Web stranica za naručivanje usluga kozmetičkog salona

Kulić, Filip

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij

WEB STRANICA ZA NARUČIVANJE USLUGA KOZMETIČKOG SALONA

Završni rad

Filip Kulić

Osijek, 2016

SADRŽAJ

1. UVOD 1
1.1. Zadatak završnog rada 1
2. WEB STRANICA 2
2.1. HTML 3
2.1.1. HTML oznake 4
2.2. CSS 7
2.2.1. CSS sintaksa i selektori 7
2.2.2. CSS boje 9
2.2.3. CSS model pravokutnika 9
2.3. JavaScript 10
2.3.1. JavaScript varijable 10
2.3.2. JavaScript funkcije i događaji 11
2.4. Bootstrap 12
2.4.1. Responzivni dizajn 13
2.4.2. Grid system 13
3. WEB APLIKACIJA 15
3.1. AngularJS i TypeScript 15
3.2. Postavljanje Angular aplikacije 16
3.3. Komponente 18
3.4. Servisi 20
3.5. Usmjerivanje 21
4. SUČELJE I PRIMJENA 22
5. ZAKLJUČAK 26
LITERATURA 27
SAŽETAK 28
ABSTRACT 29
ŽIVOTOPIS 30
PRILOZI 31

1. UVOD

Tema završnog rada je „Web stranica za naručivanje usluga kozmetičkog salona“, odnosno ovaj završni rad opisuje tehnologije potrebne za izradu web stranice i web aplikacije, kao i rezultat dobiven u slučaju ovog završnog rada.

Poglavlje WEB STRANICA opisuje tehnologije potrebne za izradu web stranice. Web stranica koju je potrebno izraditi treba biti responzivna (engl. *responsive*), odnosno web stranica se treba prilagođavati veličini zaslona na kojemu se prikazuje. Web stranica sadržavat će sljedeće informacije o kozmetičkom salonu: ponudu usluga, informacije o djelatnicima, galerija slika te kontakt informacije.

Poglavlje WEB APLIKACIJA opisuje osnove korištenja AngularJS frameworka, odnosno osnovne usluge koje AngularJS *framework* pruža kako bi olakšao izradu web aplikacije. Web aplikacija omogućavat će naručivanje, to jest rezervaciju termina za odabranu uslugu.

Poglavlje SUČELJE I PRIMJENA opisuje funkcionalnosti i način korištenja web stranice i web aplikacije.

1.1. Zadatak završnog rada

Napraviti AngularJS aplikaciju za vođenje kozmetičkog salona. Stranica će imati responzivni dizajn koji će omogućiti korisnicima rezerviranje usluga.

2. WEB STRANICA

Web stranica dokument je prikladan za *World Wide Web* i web preglednike, odnosno web stranica niz je dokumenata napisanih u jeziku za označavanje (engl. *markup language*). Web preglednici koordiniraju različite web resurse: *markup* datoteke, *style sheet*-ove, skripte, itd.

U hrvatskom *cyber slangu* ne postoje različiti izrazi za engleske nazive *web site* i *web page*, tako da je ponekad nejasno odnosi li se izraz 'web stranice' samo na dokumente (*web page*) ili na kolekciju dokumenata i resursa koji čine *web site* pa većina surfera i webmastera radije koristi engleske izraze za stranicu (*page* - HTML dokument) i stranice (*site* - kolekcija HTML dokumenata i multimedijalnih resursa) [1].

Dakle, web stranica kao HTML dokument je koji omogućuje prezentaciju teksta, poveznica je multimedijskih elemenata kao što su slike u digitalnom formatu, animacije, zvuk, video, itd.

2.1. HTML

Osnova je svake web stranice njezin *markup*, odnosno njen sadržaj. HTML kratica je za *HyperText Markup Language.* Kako je *markup* jezik, ili u slobodnom prijevodu jezik za označavanje, koji se sastoji od oznaka (engl. *markup tags*), tako je i HTML dokument sastavljen od HTML oznaka (engl. *oznaka*). HTML uz pomoć HTML oznaka opisuje strukture web stranica i omogućuje objavu *online* dokumenata s naslovima, tekstovima, tablicama, listama, slikama, i drugim elementima. Također omogućuje reprodukciju video zapisa, zvuka i omogućuje umetanje aplikacija direktno u dokument.

Svaki HTML dokument započinje deklaracijom da se radi o HTML dokumentu, koja je u slučaju HTML verzije 5 (HTML5) <!DOCTYPE html>, a cijeli dokument se nalazi unutar <html></html> oznaka, odnosno unutar ovih oznaka nalazi se cijeli *markup* za određenu stranicu.

Prvi javno dostupan opis HTML-a je dokument zvan HTML tags (oznake), prvi put se spominje na internetu od strane Tim Berners-Leeja krajem 1991. Taj opis se sastoji od 20 elemenata početnog, relativno jednostavnog dizajna HTML-a. Trinaest tih elemenata još uvijek postoji u HTML4. Postanak mnogih svojih oznaka duguje jednom od ranih jezika za formatiranje teksta, runoff-u. Runoff je razvijen u ranim 1960-im za CTSS (Kompatibilni Time-Sharing System) operacijski sustav. Runoff je kasnije inkorporiran u UNIX operativni sustav u naprednije formatirajuće programe kao što su roff, nroff i troff. Svaka nova verzija HTML-a je razvijana tako da ostane čitljiva na svim web preglednicima. Tim Berners-Lee je, nakon što je u listopadu 1994. napustio CERN (Europsku organizaciju za nuklearno istraživanje), osnovao organizaciju World Wide Web Consortium koja se bavi standardizacijom tehnologija korištenih na webu poznatija kao W3C [2].

```
<!DOCTYPE html>
<html>
<head>
    <title>Ime stranice</title>
</head>
<body>

    <h1>Naslov</h1>
    <p>Prvi paragraf.</p>

</body>
</html>
```

Sl. 2.1. *Primjer HTML dokumenta*

2.1.1. HTML oznake

Kako bi unutar HTML dokumenta definirali što je što koristimo HTML oznake, to jest ključne riječi koje su okružene s oznakama veće i manje. Struktura je HTML dokumenta takva da se sastoji od zaglavlja (engl. *head*) i tijela (engl. *body*) koji se nalaze unutar html oznaka. Unutar *head* elementa nalaze se specifikacije i opći podaci o stranici kao što je naslov (engl. *title*) koji se nalazi unutar <title></title> oznaka, meta podaci (engl. *metadata*) unutar <meta> oznaka, linkovi do *stylesheet*-ova te bitnije skripte. Ono što se nalazi unutar <head></head> oznaka neće se prikazati na web stranici (osim naslova koji će se prikazati kao ime kartice). Ono što se prikazuje na stranici nalazi se unutar <body></body> oznaka.

