najbolje prihvaćena u to doba, PHP je i dalje rastao u popularnosti kao CGI alat ali još uvijek ne kao programski jezik (Tatroe, MacIntyre, Lerdorf, 2015).

U listopadu 1995. Rasmus se odlučuje ponovno prepisati cijeli kod te objaviti novu verziju PHP-a koja vraća PHP ime te daje jeziku dizajn koji naliči na strukturu C programskog jezika što je omogućilo lakšu adaptaciju programerima koji su upoznati s C, Perl i sličnim programskim jezicima. Osim toga započinje se istraživanje za potencijalnom implementacijom za Windows NT jer je PHP u to doba bio ograničen na UNIX te POSIX sustave (PHP, 2019).

Sljedeći datum koji je ključan kada se radi o povijesti PHP-a je travanj 1996. godine a ključan je zbog toga što je Rasmus napravio novu „verziju“ PHP koja kombinira dva imena PHP/FI. Ovom novom generacijom PHP počinje svoju transformaciju iz alata u programski jezik. Nova verzija dobila je podršku za DBM (eng. data base management) – upravljanje bazom podataka, mSQL i Postgres95 baze podataka, kolačiće, korisnički definiranu funkcijsku podršku te još mnogo toga. Dva mjeseca poslije, točnije u lipnju 1995. godine PHP je i službeno postao PHP 2.0.

Do sredine 1997. godine verzija 2 PHP-a prilično je narasla te privukla velik broj korisnika, no još uvije su postojali problemi sa stabilnošću u temeljnom mehanizmu za obrađivanje. Na projektu je još uvijek uglavnom radio samo jedan čovjek uz nekoliko doprinosa sa strane. No, tad su se Zeev Suranski i Andi Gutmans iz Tel Aviva ponudili da ponovno napišu temeljni mehanizam za obrađivanje, koji je na kraju postao temelj verzije 3 PHP-a. Nakon toga mnogo drugih programera se ponudilo za malu pomoć na ostalim dijelovima PHP-a i tako se projekt promijenio iz samostalnog napora uz malu pomoć drugih u pravi projekt otvorenog izvornog koda s brojnom zajednicom programera širom svijeta.

Upravo ovakva inicijativa je bila potrebna kako bi se PHP počeo znatno brže razvijati, te sada kada je zajednica bila utemeljena niz programera je počeo raditi na verziji 4.0 PHP-a. Ti programeri su imali cilj napraviti neke izmjene u arhitekturi (izmjene apstrakcije sloja između jezika i Web poslužitelja, dodatak mehanizma za sigurnost niti te dodatak naprednijeg dvostupanjskog sustava obrađivanja oznaka). PHP 4.0 pušten je u distribuciju 22. svibnja 2000. godine. PHP je tako nastavio svoj razvoj te je u trenutku pisanja ovog završnog rada bila aktualna verzija 7.3 koja je puštena u distribuciju 6. prosinca 2018. godine te je podržana sve do 6. prosinca 2021 (Tatroe, MacIntyre, Lerdorf, 2015).

4.7.2 Programski okviri unutar PHP

Programski okviri sami po sebi su prilično korisni te web programiranje čine mnogo organiziranije i bolje u puno različitih načina. Pisanje programskog koda koje može oduzeti sate te stotine linija koda je uz pomoć programskih okvira pretvoreno u samo nekoliko minuta, što samim programskim okvirima daje epitet produktivnijeg programiranja. Dugo vremena PHP kao programski jezik nije bio smatran zadovoljavajućim za pisanje velikih web aplikacija jer je bio popularan za manje projekte te osobito jer je programiranje većih web aplikacija bilo rezervirano za programske okvire kao što su Spring, Ruby on Rails ili Django. No situacija se promijenila te je postalo jasno da PHP ima mnogo veće mogućnosti pojavljivanjem PHP-ovih programskih okvira kao što su: Laravel, Symfony, CakePHP ili npr. Zend programski okvir.

Razvoj ovih programskih okvira je bio prilično brz i stabilan, sami izvorni kod je pisan objektno orijentiranim principima te je pisan u PHP 5.0 verziji što je sami kod učinilo elegantnim te lako održivim. Uvođenje programskih okvira za PHP je PHP gurnulo u sami vrh programskih jezika za razvoj web aplikacija pa su tako najpopularniji programski jezici za razvoj web aplikacija: PHP, Ruby, Python i Java (Prokofyeva, Boltunova, 2017).

4.7.3 Usporedba programskih okvira unutar PHP-a

U nastavku teksta opisani će biti Symfony, CodeIgniter, Laravel te Phalcon koji se svi svrstavaju pod PHP programske okvire. Symfony je jedan od PHP programskih okvira koji su se razvili u ranoj fazi pojeve programskih okvira kod PHP-a. Kreator Symfony-a je Fabien Potencier koji je prvu verziju ovog PHP programskog okvira pustio u distribuciju 2005. godine. U vrijeme pisanja posljednja verzija Symfony-a je bila 4.3 te je zahtijevala 7.1 verziju ili višu instaliranu na računalu na kojemu se planira postaviti programski okvir Symfony. Neke od značajki koje vrijedi istaknuti kod Symfony-a su: mogućnost instalacije programskog okvira kao samostalnog alata, ugrađeno objektno relacijsko pridruživanje (eng. mapping) (ORM), komponente za korisničku ovjeru (eng. authentication) i autorizaciju itd.

CodeIgnite je također jedan od rano razvijenih programskih okvira unutar PHP-a čija je prva verzija puštena u distribuciju 2006. godine. Za razvoj ovog programskog okvira odgovoran je Rick Ellis, rock muzičar koji je svoju profesiju zamijenio programiranjem. Sami poticaj za razvoj ovog programskog okvira je bilo Ellis-ovo ne zadovoljstvo u tadašnje vrijeme postojećim programskim okvirima unutar PHP-a.

Za ovaj programski okvir ne postoji instalacijski alat kako bi se instalirana njegova stabilna verzija nego je cijela struktura programskog okvira zapakirana u kompresijski paket i to je sve što programer treba kako bi počeo koristiti CodeIgnite programski okvir (Samra, 2015).

Nadalje Laravel je programski okvir koji se svrstava u novije PHP programske okvire čija je prva verzija puštena u distribuciju 9. lipnja 2011. godine pod verzijom 1 „beta“. Aktualna verzija Laravela u vrijeme pisanja ovog završnog rada je verzija 5.4 koja je puštena u distribuciju 24. siječnja 2017. godine te za svoj rad zahtijeva verziju 5.6.4 ili više PHP-a. Laravelov kreator je Taylor Otwell koji je kao razlog za kreiranje programskog okvira Laravel naveo nedostatak osnovnih funkcionalnosti kao npr. korisnička ovjeru u CodeIngniter programskom okviru. Bitno je naglasiti da Symfony i Laravel imaju mnogo zajedničkih funkcionalnosti te se zato često i uspoređuju (Samra, 2015).

Phalcon je kao i Laravel jedan od novijih PHP okvira koji je pušten u produkciju 2012. godine. Verzija Phalcon-a u vrijeme pisanja ovog rada je bila 3.4.4 te za stabilan rad zahtijeva instaliran PHP 7.0 i više. Kreator Phalcon-a je Andres Gutierrez koji se odlučio Phalcon napisati kao PHP produžetak u C programskom jeziku. Taj produžetak je učitan u memoriju samo jednom te je njegov API prikazan programeru (Samra, 2015).

5. Laravel

5.1 Razvoj Laravela kao PHP programskog okvira

Cilj ovoga podnaslova će biti pobliže opisati razvoj Laravela kao programskog okvira kroz njegovu povijest te razvoj kroz verzije. Sama povijest će reći nešto o nastanku Laravela njegovim početnim funkcionalnostima te daljnjem razvoje, a poseban podnaslov će se posvetiti svim prethodnim verzijama te usporedbi između njih.

5.1.1 Povijest Laravela

Za sami Laravel se može reći da je na scenu došao prilično kasno, u prijevodu na tržištu je već postojalo mnogo opcija kada se govori u smislu PHP programskih okvira (npr. CodeIgniter, Symfony, FuelPHP itd.). Ali, programeri koji su razvili Laravel su taj vremenski „deficit“ nadoknadili sjajnim razvojem izbjegavši sve greške koje je dosta programskih okvira napravilo. Kao npr. pokušavanje izbjegavanja problema ostalih programskih okvira, građenja na postojećim programskim okvirima ili nuđenje velike palete nefleksibilnih biblioteka. Naime, Laravel nudi pametne komponente vođene upravljačkim programima (eng. drivers) koje programeri mogu koristiti kako bi gradili aplikacije na svoj način (Bean, 2015).

Laravel je alat koji koristi postojeće komponente kako bi omogućio kohezivni sloj koji omogućuje gradnju web aplikacije u puno strukturalnijem i pragmatičnijem načinu. Samu inspiraciju Laravel je povukao iz mnogo drugih popularnih programskih okvira kao što su: CodeIgniter, Yii, ASP.NET MVC, Ruby on Rails, Sinatra te ostalih. Osim inspiracije Larave je od svakog od tih programskih okvira je preuzeo najbolje ideje kao bi izgradio svoju strukturu te funkcionalnost. Bitno je naglasiti da većina ovih programskih okvira koristi paradigmu MVC – kontrolor preglednih modela (eng. Model View Controller).

S obzirom da je Laravel relativno novi programski okvir postojala je potreba za dokazivanjem da je on zapravo u sposobnosti natjecati se s drugim programskim okvirima. To se upravo i dogodilo od verzije 3 kada je Laravel ekstremno porastao u popularnosti te postao jedan od najpopularnijih te široko korištenih PHP programskih okvira u jako kratkom vremenskom roku (Bean, 2015).

5.1.2 Usporedba te osvrt na prethodne verzije Laravela

Kritički osvrt na prethodne verzije Laravela će se napraviti samo od verzije 4 pa nadalje jer nema smisla se vraćati u povijest u verzije koje se više ne održavaju te za koje se ne pruža podrška ni od strane programera koji razvijaju programski okvir Laravel. Pa ovako, u samoj verziji 4.1 Laravela moguće je reći da su uvedene nove SSH komponente koje su omogućile olakšano spajanje na udaljene servere te izvršavanje naredbi na njima, prioritet izvršavanje naredbi u redu čekanja, fleksibilni podsjetnici za lozinke, unapređeni mehanizmi za sesije itd. Larevel 4.2 je zahtijevao minimalnu verziju 5.4 PHP-a a u sebi je od novih značajki uključio „Laravel Homestead“ koji se može opisati kao razvojno okruženje za izgradnju robusnih Laravel i PHP aplikacija. Nadalje osim „Homested-a“ verzija 4.2 je uključivala „Laravel Cashier“ koji je zapravo biblioteka za upravljanje pretplatama, mail API upravljačke programe, potvrdu migracije te ostale značajke.

Sljedeća prilično značajna verzija Laravela je 5.0 verzija koja je korisnicima predstavila novu aplikacijsku strukturu za zadane (eng. default) Laravelove projekte. Ova nova struktura služi kao bolji temelj za građenje robusnih aplikacija u Laravelu. Neke od novih stvari koje su bile uključene u ovu verziju su: nova datotečna struktura, datoteka ugovori (eng. contracts), putevi pred-memorije (eng. route cache), putevi središnjice (eng. route middleware), kontrolor metoda ubrizgavanja, događaj objekti, red čekanja u bazi podataka, Tinker itd.

Laravel se trenutno nalazi na verziji 5.8 koja je puštena u distribuciju u veljači 2019. godine.

Verzija	Datum distribucije	Popravak grešaka do	Popravak sigurnosti do
5.0	04. veljače 2015.	04. kolovoza 2015.	04. veljače 2016.
5.1 (LTS)	09. lipnja 2015.	09. lipnja 2017.	04. lipnja 2018.
5.2	21. prosinac 2015.	21. lipanj 2016.	21. prosinac 2016.
5.3	23. kolovoz 2016.	23. siječanj 2017.	23. kolovoz 2017.
5.4	24. siječanj 2017.	24. srpanj 2017	24. siječanj 2018.
5.5 (LTS)	30. kolovoz 2017.	30. kolovoz 2019.	30. kolovoz 2020.

Verzija	Datum distribucije	Popravak grešaka do	Popravak sigurnosti do
5.6	07. veljača 2018.	07. kolovoz 2018.	07. veljača 2019.
5.7	04. rujan 2018.	04. ožujak 2018.	04. rujan 2019.
5.8	26. siječanj 2019.	26. kolovoz 2019.	26. siječanj 2020.

Tablica 1. Prikaz svih pod verzija Laravelove 5 inačice. Izrada autora prema https://laravel.com/docs/5.8/releases

Na tablici 1 je moguće vidjeti sve verzije (5) Laravela počevši od 5.0 verzije pa sve do posljednje verzije koja je 5.8 verzija. Osim toga tablica prikazuje još dva dodatna podatka a to su podaci o datumima do kada tim Laravela podržava tj. popravlja greške (eng. bugs) te sigurnost (eng. security).

5.2 Arhitektura i struktura Laravela

5.2.1 Osnovni koncepti Laravela

5.2.1.1 Usmjeravanje i središnjica

Za početak fokus će se staviti na usmjeravanje (eng. routing), naime najjednostavnije usmjeravanje kada se radi o primjeru Laravela prihvaća adresu na koju treba ići te završetak (eng. closure) (Laravel, 2019).

```
Route::get('foo', function () {
      Return 'Hello World';
});
```

Na prethodnom isječku koda je moguće vidjeti upravo ono što je objašnjeno u prethodnoj rečenici, usmjeravanje kao prvi parametar prima adresu koja je određena sa “foo” u ovome slučaju te funkciju koja će vratiti neku vrijednost na kraju a čija se vrijednost smatra zatvaračem usmjeravanja.

Kao i većina usmjeravanja tj. preusmjeravanja koja postoje u drugim programskim jezicima i Laravel ima nekoliko metoda koje se koriste za usmjeravanje.

```
Route::get($uri, $callback);
Route::post($uri, $callback);
Route::put($uri, $callback);
Route::patch($uri, $callback);
Route::delete($uri, $callback);
Route::options($uri, $callback);
```

Na prethodnom isječku koda je moguće vidjeti upravo šest Laravelovih metoda koje se koriste za usmjeravanje. Nadalje kod usmjeravanja je bitno spomenuti da se mogu koristiti i parametri usmjeravanja (Laravel, 2019).

```
Route::get('user/{id}', function ($id) {
      Return 'User '.$id;
});
```

Primjer koji je prikazan na isječku koda iznad, korisnika usmjerava na stranicu "user/{id}" što znači da će osoba koja se nalazi na toj stranici unosom određenog id-a korisnika koji postoji u bazi biti u mogućnosti prikazati njegovu stranicu a funkcija koja se koristi kao drugi parametar će ispisati "User {id}" tj. tekst te „id" korisnika koji je pronađen.

Imenovana usmjeravanja omogućavaju skraćenije iskorištavanje preusmjeravanja u ostatku projekta odnosno programa koji je kreiran s Laravelom (Laravel, 2019).

```
Route::get('user/profile', function () {
      //
})->name('profile');
```

Moguće je primijetiti da na prethodnom isječku koda postoji kreirano usmjeravanje na stranicu "user/profile" sa funkcijom kao drugim parametrom a na samome kraju nalazi se funkcionalnost imenovanja tog cijelog usmjeravanja.

```
$url = route('profile');
Return redirect()->route('profile');
```

Samo iskorištavanje imenovanja usmjeravanja odnosi se na zgodnu generaciju URL-ova za specifičnih usmjeravanja. Osim toga imenovanje usmjeravanja je moguće specificirati i za akcije kontrolora (eng. controllers) (Laravel, 2019).

