

SVEUČILIŠTE U SPLITU
FAKULTET ELEKTROTEHNIKE, STROJARSTVA I
BRODOGRADNJE

ZAVRŠNI RAD

RezervirajStol.hr

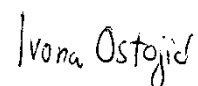
Ivona Ostojić

Split, rujan 2018.

IZJAVA

Ovom izjavom potvrđujem da sam završni rad s naslovom (RezervirajStol.hr) pod mentorstvom (prof. dr. sc. Maja Štula) pisala samostalno, primijenivši znanja i vještine stečene tijekom studiranja na Fakultetu elektrotehnike, strojarstva i brodogradnje, kao i metodologiju znanstveno-istraživačkog rada, te uz korištenje literature koja je navedena u radu. Spoznaje, stavove, zaključke, teorije i zakonitosti drugih autora koje sam izravno ili parafrazirajući navela u završnom radu citirala sam i povezala s korištenim bibliografskim jedinicama.

Ivona Ostojić





SVEUČILIŠTE U SPLITU
FAKULTET ELEKTROTEHNIKE, STROJARSTVA I
BRODOGRADNJE



Preddiplomski studij: Računarstva

Oznaka programa: 120

Akadska godina: 2017./2018.

Ime i prezime: **IVONA OSTOJIĆ**

Broj indeksa: 264-2015

ZADATAK ZAVRŠNOG RADA

Naslov: **RezervirajStol.hr**

Zadatak: Osmisliti, projektirati i realizirati web aplikaciju za rezervaciju stolova u restoranima. Na serverskoj strani aplikacije koristiti Node.js platformu za razvoj web aplikacija. Voditi računa o dizajnu korisničkog sučelja prilagodljivog različitim dimenzijama uređaja kojima korisnici pristupaju web sadržajima.

Prijava rada: 22.02.2018.

Rok za predaju rada: 20.10.2018.

Rad predan: 10.09.2018.

Datum obrane: 13.09.2018.

Mentor:

Prof. dr. sc. Maja Štula

Sadržaj

1	UVOD	1
2	PRIMJENJENE TEHNOLOGIJE	2
2.1	Općenito o HTML-u i CSS-u	2
2.2	Bootstrap.....	3
2.3	JavaScript.....	5
2.4	NodeJS.....	8
2.4.1	V8 <i>engine</i>	8
2.4.2	Neblokirajući način rada i petlja događaja	8
2.4.3	NPM	10
2.4.3.1	ExpressJS	10
2.5	MongoDB	11
3	REALIZACIJA PRAKTIČNOG DIJELA RADA.....	13
3.1	Pregled sustava	13
3.2	Klase korisnika	14
3.3	Korisnički zahtjevi.....	14
3.3.1	Funkcionalni zahtjevi	14
3.3.2	Nefunkcionalni zahtjevi	16
3.4	Slojevi aplikacije	16
3.4.1	Klijentski dio	16
3.4.2	Serverski dio.....	17
3.4.3	Baza podataka	18
3.4.3.1	Sigurnost podataka.....	18
3.5	Korisnička sučelja.....	19
3.5.1	Javno sučelje	20
3.5.2	Privatna sučelja	21

4	ZAKLJUČAK	24
	LITERATURA.....	25
	SAŽETAK.....	26
	SUMMARY	27

1 UVOD

U današnjem svijetu, sve je veći razvoj web aplikacija različitih primjena. Web aplikacija je korisničko – poslužiteljski računalni program koji korisnik (klijent) pokreće unutar svog web preglednika. Ponašanje web aplikacija, slično je ponašanju desktop ili mobilnih aplikacija. Osnovno obilježje je mogućnost jednostavnog ažuriranja i dodavanja sadržaja. Korisnici mogu vršiti pretragu po aplikaciji po određenim kriterijima, ostavljati komentare, ispunjavati ankete ili forme i sl.

Zbog jednostavnosti korištenja i olakšavanja svakodnevnih životnih potreba, ovakvim aplikacijama raste popularnost i broj korisnika. Iz tog razloga, proizašla je i ideja aplikacije ovog završnog rada, sustav RezervirajStolHr. Sustav omogućava korisnicima rezervaciju u omiljenom restoranu na način da pomoću tlocrta stolova točno mogu izabrati gdje žele biti smješteni, ukoliko je određeno mjesto slobodno u željenom terminu. Time se olakšava osoblju i njima samima proces rezerviranja.

Ovaj rad objašnjava korištene tehnologije i radne okvire pri izradi same aplikacije kao što su HTML, CSS, Bootstrap, JavaScript, NodeJS i MongoDB baza podataka. Osim toga, objašnjava funkcionalne i nefunkcionalne zahtjeve aplikacije, općeniti prikaz i podjelu sustava te korisnička sučelja.