Unutar *body* oznaka nalaze se razne oznake koje označavaju različite elemente. Kako bi se neki tekst definirao kao paragraf, ispred se teksta stavlja oznaka koja otvara (engl. *opening tag*), a nakon teksta oznaka koja zatvara element (engl. *closing tag*). Naslovi se u HTML dokumentu označavaju po važnosti gdje je <h1></h1> oznaka za najvažniji naslov, a <h6></h6> oznaka za najnevažniji naslov. HTML oznake najčešće dolaze u parovima kao što su i , no neke HTML oznake imaju samo jednu oznaku. Također, postoje i neke HTML oznake koje služe za oblikovanje teksta, to su na primjer oznaka
 koja lomi tekst u novi red ili oznaka koja služi za podebljavanje teksta. U tablici 2.1. prikazane su neke od osnovnih oznaka.

Tab 2.1. *Osnovne HTML oznake*

Oznaka	Opis
<head> … </head>	Zaglavlje HTML dokumenta
<body> … </body>	Tijelo HTML dokumenta
<title> … </title>	Naslov stranice
<meta/>	*Meta* podaci
<link/>	Linkovi do *stylesheet*-ova
<script> … </script>	Oznake za skripte
…	Najznačajniji naslov
…	Drugi po važnosti naslov
…	Najmanje značajan naslov
…	Oznaka za paragraf
…	Oznaka za kontejner (grupiranje elemenata)

Svaka od HTML oznaka ima i mogućnost stavljanja atributa. Atributi pružaju dodatne informacije o HTML elementima, a specificiraju se u oznakama za otvaranje elementa (engl. *opening tag*). Primjerice, atribut „lang“ pridružuje se <html> oznaci kako bi se pružila dodatna informacija o jeziku i dijalektu web stranice. Ukoliko se želi specificirati da je web stranica na hrvatskom jeziku i hrvatskom dijalektu, html oznaci pridružit će se „lang“ atribut kao što je prikazano na slici 2.2.

```
<html lang="hr-HR">
```

Sl. 2.2 *Specificiranje lang atributa na html oznaci*

Atributi su posebno važni kod linkova, odnosno <a> oznaka koje se značajno razlikuju od <link/> oznaka. <a> oznake su hiperlinkovi (engl. *hyperlink*), dok <link/> oznake specificiraju gdje se nalaze *stylesheet* dokumenti. Najvažniji atribut kod <a> oznake je „href“ atribut koji određuje URL. Dakle, ukoliko bi „href“ atribut bio postavljen na www.etfos.unios.hr, korisnik bi klikom na <a> element bio preusmjeren na taj URL. Također, postoji i atribut „target“ koji specificira kako će se korisniku otvorit hiperlink, a osim na <a> oznaku može se primijeniti i na <area>, <base> te <form> oznake. U tablici 2.2. prikazane su moguće vrijednosti „target" atributa.

Tab. 2.2. *Vrijednosti „target“ atributa*

Vrijednosti target atributa	Opis
_blank	Hiperlink otvara u novom prozoru ili kartici
_self	Hiperlink otvara u istom prozoru
_parent	Hiperlink otvara u roditeljskom prozoru
_top	Hiperlink otvara u punoj veličini
framename	Hiperlink otvara u imenovanom prozoru

Kako bi se unutar HTML dokumenta omogućio prikaz slika koriste se <img> oznake sa svojim atributima. Također, uz pomoć <video> oznaka i <audio> oznaka moguće je omogućiti prikaz video i audio zapisa. Na slici 2.3. prikazan je način korištenja ovih medijskih oznaka.

```
<img src="slika.jpg" alt="Alternativni tekst ukoliko se slika ne učita">

<video width="320" height="240" autoplay>
    <source src="film.mp4" type="video/mp4">
    <source src="film.ogg" type="video/ogg">
</video>

 <audio controls>
    <source src="audiozapis.ogg" type="audio/ogg">
    <source src="audiozapis.mp3" type="audio/mpeg">
</audio>
```

Sl. 2.3. *Primjer korištenja medijskih HTML oznaka*

Jedne od najčešće korištenih HTML oznaka su oznake za liste. Uz pomoć oznaka za liste mogu se napraviti označene, neoznačene i podatkovne liste. U tablici 2.3. prikazane su oznake za liste.

Tab. 2.3. *HTML oznake za liste*

Oznaka	Opis
...	Neoznačena lista
...	Označena lista
...	Stavka unutar označene ili neoznačene liste
<dl> ... </dl>	Opisna lista
<dt> ... </dt>	Stavka unutar opisne liste
<dd> ... </dd>	Opis stavke opisne liste

Oznake za liste nerijetko se koriste za izradu navigacije na stranici tako što se unutar oznaka stavljaju hiperlinkovi uz pomoć <a> oznaka. Primjer navigacije uz pomoć liste prikazan je na slici 2.4.

```
<ul>
  <li><a href="#poc">Početna</a></li>
  <li><a href="#oNama">O nama</a></li>
  <li><a href="#kontakt">Kontakt</a></li>
</ul>
```

Sl. 2.4. *Prikaz korištenja liste kao navigacije*

2.2. CSS

CSS (od engl. *Cascading Style Sheets*) stilski je jezik koji označava upotrebu uzoraka stilova koji se mogu koristiti u HTML dokumentima, odnosno jezik koji definira način prikazivanja HTML elemenata. CSS napravljen je od strane W3C sa svrhom odjeljivanja sadržaja od uređivanja, to jest kako bi se HTML dokumenti mogli orijentirati primarno na sadržaj.

Najveća prednost CSS-a kvalitetni je doprinos boljem dizajnu web stranica, a samim time omogućava bolju preglednost te lakše snalaženje korisniku. Osim estetski boljeg ugođaja, CSS olakšava i ubrzava uređivanje web stranica jer odvaja sadržaj stranica od uređivanja stila.

Stilovi, odnosno CSS kod može se definirati na različite načine: unutar HTML dokumenta, u vanjskoj datoteci ili u liniji HTML elementa. Također, unutar jednog dokumenta mogu se pozivati i višestruki vanjski stilovi. Stilovi se redaju kaskadno što znali da će svaki sljedeći stil imati prednost:

1) Vrijednost Internet preglednika (engl. browser default)
2) Vanjski stilski uzorak (engl. external Style Sheet)
3) Unutarnji stilski uzorak (engl. internal Style Sheet)
4) Linijski stil (engl. inline Style)

2.2.1. CSS sintaksa i selektori

Set pravila CSS-a, to jest određivanje stila nekog HTML elementa, sastoji se od selektora i deklaracijskog bloka. Selektor pokazuje na HTML element, a deklaracijski blok sadrži jednu ili više deklaracija odvojenih točka-zarezom, a svaka deklaracija sadrži CSS atribut i vrijednost. CSS selektori koriste se kako bi se odabrali HTML elementi ovisno o imenu elementa, klasi, *idu*, ili atributu. Element selektor (engl. *element selector*) selektor je koji je zapravo ime elementa.

```
p {
    text-align: center;
    font-family: "Open Sans", sans-serif;
}
```