Nadalje, kako bi se središnjica pridružila svim rutama unutar grupe koristi se metoda središnjice prije nego se sama grupa definira. Središnjice se izvršavaju u poretku u kojemu su navedene unutar polja.

```
Route::middleware(['first', 'second'])->group(function () {
      Route::get('/', function () {
      });

      Route::get('user/profile', function () {
      });
});
```

5.2.1.2 Kontrolori, zahtjevi i odgovori

Umjesto definiranja logike svih zahtjeva unutar datoteka usmjeravanja moguće je tu logiku te njezino ponašanje organizirati uz pomoć kontrolorskih klasa (eng. controller classes). Potrebno je naglasiti da kontrolori imaju mogućnost grupirati povezanu logiku zahtjeva unutar jedne klase. Kontrolori se unutar strukture Laravela 5 mogu pronaći pod app/Http/Controllers. Na sljedećem isječku koda je moguće vidjeti primjer jednostavne kontrolorske klase koja produžuje (eng. extends) baznu kontrolorsku klasu koja nalazi unutar Laravela. Ta produžena klasa pruža nekoliko pogodnih metoda kao što je metoda središnjice koja se može koristi kako bi se središnjica vezala uz kontrolorske akcije.

```
<?php
namespace App\Http\Controllers;

use App\Http\Controllers\Controller;
use App\User;
class UserController extends Controller {
      public function show ($id)
      {
            return view('user.profile', ['user' =>
            User::findOrFail($id)]);
      }
}
?>
```

Što se tiče HTTP zahtjeva (eng. requests) kako bi se dohvatila trenutna instanca HTTP zahtjeva potrebno je koristiti ubrizgavanje zavisnosti (eng. dependency injection) na način da se uz pomoć automatskog završavanja (eng. type-hint) u Illuminate/Http/Request definira klasa. Nakon toga nadolazeća instanca zahtjeva će automatski biti ubrizgana uz pomoć servisnog kontejnera (eng. service container). Na sljedećem isječku koda je moguće vidjeti način na koji se definira pristupanje zahtjevu.

```
<?php
namespace App\Http\Controllers;

use Illuminate\Http\Request;

class UserController extends Controller
{
      public function store(Request $request)
      {
            $name = $request->input('name');
      }
}
?>
```

Odgovori (eng. responses) se mogu podijeliti na nekoliko dijelova, a to su kreiranje odgovora, preusmjeravanja te ostali tipovi odgovora. Kada se govori o kreiranju odgovora onda je bitno naglasiti da sve rute i kontrolori moraju vratiti odgovor koji će biti poslan nazad korisničkom pretraživaču. Laravel pruža nekoliko načina prema kojima se može vratiti odgovor korisniku, a najjednostavniji je vraćanje nekog teksta od rute ili kontrolora. Programski okvir će automatski pretvoriti tekst u potpuni HTTP odgovor.

```
Route::get('/', function () {
      return 'Hello World';
});
```

5.2.1.3 Generiranje URL-a i sesije

Laravel pruža nekoliko pomoćnika koji bi trebali asistirati prilikom generiranja URL-a za aplikaciju. Oni su većinom od pomoći kada se kreiraju poveznice unutar predložaka, API odgovora ili kada se generira direktni odgovor prema drugom dijelu aplikacije. URL pomoćnik može biti korišten kako bi generirao proizvoljne URL-ove za aplikaciju. Ti generirani URL-ovi će automatski koristiti shemu (HTTP ili HTTPS) te usluge posluživanja iz trenutnog zahtjeva.

```
$post = App\Post::find(1);
echo url('/post/{$post->id}');
```

S druge strane što se tiče sesije, ona se većinom dovodi u priču kada se govori o kontekstu HTTP aplikacija koje su bez stanja. Sesije kreiraju način spremanja informacija o korisnicima preko više zahtjeva. Sesijski konfiguracijski dokument se nalazi u config/session.php. Laravel kod sesija omogućuje konfiguriranje te biranje određenog upravljačkog programa (eng. drivera) pa je tako moguće birati između: file, cookie, database, memcached/redis ili array upravljačkog programa.

5.2.1.4 Validacija

Unutar Laravela nudi se nekoliko različitih pristupa za validaciju podataka koji dolaze u aplikaciju. Prema zadanim postavkama Laravelova osnovna kontrolorska klasa koristi „ValidateRequests“ svojstvo koje pruža zgodnu metodu koja se koristi za validaciju HTTP zahtjeva sa raznolikim validacijskim pravilima. Kako bi se sama validacija brzo postavila potrebno je proći kroz nekoliko koraka: definiranje putanja (eng. routes), kreiranje kontrolora (eng. controllers), pisanje validacijske logike, prikazivanje validacijskih grešaka.

5.2.1.5 Rukovanje greškama

Prilikom kreiranja novog Laravel projekta rukovanje greškama i izuzecima (eng. exceptions) je već konfigurirano za programera koji će raditi na aplikaciji. Kroz putanju app/exceptions/handler moguće je doći do klase gdje se svi izuzeci koji su pokrenuti od strane aplikacije ispisuju te prikazuju korisnicima. Unutar Laravelove dokumentacije moguće je detaljnije istražiti tu klasu i saznati sve o njezinim funkcionalnostima.

5.2.1.6 Predlošci

Za kreiranje predložaka unutar Laravela koristi se Blade – mehanizam namijenjen upravo za tu svrhu. Blade je jako moćan mehanizam te za razliku od drugih PHP mehanizama za kreiranje predložaka Blade ne zabranjuje pisanje „čistog“ PHP koda unutar svojih predložaka.

5.2.1.7 Lokalizacija

Laravelove lokalizacijske značajke pružaju zgodan način vraćanja teksta u različitim jezicima što zapravo aplikaciju čini mnogo fleksibilnijom u podržavanju više jezičnosti. Jezični tekstovi su spremljeni unutar datoteka koje se nalaze u resources/lang. Kako bi se postavila bazne jezične postavke potrebno je prvo definirati „locale“ putanju koja će postaviti aktivni jezik u trenutnom vremenu pomoću metode „setLocale“.

```
Route::get('welcome/{locale}', function ($locale) {
     App::setLocale($locale);
});
```

```
'fallback_locale' => 'en'
```

6. Izrada web aplikacije korištenjem PHP programskog okvira Laravel

6.1 Zahtjevi i karakteristike web aplikacije

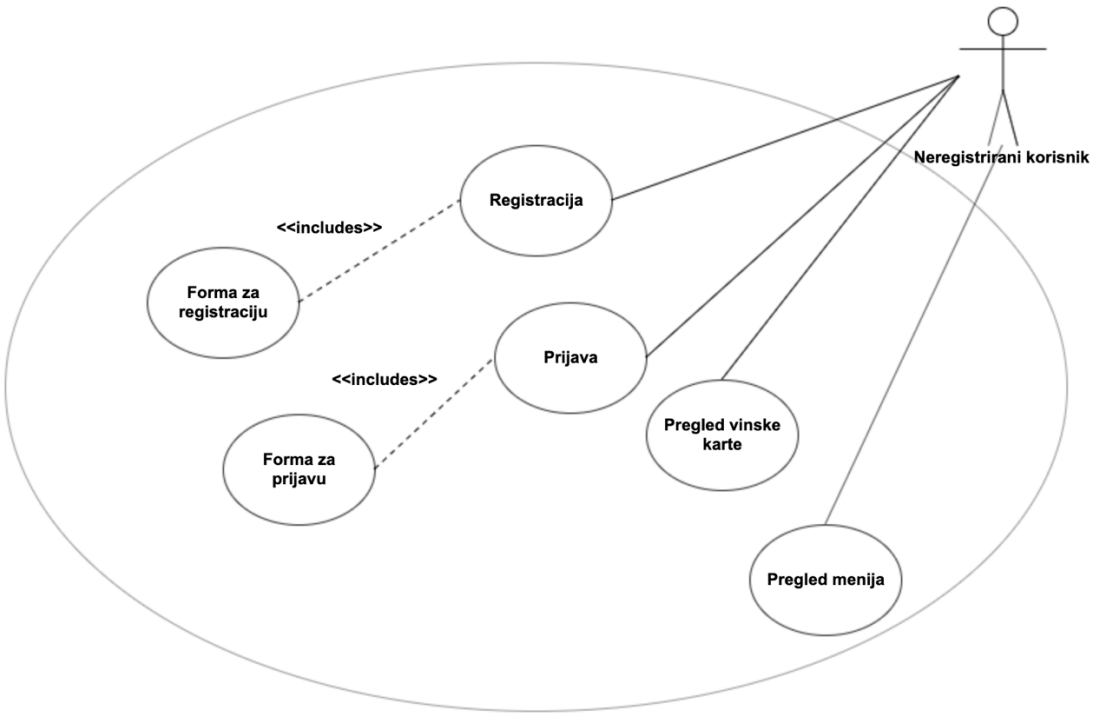
Kratki opis projekta:

Sustav koji služi za upravljanje visoko klasnim restoranom.

Uloge:

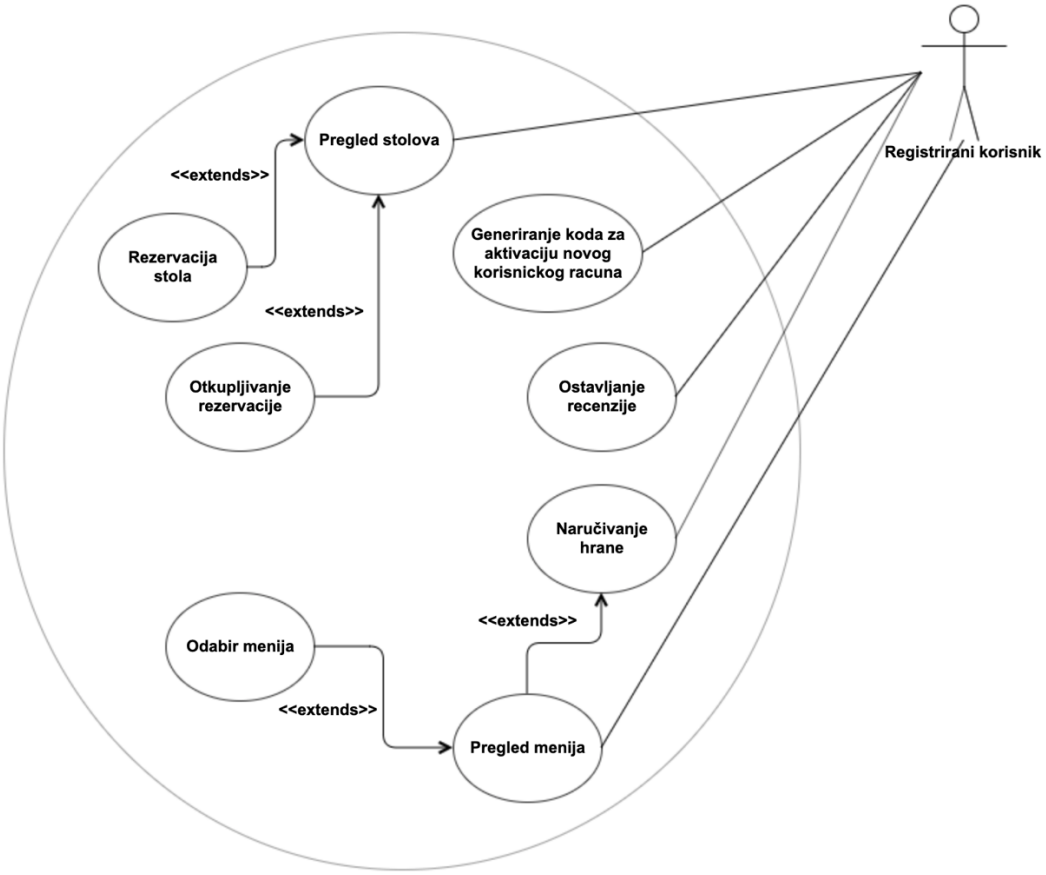
Neregistrirani korisnik, registrirani korisnik, moderator, administrator.

Kako bi se bolje pojasnilo cjelokupno funkcioniranje aplikacije te definirali procesi pojedinačnih uloga, unutar ovoga podnaslova uz pomoć dijagrama slučajeva korištenja za svaku ulogu će se prikazati njezina detaljna razrada. Prva uloga koja će biti prikaza na je uloga neregistriranog korisnika, ova uloga može odabrati jedan od četiri moguća slučaja a to su registracija koja uključuje pristup registracijskoj formi, prijava koja uključuje pristup formi za prijavu, pregled vinske karte i pregled menija.



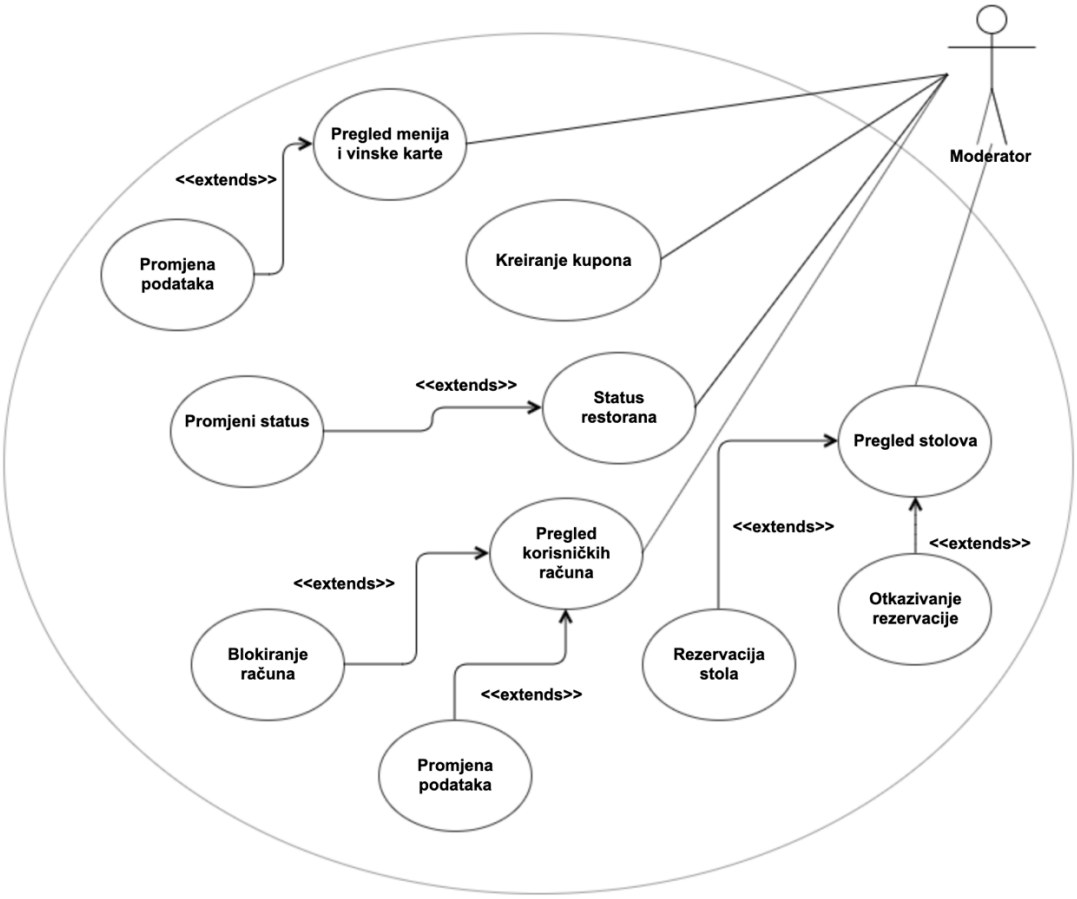
Slika 8. Dijagram slučaja korištenja za neregistriranog korisnika. Izrada autora.

Nakon dijagrama slučaja korištenja za neregistriranog korisnika prikazati će se dijagram za registriranog korisnika. Registrirani korisnik ima više slučajeva koji se mogu ostvariti kao što su; generiranje koda za registraciju novog korisnika, ostavljanje recenzije, naručivanje hrane koje produžuje slučaj pregleda menija, pregled menija koji produžuje slučaj odabira menija, pregled stolova koji produžuje dva slučaja: rezervaciju stolova i otkupljivanje rezervacija.