2 PRIMJENJENE TEHNOLOGIJE

U ovom poglavlju obrađivat će se tehnologije korištene za izradu praktičnog dijela rada. Za prezentaciju i obradu podataka na klijentskoj strani korišten je Bootstrap (*HTML* i *CSS framework*) zajedno s JavaScriptom, a za obradu podataka na serverskoj strani korišten je NodeJS s dodatkom ExpressJS u kombinaciji s MongoDB bazom podataka. Sve navedene tehnologije su javno dostupne i besplatne za korištenje.

2.1 Općenito o HTML-u i CSS-u

HTML (*HyperText Markup Language*) je *markup* jezik koji se koristi za izradu web stranica. *Markup* je jezik koji kombinira sadržaj koji se objavljuje s dodatnim podacima o samom sadržaju (npr. veličina slova i font, položaj na stranici, boje itd.). *HTML* je jednostavan za uporabu i lako se uči, zbog čega je opće prihvaćen i popularan, a svoju raširenost je stekao upravo zbog te jednostavnosti i zato što je od početka zamišljen kao besplatan i dostupan svima.

Prikaz hipertekst dokumenta omogućava sam web preglednik, a uloga *HTML* jezika jest uputiti web preglednik kako će prikazati taj dokument. Osnovni građevni element svake stranice su oznake (*tags*) koji opisuju kako će se nešto prikazati u pregledniku. Ovaj jezik nije programski jezik jer njime ne možemo izvršiti nikakve zadaće niti najjednostavnije matematičke operacije, već on služi isključivo za opis hipertekstnih dokumenata. Ekstenzije html datoteka su .html ili .htm. [1]

CSS (*Cascading Style Sheets*) je jezik stila (*stylesheet language*) koji se koristi za opisivanje izgleda stranice dokumenta napisanog *markup* jezikom. Obično se koristi u kombinaciji s HTML-om ili XHTML-om, ali može se primijeniti i na druge markup jezike kao npr. XML.

CSS ima jednostavnu sintaksu u kojoj koristi riječi engleskog jezika za definiranje stila te sadrži niz pravila od kojih se svako pravilo sastoji od jednog ili više selektora odvojenih zarezom i deklaracijskog bloka koji se sastoji od liste deklaracija odvojenih točka-zarezom (;) zatvorenih u zagradama ({}). Svaka deklaracija se sastoji od svojstva, dvotočke (:) i vrijednosti (npr. `background-color: yellow`). Kako bi stil bio primjenjiv, potrebno je CSS datoteke uključiti u *HTML* dokument. Ekstenzije css datoteka su `.css`. [2]

2.2 Bootstrap

Bootstrap je besplatan *open-source front-end* okvir (*framework*) za izradu web stranica i aplikacija koji održava mali tim razvojnih programera na GitHubu. Sadrži HTML i CSS temeljne predloške za tipografiju, oblike, gumbe, navigaciju i ostale komponente sučelja, kao i dodatna JavaScript proširenja. Za razliku od mnogih web okvira, bavi se samo razvojem *front-end-a*. [3]

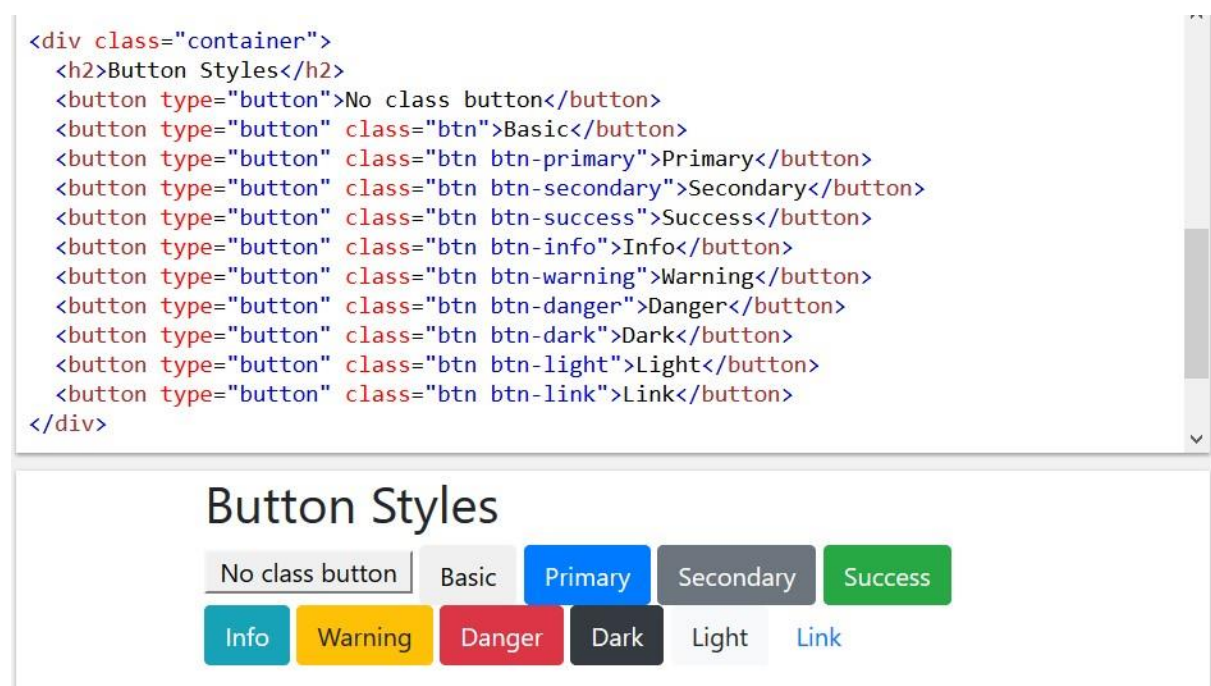
Prije Bootstrapa, koristile su se različite tehnologije što je vodilo do nedosljednosti i visokog tereta održavanja stranica i aplikacija, stoga su Mark Otto i Jacob Thornton, dizajneri i programeri na Twitter-u, stvorili „*Twitter Blueprint*“. Cilj je bio postizanje dosljednosti među korištenim alatima za razvoj sučelja. Nekoliko mjeseci nakon, mijenjaju naziv u Bootstrap i objavljuju ga kao *open-source* projekt. [4]