Sl. 2.5. *Sintaksa element selektora za p element*

Id selektor koristi *id* atribut HTML elementa kako bi odabrao specifični element. *Id* nekog elementa treba biti jedinstven ovisno o stranici, dakle *id* selektor koristi se kako bi se odabrao jedan jedinstveni element. Unutar HTML dokumenta *id se* dodjeljuje elementu uz pomoć *id* atributa. Unutar CSS dokumenta *id* selektor se piše tako da se ispred imena *ida* nekog elementa stavi znak ljestvi (#).

```
#o-nama {
    text-align: center;
    font-family: "Open Sans", sans-serif;
}
```

Sl. 2.6. *Sintaksa id selektora za o-nama id*

Selektor klase (engl. *class selector*) odabire elemente sa određenim *class* atributom. Unutar HTML dokumenta klasa se dodjeljuje elementima uz pomoć *class* atributa, a unutar CSS dokumenta *class* selektor se piše tako da se ispred imena klase stavlja točka (.).

```
.center {
    text-align: center;
}
```

Sl. 2.7. *Sintaksa class selektora za klasu center*

CSS selektore moguće je i grupirati tako da se ispred deklaracijskog bloka različiti selektori odvoje zarezom. Selektore je moguće kombinirati uz pomoć *CSS combinatora*. CSS3 definira četiri različita *combinatora* prikazanih u tablici 2.4.

Tab. 2.4. *CSS combinatori*

Ime *combinatora*	Selektor	Opis
descendant selector	razmak	Odabire potomke specificiranog elementa
child selector	>	Odabire djecu specificiranog elementa
adjacent sibling selector	+	Odabire susjedne blizance specificiranog elementa
general sibling selector	~	Odabire sve blizance specificiranog elementa

2.2.2. CSS boje

Boje se u CSS-u najčešće specificiraju kao ime boje na engleskom, RGB vrijednost ili kao HEX vrijednost. HTML i CSS podržavaju 140 standardnih imena za boje na engleskom (npr. crvena = *red,* zelena = green, itd.).

Sintaksa za RGB vrijednosti jest rgb(crvena, zelena, plava) gdje svaka od 3 navedene boje predstavlja intenzitet iste, a predstavlja se brojem iz intervala od 0 do 255. Dakle, „rgb(255,0,0)“ RGB vrijednost bila bi prikazana kao crvena jer je atribut crvena najveća moguća vrijednost dok su druga dva atributa 0. Nijanse sive se u RGB standardu definiraju kao jednake vrijednosti sva tri atributa, što bi značilo da je „rgb(0,0,0)“ crna boja, a „rgb(255,255,255)“ bijela boja.

HEX standard vrlo je sličan RGB standardu, a specificira se korištenjem heksadekadskih vrijednosti u intervalu od 00 do FF(dekadski od 0 do 255) za vrijednost crvene, zelene i plave. Plava boja u HEX standardu bila bi „ #0000FF“ jer je FF najveća vrijednost plave, a vrijednost crvene i zelene postavljene su na 00.

Aktualna verzija CSS-a, CSS3, definira i RGBA boje, HSL boje, HSLA boje te prozirnost (engl. *opacity*).

2.2.3. CSS model pravokutnika

HTML elementi se mogu smatrati pravokutnicima (engl. *box*). Izraz *box model* koristi se kada se priča o dizajnu i strukturi. *Box model* ustvari je pravokutnik koji okružuje svaki HTML element, a sastoji se od margina (engl. *margins*), granica (engl. *borders*), *padding*a i sadržaja. *Padding* čisti područje oko sadržaja, granica je linija koja se nalazi između *padding*a i margine, a margina čisti područje izvan granice. *Padding* i margin su prozirni, dok granica može imati različite stilove. Slika 2.5. grafički prikazuje CSS *box model*.

Sl. 2.8. *Grafički prikaz CSS box modela*

2.3. JavaScript

JavaScript je skriptni programski jezik, koji se izvršava u web pregledniku na strani korisnika. Napravljen je da bude sličan Javi, zbog lakšega korištenja, ali nije objektno orijentiran kao Java, već se temelji na prototipu i tu prestaje svaka povezanost s programskim jezikom Java. Izvorno ga je razvila tvrtka Netscape (www.netscape.com). JavaScript je primjena ECMAScript standarda [4].

Cilj je kreiranja JavaScripta bio dodavanje interaktivnosti HTML stranicama. Javascript je interpreter, to jest kod napisan u JavaScriptu izvršava se naredbu po naredbu bez prethodnog prevođenja (engl. *compile*).

2.3.1. JavaScript varijable

Varijable u JavaScriptu nemaju tip podataka, odnosno svaka varijabla može biti bilo koji tip podataka te je tip moguće i promijeniti bez ikakve dodatne sintakse. U JavaScriptu varijable se deklariraju ključnom riječi var (u novoj verziji ECMAScripta, ES6, ključna riječ za deklaraciju je let, ali ES6 još uvijek nije podržan u velikoj većini Internet preglednika).

```
var boja = "plava", broj = 5;
var polje = [3, 5, 7];
var auto = {marka: "opel", snaga: 80};
```

Sl. 2.9. *Deklaracije varijabli*

U primjeru prikazanom na slici 2.9. varijabla boja je string, varijabla broj je broj, varijabla polje je polje brojeva, a varijabla auto je objekt.

2.3.2. JavaScript funkcije i događaji

JavaScript ne bi bio programski jezik bez funkcija. Funkcije u JavaScriptu blokovi su JavaScript koda koji se mogu izvršiti kada se to od njih traži. Na primjer, funkcija može biti izvršena kada uslijedi neki događaj (engl. *event*). Događaj može biti nešto što Internet preglednik učini ili što korisnik učini. Primjer nekih događaja su: dovršeno učitavanje HTML stranice, HTML *input* polje je promijenilo vrijednost ili klik mišem.

Funkcije u JavaScriptu se definiraju s ključnom riječi „function“, a može biti standardna deklarirana funkcija ili funkcijski izraz (engl. *function expression*). Deklarirane se funkcije izvršavaju kada se pozovu. Sintaksa deklarirane funkcije prikazana je na slici 2.9.

```
function mojaFunkcija(a, b) {
    return a * b;
}
```

Sl. 2.10. *Deklaracije funkcije u JavaScriptu*

Funkcijske je izraze moguće spremiti u varijable, a te varijable kasnije koristiti kao funkcije. Funkcije koje se spremaju u varijable ne trebaju ime funkcije pa se zovu i anonimne funkcije i uvijek se pozivaju koristeći ime varijable.

```
var x = function (a, b) {return a * b};
var z = x(1, 2);
```

Sl. 2.11. *Funkcijski izraz, varijabla kao funkcija*

JavaScript omogućuje izvođenje koda pojavom događaja. HTML omogućuje dodavanje *event handlera* u HTML elemente. Naravno, moguće je kombinirati događaje, odnosno pojavom događaja izvršiti neku funkciju.

```
<button onclick="pokaziVrijeme()">Vrijeme?</button>

<script>
    function pokaziVrijeme() {
        document.getElementById("demo").innerHTML = Date();
    }
</script>

<p id="demo"></p>
```

Sl. 2.12. *Primjer event handlera*

Primjer JavaScript i HTML koda prikazanog na slici 2.11. prikazat će trenutno vrijeme u paragrafu s *idem* demo nakon klika na dugme „Vrijeme?“.