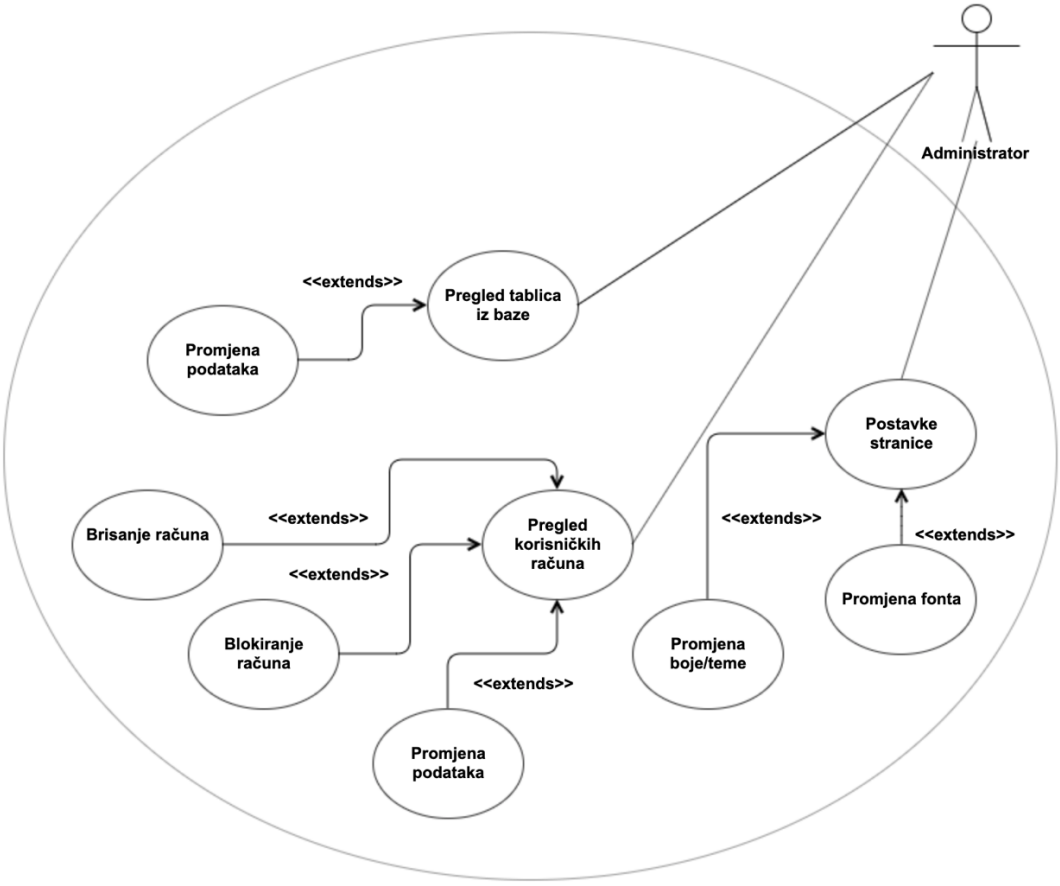
Slika 9. Dijagram slučaja korištenja za registriranog korisnika. Izrada autora.

Sljedeći dijagram slučaja korištenja je dijagram koji se odnosi na moderatora. Moderator za razliku od neregistriranog i registriranog korisnika može ostvariti slučaj koji se odnosi na pregled korisničkih računa te koji produžuje promjenu podataka i blokiranje računa. Osim toga moderator može ostvariti slučajeve kao što su pregled kupona, pregled menija i vinske karte koji produžuje slučaj promjene podataka, pregled stolova koji produžuje otkazivanje rezervacije i rezerviranje stola i slučaj statusa restorana koji produžuje promjenu statusa.



Slika 10. Dijagram slučaja korištenja za moderatora. Izrada autora.

Posljednji dijagram slučajeva korištenja je dijagram koji se odnosi na ulogu administratora. Naime administrator može ostvariti slučajeve kao što su pregled korisničkih računa koji produžuje blokiranje računa, promjenu podataka te brisanje računa, postavke stranice koji produžuje promjenu boje odnosno teme i promjenu fonta te slučaj pregleda svih tablica iz baze koji produžuje promjenu podataka.



Slika 11. Dijagram slučaja korištenja za administratora. Izrada autora.

6.2 Tehnologije i alati potrebni za izradu

Kada se govori o alatima koji su korišteni u izradi programskog rješenja ovoga rada onda je moguće prvenstveno istaknuti Mac OS kao sustav te alate kao što su:

- Visual studio code
- iTerm (terminal)
- Visual paradigm (ERA)
- Gliffy (UML)

Tehnologije odnosno programski jezici te programski okviri koji su korišteni u svrhu izrade ovog programskog rješenja su:

- HTML, SCSS
- Vue.js
- PHP
- Laravel

6.3 Opis implementacije web aplikacije

6.3.1 Model baze podataka

Model baze podataka je prva stavka koja se mora kreirati odnosno definirati prije početka izrade same aplikacije. Pa stoga sami model se kreće graditi iz tablice korisnik te tip korisnika koje će se koristiti za spremanje podataka o korisnicima te određivanju njihovog tipa (administrator, moderator, registrirani korisnik). Nadalje, kako bi se sama registracija mogla realizirati potrebno je kreirati i tablicu kodova za registraciju koji će služiti za potvrdu odnosno aktivaciju korisničkog računa. Nakon prijave i registracije potrebno je osigurati i aplikacijske funkcionalnosti za neregistriranog korisnika. Jedna od tih funkcionalnosti je pregled menija te vinske karte. Za pregled menija i vinske karte kreirane su tablice meni, vinska karta te stavka koje su povezane tablicama koje realiziraju veze više na više jer meni i vinska karta mogu imati više od jedne stavke na sebi a sama stavka može se pojaviti samo jednom. Tablica stol će se koristiti za spremanje podataka o svim stolovima koji su trenutno aktivni unutar restorana, rezervacija stola će se obavljati putem tablice rezervacija koja će u sebi sadržavati ključ registracije te vanjski ključ stola. S obzirom da će korisnici prikupljati bodove te uz pomoć tih bodova biti u mogućnosti otkupiti određenu rezervaciju unutar aplikacije će postojati komunikacijski kanal putem kojeg će korisnici moći razmijeniti poruke te dogovoriti moguće otkupljivanje rezervacije za restoranske bodove. Nakon obavljene večere svaki korisnik aplikacije može ostaviti recenziju koja će se odnositi na dojmove o hrani, piću, usluzi te ukupnom dojmu restorana. Za kreiranje recenzija koristiti će se tablica recenzije koja će svaku recenziju povezati s korisnikom kako bi se znalo koji korisnik je ostavio koju recenziju.

Za restoranske bodove korisnik će biti u mogućnosti kupovati kupone koje restoran nudi, sami kuponi spremati će se u tablicu kuponi a vlasništvo kupona će biti realizirano uz pomoć treće tablice za vezu više na više između tablice korisnik i tablice kuponi. U nastavku će biti prikazane sve tablice koje su prethodno opisane a nakon toga će se sve sažeti unutar slike ERA modela.

Naziv stupca	Tip	Osobine	Opis
id_korisnik	integer(255)	pk	
ime	varchar(255)		
prezime	varchar(255)		
email	varchar(255)		
lozinka	varchar(255)		
tajno_pitanje	varchar(255)		
email_kod	varchar(255)		
zakljucan	integer(10)		
bodovi	integer(255)		

Tablica 2. Korisnik. Izrada autora.

Naziv stupca	Tip	Osobine	Opis
id_tip	integer(255)	pk	
naziv_tipa	varchar(255)		

Tablica 3. Tip korisnika. Izrada autora.

Naziv stupca	Tip	Osobine	Opis
korisnikid_korisnik	integer(255)	fk	
tip_korisnikaid_tip	integer(255)	fk	

Tablica 4. Korisnik tip. Izrada autora.

korisnik	1..*	korisnik tip
id_korisnik	<->	korisnikid_korisnik

Tablica 5. Vanjski ključ korisnik - korisnik tip. Izrada autora.

Vanjski ključ koji se nalazi u tablici tip korisnika povezuje tablicu korisnika te tablicu tip korisnika unutar tablice korisnik tip vezom jedan prema više iz razloga što jedan korisnik može u isto vrijeme imati dvije uloge a jedna uloga može biti određena za više korisnika. Na taj način se sprječava da se ponovi ista kombinacija korisnika te uloge što znači da ako je određeni korisnik administrator jednom ne može tu ulogu obnašati putem dva ili više različitih id-eva.

tip korisnika	1..*	korisnik tip
id_tip	<->	tip_korisnikaid_tip

Tablica 6. Vanjski ključ tip korisnika - korisnik tip. Izrada autora.

Ista situacija se nalazi i s druge strane gdje se tablica tip korisnika povezuje s tablicom korisnik putem tablice korisnik tip vezom jedan na više. Kao što je već objašnjeno svaki tip korisnika može se koristiti više puta a to osigurava veza jedan na više prema tablici korisnik tip, u suprotnome tip korisnika se samo jednom može ponoviti u kombinaciji s određenim id-em korisnika.

Naziv stupca	Tip	Osobine	Opis
id_stol	integer(255)	pk	
naziv	varchar(255)		
status	varchar(255)		

Tablica 7. Stol. Izrada autora.

Naziv stupca	Tip	Osobine	Opis
id_rezervacije	integer(255)	pk	
datum	date		
vrijeme_od	time(7)		
vrijeme_do	time(7)		
stolid_stol	integer(255)	fk	
korisnikid_korisnik	integer(255)	fk	

Tablica 8. Rezervacija. Izrada autora.

korisnik	1..*	rezervacija
id_korisnik	<->	korisnikid_korisnik

Tablica 9. Vanjski ključ korisnik - rezervacija. Izrada autora.

Tablica korisnik je povezana s tablicom rezervacija vezom jedan na više te unutar te tablice ima svoj vanjski ključ korisnikid_korisnik. Vezom jedan na više se osigurava da svaki korisnik može rezervirati više stolova odnosno napraviti jednu ili više rezervacija a s druge strane se osigurava da svaka rezervacija može biti kreirana od strane samo jednog korisnika.

stol	1..*	rezervacija
id_stol	<->	stolid_stol

Tablica 10. Vanjski ključ stol - rezervacija. Izrada autora.

Kao i tablica korisnik tablica stol je povezana s tablicom rezervacija vezom jedan na više te unutar te tablice ima svoj vanjski ključ stolid_stol koji osigurava tu vezu. Prema definiranoj vezi tablica stol može imati jedan stol definiran unutar više rezervacija a jedna rezervacija sadrži samo jedan stol. Ovaj se slučaj može još proširiti tako da se omogući da jedna rezervacija sadrži i više stolova ali za time nije bilo potrebe s obzirom da aplikacija i bez toga ima visoku razinu složenosti.

Naziv stupca	Tip	Osobine	Opis
id_koda	integer(255)	pk	
kod	varchar(255)		
korisnikid_korisnik	integer(255)	fk	

Tablica 11. Registracija kod. Izrada autora.

korisnik	1..*	registracija kod
id_korisnik	<->	korisnikid_korisnik

Tablica 12. Vanjski ključ korisnik - registracija kod. Izrada autora.

Tablica registracija kod koja se koristi za spremanje kodova aktivacije novo registriranih korisničkih računa u sebi sadrži vanjski ključ koji je povezan s primarnim ključem iz tablice korisnik. Od tablice korisnik prema tablici registracija kod moguće je prepoznati vezu jedan na više. Tom vezom se osigurava da jedan korisnik može generirati više od jednog koda za aktivaciju korisničkog računa odnosno jedan kod je povezan samo sa jednim korisnikom.

Naziv stupca	Tip	Osobine	Opis
id_kupon	integer(255)	pk	
naziv	varchar(255)		
opis	varchar (255)		
bodovna_cijena	varchar(255)		

Tablica 13. Kuponi. Izrada autora.

Naziv stupca	Tip	Osobine	Opis
korisnikid_korisnik	integer(255)	fk	
kuponiid_kupon	integer (255)	fk	

Tablica 14. Korisnik kupon. Izrada autora.

kuponi	1..*	korisnik kupon
id_kupon	<->	kuponiid_kupon

Tablica 15. Vanjski ključ kuponi - korisnik kupon. Izrada autora.

Tablica kupona i tablica korisnika se povezuju putem tablice korisnik kupon vezom više na više. Gledano iz perspektive tablice kuponi prema tablici korisnik kupon formirana je veza jedan na više kojom se osigurava da svaki kupon može biti kupljen od strane više korisnika.

korisnik	1..*	korisnik kupon
id_korisnik	<->	korisnikid_korisnik

Tablica 16. Vanjski ključ kuponi - korisnik kupon. Izrada autora.

S druge strane tablica korisnik je kao i tablica kuponi povezana s vezom jedan na više s tablicom korisnik kupon. Što znači da jedan korisnik može kupiti više kupona, na taj način se osigurava konzistentnost postizanja veze više na više.

Naziv stupca	Tip	Osobine	Opis
id_recenzije	integer(255)	pk	
recenzija	varchar(255)		
korisnikid_korisnik	integer(255)		

Tablica 17. Recenzije. Izrada autora.

korisnik	1..*	korisnik kupon
id_korisnik	<->	korisnikid_korisnik

Tablica 18. Vanjski ključ korisnik - recenzije. Izrada autora.

Naime kako aplikacija nudi mogućnost ostavljanja recenzije u svrhu prikupljanja povratnih informacija od korisnika restorana te informacije odnosno recenzije se spremaju u tablicu recenzije. Tablica recenzije unutar svojih atributa sadrži i vanjski ključ korisnikid_korisnik koji se referencira na primarni ključ iz tablice korisnik.

Na taj način se osigurava veza jedan na više od tablice korisnik prema tablici recenzije što znači da jedan korisnik može ostaviti više recenzija a jedna recenzija može biti napisana samo od strane jednog korisnika.

Naziv stupca	Tip	Osobine	Opis
id_poruke	integer(255)	pk	
poruka	varchar(255)		
korisnikid_korisnik	integer(255)	fk	
korisnikid_korisnik2	integer(255)	fk	

Tablica 19. Poruka. Izrada autora.

korisnik	1..*	korisnik poruka
id_korisnik	<->	korisnikid_korisnik

Tablica 20. Vanjski ključ korisnik – poruka 1. Izrada autora.

korisnik	1..*	korisnik poruka
id_korisnik	<->	korisnikid_korisnik2

Tablica 21. Vanjski ključ korisnik – poruka 2. Izrada autora.

Od tablice korisnik do tablice poruka kreirana je veza jedan na više koja osigurava da jedan korisnik može poslati odnosno pohraniti više poruka u bazu podataka a jedna poruka može biti poslana od strane jednog korisnika samo jednom korisniku te upravo iz tog razloga ova veza postoji dva puta. I korisnika je potrebno referencirati još jednom kako bi se osim pošiljatelja mogao utvrditi i primatelj.

Naziv stupca	Tip	Osobine	Opis
id_meni	integer(255)	pk	
naslov	varchar(255)		

Tablica 22. Meni. Izrada autora.

Naziv stupca	Tip	Osobine	Opis
id_stavke	integer(255)	pk	
naziv	varchar(255)		

Tablica 23. Stavka. Izrada autora.

Naziv stupca	Tip	Osobine	Opis
meniid_meni	integer(255)	fk	
stavkaid_stavke	integer(255)	fk	

Tablica 24. Meni stavka. Izrada autora.

meni	1..*	meni stavka
id_meni	<->	meniid_meni

Tablica 25. Vanjski ključ meni – meni stavka. Izrada autora.

Aplikacija restorana će imati mogućnost kreiranja više menija, upravo zato i postoji veza jedan na više od tablice meni prema tablici meni stavka koja osigurava da jedan meni može imati više stavki te da određena stavka može biti samo jednom na određenom meniju.

stavka	1..*	meni stavka
id_stavke	<->	stavkaid_stavke

Tablica 26. Vanjski ključ stavka – meni stavka. Izrada autora.

Kao što od tablice meni prema tablici meni stavka postoji veza jedan na više tako i s druge strane od tablice stavka prema tablici meni stavka postoji veza jedan na više koja osigurava da se jedna stavka može nalaziti na više menija odnosno kao što je već spomenuto jedna stavka može se samo jednom nalaziti na određenom meniju.

Naziv stupca	Tip	Osobine	Opis
id_karte	integer(255)	pk	
naslov	varchar(255)		

Tablica 27. Vinska karta. Izrada autora.