2012. izdan je Bootstrap 2, koji je dodao prilagodljivi sustav rešetke s dvanaest stupaca, ugrađenu podršku za *Glyphicons* (ikone), nekoliko novih komponenti, kao i promjene na mnogim postojećim komponentama. 2013. izdan je Bootstrap 3 koji je sadržavao *mobile-first* dizajn i redizajnirane komponente za upotrebu *flat* dizajna. *Mobile-first* dizajn znači da se prvo radi dizajn stranice za uređaje s najmanjom veličinom ekrana, npr. mobitele, te se kasnije taj dizajn prilagođava za uređaje s većim ekranima, a *flat* dizajn je stil dizajn sučelja s naglaskom na minimalnu uporabu stilskih elemenata koji daju iluziju tri dimenzije (kao što je korištenje sjena, gradijenta ili tekstura) i usmjeren je na minimalističku upotrebu jednostavnih elemenata, tipografije i boja. 2014. Mark Otto je najavio da je Bootstrap 4 u razvoju. Prva beta verzija Bootstrap 4 izdana je 2015, a stabilna inačica izdana je 2018. godine u osnovi poboljšavajući stilove ispisa, okvira (*border utilities*) i pružajući veću kontrolu nad *flexbox*

komponentama. U praktičnom dijelu ovog rada korištena je zadnja izdana verzija (Bootstrap 4). [3]

Korištenje Bootstrap tehnologije omogućuju nam dva dokumenta. Jedan dokument stila (*stylesheet*) koji uključujemo prije svih ostalih dokumenata stilova i drugi u kojem se nalazi JavaScript kôd (*script*) za upravljanje sučeljem. Također, da bi dinamičke komponente mogle funkcionirati, potrebno je uključiti jQuery i Popper.js datoteke prije JavaScript datoteke. Uključivanje ovih dokumenata u projekt može se izvršiti tako da ih se uključi putem poveznice (na Bootstrapovoj stranici nudi se i gotov početni predložak) ili skine na računalo. Kada se to izvrši, samo korištenje Bootstrapa je vrlo jednostavno.

Osnovni koncept je dodavanje klasa elementima koji na taj način preuzmu stil te klase definirane u Bootstrap dokumentima. (Slika 2.1)



Slika 2.1 Primjer klasa primijenjenih na element „button“ [14]

Osim toga, još jedna prednost ove tehnologije je što omogućava izradu stranica prilagodljivih različitim dimenzijama ekrana, a budući da danas korisnici pristupaju stranicama s različitih uređaja s različitim dimenzijama ekrana potrebno je stranicu prilagoditi tim veličinama. Bootstrap to automatski odrađuje i na taj način se programer ne mora brinuti

za to. Ako je sve dobro povezao, kod promjene veličine ekrana svi elementi se sami poslože i prilagode ekranu. [4]