2.4. Bootstrap

Originalno nazvan *Twitter Blueprint,* kojeg su razvili Mark Otto i Jacob Thornton iz Twittera kako bi si olakšali izradu korisničkog sučelja. Pošto se ideja svidjela nekolicini programera, nastavili su sa razvitkom ovog *frameworka*, a ime se mijenja u Bootstrap. Početkom 2012. najavljen je Bootstrap 2 koji uvodi sustav mreže 12 stupaca i komponente za responzivni dizajn (RWD od engl. Responsive Web Design). Bootstrap ubrzo postaje najpopularniji projekt na GitHubu te postaje najpopularniji *framework* za RWD. Današnja aktualna verzija Bootstrapa je 3.3.6, a verzija 4 je dostupna za beta testiranje te bi trebala izaći krajem 2016. ili početkom 2017. godine.

2.4.1. Responzivni dizajn

Responzivni dizajn pristup je web dizajnu koji se fokusira na osiguravanje optimalnog i kvalitetnog pregleda web sadržaja bez obzira na veličinu zaslona uređaja na kojemu se gleda web stranica. Stranica dizajnirana kao responzivna prilagođava izgled i sadržaj koristeći fluid, *grid* (engl. *proportion-based grids*) i CSS3 medijske zahtjeve (engl. *media queries*).

Kreator je pojma RWD Ethan Marcotte spomenuvši ga prvi put u članku objavljenom u svibnju 2010. godine na portalu A list apart. Teoriju i primjenu RWD-a opisao je u knjizi „Responsive Web Design“ 2011. godine. Britanski „Top Web Design Trends“ godinu dana kasnije pojam RWD uvršten je na 2. mjesto trendova. Uskoro RWD postaje standard web dizajna te 2013. godina biva proglašena godinom RWD-a.

2.4.2. Grid system

Glavni je dio svakog *frameworka,* namijenjenog RWD-u, *Grid sistem*, odnosno sustav redova i stupaca. *Grid system* koristi se u izradi stranica kroz seriju redova i stupaca. Redovi se moraju staviti u „.container“ ili „.container-fluid“. Stupci se definiraju kao broj željenih stupaca od 12 dostupnih stupaca u jednome redu. U tablici 2.5. prikazane su Bootstrap klase vezane uz *grid system*.

Tab. 2.5. *Bootstrap grid system klase*

Ime klase	Opis
container	*Container* fiksne širine
container-fluid	*Container* širine *viewporta*
row	Redak
col-lg-*	Broj stupaca (* - predstavlja neki broj od 1 do 12) za ekrane širine 1200px ili više
col-md-*	Broj stupaca (* - predstavlja neki broj od 1 do 12) za ekrane širine od 992px ili više
col-sm-*	Broj stupaca (* - predstavlja neki broj od 1 do 12) za ekrane širine 768px ili više
col-xs-*	Broj stupaca (* - predstavlja neki broj od 1 do 12) za ekrane širine manje od 768px

Svaki redak sadrži maksimalno 12 stupaca koji su široki ovisno o klasi koja definira pojedini stupac, odnosno jedan stupac može biti širine 4 na velikom ekranu, a širine 10 na manjem ukoliko se dodjele obje klase tom stupcu. Ukoliko je definirana samo jedan širina stupca za neku veličinu zaslona, nakon *breakpointa* širine tog zaslona taj će stupac biti širine 12, odnosno biti će sam u retku.

```
<div class="container">
    <div class="row">
        <div class="col-md-1 col-sm-5">Prvi stupac</div>
        <div class="col-md-1 col-sm-7">Drugi stupac</div>
        <div class="col-md-1 col-sm-5">Treći stupac</div>
        <div class="col-md-8 col-sm-7">Četvrti stupac</div>
    </div>
</div>
```

Sl. 2.13. *Sintaksa grid systema*

3. WEB APLIKACIJA

Web aplikacija je programsko rješenje kojemu se pristupa uz pomoć Internet preglednika, odnosno aplikacija koja omogućuje neku uslugu putem Internet preglednika. Web aplikacija može biti *front-tend* aplikacija, *back-end* aplikacija ili kombinacija. *Front-end* aplikacija je ona aplikacija koja se izvodi u Internet pregledniku, odnosno na strani korisnika (engl. *client side*), a *back-end* aplikacija je ona aplikacija koja se izvodi na serveru (engl. *server side*).

U naslovu teme ovog završnog rada stoji „Web stranica za naručivanje usluga kozmetičkog salona**“**, a u opisu „Napraviti AngularJS aplikaciju za vođenje kozmetičkog salona…“. Dakle, ideja jest napraviti web aplikaciju koja će omogućiti korisniku rezerviranje termina za određenu uslugu.

Aplikacija će omogućiti korisniku odabir kategorije usluge, a ovisno o odabranoj kategoriji omogućit će odabir usluge iz te kategorije te će korisnik moći odabrati dva termina koja mu odgovaraju. Nakon potvrde odabrane usluge i termina, korisnik će unijeti podatke uključujući e-mail i broj telefona (mobitela) kako bi se osoblje salona moglo javiti radi potvrde termina ili korekcije termina.

3.1. AngularJS i TypeScript

AngularJS (spominje se još kao Angular i Angular.js) je *open-source front-end* JavaScript framework kojeg održava Google i zajednica individualaca i kompanija kako bi olakšali izradu *single-page* aplikacija. Trenutna stabilna verzija Angulara je 1.5.8., dok je Angular 2, odnosno AngularJS 2.0 u *beti*, ali iako je u *beti* preporuča se izrada aplikacija u verziji 2 jer je puno brži, a stabilna verzija najavljena je da će izaći u roku od par mjeseci od vremena pisanja ovoga rada.

Iako je Angular JavaScript framework, većina dokumentacije, kao i sam framework, pisani su za TypeScript. TypeScript besplatni je i *open-source* programski jezik razvijan i održavan od strane Microsofta. TypeScript striktni je *superset* JavaScripta koji unosi *static typing* i *class-based* objektno orijentirano programiranje. Pošto je TypeScript *superset* JavaScripta, ne može se izvoditi direktno u Internet pregledniku već ga je potrebno prije upotrebe kompajlirati u JavaScript.