Naziv stupca	Tip	Osobine	Opis
vinska_kartaid_karte	integer(255)	fk	
stavkaid_stavke	integer(255)	fk	

Tablica 28. Vinska karta stavka. Izrada autora.

vinska karta	1..*	vinska karta stavka
id_karte	<->	vinska_kartaid_karte

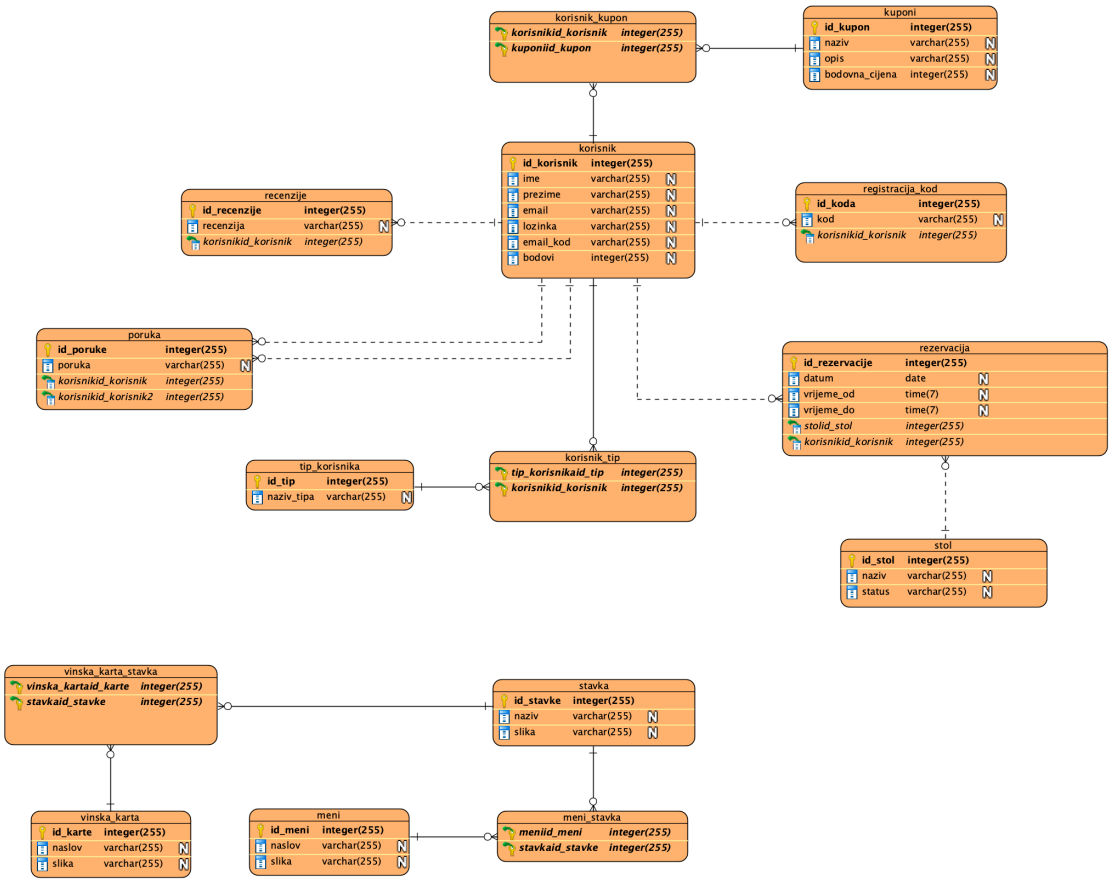
Tablica 29. Vanjski ključ vinska karta – vinska karta stavka. Izrada autora.

stavka	1..*	vinska karta stavka
id_stavke	<->	stavkaid_stavke

Tablica 30. Vanjski ključ stavka – vinska karta stavka. Izrada autora.

Koncept povezanosti tablica vinske karte i tablice stavka je identičan konceptu povezanosti tablice meni i tablice meni stavka. Sažeto rečeno tablice vinska karta i stavka su vezom više na više povezane uz pomoć tablice vinska karta stavka što omogućuje da se više stavki nalazi na više različitih menija ali u isto vrijeme da se jedna stavka samo jednom pojavljuje na određenom meniju.

Na kraju ovoga podnaslova moguće je sve spomenute tablice sa svim njihovim vezama prikazati preko jednostavnog vizualnog prikaza ERA modela koji se može vidjeti na sljedećoj slici.



Slika 12. ERA model. Izrada autora.

6.3.2 Povezivanje te korištenje baze podataka

Kako bi aplikacija bila funkcionalna potrebno je implementirati modele baze podataka. No prije nego se modeli baze podataka implementiraju potrebno je aplikaciju povezati s bazom podataka, a kako bi se to postiglo unutar aplikacijskih postavki moraju se definirati određeni zahtjevi. Laravel sadrži nekoliko elemenata koji se moraju konfigurirati kako bi se uspostavila veza između aplikacije i baze podataka. U ovome slučaju odlučeno je da će se za bazu podataka koristiti sqllite koji se jednostavno kreira uz pomoć jedne naredbe a to je: „touch database/database.sqlite“. To znači da će se unutar datoteke „database“ kreirati dokument s završetkom „.sqlite“ koji simbolizira na dokument za spremanje podataka.

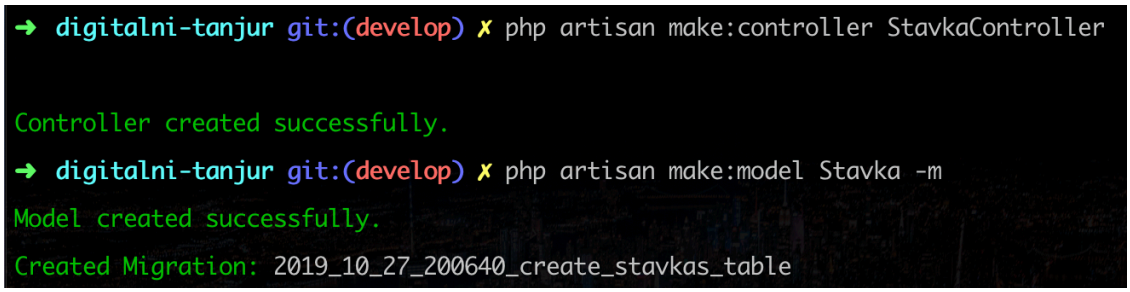
Nadalje sada kada je baza podataka kreirana potrebno je uspostaviti vezu s tom bazom, a to se radi na način da se prvo unutar „.env“ dokumenta definira „DB_CONNECTION“ koji će označiti vrstu baze podataka koja će se koristiti za ovu web aplikaciju.

```
DB_CONNECTION=sqlite
```

Nakon konfiguracije vrste baze podataka potrebno je definirati putanju do dokumenta koji je prethodno kreiran za bazu. Ovaj puta potrebno je otvoriti dokument „database.php“ te unutar „sqlite“ odjeljka postaviti već spomenutu putanju.

```
'connections' => [
        'sqlite' => [
            'driver' => 'sqlite',
            'url' => env('DATABASE_URL'),
            'database' => env('DB_DATABASE',
              database_path('database.sqlite')),
            'prefix' => '',
            'foreign_key_constraints' =>
              env('DB_FOREIGN_KEYS', true),
        ]
];
```

Što se same konfiguracije tiče ništa više nije potrebno uraditi barem kada je u pitanju sqlite baze podataka, u nekim drugim slučajevima više podataka je potrebno definirati kako bi se veza na bazu uspješno ostvarila. Nakon povezivanja potrebno je kreirati tablice, to je moguće napraviti na nekoliko načina: korištenjem nekog od vanjskih programa koji čitaju sqlite dokument te putem sučelja kreiraju tablicu ili putem terminala. U ovome slučaju tablice će se kreirati putem terminala te na isti način će se popunjavati s podacima. Kako bi se tablica kreirala potrebno je kreirati njezin kontrolor te model tablice, a to se postiže uz dvije naredbe (u ovome primjeru kreirati će se tablica stavke koja će kasnije stvarno biti korištena unutar aplikacije): „php artisan make:controller StavkaController“ i „php artisan make:model Stavka -m“ kod druge naredbe „-m“ se koristi kako bi se kreirala jedinstvena migracija koja će u konačnici omogućiti kreiranje tablice Stavka.

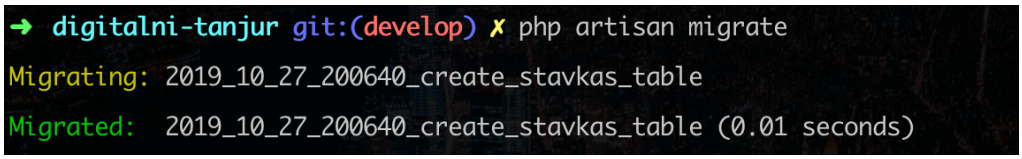


Slika 13. Kreiranje kontrolora, modela/migracije. Izrada autora.

Kada je migracija kreirana moguće je unutar migracijskog dokumenta definirati strukturu tablice, kod same strukture detaljno se mogu definirati redovi tablice te vrsta podataka koja se namjerava unositi u njih. Još je bitno spomenuti da se prilikom kreacije migracijskog dokumenta automatski dodaje datum izrade na početak imena kako bi se migracije mogle razlikovati u slučaju da postoje migracije s sličnim odnosno istim imenima. Na sljedećem isječku koda je moguće vidjeti migracijski dokument tablice stavka koji definira red „id_stavke“ koji će biti tipa „bigIncrements“ što znači da će se prilikom unosa podataka u tablicu to polje brojčani podatak automatski povećavati te red „naziv“ koji je tipa „string“ odnosno tipa koji je namijenjen za unos znakovnih elemenata.

```
public function up()
{
      Schema::create('stavkas', function (Blueprint $table) {
            $table->bigIncrements('id_stavke');
            $table->string('naziv', 255)->unique();
            $table->timestamps();
      });
}
```

Poslije definicije strukture same tablice potrebno je migraciju i migrirati kako bi se tablica kreirala unutar datoteke koja je prethodno kreirana u svrhu baze podataka, a to se čini pomoću jednostavne naredbe „php artisan migrate“.



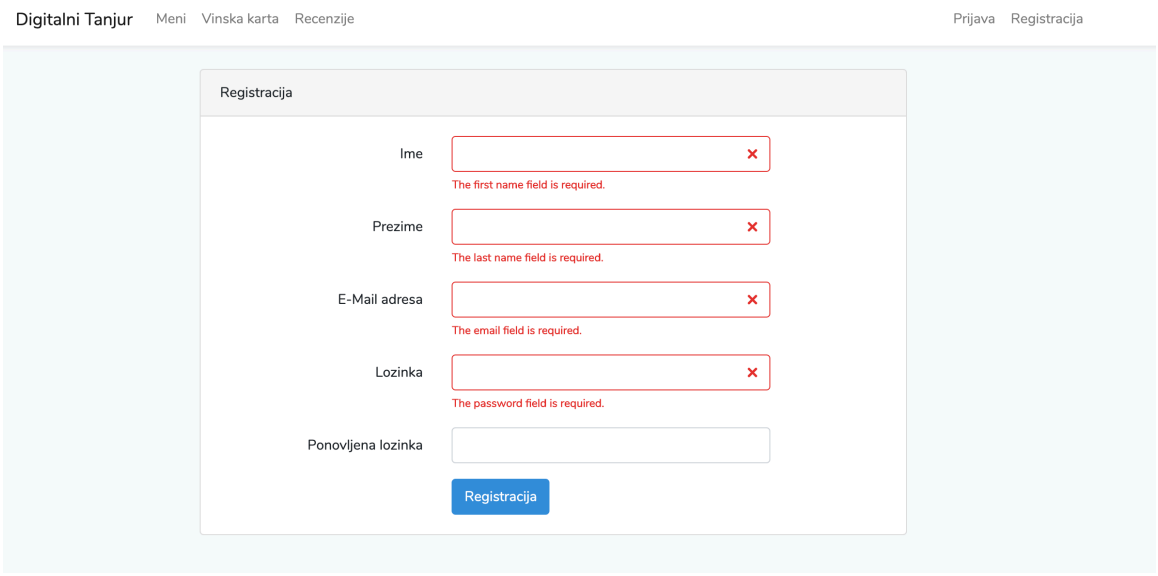
Slika 14. Migriranje prethodno definiranog migracijskog dokumenta. Izrada autora.

Nakon toga tablica je kreirana i spremna za korištenje, a prvo korištenje će biti u svrhu unosa podataka. Unos podatak se može vršiti uz pomoć alata koji je uključen u PHP a zove se „tinker“. Za početak je moguće kreirati novu varijablu „unos“ u koju će se spremiti nova instanca kontrolora stavka koja će moći dodjeljivati vrijednosti novim unosima u tablici te na kraju te vrijednosti spremiti u tablicu. Bitno je napomenuti kako se prilikom kreiranja svakog novog unosa u red tablice mora definirati nova instanca jer inače će se trenutna instanca samo prebrisati te unijeti s podacima iz posljednjeg unosa.

Slika 15. Unos podataka u tablicu uz pomoć tinker-a. Izrada autora.

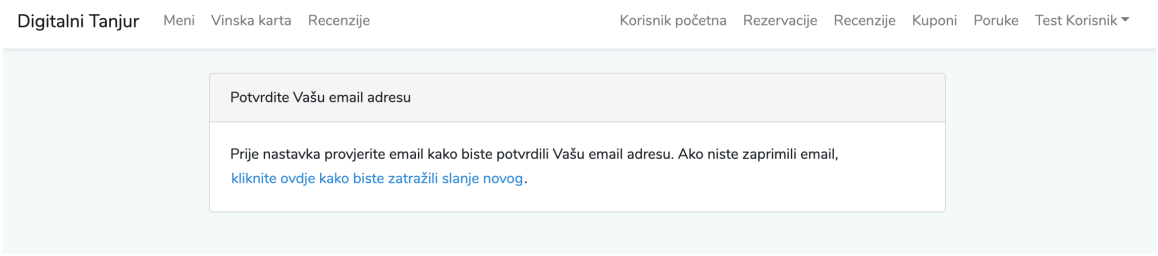
6.3.3 Neregistrirani korisnik

Kod neregistriranog korisnika prvo je bitno spomenuti prijavu i registraciju, sama registracija zahtjeva popunjavanje nekoliko podataka kao što su: ime, prezime, email adresa, lozinka te ponovljena lozinka.



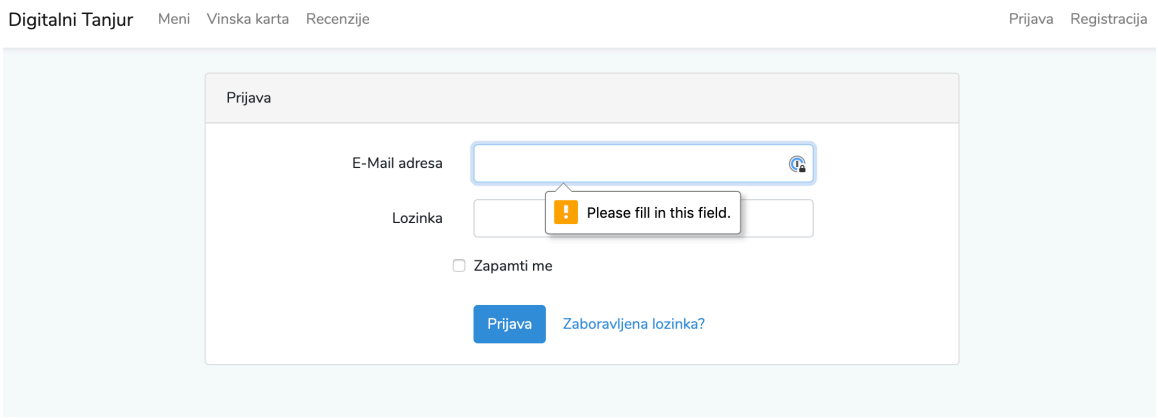
Slika 16. Registracijska forma. Izrada autora.

Osim toga bitno je napomenuti da se kod registracijske forme vrši validacija podataka uz pomoć serverske ali i klijentske strane (AJAX). Nakon popunjavanja registracijske forme korisniku se šalje email putem kojeg mora potvrditi svoju registraciju, taj dio se također reflektira na početnoj korisničkoj stranici koja se prikazuje odmah nakon uspješne registracije.



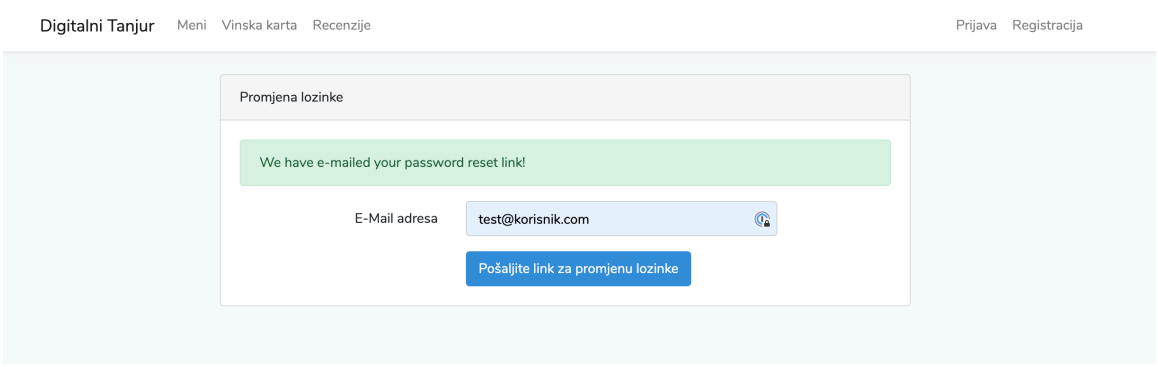
Slika 17. Obavijest o slanju nove potvrde emaila. Izrada autora.