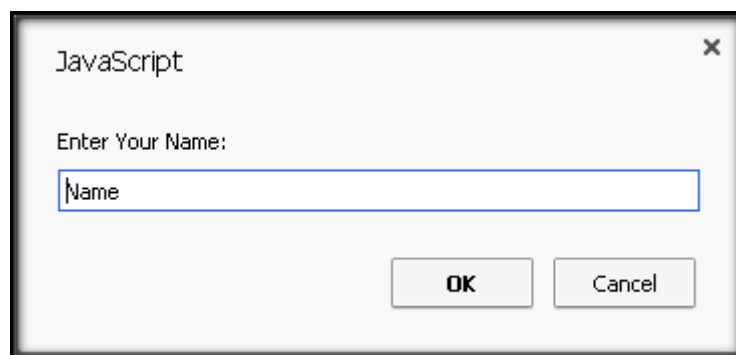
2.3 JavaScript

JavaScript je skriptni objektno-orijentirani programski jezik koji se izvršava u pregledniku na korisničkoj strani. To što je skriptni znači da se kôd ne kompajlira već izvršava u trenutku kad se pozove pomoću interpretera integriranog u web pregledniku. Razvila ga je tvrtka Netscape 1995. godine uz originalni naziv „*LiveScript*“. 1997. ga standardizira ECMA (*European association for standardizing information and communication systems*), a zadnja izdana verzija standarda je 7.0 2016. godine.[5]

Uz HTML i CSS, JavaScript je jedna od osnovne tri tehnologije World Wide Web-a. Omogućava izradu interaktivnih web stranica te tako i aplikacija kroz različite mogućnosti kao što su reagiranja na događaje (*events*), mijenjanje HTML elemenata preko DOM (*Document Object Model*) stabla i sl. Uključivanje JavaScript koda u HTML stranice vrši pomoću taga `<script>` te ih se može uključiti proizvoljan broj unutar *head* ili *body* dijela stranice. One navedene u *head* dijelu pišemo kada želimo da reagiraju na određeni događaj. Te skripte pretraživač učitava u memoriju bez izvršavanja i čeka da se pozovu, dok one u *body* dijelu izvrši kada na njih naiđe pri parsiranju HTML dokumenta. Također, kôd se ne mora direktno pisati unutar *script* taga, već se može smjestiti u zasebnu .js datoteku. U tom slučaju *script* tagu dodamo atribut *src* s imenom skripte. Kad parser u *body* dijelu naiđe na *script* tag, tada prestaje izvršavati HTML kôd i izvršava JS kôd, a isto pravilo vrijedi i ako naiđe na vanjsku uključenu skriptu. To može utjecati na performanse učitavanja stranice pa kako bi se to izbjeglo, skripta se uključuje na kraju HTML-a te se taj kôd dohvaća tek nakon učitavanja cijelog HTML koda, ali novi problem nastaje ako JS mijenja DOM stranice što onda dovodi do ponovnog renderiranja cijele stranice. Drugo rješenje je uključivanje vanjske skripte u *head* dio uz korištenje *async* i *defer* atributa. Asinkrone skripte ne blokiraju HTML kôd i kreiranje DOM stabla i kod njih nije određen redoslijed izvršavanja. *Defer* atribut također ne blokira parsiranje, ali za razliku od asinkronih, postoji redoslijed izvršavanja koji se određuje navođenjem skripti.[5]

Tipovi elemenata koji se razlikuju u JavaScriptu su varijable, literali, operatori, kontrolne strukture i objekti. Za razliku od statičkih programskih jezika u kojima je potrebno definirati tipove varijabli prije korištenja, u JavaScriptu se određuju dinamički prilikom izvršavanja aplikacije. Tip varijable se određuje na osnovu sadržaja. Može se deklarirati s ključnom riječi `var`. `Var` deklarira lokalnu varijablu (u funkciji) i bez nje je varijabla globalna. Također, varijable možemo deklarirati i ključnom riječi `let`. `Let` definira doseg varijable unutar bloka (`{ }`) u kojem je varijabla deklarirana.[5]

U JavaScriptu je moguće definirati tri vrste privremenih prozora: prozor za upozoravanje (*alert box*), prozor za potvrdu (*confirm box*) i prozor za unos (*prompt box*) (Slika 2.2). Funkcije koje se koriste definiraju se na početku dokumenta, u HEAD dijelu, te se na taj način osigurava da se učitaju prije poziva. Definiranje se vrši tako da se funkciji dodijeli ime, definiraju argumenti i u tijelu napišu sve naredbe koje želimo da funkcija izvrši.



Slika 2.2 Prozor za unos[15]

Objekti koji se razlikuju u JavaScriptu su ugrađeni i vlastiti. Neki od ugrađenih (predefiniranih) koji nisu vezani u DOM strukturu HTML dokumenta su niz, datum, `string`, `math` i `boolean`. Ugrađeni objekti vezani u DOM strukturu su `document`, `body`, `button`, `table`... i sl. Svaki JavaScript DOM objekt odgovara elementu DOM stabla u HTML dokumentu. JavaScript također uključuje i podršku za rad s preglednikom kroz pet ugrađenih objekata, a to su `navigator`, `screen`, `window`, `location` i `history`. Vlastiti objekti su oni koje korisnik sam stvara na dva načina. Jedan je stvaranjem direktne instance objekta korištenjem ugrađenog složenog tipa podataka `Object` (Slika 2.3), a drugi stvaranjem predloška objekt (Slika 2.4).

```

osobaObj=new Object();
osobaObj.ime="John";
osobaObj.prezime="Doe";
osobaObj.dob=50;
osobaObj.boja_ociju="blue"
function jelo(a)
{ alert("Ja jedem" + a);}
osobaObj.jelo=jelo;