Pri izradi ovog završnog rada korišten je Angular 2 (u daljnjem tekstu Angular) framework, a aplikacija je pisana u TypeScriptu.

3.2. Postavljanje Angular aplikacije

Postavljanje svake Angular aplikacije kreće od kreiranja sljedećih konfiguracijskih datoteka: „package.json“, „tsconfig.json“ te „typings.json“. „Package.json“ datoteka određuje npm *dependecije*, to jest verzije paketa koji su potrebni u projektu, „tsconfig.json“ definira kako TypeScript kompajler generira JavaScript datoteke, a „typings.json“ pruža dodatne definicijske datoteke TypeScript kompajleru kako bi ih kompajler mogao prepoznati.

```
{
    "name": "web-narudzbe",
    "email": "fkulic@etfos.hr",
    "version": "1.0.0",
    "author": "Filip Kulic",
    "description": "Paket za web aplikaciju narucivanja",
    "scripts": {
        "start": "concurrently \"npm run tsc:w\" \"npm run lite\" ",
        "tsc": "tsc",
        "tsc:w": "tsc -w",
        "lite": "lite-server",
        "typings": "typings",
        "postinstall": "typings install"
    },
    "license": "ISC",
    "dependencies": {
        "angular2": "2.0.0-beta.15",
        "systemjs": "0.19.26",
        "es6-shim": "^0.35.0",
        "reflect-metadata": "0.1.2",
        "rxjs": "5.0.0-beta.2",
        "zone.js": "0.6.10"
    },
    "devDependencies": {
        "concurrently": "^2.0.0",
        "lite-server": "^2.2.0",
        "typescript": "^1.8.10",
        "typings": "^0.7.12"
    }
}
```

Sl. 3.1. *package.json datoteka*

```
{
  "compilerOptions": {
    "target": "es5",
    "module": "system",
    "moduleResolution": "node",
    "sourceMap": true,
    "emitDecoratorMetadata": true,
    "experimentalDecorators": true,
    "removeComments": false,
    "noImplicitAny": false
  },
  "exclude": [
    "node_modules",
    "typings/main",
    "typings/main.d.ts"
  ]
}
```

Sl. 3.2. tsconfig.*json datoteka*

```
{
  "ambientDependencies": {
"es6-shim": "github:DefinitelyTyped/DefinitelyTyped/es6-shim/es6-shim.d.ts#7de6c3dd94feaeb21f20054b9f30d5dabc5efabd"
}
```

Sl. 3.3. typings.*json datoteka*

Nakon postavljanja svih konfiguracijskih datoteka potrebno je instalirati pakete koji su navedeni u package.json datoteci. Instalacija paketa obavlja se uz pomoć npm-a (*Node Package Manager*) te ukoliko na računalu nema instaliran Node, potrebno ga je instalirati. Kako bi se instalirali paketi potrebno je u konzoli (engl. *command prompt*), odnosno terminalu, iz direktorija u kojemu se nalaze konfiguracijske datoteke uz pomoć narede „npm install“ instalirati pakete. Ukoliko se typings direktorij nije pojavio, potrebno ga je ručno instalirati naredbom „npm run typings install“. Unutar trenutnog direktorija pojavio se i direktorij „node_modules“ koji sadrži sve potrebne pakete za izradu ove aplikacije.

Unutar direktorija gdje se nalaze konfiguracijske datoteke dobra je praksa kreirati novi direktorij imena „app“ kako bi kodove i datoteke aplikacije odvojili od web stranice. Unutar app direktorija potrebno je kreirati main.ts datoteku te glavnu komponentu aplikacije- datoteku app.component.ts. Main.ts datoteka je datoteka koja *bootstrapira*, odnosno učitava, glavnu komponentu aplikacije app.component.ts.

```
import {bootstrap} from "angular2/platform/browser";
import {AppComponent} from "./app.component";

bootstrap(AppComponent);
```

Sl. 3.4. *Kod main.ts aplikacije*

Bootstrap naredba učitava i pokreće ono što se nalazi unutar argumenata funkcije *bootstrap*. U ovom slučaju naredba „bootstrap“ učitava i pokreće „AppCompponent“. Kako bi „bootstrap“ funkcija radila, potrebno je napraviti *import bootstrap* funkcije iz *browser* datoteke koja se skinula pri instalaciji. Naredba „import“ unutar vitičastih zagrada prima ime klase koja se *importa* u kod, a nakon ključne riječi „from“ unutar navodnika (jednostrukih ili dvostrukih) prima putanju do datoteke u kojoj se klasa nalazi. Ukoliko putanja počinje s „ **./** „ radi se o relativnoj putanji, to jest o putanji u odnosu na direktorij u kojemu se datoteka nalazi. Ukoliko putanja ne počinje s “ **./** „ radi se o putanji koja kreće iz direktorija „node_modules“. Također, kod „import“ izjava, to jest putanje datoteke, ne smije se stavljati ekstenzija, već će kompajler sam dodati ekstenziju pri kompajliranju.

Kompajler će izbacivati greške pošto još ništa ne postoji unutar „app.component.ts“, no to nije problem jer će greške nestati čim se napiše kod unutar iste datoteke.

3.3. Komponente

Angular se bazira na komponentama (engl. *component*). Komponente su osnovni građevinski blokovi koji grade svaku Angular aplikaciju. Komponenta kontrolira dio zaslona, to jest pogled, kroz njegov *template*. Komponenta je zapravo klasa s „@Component“ *decoratorom* koji se mora uz pomoć *import* naredbe učitati u kod. *Decoratori* su funkcije Angulara koji daju neke mogućnosti klasama. Primjerice, „@Component“ definira klasu kao komponentu, „@Input“ i „@Output“ omogućuju komunikaciju između roditeljskih i dječjih komponenti. Na slici 3.5. prikazana je app komponenta.

```
import {Component} from "angular2/core";

@Component({
    selector: "nar-app",
    templateUrl: "app/app.component.html",
    styleUrls: ["app/app.component.css"]
})
export class AppComponent {
    appTitle: string = "Naruči se";
}
```

Sl. 3.5. *Kod app.component.ts datoteke*

Kako bi se omogućio *import* ove komponente, ispred ključne riječi *class* stavlja se ključna riječ *export*. Da bi neka klasa bila komponenta, ispred klase stavlja se već spomenuti „@Component“ *decorator*. *Component decorator* prima jedan ili više parametara. Parametar *selector* prima *string* koji je zapravo HTML *tag* koji se stavlja unutar HTML dokumenta, a govori gdje će se aplikacija nalaziti. *Template* komponente može se definirati ključnom riječi „template“ unutar datoteke komponente ili se može definirati u vanjskoj datoteci ključnom riječi „templateUrl“. CSS se definira ključnom riječi „styles“ unutar datoteke komponente te prima polje *stringova* ili se definira ključnom riječi „styleUrls“ te prima polje *stringova* koji su putanje do CSS datoteka. Atributi se unutar klase definiraju imenom atributa te se specificira tip podatka. Također je moguće i inicijalizirati atribut u istom retku uz pomoć operatora jednako (=).