Dok god novoregistrirani korisnik ne potvrdi svoj email on ne može dalje pristupiti ostalim funkcionalnostima stranice. Nadalje neregistrirani korisnik osim registracije u svojim opcijama ima i mogućnost prijave.



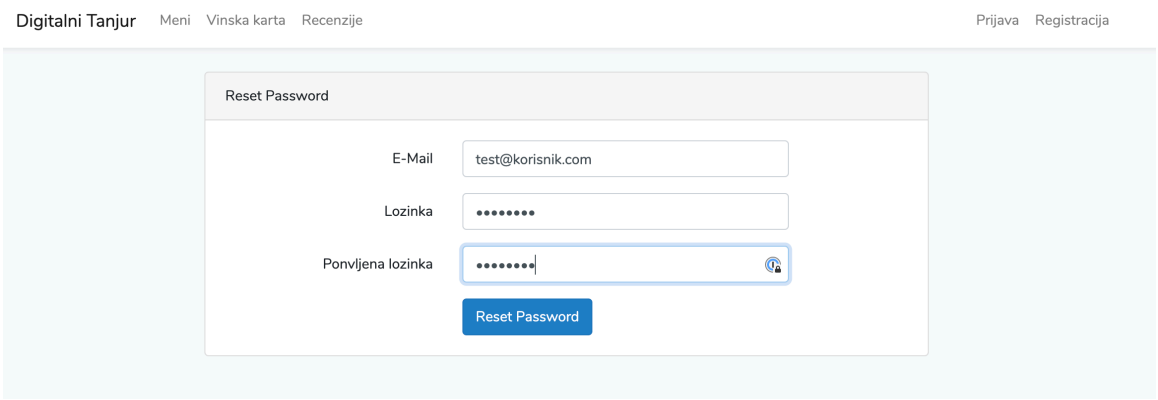
Slika 18. Forma prijave. Izrada autora.

Forma za prijavu jednostavnog je karaktera te od korisnika zahtjeva da ispuni samo email adresu te lozinku kako bi se prijavu. Korisnika je također moguće i zapamtiti tako da se odabere opcija „Zapamti me“. U slučaju da je korisnik zaboravio svoju lozinku, može zatražiti kreiranje nove lozinke.



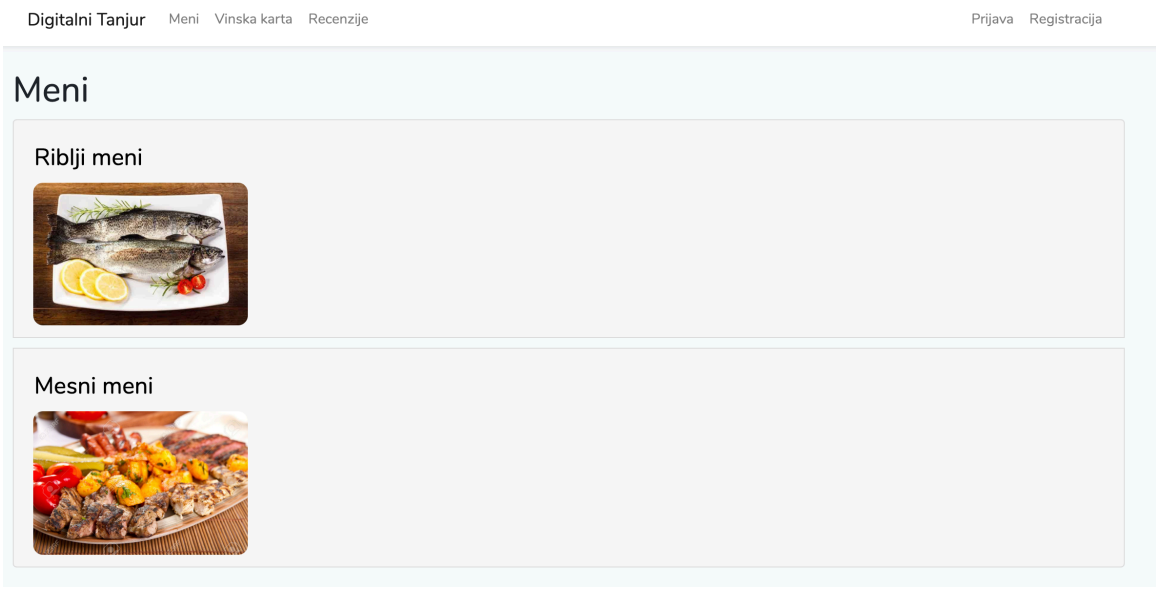
Slika 19. Forma za slanje zahtjeva za promjenu lozinke. Izrada autora.

Korisnik mora samo unijeti email s kojim se registrirao kako bi od aplikacije zaprimio email s linkom koji će ga proslijediti na stranicu gdje će moći unijeti novu lozinku.



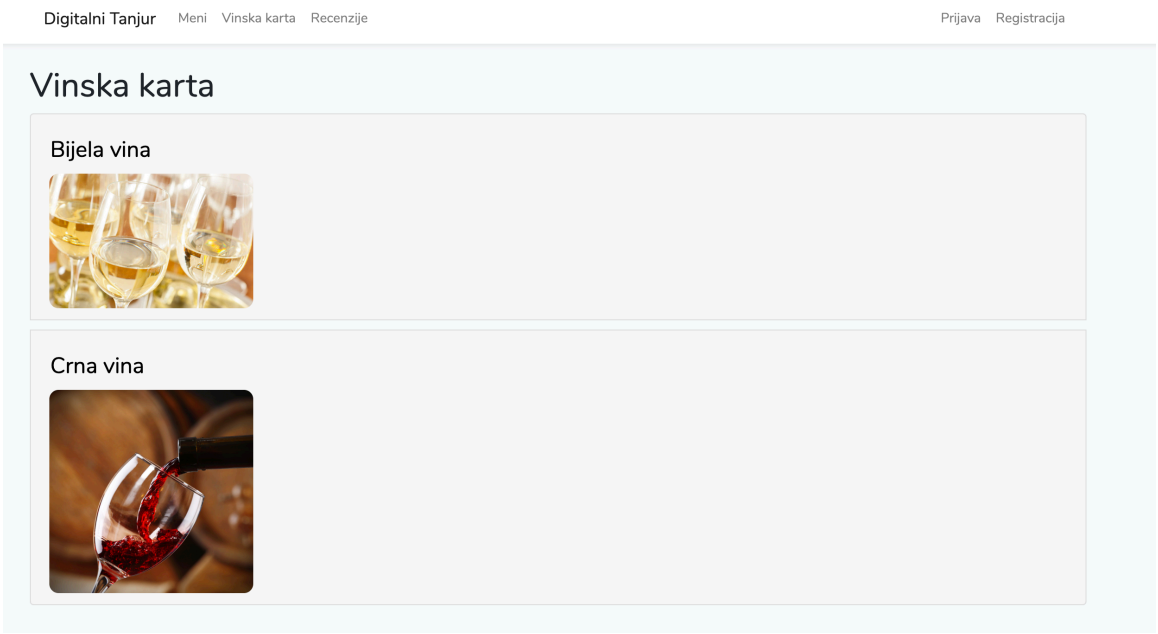
Slika 20. Forma za unos nove lozinke. Izrada autora.

Osim što mora unijeti novu lozinku korisnik mora unijeti još jednom email s kojim se registrirao te ponoviti unesenu lozinku. Neregistrirani korisnik također kao i svi ostali korisnici (moderator, administrator, registrirani korisnik) može pregledavati meni, vinsku kartu te recenzije koje su ostavili ostali korisnici.



Slika 21. Pregled menija. Izrada autora.

Slika 21 prikazuje pregled menija na kojemu se mogu uočiti podaci kao što je naslov stranice, naslov menija te slika koja predstavlja svaki određeni meni.

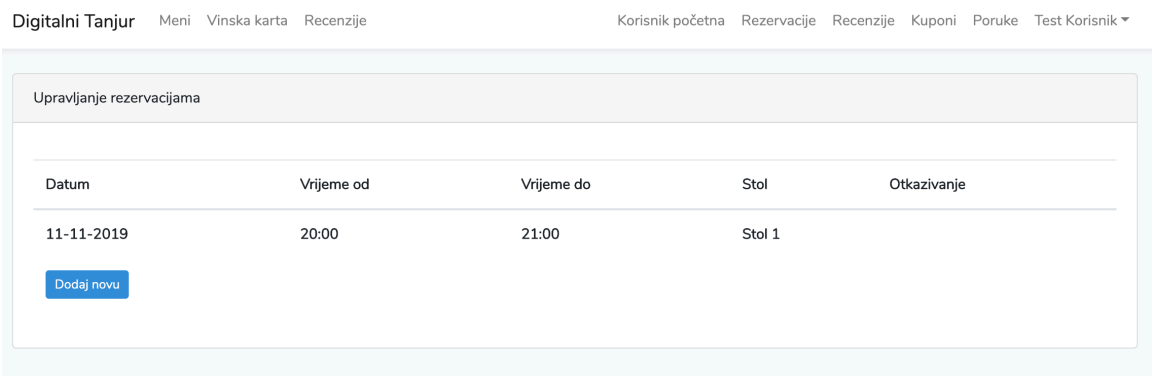


Slika 22. Pregled vinske karte. Izrada autora.

Slika 22 prikazuje pregled vinske karte na kojemu se mogu uočiti isti podaci kao i kod pregleda menija.

6.3.4 Registrirani korisnik

Registrirani korisnik ima pristup svim pregledima kao i neregistrirani korisnik, a osim ti pregleda postoje i korisnički specifične opcije kao što su: rezervacije, recenzije, kuponi i poruke. Za početak promotriti će se opcija rezervacija gdje korisnik odabirom te opcije unutar navigacije odlazi na stranicu (/korisnik/rezervacije) prikazanu na sljedećoj slici.

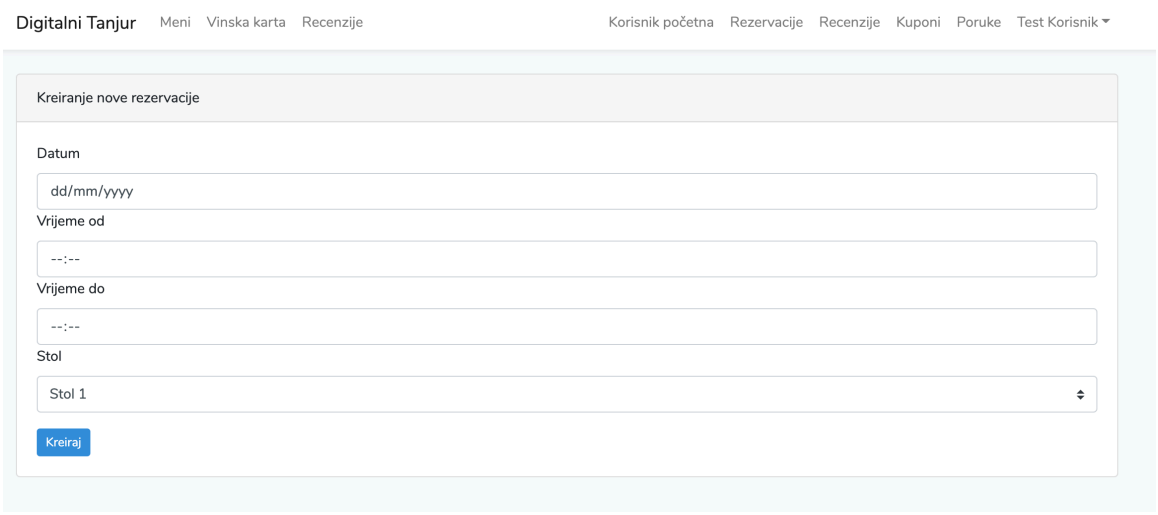


Slika 23. Pregled rezervacija. Izrada autora.

Sami pregled rezervacija se postiže pomoću RezervacijaController.php dokumenta koji kroz funkciju „index“ prosljeđuje korisnika aplikacije na predložak što je vidljivo na sljedećem isječku koda.

```
public function index()
{
      return view('moderator.rezervacije.index')
      ->with('rezervacije', Rezervacija::all())->with('stolovi',
      Stol::all())->with('korisnik', Auth::user()->id);
}
```

Na stranici je vidljivo da se prikazuju podaci o datumu rezervacije, vremenu od kada rezervacija vrijedi, vremenu do kada rezervacija vrijedi te nazivu stola koji je rezerviran. Ispod pregleda informacija o rezervaciji, nalazi se gumb koji nosi naziv „Dodaj novu“ te se koristi u svrhu dodavanja nove rezervacije. Klikom na gumb korisnik odlazi na stranicu sa putanjom: /korisnik/rezervacije/create.



Slika 24. Pregled forme za kreiranje nove rezervacije. Izrada autora.

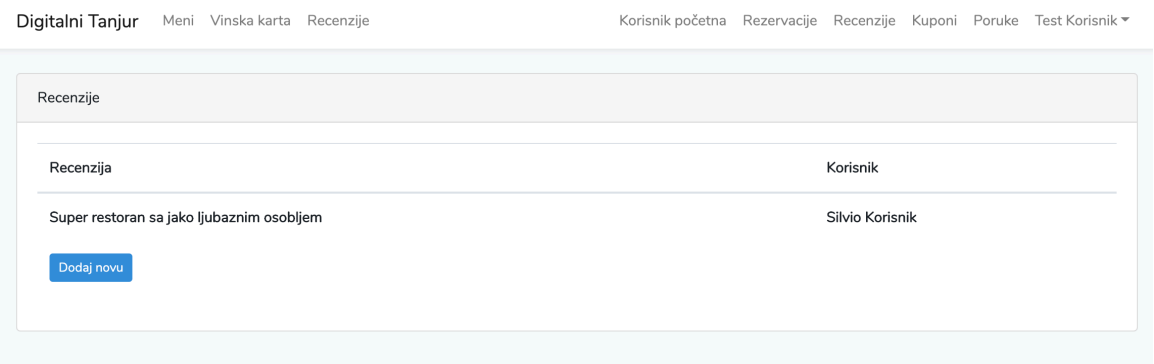
Forma za kreiranje nove rezervacije od korisnika zahtjeva unos četiri podatka a to su datum kada želi napraviti rezervaciju, vrijeme kada rezervacija započinje, vrijeme kada rezervacija završava te stol koji želi rezervirati. Nakon unosa podataka vrši se nekoliko provjera gdje je prva od njih provjera da li je stol rezerviran u odabrano vrijeme.

```
foreach($rezervacije as $rezervacija) {
      if ($rezervacija->datum == $request->datum && $rezervacija
      ->vrijeme == $request->vrijeme) {
            return redirect()
            ->route('korisnik.rezervacije.index')->with('poruka',
            'Odabrani stol je rezerviran u to vrijeme!');
      }
}
```

Osim provjere da li je stol rezerviran u određeno vrijeme potrebno je provjeriti da li se datum ili vrijeme rezervacije nalaze u prošlosti, da li je vrijeme početka rezervacije kasnije od vremena kraja rezervacije, da li je raspon između ta dva vremena veći od dva te da li je odabrani stol blokiran/rezerviran od strane moderatora. Sve to se postiže na način prikazan u sljedećem isječku koda. Također bitno je naglasiti da se za svaki ne ispunjeni uvjet korisnika vraća na početnu stranicu uz poruku koja ga informira što je krivo unio.