```

Slika 2.3 Stvaranje direktne instance Object[5]

```

function osoba(ime,prezime,dob,boja_ociju)
{
    this.ime=ime;
    this.prezime=prezime;
    this.dob=dob;
    this.boja_ociju=boja_ociju;
}
function vrati_ime(){return this.ime;}
myFather=new osoba("John","Doe",50,"blue");
alert(myFather.vrati_ime());
myMother=new osoba("Sally","Rally",48,"green");

```

Slika 2.4 Korištenje predloška objekt[5]

Objekt *Event* definira događaj koji se dogodio na HTML DOM elementu. Ne podržavaju svi elementi sve događaje. Obično se događaji kombiniraju s nekom funkcionalnošću isprogramiranom u funkciji. Događaju se pristupa preko predefiniranih atributa koji označavaju određeni događaj (onclick, onkeypress...).[5]

2.4 NodeJS

NodeJS je *open-source* platforma temeljena na JavaScriptu i izgrađena pomoću Google Chrome JavaScript V8 *engine*. Koristi se za razvijanje I/O (ulazno/izlaznih) web aplikacija. Potpuno je besplatan i koriste ga tisuće programera dijelom svijeta. [6]

Razvio ga je Ryan Dahl 2009. godine (otprilike trinaest godina nakon predstavljanja prvog JavaScript okruženja na serverskoj strani). Dahl je kritizirao ograničene mogućnosti Apache HTTP Servera poput kodiranja kod kojeg kôd blokira cijeli proces, potrebu višestruke izvršne snage u slučaju izvršavanja istodobnih veza i nemogućnost mnogo istovremenih povezivanja (do 10 000 i više).

NodeJS omogućava da se aplikacija može pokrenuti i kao samostalna aplikacija na računalu, ne samo pomoću preglednika. Tako s JavaScriptom možemo stvoriti puno više od same interaktivne web stranice. [7]

2.4.1 V8 engine

V8 je Googleov JavaScriptov *engine* s visokim performansama otvorenog koda, napisan u C++. Upotrebljava se u Google Chromeu, pregledniku otvorenog koda iz Googlea i među ostalima u NodeJS. Implementira ECMAScript kako je navedeno u ECMA-262 i radi na Windows 7 ili novijim, MacOS 10.12+ i Linux sustavima koji koriste IA-32, ARM ili MIPS procesore. V8 se može izvoditi samostalno ili se može ugraditi u bilo koju aplikaciju C++. [8]

2.4.2 Neblokirajući način rada i petlja događaja

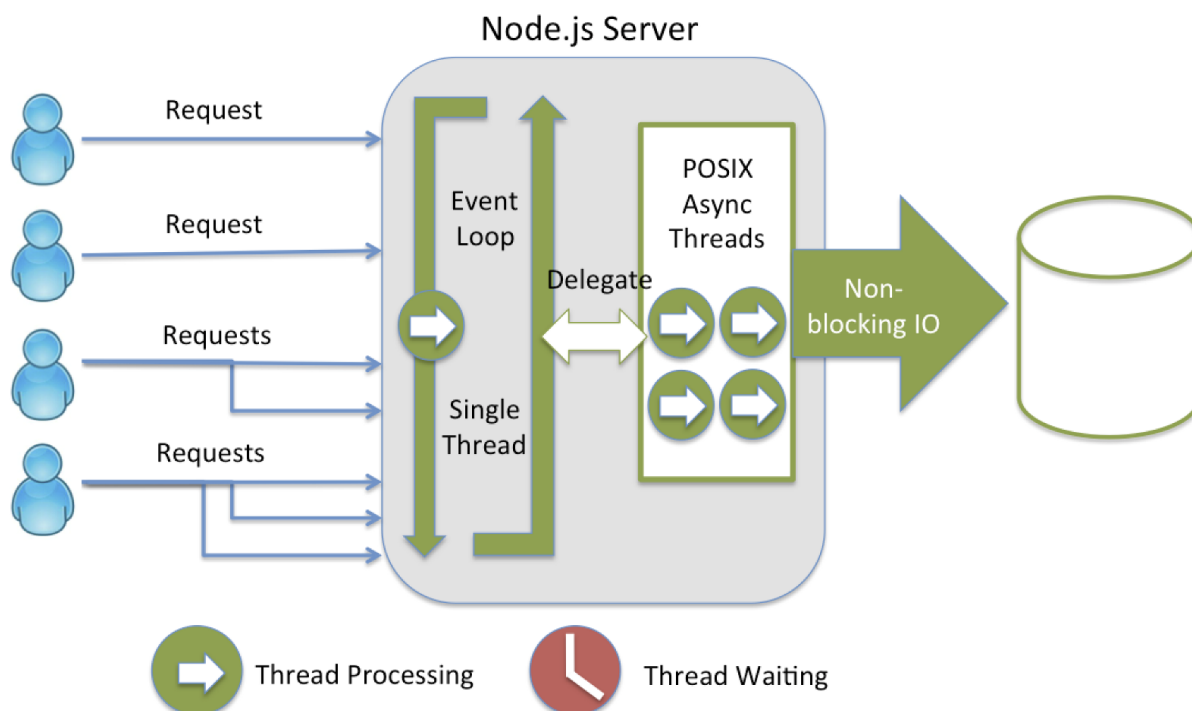
Prednost NodeJS-a je što koristi neblokirajući (asinkroni) način rada. Događaj koji se poziva ne blokira proces, već se izvršavanje nastavlja. To ga čini laganim i učinkovitijim.