U Angularu, aplikacije pravimo koristeći komponente te ih nerijetko treba ugnijezditi (engl. *nest*) jedne u druge kako bi mogli koristiti njene mogućnosti unutar druge komponente. Kako bi jedna komponenta ugnijezdila drugu, potrebno je *importati* komponentu te unutar „@Component“ *decoratora* navesti komponentu kao *directive. Directive* parametar prima polje, što znači da je moguće ugnijezditi više komponenti koje je potrebno odvojiti zarezom. Kako bi ugniježđena (engl.) komponenta primila podatke: koristi se „@Input“ *decorator* ispred atributa ili metoda (funkcija). Ukoliko je potrebno da ugniježđena komponenta šalje podatke roditeljskoj komponenti koristi se „@Output“ *decorator* koji uz pomoć *EventEmittera* omogućuje slanje atributa ili metoda ugniježđenoj komponenti.

3.4. Servisi

Servisi (engl. *services*) su u Angularu način „nasljeđivanja“ klase u više komponenti. Servisi se koriste kada neki podaci ili logika nisu povezani isključivo samo s jednim pogledom ili komponentom ili kad se pristup nekim podacima ili logici želi omogućiti više komponenti. Kako bi neka klasa postala servis koristi se „@Injectable“ *decorator* koji kreira jedinstvenu instancu servisa (engl. *singleton*) i omogućuje Angular *injectoru* upravljanje servisom. Ukoliko komponenti treba neki servis, komponenta definira servis kao zavisnost (engl. *dependency*) kako bi dobila pristup instancama objekata koji su joj potrebni. Ovaj proces se naziva *dependency injection*. Kako Angular upravlja jedinstvenom instancom servisa, sva logika i podaci dijelit će se u svim klasama koji koriste neki servis.

U slučaju ovog završnog rada, odnosno ove web aplikacije, servis pruža podatke o uslugama komponentama koje ih trebaju.

```
import {Injectable} from "angular2/core";
import {IUsluge} from "./usluge";

@Injectable()
export class UslugeService {
    getUsluge(): IUsluge[] {
                return [
                {
                    "id": 1,
                    "ime": "Tretman lica i dekoltea",
                    "cijena": 180,
                    "trajanje": 30
                },
                {
                    "id": 2,
                    "ime": "Tretman lica i dekoltea (ampula ,ultrazvuk i masaža)",
                    "cijena": 200,
                    "trajanje": 30
                },
                {
                    "id": 3,
                    "ime": "Klasični tretman lica (bez masaže)",
                    "cijena": 130,
                    "trajanje": 30
                }
            ];
    }
}
```

Sl. 3.6. *Servis koji pruža podatke o uslugama*

Ovaj servis koristi *interface*. *Interface* specificira povezani skup atributa i metoda, a klasa može koristiti *interface* tako što ga implementira. U slučaju ove apliklacije, *interface* se koristi kao tip podatka. Na slici 3.7. prikazano je korištenje *interfacea* kao tipa podatka koji sadrži *id,* ime, cijenu te trajanje usluge.

```
export interface IUsluge {
    id: number,
    ime: string,
    cijena: number,
    trajanje: number
}
```

Sl. 3.7. *Interface kao tip podatka*

3.5. Usmjerivanje

Angular aplikacije su *single-page* aplikacije, to jest Angular dinamički mijenja poglede, odnosno strukturu HTML dokumenta. Usmjerivanje (engl. *routing*) omogućuje upravljanje pogledima, odnosno odlučuje kada će se koji pogled prikazati. Za komponente koje trebaju koristiti usmjerivanje treba konfigurirati put (engl. route), definirati događaje koji mijenjaju pogled te povezati puteve s događajima. Dakle, kada korisnik odabere neku opciju, odnosno dogodi se događaj koji je povezan s putem pogleda neke komponente, taj pogled će korisnik vidjeti.

Angular *router* vanjski je opcionalni *NgModul* naziva *RouterModule*. Kako bi neka komponenta mogla mjenjati pogled unutar nje, potrebno je pozvati *ROUTER_PROVIDERS, RouteConfig, ROUTER_DIRECTIVES* uz pomoć *import* naredbe, unutar „@Component“ *decoratora* pod atribut *directives* navesti *ROUTER_DIRECTIVES*, a pod *providers* navesti *ROUTER_PROVIDERS*. Također, potrebno je dodati i „@RouteConfig“ *decorator* u kojemu treba postaviti puteve do svih pogleda koje je potrebno mjenjati. Put uz koji je atribut „useAsDefault: true“ prikazivat će se kao *default* pogled, odnosno kao prvi pogled nakon učitavanja aplikacije.

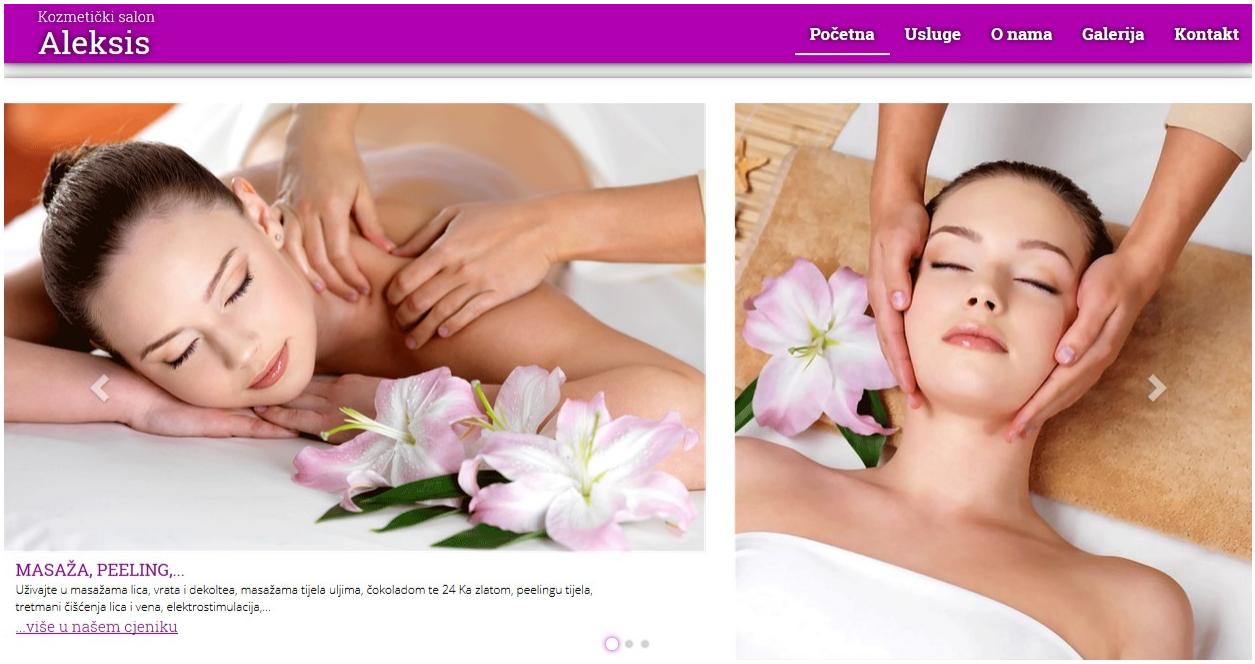
```
@RouteConfig([
    {path: "/defaultpogled", name: "defaultpogled", component: DefaultComponent, useAsDefault: true},
    {path: "/pogled2", name: "pogled2", component: Pogled2Component}
])
```