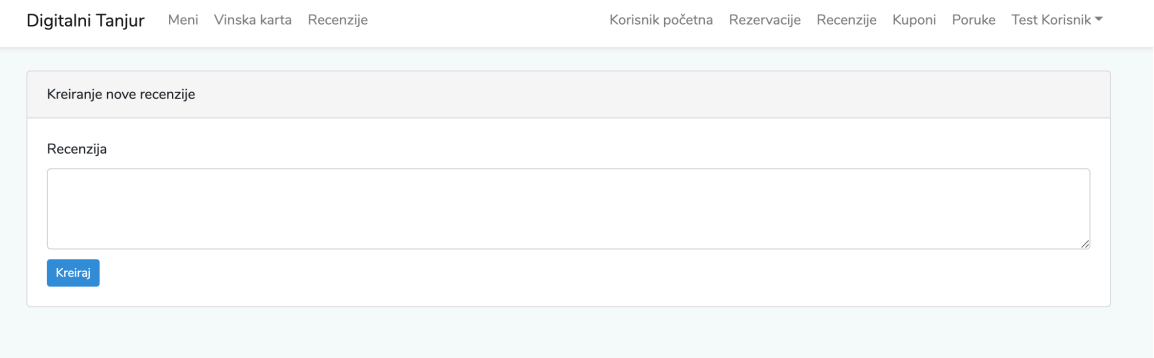
```
if ($request->datum > $current) {
      if ($stol->status == 'Slobodan') {
            $korisnikData = User::find(Auth::user()->id);
            if ($request->vrijemeDo > $request->vrijemeOd) {
                  if ((strtotime($request->vrijemeDo)
                  - strtotime($request->vrijemeOd))/3600 <= 2) {

                        $novaRezervacija = Rezervacija::create([
                              'datum' => $request->datum,
                              'vrijeme_od' => $request->vrijemeOd,
                              'vrijeme_do' => $request->vrijemeDo,
                              'stol_id' => $request->stol,
                              'korisnik_id' => Auth::user()->id
                          ]);
                        $korisnikData->bodovi =
                        $korisnikData->bodovi + 35;
                        $korisnikData->save();
                  } else {
                        return redirect()
                        ->route('korisnik.rezervacije.index')
                        ->with('poruka', 'Rezervacija ne može
                        trajati više od 2 sata');
                  }
            } else {
                  return redirect()
                  ->route('korisnik.rezervacije.index')
                  ->with('poruka', 'Vrijeme od veće od vremena do
                  što nije ispravan unos');
            }
      } else {
            return redirect()
            ->route('korisnik.rezervacije.index')
            ->with('poruka', 'Odabrani stol je pod rezervacijom od
            strane administratora odaberite neki drugi stol!');
       }
} else {
      return redirect()->route('korisnik.rezervacije.index')
      ->with('poruka', 'Datum mora biti današnji ili kasniji!');
}
```

Recenzije su također jedna od opcija koje registrirani korisnik može odabrati a nalaze se na stranici sa putanjom /korisnik/recenzije. Kao i kod rezervacija korisnik na ovoj stranici može vidjeti sve recenzije koje su ostali korisnici ostavili prethodno unutar aplikacije.



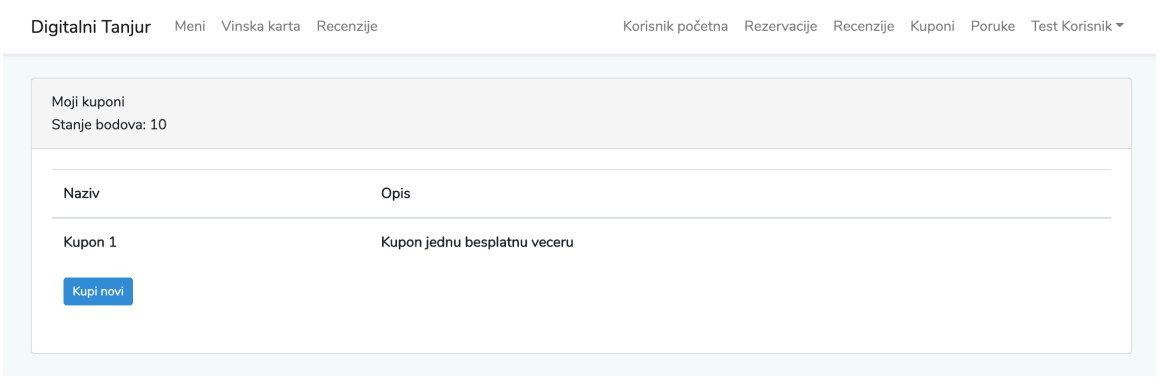
Slika 25. Pregled recenzija. Izrada autora.

Nakon pregleda, klikom na gumb „Dodaj novu“ korisnik odlazi na stranicu koja u sebi sadrži jednostavnu formu koja se koristi za ostavljanje recenzije, sama forma ne sadrži nikakvu validaciju te bilo koji unos će se potvrditi i spremiti unutar baze podataka.



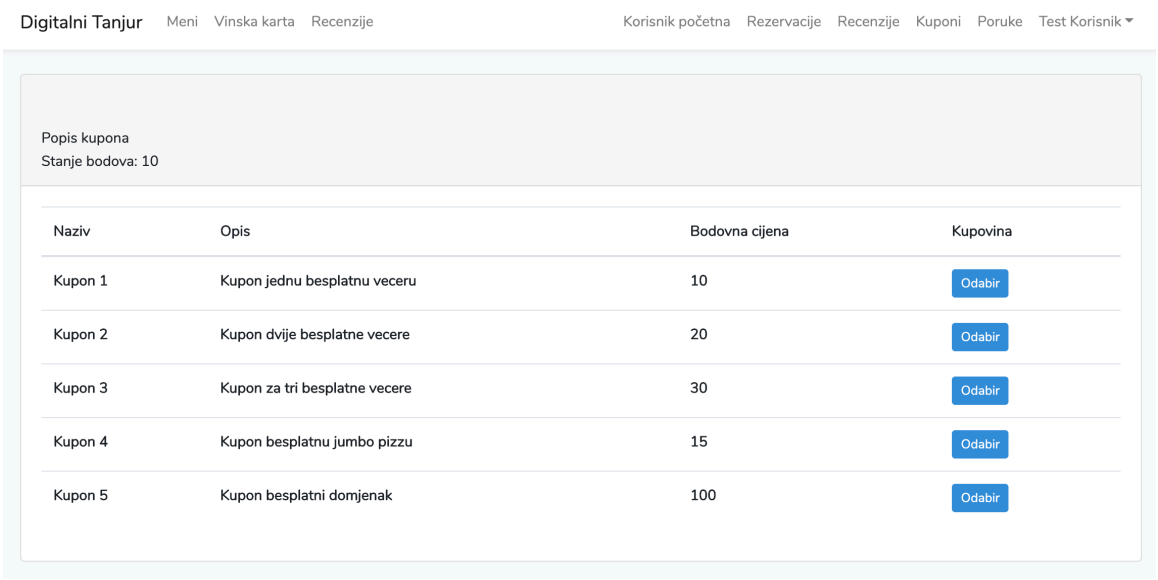
Slika 26. Pregled forme za kreiranje nove recenzije. Izrada autora.

Za svaku ostavljenu recenziju korisnik kao nagradu dobiva 20 bodova koji mu se pribrajaju u ukupnu sumu bodova koju koristi za kupovinu kupona. Pa tako odabirom opcije kuponi korisnik odlazi na stranicu (/korisnik/kuponi) kupona koja sadrži sve kupone koje korisnik ima „kupljene“.



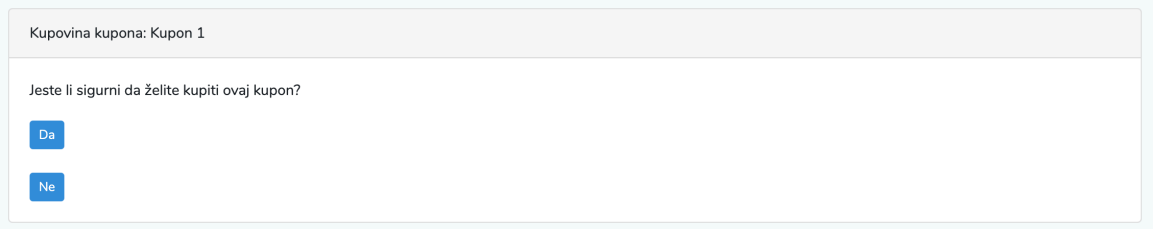
Slika 27. Pregled kupona koje je korisnik prethodno kupio. Izrada autora.

Stranica kupona daje informacije o kuponima a to su informacije o nazivu, o opisu kupona te informacije o trenutnom stanju bodova korisnika. Osim toga na kraju popisa kupona nalazi se gumb „Kupi novi“ koji korisnika vodi na stranicu sa putanjom /korisnik/popisKupona.



Slika 28. Pregled kupona dostupnih za kupovinu. Izrada autora.

Popis kupona koji su dostupni za kupovinu prikazuje naziv kupona, opis kupona te bodovnu cijenu. U slučaju da korisnik želi kupiti neki kupon dovoljno je napraviti klik na gumb „Odabir“ u pripadajućem redu kupona. U tome slučaju korisnik će biti preusmjeren na stranicu gdje će mu se ponuditi opcija prihvaćanja ili odbijanja kupovine.

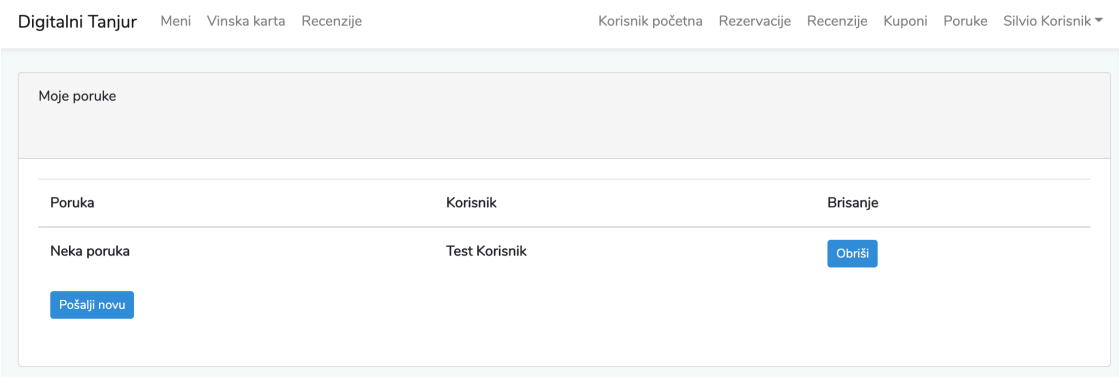


Slika 29. Potvrda kupovine kupona. Izrada autora.

U slučaju da korisnik odbije kupnju kupona biti će preusmjeren na početnu stranicu koja sadrži popis kupona, a u slučaju da pristane na kupovinu forma će se poslati sa podacima o kuponu (id kupona) te će se kroz funkciju „store“ kreiranu unutar KuponiController.php kreirati novi unos korisnika i kupona koji je kupio. Nadalje, ako korisnik nema dovoljno bodova to će biti provjereno i on neće moći kupiti određene kupone.

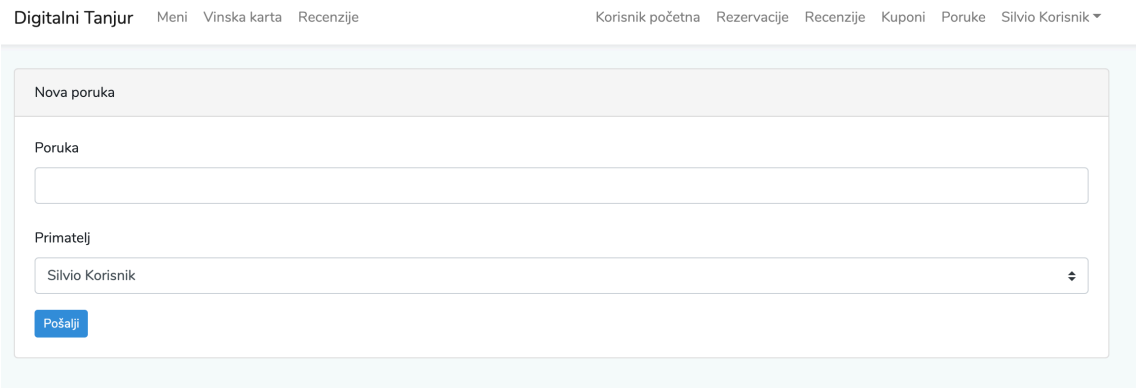
```
public function store(Request $request, $id)
{
      $korisnik = User::find(Auth::user()->id);
      $kupon = Kuponi::find($id);

      if ($korisnik->bodovi >= $kupon->bodovna_cijena) {
            $noviUnos = KorisnikKupon::create([
            'korisnik_id' => $korisnik->id,
            'kupon_id' => $kupon->id,
            ]);
      if ($korisnik->bodovi != 0) {
            $korisnik->bodovi = $korisnik->bodovi - $kupon
            ->bodovna_cijena;
            $korisnik->save();
      }

      return redirect()->route('korisnik.kuponi.popis')
      ->with('poruka', 'Uspješno ste kupili kupon!');
      } else {
            return redirect()->route('korisnik.kuponi.popis')
            ->with('poruka', 'Nemate dovoljno bodova za
      kupovinu!');
      }
}
```

Posljednja opcija koju je potrebno objasniti kod registriranog korisnika je opcija poruka. Naime svaki korisnik unutar aplikacije može sebi ili nekom drugom korisniku poslati poruku te za to dobije 5 bodova. Poruke se također mogu i brisati ali za svaku obrisanu poruku se gubi 5 bodova a ako korisnik nema dovoljno bodova poruke se ne mogu brisati dalje.



Slika 30. Prikaz popisa poruka. Izrada autora.

Odabirom gumba „Pošalji novu“ korisnik odlazi na stranicu sa putanjom /korisnik/poruke/create na kojoj se nalazi forma sa svim potrebnim poljima za unos kako bi se poruka uspješno poslala.



Slika 31. Prikaz forme za slanje poruke. Izrada autora.

Samo spremanje i brisanje poruke vrši se na već standardni način, uz pomoć funkcija koje se nalaze unutar PorukaController.php dokumenta. Na sljedećem isječku moguće je vidjeti funkciju „store“ koja se koristi kako bi se kreirao novi unos poruke unutar baze za određenog korisnika koji šalje te korisnika koji prima poruku.

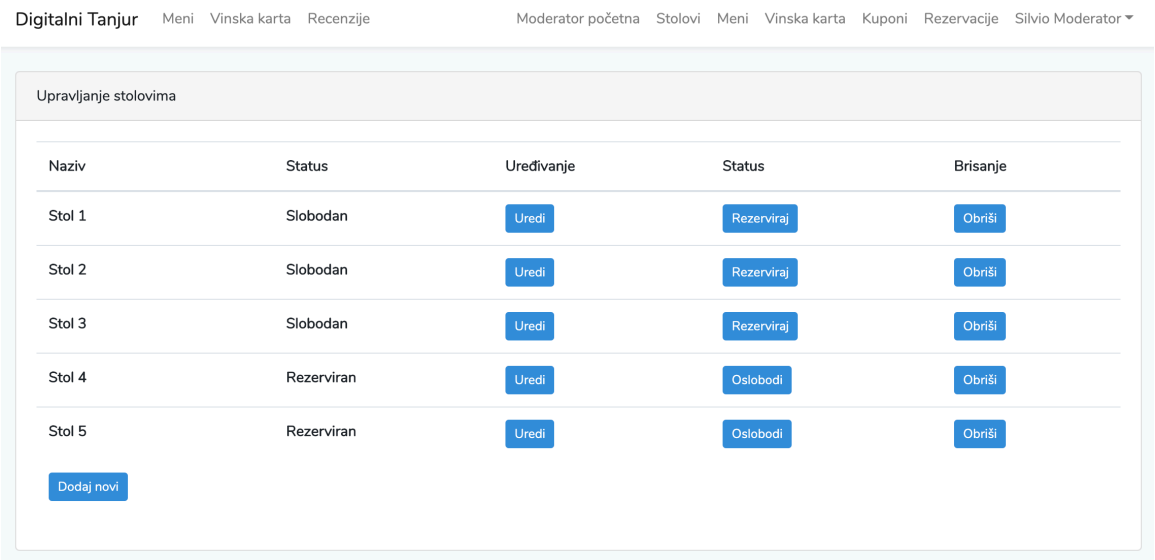
```
public function store(Request $request)
{
	$korisnik = User::find(Auth::user()->id);
	$noviUnos = Poruka::create([
		'poruka' => $request->poruka,
		'primatelj_id' => $request->korisnik,
		'posiljatelj_id' => Auth::user()->id
	]);

	$korisnik->bodovi = $korisnik->bodovi + 5;
	$korisnik->save();

	return redirect()
	->route('korisnik.poruke.index')
	->with('poruka', 'Poruka uspješno poslana!');
}
```

Osim „store“ funkcije bitno je napomenuti da se za brisanje koristi funkcija „delete“ koja se može vidjeti na isječku koda ispod.

```
public function delete(Request $request, $id)
{
	$korisnikData = User::find(Auth::user()->id);
	$poruka = Poruka::find($id);
	$poruka->delete();

	if ($korisnikData->bodovi != 0) {
		$korisnikData->bodovi = $korisnikData->bodovi - 5;
		$korisnikData->save();
	}

	return redirect()->route('korisnik.poruke.index');
}
```

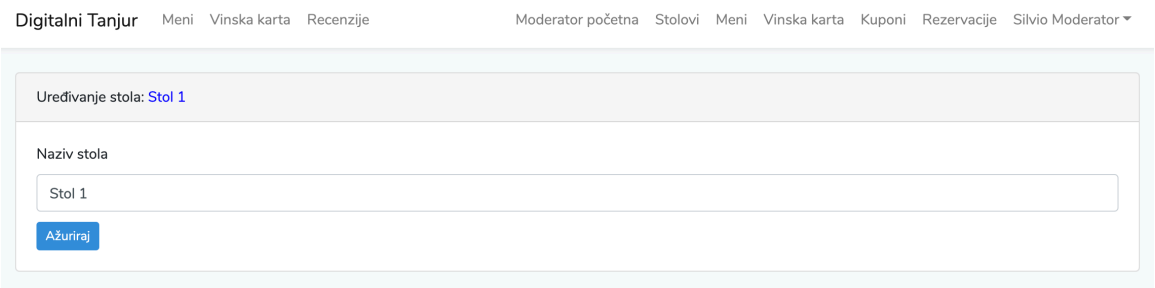
6.3.5 Moderator

Većinu podataka koji su do sada bili prikazani kod korisnika moderator može moderirati unutar svog sučelja. Pa se tako svi podaci o stolovima mogu uređivati na stranici s putanjom /moderator/stolovi.