Općenito, I/O se odnosi na ulaze i izlaze, sve od čitanja/pisanja lokalnih datoteka do izrade HTTP zahtjeva *API*-ju, čije izvršavanje traje dugo i blokira druge funkcije. Kod tradicionalnih blokirajućih modela, čeka se izvršavanje zahtjeva prvog korisnika i tek kad se

taj zahtjev izvrši i obradi, izvršava se idući. Ako se radi o web poslužitelju, započinje se nova nit kod svakog novog korisnika i ukoliko su sve niti zauzete, mora se čekati na otvaranje nove niti.

Kod neblokirajućeg modela (Slika 2.5), nije potrebno čekati izvršavanje jednog zahtjeva da bi se pokrenuo novi. NodeJS koristi samo jednu nit i temelji se na pokretanju događaja s povratnim pozivima funkcija zbog signalizacije završetka zadatka. Takve funkcije nazivaju se *callback* funkcije, a pozivi funkcija *callback* pozivi.

Pozivanje događaja izvršava se kroz petlju događaja. To je konstrukt koji čeka i odašilje poruke i događaje. JavaScript ima jedan pozivni stog u kojem prati što funkcija trenutačno provodi i koja će funkcija biti izvršena nakon toga. Kad se namjerava izvršiti funkcija, dodaje se na pozivni stog. Zatim, ako ta funkcija pozove neku drugu funkciju - druga će funkcija biti na vrhu prvog u pozivnom stogu. Petlja događaja je neprekidni proces koji provjerava je li pozivni stog prazan. Ako se u redu čekanja događaja nešto čeka, premjestit će se u pozivni stog. Ako ne, onda se ništa ne događa. Odvajanje pozivatelja funkcije od odgovora omogućava da se izvode druge stvari dok se čeka da operacija obavi rad i da se pokrene njezin *callback*. [9]



Slika 2.5 Način rada NodeJS[16]

2.4.3 NPM

Npm (*Node Package Manager*) je upravitelj paketa za NodeJS pakete ili module. Sastoji se od tri glavne komponente: Internet stranice, sučelja komande linije i registra. Sadrži više od 600 000 paketa koji pomažu bržem razvoju aplikacija. U paketu se nalaze sve datoteke koje su potrebne za modul, a moduli su JavaScript biblioteke koje se mogu uključiti u projekt.

U NodeJS-u, paket je datoteka ili direktorij koji opisuje package.json. Paket package.json definira paket, a modul je bilo koja datoteka ili direktorij koju NodeJS može učitati pomoću funkcije `require()`. Mjesto u kojem se nalaze moduli koje NodeJS traži je mapa `node_modules`.

Na primjer, ako se stvori datoteka na `node_modules / foo.js`, a zatim postoji program koji sadrži `var f = require ('foo.js')`, učitao bi se modul. Međutim, `foo.js` neće biti "paket" u ovom slučaju jer nema package.json. S druge strane, ako se stvori paket koji nema datoteku `index.js` ili "glavno" u datoteci package.json, onda to nije modul. Čak i ako je instaliran u `node_modules`, ne može biti argument za `require()`.

Instaliranje, tj. skidanje paketa u projekt provodi se na način da se u komandnoj liniji upiše `npm install <ime_paketa>` (Slika 2.6). Sukladno tome, uklanjanje paketa vrši se upisivanjem `npm uninstall <ime_paketa>`. [10]

```
> npm install <package_name>
```

Slika 2.6 Instaliranje paketa[17]

Neki od najpoznatijih i najkorištenijih NodeJS paketa su ExpressJS, Async.js, Request – Simplified HTTP Client, Passport, React, Pug, MongoDB, Angular...i dr. [11]