Sl. 3.8. *Definiranje puteva uz pomoć „@RouteConig“ decoratora*

4. SUČELJE I PRIMJENA

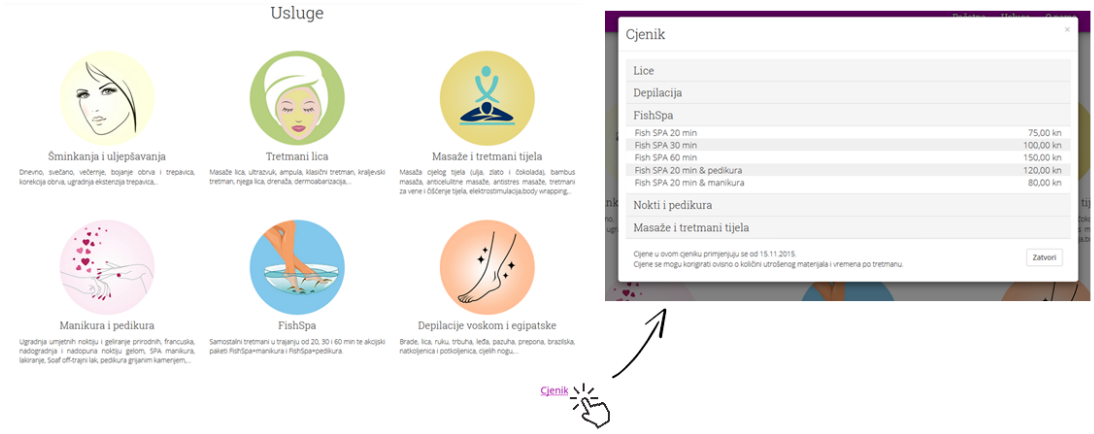
Web stranica dobivena kao rezultat ovog završnog rada je web stranica koja služi za informiranje klijenata o uslugama koje nudi kozmetički salon, a web aplikacija olakšava rezervaciju termina za odabranu uslugu. Prilikom izrade web stranice i web aplikacije korištene su sve opisane tehnologije.

Pri prvom učitavanju web stranice može se vidjeti navigacija i izmjenjivač slika koji prikazuje bit kozmetičkog salona. Klikom na jedan od *linkova* na navigaciji, preglednik će automatski skliznuti (engl. *scroll*) do tog dijela web stranice.



Sl. 4.1. *Pogled pri prvom učitavanju web stranice*

Kada korisnik odskliže prema dolje ili pritisne na navigaciji na *link* usluge, ugledat će podnaslov usluge, to jest slikoviti prikaz usluga koje kozmetički salon nudi. Također, klikom na *link* „Cjenik“ pojavit će se iskočni prozor unutar same stranice koji prikazuje cjenik po kategorijama. Ovisno na kliknutu uslugu, prikazat će se isključivo usluge koje spadaju pod tu kategoriju.

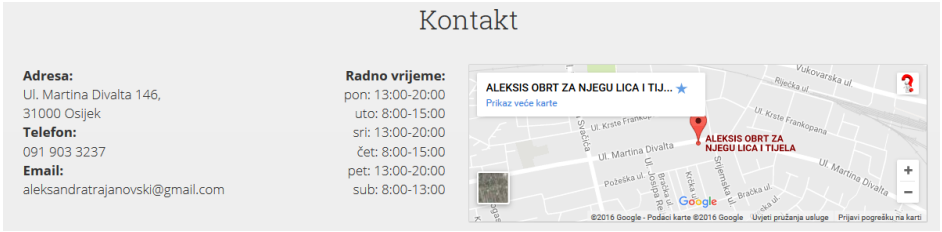


Sl. 4.2. *Pogled na „Usluge“ i cjenik*

Također, tu su i dijelovi stranice „Galerija“ i „O nama“ prikazani na slici 4.3. Klikom na jednu od slika u galeriji, otvorit će se uvećan prikaz slika s kontrolama za projekciju.

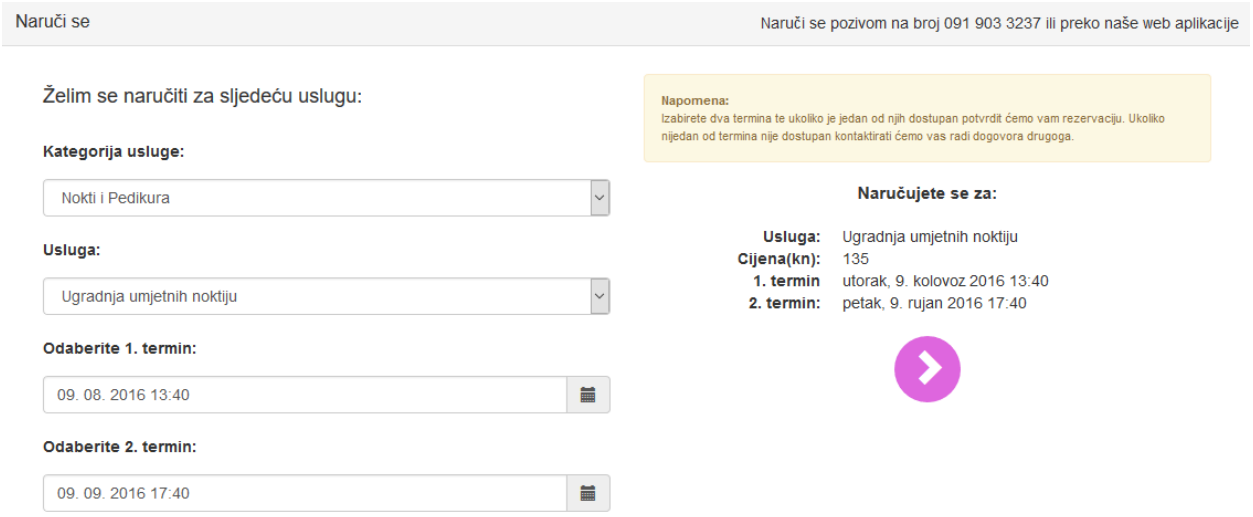
Sl. 4.3. *Dijelovi stranice „O nama“ i „Galerija“*

Ispod galerije nalaze se i kontakt informacije te *Google maps* karta s lokacijom salona.