Slika 32. Prikaz svih stolova. Izrada autora.

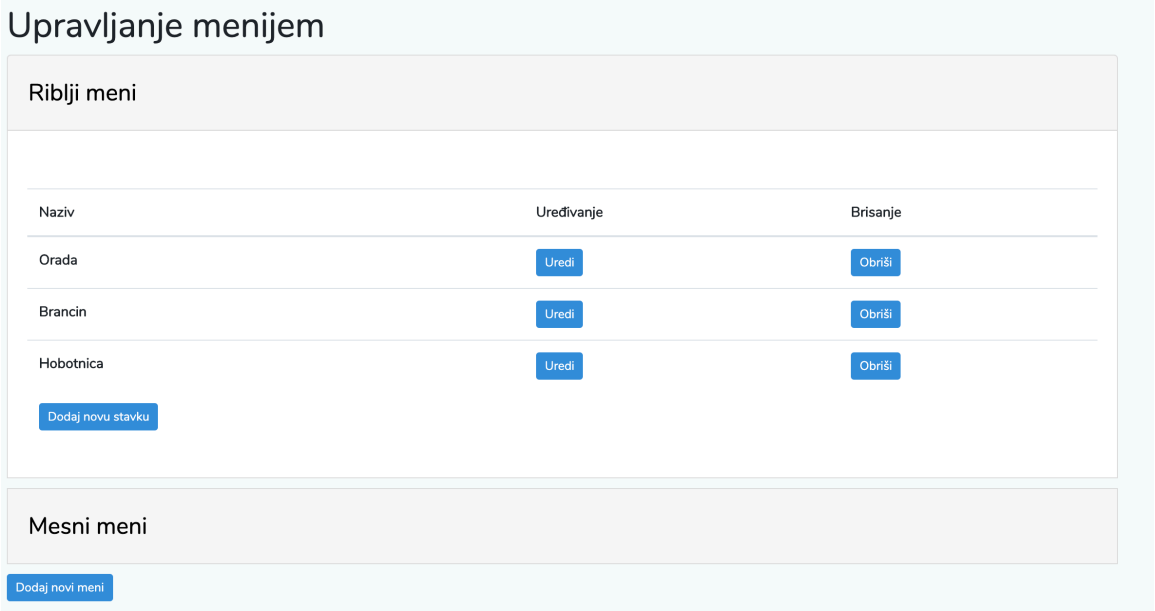
Sa slike 32 je moguće primijetiti da moderator može uređivati naziv stola, mijenjati status stola te brisati stolove. Sama stranica za uređivanje sadrži mogućnost izmjene samo jednog podataka a to je naziv stola.



Slika 33. Prikaz forme za uređivanje naziva stola. Izrada autora.

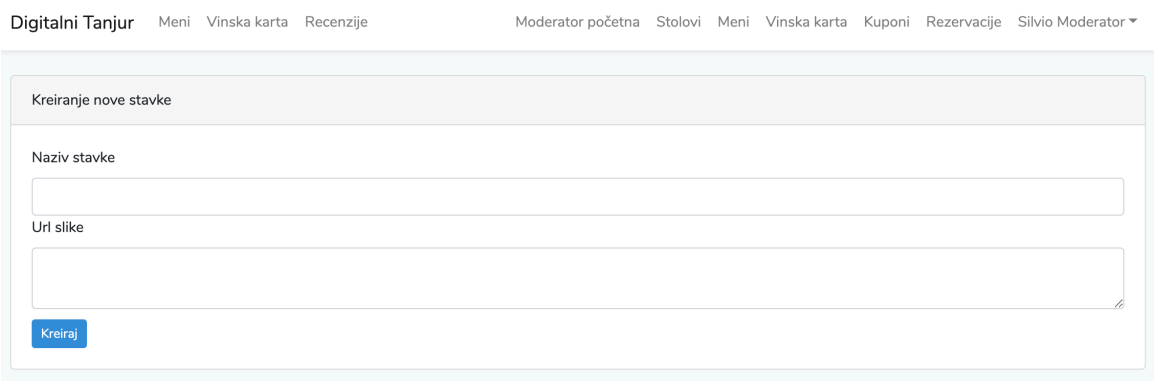
Forma za kreiranje novog stola izgleda identično samo što umjesto funkcije „update“ koja se koristi za promjenu postojećih podataka koristi funkciju „create“ i „store“ kako bi unijela podatke u bazu.

Kao i kod uređivanja stolova iste opcije se pojavljuju osim opcije promjene statusa što možemo vidjeti i na sljedećoj slici.



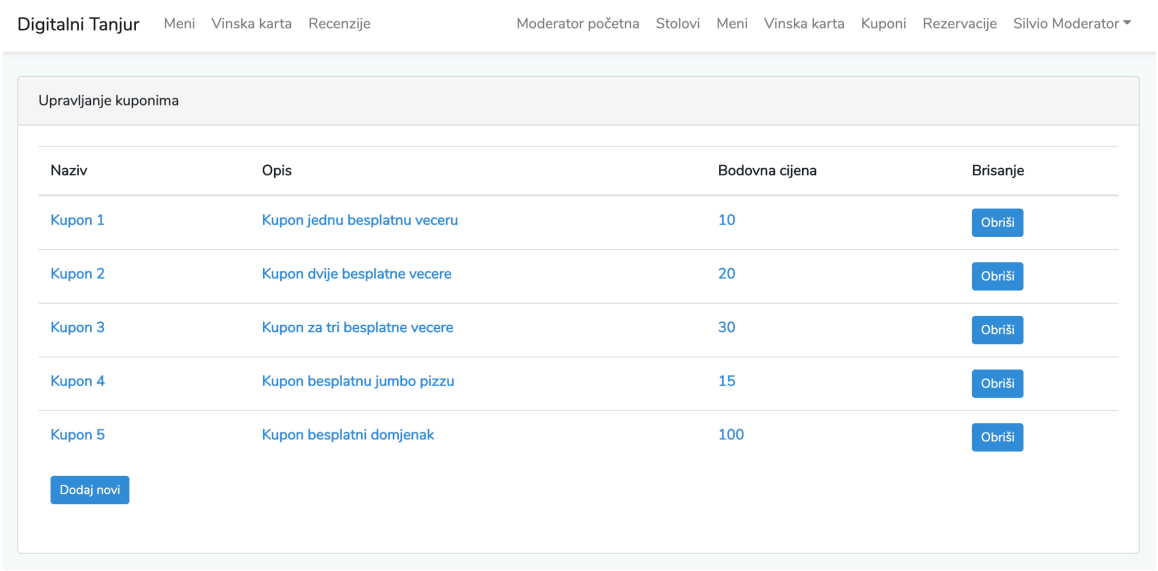
Slika 34. Prikaz stavki prema meniju. Izrada autora.

Bitno je napomenuti da kod opcije dodavanja nove stavke korisnik osim naziva stavke unosi i URL slike koja će prikazati uz tu stavku.



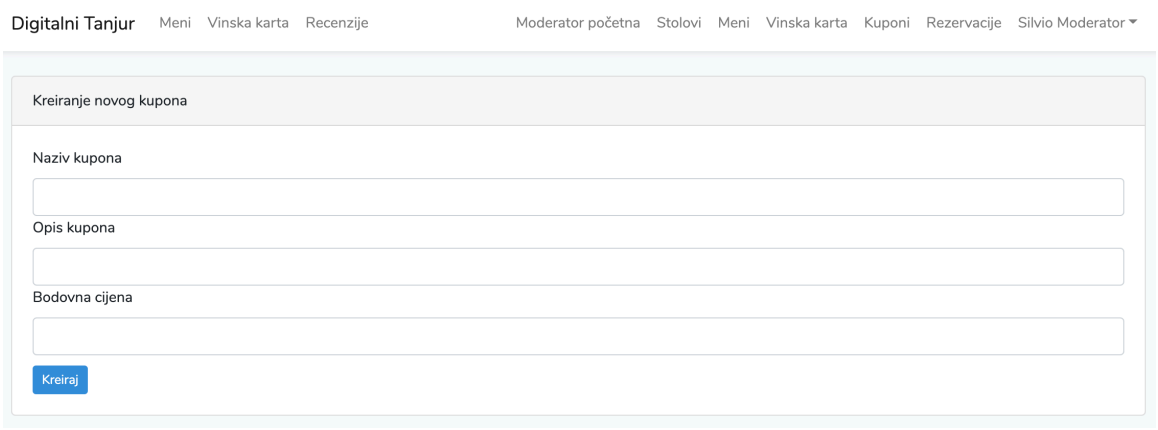
Slika 35. Prikaz forme za kreiranje nove stavke menija. Izrada autora.

Sama vinska karta ima identične opcije kao i meni stoga nema smisla dva puta prikazivati istu stvar. Nego se može odmah prijeći na opciju kupona koje moderator može uređivati. Svi podaci koji se nalaze na stranici s putanjom /moderator/kuponi se mogu individualno mijenjati klikom na sami podatak, osim mijenjanja podataka podatke je moguće obrisati ali i dodati.



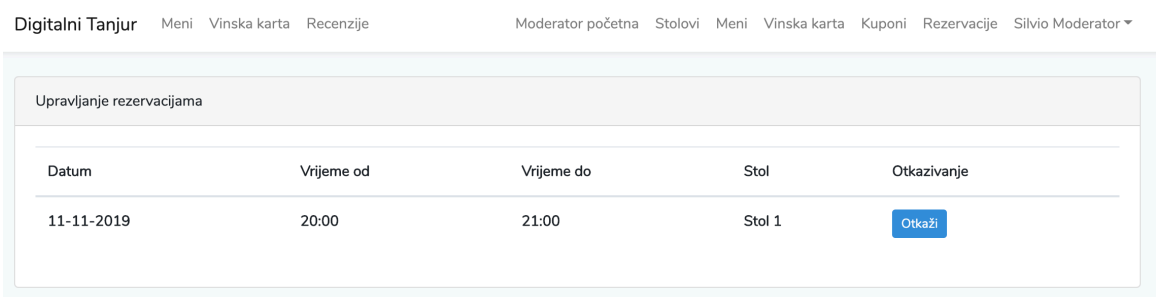
Slika 36. Prikaz svih kupona. Izrada autora.

Dodavanje novog kupona se izvršava na stranici koja ima putanju /moderator/kuponi/create na kojoj se nalazi forma koja zahtjeva unošenje naziva kupona, opisa kupona te bodovne cijene.



Slika 37. Prikaz forme za kreiranje novog kupona. Izrada autora.

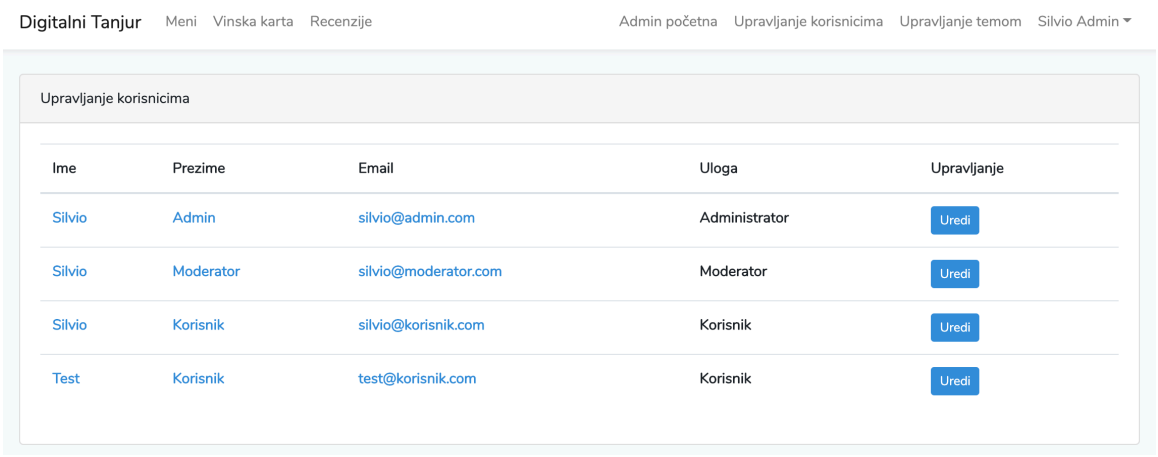
Naposljetku je moguće još navesti da moderator može pregledati sve rezervacije te iste otkazati u slučaju da za to postoji određeni razlog.



Slika 38. Prikaz svih rezervacija. Izrada autora.

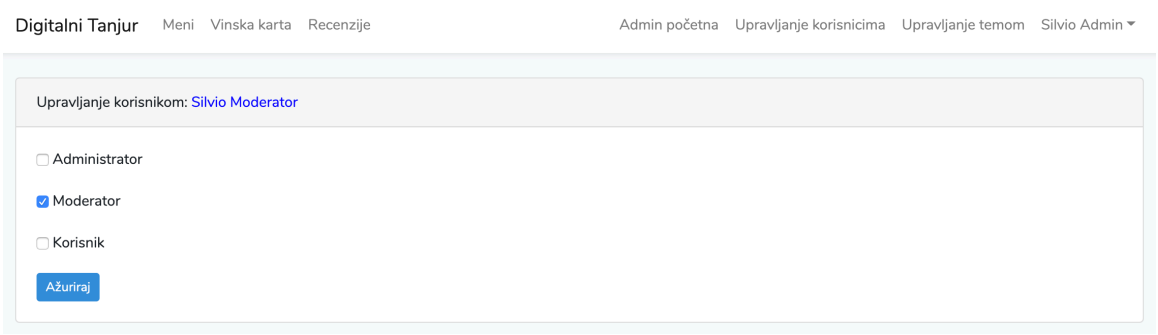
6.3.6 Administrator

Uloga administratora ima samo dvije glavne opcije a to su upravljanje korisnicima te upravljanje temom. Odabirom opcije upravljanja korisnicima kroz navigaciju otvara se stranica sa putanjom /admin/korisnici na kojoj se nalazi popis svih korisnika sa svim njihovim podacima te opcijom uređivanja uloge i opcijom promjene svakog podatka o korisniku (osim lozinke).