2.4.3.1 ExpressJS

Express je minimalan i fleksibilan okvir za web aplikacije NodeJS koji pruža robusni skup značajki za razvoj web i mobilnih aplikacija. Omogućuje brz razvoj mrežnih aplikacija

baziranih na NodeJS-u. Neke od osnovnih značajki *Express* okvira su omogućavanje postavljanja *middlewarea* za odgovaranje na HTTP zahtjeve, određivanje tablice usmjerenja koja se koristi za izvođenje različitih radnji na temelju HTTP metode i URL-a i omogućavanje dinamičkog prikazivanja HTML stranica prosljeđivanjem argumenata predlošcima. [10]

Komanda za instalaciju Express paketa je `>npm install express`. Sljedeći dio koda (Slika 2.7) prikazuje jednostavan primjer Hello World NodeJS programa uz Express dodatak:

```
const express = require('express') // uključivanje modula
express
const app = express()

app.get('/', (req, res) => res.send('Hello World!')) //
usmjerenje

app.listen(3000, () => console.log('Example app listening on
port 3000!')) // pozivanje servera na portu 3000
```

Slika 2.7 Hello World program[17]

2.5 MongoDB

MongoDB je vodeća *NoSQL open-source* baza podataka koja koristi *document-oriented* model podataka. Zapis u MongoDB-u (Slika 2.8 MongoDB zapis podataka) je dokument koji je struktura podataka sastavljena od parova polja i vrijednosti. MongoDB dokumenti su slični JSON objektima. Vrijednosti polja mogu uključivati druge dokumente, polja i polja dokumenata.

```
{
  name: "sue",
  age: 26,
  status: "A",
  groups: [ "news", "sports" ]
}
```

← field: value
← field: value
← field: value
← field: value

Slika 2.8 MongoDB zapis podataka [18]

Unutar instance MongoDB može postojati nula ili više baza podataka, od kojih svaka služi kao mjesto pohrane za ostalo. Baza podataka može imati nula ili više zbirki. Zbirka se može smatrati poput tablice u *SQL* bazi te može biti indeksirana, što poboljšava pretraživanje i razvrstavanja. Ona se sastoji od nula ili više dokumenata. Opet, dokumente se može smatrati poput redova. Svaki dokument se sastoji od jednog ili više polja, što odgovara stupcima. Podatke dobivljamo pomoću pokazivača koji omogućava lakše prebrojavanje, preskakanje i druge funkcionalnosti za obradu podataka.

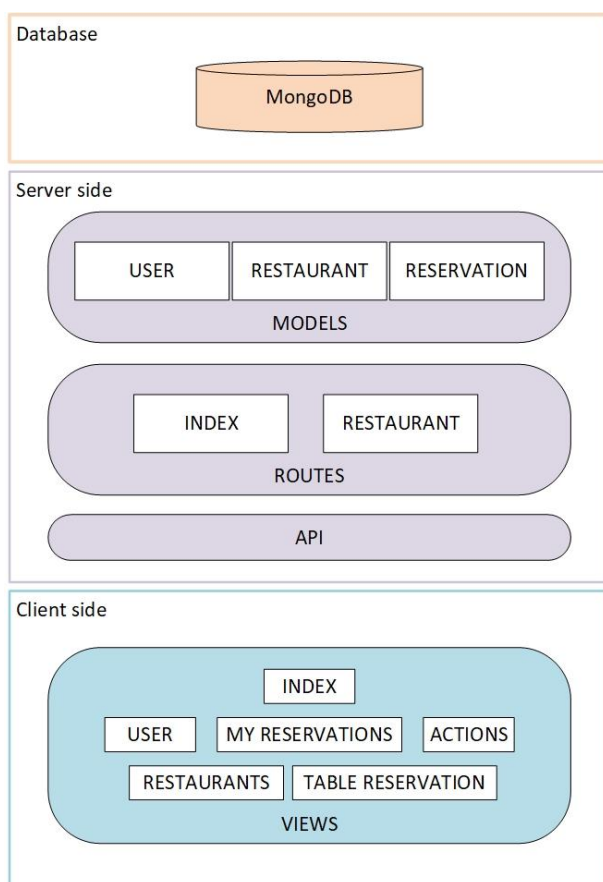
Bazom se upravlja pomoću terminala unutar kojeg se pokreće baza. Na bazu se spaja naredbama `mongod` i `mongo`, a zatim uz pomoć određenih funkcija poziva se određena baza koja se želi koristiti ili stvoriti te kolekcije koje se želi uređivati. Da bi se prikazale sve baze koje postoje u sustavu koristi se naredba `show collections`. Osnovna funkcija za korištenje baze je `use <ImeBaze>`. Na taj način se ulazi u bazu, odnosno ona se stvara ukoliko ne postoji. Naredba za stvaranje kolekcije je `db.createCollection('ImeKolekcije')`, a kako bi joj se pristupilo u bazi, koristi se `db.<ImeKolekcije>.<funkcija>`. Ukoliko se žele ispisati svi podaci, unutar sučelja terminala unosi se `db.<ImeKolekcije>.find({})`. Ako se traži samo jedan određeni podatak, pretražujemo ga na način da upišemo atribut/e unutar `find`: `db.<ImeKolekcije>.find({ atribut: vrijednost })`. Osim funkcije `find`, MongoDB nudi i ostale poput: `findOne`, `insert`, `delete`, `update`, `count` i druge. [12]