Sl. 4.4. *Dio stranice „Kontakt“*

Nakon „Kontakt“ dijela stranice, korisnik dolazi do aplikacije. Aplikacija omogućuje odabir kategorije, a ovisno o odabranoj kategoriji omogućuje korisniku odabir usluge. Korisnik također treba odabrati dva termina. Na desnom dijelu aplikacije prikazati će se odabrana usluga i termini te klikom na gumb sa znakom više (>) korisnik potvrđuje usluge i termin te se mijenja prikaz aplikacije. Ukoliko nisu unesena sva polja, aplikacija će upozoriti korisnika.

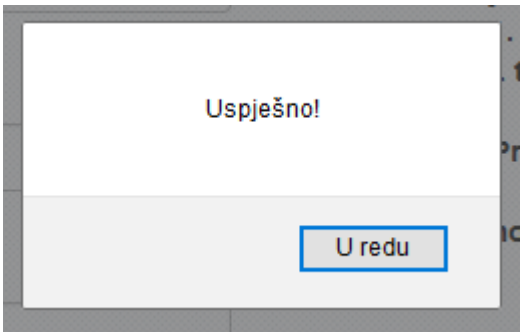


Sl. 4.5. *Prvi pogled aplikacije – odabir usluge i termina*

Sljedeći prikaz je prikaz na kojem korisnik unosi osobne podatke.

Sl. 4.6. *Drugi pogled aplikacije – unos osobnih podataka*

Nakon unosa svih podataka i pritiska na gumb s kvačicom ispisat će se poruka „Uspješno“.



Sl. 4.7. *Aplikacija obavještava korisnika o uspješno poslanoj rezervaciji*

5. ZAKLJUČAK

U ovom završnom radu napravljena je responzivna web stranica i web aplikacija za naručivanje usluga kozmetičkog salona. Za izradu web stranice korištene su tehnologije; HTML, CSS, JavaScript. Web stranica je responzivna, odnosno rađena je uz pomoć Bootstrap *frameworka* koji uz pomoć *grid systema* značajno olakšava izradu responzivne web stranice. Kako bi se olakšalo rezerviranje usluga kozmetičkog salona izrađena je web aplikacija uz pomoć AngularJS 2 *frameworka*. AngularJS 2 bazira se na izradi komponenti. Komponente je moguće ugnježđivati jedne u druge te uz pomoć raznih *decoratora* omogućiti interakciju između istih. Kako bi se olakšao pristup podacima o uslugama svim komponentama napravljen je servis koji koristi *interface* kao tip podatka. S *interfaceom* Angular omogućuje izradu vlastitih tipova podataka. Rezultat završnog rada je web stranica koja se prilagođava zaslonu te web aplikacija koja omogućuje rezervaciju termina za određenu uslugu.

LITERATURA

[1] *Wiki* o web stranicama, https://hr.wikipedia.org/wiki/Web_stranice, lipanj 2016.

[2] *Wiki* o HTML-u, https://hr.wikipedia.org/wiki/HTML, lipanj 2016.

[3] WWW konzorcij za postavljanje web standarda, https://www.w3.org/, lipanj 2016.

[4] *Wiki* o JavaScriptu, https://hr.wikipedia.org/wiki/JavaScript, lipanj 2016.

[5] Wiki o responzivnom dizajnu, https://en.wikipedia.org/wiki/Responsive_web_design, lipanj 2016.

[6] Stranica Odjela za matematiku, Kolegij Web programiranje, http://www.mathos.unios.hr/wp/wp2009-10/wp.html, lipanj 2016.

[7] Blog tvrtke Avalon, https://www.avalon.hr/blog/2013/11/responsive-web-design-prestiz-ili-potreba/, lipanj 2016.

[8] Službena stranica Bootstrap *frameworka*, http://getbootstrap.com/, kolovoz 2016.

[9] Službena stranica AngularJS *frameworka*, https://angular.io/docs/ts/latest/, kolovoz 2016.

SAŽETAK

U ovom završnom radu cilj je bio izrada funkcionirajuće web stranice koja se prilagođava zaslonu te web aplikacije koja olakšava rezervaciju termina za neku uslugu. Pri odabiru tehnologije za izradu responzivne web stranice, uz HTML, CSS i JavaScript koji čine osnovu izrade web stranica, bilo je ključno odabrati *framework* koji ima ugrađen sustav mreže (engl. *grid system*) te koji ima dobru dokumentaciju. Odabran je Bootstrap koji ima daleko najveću podršku te najviše dokumentacije kao najpopularniji projekt na GitHubu. Uz pomoć bootstrap, ne izostavljajući HTML, CSS, i JavaScript, izrađena je funkcionalna web stranica koja se prilagođava zaslonu. Odabiru tehnologije za izradu web aplikacije pao je na Angular, najpopularniji JavaScript *framework*. Uz pomoć Angulara izrađena je aplikacija koja omogućuje odabir usluge ovisno o izabranoj kategoriji te odabir dva termina.

Ključne riječi: Angular, Bootstrap, responzivan dizajn, web aplikacija, web stranica

ABSTRACT

Web page for appointment arranging for beauty salon

The main task of this Bachelor thesis was developing a functional web site which adapts its structure depending on the size of the screen and to develop a web application which allows the user to arrange an appointment for a service. When choosing the technology for development of a responsive web site, not counting HTML, CSS or JavaScript, it was crucial to choose a framework which had grid system embedded into it and also a framework which had lots of documentation, hence Bootstrap was chosen. Bootstrap was a logical choice since it's the most popular project on GitHub. A functional page was created using Bootstrap and also elementary technologies of front-end; HTML, CSS and JavaScript. Choosing a technology for web application hasn't been that hard since Angular is the most popular JavaScript framework. So, the web application was made using Angular, which allows the user to arrange two appointments and a service according to the category.

Keywords: Angular, Bootstrap, responsive design, web application, web page

ŽIVOTOPIS

Filip Kulić rođen je 25.01.1995. godine u Osijeku. Nakon završene Osnovne škole Jagode Truhelke, upisuje Elektrotehničku i prometnu školu Osijek, smjer tehničar za računalstvo gdje maturira 2013. godine. Iste godine upisuje sveučilišni preddiplomski studij računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. Govori i piše engleski jezik. Poznaje rad na računalu te se odlično služi MS Office alatima. Također, poznaje i razne programske i skriptne jezike poput C-a, C++-a, C#-a, HTML-a, CSS-a, JavaScripta i SQL-a.

Potpis:

_____

PRILOZI

Na CD-u priloženom uz Završni rad nalaze se:

- Dokumenti:
  - Završni rad.docx
  - Završni rad.pdf
- Direktorij: „web-stranica“ sa svim datotekama izrađenim pri izradi završnog rada