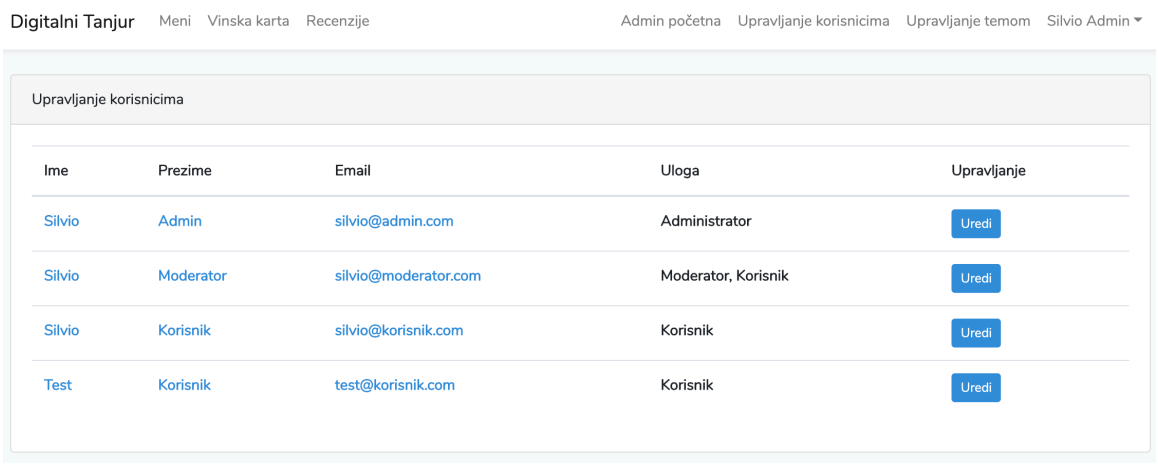
Slika 39. Prikaz svih korisnika. Izrada autora.

Klikom na bilo koji podatak otvara se forma koja sadrži jedno polje za unos uz pomoć kojeg je moguće promijeniti bilo koji podatak o bilo kojem korisniku. No, forma za uređivanje uloge korisnika je malo drugačijeg karaktera nego forme koje su već do sada prikazane stoga će se ona izdvojiti i prikazati.



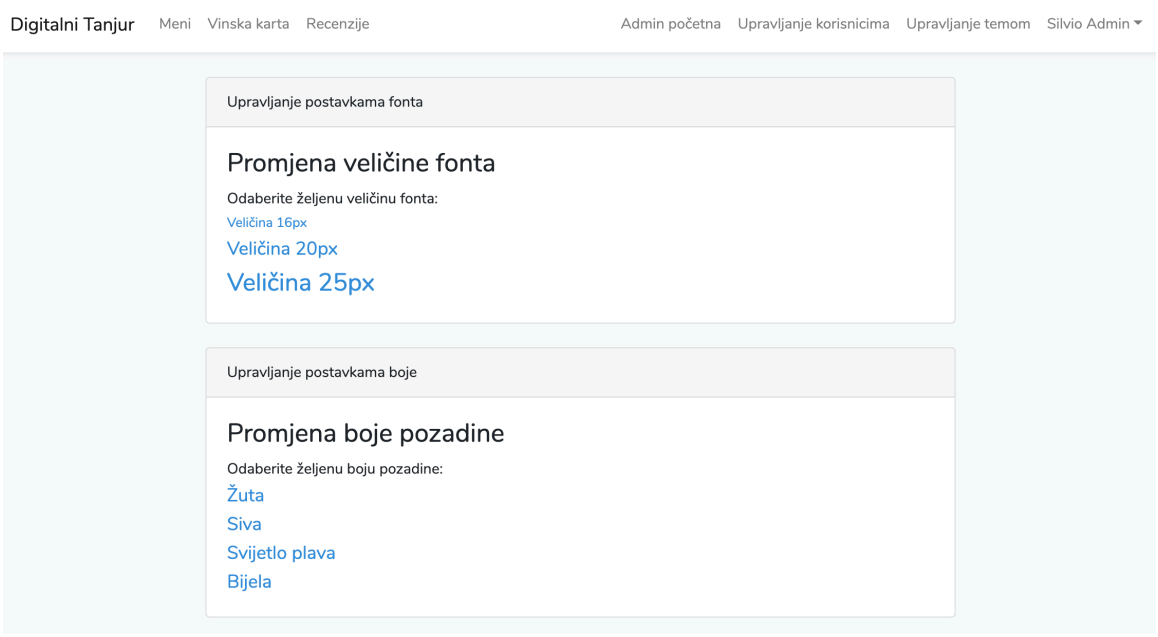
Slika 40. Prikaz forme za uređivanje uloge korisnika. Izrada autora.

Ako se korisniku „Silvio Moderator“ promjeni da može u isto vrijeme biti i korisnik i moderator te će se reflektirati na stranici popisa korisnika.



Slika 41. Prikaz popisa svih korisnika s izmjenom podataka. Izrada autora.

Posljednja opcija koju je bitno napomenuti je opcija upravljanja temom u čijem se slučaju trebaju javiti promjena veličine fonta ili promjena boje pozadine. Klikom na bilo koju veličinu fonta, font se mijenja na svim stranicama kao i boja pozadine ovisno o odabiru, a to je moguće jer se svi podaci koje korisnik odabere spremaju u kolačić (eng. cookie) i primjenjuju na globalnoj razini.



Slika 42. Prikaz opcija za uređivanje svih stranica aplikacije. Izrada autora.

Zaključak

Kroz proces izrade web aplikacije uz pomoć programskog okvira Laravel mogu se izvući razni zaključci a jedan od njih je da je Laravel jako koristan programski okvir jer prilikom svoje instalacije postavlja cijelu strukturu dokumenata. Sama struktura je smislena te nakon samo kratkog vremena korištenja lako se može snaći unutar nje. Nadalje Laravel nudi postavljanje registracije i prijave uz doslovno jednu liniju koja se poziva unutar terminala te koja generira sve što je potrebno kako bi se dobila funkcionalna registracija i prijava. Kreiranje kontrolora, modela, tablica odnosno migracija i seeder-a je također lako shvatiti nakon nekoliko uspješno odrađenih primjera. Samo komuniciranje s bazom je također prilično bezbolan proces s obzirom da je potrebno izmijeniti nekoliko podataka kako bi se napravila veza na bazu dok se kreiranje tablica vrši putem migracija. Popunjavanje tablica se također može napraviti uz pomoć seeder-a koji se mogu pokrenuti svaki puta prilikom kreiranja svježe migracije. Ako se u obzir uzmu svi navedeni faktori moguće je donijeti zaključak da bi se za implementaciju ovakve aplikacije potrošilo dvostruko ako ne i trostruko više vremena kada bi se radila bez programskog okvira kao što je to Laravel. Osim vremena potrebno je uzeti i faktore organizacije projekta te praktičnosti implementacije.

Zaključak koji se na kraju može donijeti je taj da je sami programski jezik PHP jako moćan i omogućuje razne varijacije implementiranja a postaje još moćniji ako ga se koristi kroz neki njegov programski okvir kao što je to Laravel. Ako bi se trebao preporučiti Laravel je zasigurno opcija koja se preporuča za implementaciju tipa aplikacije koja je kreirana u sklopu ovoga rada.

Literatura

1. Arnold K., Gosling J., Holmes D. (2005): *The Java Programming Language, Fourth Edition*. Addison Wesley Professional. Izvor: http://etf.beastweb.org/index.php/site/download/Java_Programming.pdf, posjećeno 26. srpnja 2019.
2. Aston B. (2015): *A brief history of JavaScript*. Izvor: https://medium.com/@benastontweet/lesson-1a-the-history-of-javascript-8c1ce3bffb17, posjećeno 05. srpnja 2019.
3. Aston B. (2015): *A brief history of JavaScript.* Izvor: https://medium.com/@benastontweet/lesson-1a-the-history-of-javascript-8c1ce3bffb17, posjećeno 20. srpnja 2019.
4. Banga S. (2019): *Web Application Arhitecture: Definitions, Models, Types and More*. Izvor: https://hackr.io/blog/web-application-architecture-definition-models-types-and-more, posjećeno 07. srpnja 2019.
5. Bean M. (2015): *Laravel Essentials.* Packt Publishing. Birmingham, UK. Izvor: https://books.google.hr/books?hl=en&lr=&id=BWO4CAAAQBAJ&oi=fnd&pg=PP1&dq=Laravel&ots=8a_aRrBMVP&sig=qYnGbPnRCgUAfI86PlyjiTW_GI8&redir_esc=y#v=onepage&q=Laravel&f=false, posjećeno 19.rujna 2019.
6. Cass S. (2018): *The 2018 top programming languages.* Izvor: https://spectrum.ieee.org/at-work/innovation/the-2018-top-programming-languages, posjećeno 12. kolovoza 2019.
7. Darveau P. F. (2018): *You should learn Vanilla JavaScript before JavaScript frameworks.* Izvor: https://hackernoon.com/you-should-learn-vanilla-javascript-before-js-frameworks-88fb727ab362, posjećeno 30. srpnja 2019.
8. Developer mozilla (2019): *CSS*. Izvor: https://developer.mozilla.org/en-US/docs/Web/CSS, posjećeno 16. srpnja 2019.
9. Downey B. A. (2013): Naučite Python. Dobar Plan, Zagreb.
10. Filipova O. (2016): *Learning Vue.js 2.* Packt Publishing, Birmingham, UK. Izvor: https://books.google.hr/books?hl=en&lr=&id=nszcDgAAQBAJ&oi=fnd&pg=PP1&dq=vue+js&ots=9nKjGhSkpM&sig=CUAOfuUm-kPrT6kD1Sv6BJItxHk&redir_esc=y#v=onepage&q=vue%20js&f=false, posjećeno 02. kolovoza 2019.

11. Flanagan D. (2006): *JavaScript: The Definitive Guide, Fifth Edition.* O'Reilly Media, Inc. California. Izvor: https://books.google.hr/books?hl=en&lr=&id=k0CbAgAAQBAJ&oi=fnd&pg=PT6&dq=javascript&ots=O3rzkmfwvZ&sig=TiPOu_jFe4__mgYcMVmN430V9BY&redir_esc=y#v=onepage&q=javascript&f=false, posjećeno 19. srpnja 2019.
12. Hejlsberg A, Wiltamuth S, Golde P (2006): *The C# Programmin Language.* Addison Wesley Professional. Izvor: https://books.google.com/books?hl=en&lr=&id=6L1Rm031qCkC&oi=fnd&pg=PA3&dq=c%23+programming&ots=5wM29CW3uH&sig=ITjeKi8D2baSb7z4aydg4C2mkLo, posjećeno 27. kolovoza 2019
13. Hoffmann J. (2017): *Flash And Its History On The Web*. Izvor: https://thehistoryoftheweb.com/the-story-of-flash/, posjećeno 05. srpnja 2019.
14. Johari A. (2019): *What is Java? A beginners guide to Java and its evolution.* Izvor: https://www.edureka.co/blog/what-is-java/, posjećeno 28. srpnja 2019.
15. Julien J. (2018): *Web Application Performance: 7 Common Problems and How to Solve Them*. Izvor: https://stackify.com/web-application-problems/, posjećeno 09. srpnja 2019.
16. Keycdn (2019): *Pinpoint website performance issues.* Izvor: https://www.keycdn.com/support/pinpoint-website-performance-issues, posjećeno 09. srpnja 2019.
17. Kolce J., Kroger M., Curic I., Saeed S., Mott J., Green David M., Buckler C. (2018): *JavaScript: Best Practice*. SitePoint Pty. Izvor: https://books.google.hr/books?id=2iReDwAAQBAJ&pg=PT30&lpg=PT30&dq=javascript+saeed&source=bl&ots=_tngbnE7Ef&sig=ACfU3U1nvp-MMbx8n0dgo5zXRYoJPWf0wA&hl=en&sa=X&ved=2ahUKEwjZ1aDw1tPkAhXDIlsKHaOpDCgQ6AEwCHoECAkQAQ#v=onepage&q=javascript%20saeed&f=false, posjećeno 19. srpnja 2019.
18. Kovačević, Ž. (2004): *C++ analiza i primjena*. Školska knjiga, Zagreb.
19. Kuprenko V. (2018): *GRUNT vs GULP vs Webpack: An elaborate comparison of tools.* Izvor: https://scotch.io/@VitalyKuprenko/grunt-vs-gulp-vs-webpack-an-elaborate-comparison-of-tools, posjećeno 10. kolovoza 2019.
20. Laravel (2019): *Release notes*. Izvor: https://laravel.com/docs/5.8/releases, posjećeno 19. kolovoza 2019.
21. Meyer A. E. (2011): *Smashing CSS*. Kompjuter biblioteka, Beograd.

22. Microsoft (2015): Introduction to the C# language and .Net Framework. Izvor: https://docs.microsoft.com/en-us/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework, posjećeno 27. kolovoza 2019.
23. Nakov S., Kolev V. i ost. (2013): Fundamentals of computer programming with C#. Introprogramming, Sofia. Izvor: https://www.introprogramming.info/wp-content/uploads/2013/07/Books/CSharpEn/Fundamentals-of-Computer-Programming-with-CSharp-Nakov-eBook-v2013.pdf, posjećeno 27. kolovoza 2019.
24. Prinz P., Kirch-Prinz U. (2005): *A complete guide to programming in C++.* Jones and Barlett Publishers, Sudbury. Izvor: https://books.google.hr/books?hl=en&lr=&id=-yhuY0Wg_QcC&oi=fnd&pg=PA2&dq=Kirch-Prinz+c%2B%2B&ots=BtNF-fBUNR&sig=1N-2I7gtKEqEjnOvU5FxJ2sg5o8&redir_esc=y#v=onepage&q=Kirch-Prinz%20c%2B%2B&f=false, posjećeno 12. kolovoza 2019.
25. Prokofyeva N., Boltunova V. (2017): Analysis and practical application of PHP frameworks in development of web information systems. Izvor: https://www.sciencedirect.com/science/article/pii/S1877050917300601, posjećeno 10. rujna 2019.
26. Prusty N. (2015): *Learning ECMAScript 6.* Packt Publishing, Birmingham UK. Izvor: https://books.google.hr/books?hl=en&lr=&id=9O13CgAAQBAJ&oi=fnd&pg=PP1&dq=ecmascript+6&ots=JLT5KiAlyq&sig=wt1I0rgQeOQ25mCnF3Yoxa9gwf4&redir_esc=y#v=onepage&q=ecmascript%206&f=false, posjećeno 30. srpnja 2019.
27. Python (2019): What is Python: Executive summary. Izvor: https://www.python.org/doc/essays/blurb/, posjećeno 14. kolovoza 2019.
28. Robberts H. (2018): *Network performance*. Izvor: https://csswizardry.com, posjećeno 09. srpnja 2019.
29. Robbins Niederst J. (2012): *Learning Web Design; A beginners guide to HTML, CSS, JavaScript and Web Graphics*. O'Reily Media, Inc., California. Izvor: https://books.google.hr/books?id=A-tltyafYmEC&printsec=frontcover#v=onepage&q&f=false, posjećeno 15. Srpnja 2019.
30. Samra J. (2015): Comparing performance of plain PHP and four of its popular frameworks. Izvor: http://www.diva-portal.org/smash/get/diva2:846121/FULLTEXT01.pdf, posjećeno 10. rujna 2019.
31. SASS (2019): *SASS/SCSS*. Izvor: https://sass-lang.com/, posjećeno 16. srpnja 2019.
32. Stackoverflow (2019): *Developer Survey Results 2019.* Izvor: https://insights.stackoverflow.com/survey/2019, posjećeno 20. srpnja 2019.

33. Stuttard D., Pinto M. (2011): *The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws, Second Edition.* John Wiley & Sons, Inc., Indianapolis, Indiana. Izvor: https://books.google.hr/books?hl=en&lr=&id=NSBHAAAAQBAJ&oi=fnd&pg=PT13&dq=web+application+security&ots=5spS_zH7CP&sig=Af5hIRfC7WdS_FX9Trs1pS_VDFA&redir_esc=y#v=onepage&q=web%20application%20security&f=false, posjećeno 09. srpnja 2019.
34. Tatroe K., MacIntyre P., Lerdorf R. (2015): *Programiranje PHP, prijevod trećeg izdanja.* Dobar Plan, Zagreb.
35. Technopedia (2019): *Java*. Izvor: https://www.techopedia.com/definition/3927/java, posjećeno 27. srpnja 2019.
36. Tomcy J. (2015): *History of Web Applications*. Izvor: https://www.javacodebook.com/2015/03/31/history-of-web-application/, posjećeno 05. srpnja 2019.
37. Uryutin O. (2018): *A brief history of web app*. Izvor: https://medium.com/@aplextor/a-brief-history-of-web-app-50d188f30d, posjećeno 05. srpnja 2019.
38. Vue.js (2019): *Vue.js.* Izvor: https://vuejs.org/, posjećeno 02. kolovoza. 2019.
39. W3School (2019): *HTML*. Izvor: https://www.w3schools.com/html/, posjećeno 15. srpnja 2019.