3 REALIZACIJA PRAKTIČNOG DIJELA RADA

Zadatak praktičnog dijela rada je izrada web aplikacije za rezervaciju stolova u restoranu. U sljedećem poglavlju opisati ćemo na koji način je realizirana. Aplikacija se sastoji od *front-end* (klijentskog) i *back-end* (serverskog) dijela. *Front-end* dio prezentira podatke, dok se *back-end* dio bavi spajanjem na server i pohranjivanjem informacija. Aplikacija radi na način da dohvaća podatke sa servera i prikazuje ih klijentu.

3.1 Pregled sustava

RezervirajStol.hr je klijent-server web aplikacija čija je svrha da korisniku omogućiti pristup informacijama ponuđenih restorana, pregleda njihove ponude i jelovnike te rezervira slobodan stol ili stolove željenog datuma u željenom terminu. Komunikacija između klijenta i servera obavlja se pomoću HTTP protokola. Klijent preko mreže šalje zahtjeve web aplikaciji koja komunicira s bazom preko poslužitelja (*Slika 3.1*).



Slika 3.1 Arhitektura sustava

3.2 Klase korisnika

Razlikujemo tri različite klase korisnika koji se mogu služiti aplikacijom. Tablica 1 opisuje funkcije pojedine klase:

Tablica 1 Opis klase korisnika

KLASA KORISNIKA	OPIS
Gost	Korisnik koji može pregledati naslovnu stranicu, ali nema pristup informacijama o restoranima, kao ni mogućnost rezervacije. Može obaviti registraciju u sustav
Korisnik	Korisnik koji ima pristup naslovnoj stranici, svom korisničkom računu, ponuđenim informacijama o restoranima te mogućnost rezervacije
Administrator	Korisnik koji ima pristup svim informacijama aplikacije, uključujući provjeravanje rada baze podataka i ažuriranje cijelog sustava

3.3 Korisnički zahtjevi

Korisnički zahtjevi detaljno prikazuju zahtjeve na jednostavan način kako bi ih krajnji korisnici mogli razumjeti. To bi bile usluge koje korisnik zahtjeva od aplikacije te ograničenja pod kojima sustav treba funkcionirati. Dijelev se na funkcionalne i nefunkcionalne.

Funkcionalni zahtjevi predstavljaju poslovne procese koje sustav pokriva, tj. funkcionalnosti koje sustav treba imati te podatke koje treba prihvaćati, obrađivati, pohranjivati i pružati (obrada ulaza i izlaza).

Nefunkcionalni zahtjevi odnose se na ponašanje sustava u vidu performansi, sigurnosti i mogućnosti korištenja (ograničenja pri pojedinim akcijama u radu).

3.3.1 Funkcionalni zahtjevi

Funkcionalni zahtjevi gosta:

- Pregled naslovne stranice – gost može pregledati informacije koje se nalaze na početnoj stranici aplikacije
- Registracija u sustav – gost može otvoriti korisnički račun u aplikaciji, nakon kojeg postaje korisnik aplikacije. Aplikacija mora zatražiti neke osnovne korisničke podatke.

Funkcionalni zahtjevi korisnika:

- Prijava u sustav – korisnik se može prijaviti u sustav sa svojim korisničkim računom
- Pregled naslovne stranice – korisnik može pregledati informacije koje se nalaze na početnoj stranici aplikacije
- Pregled akcija – korisnik može pregledati nove akcije koje se nude u restoranima
- Pregled restorana – korisnik može pregledati ponuđene restorane u aplikaciji i njihove osnovne informacije
- Pregled jelovnika pojedinog restorana – korisnik može pregledati jelovnik svakog restorana
- Mogućnost rezervacije – korisnik može obaviti rezervaciju kroz odabir stola u određenom restoranu u željenom terminu
- Pregled osobnih rezervacija – korisnik može pregledati sve rezervacije koje je obavio od trenutka stvaranja korisničkog računa
- Odjava iz sustava – korisnik se može odjaviti iz sustava te tako postaje gost

Funkcionalni zahtjevi administratora:

- Prijava u sustav – administrator se može prijaviti u sustav sa svojim imenom i lozinkom
- Upravljanje akcijama – administrator ima mogućnost pregleda akcija restorana, ali i dodavanja novih i brisanja isteklih
- Upravljanje restoranima – administrator ima mogućnost pregleda i brisanja restorana u aplikaciji te dodavanja novih restorana
- Pregled jelovnika pojedinog restorana – administrator ima mogućnost pregleda jelovnika svakog restorana

- Pregled rezervacija svakog restorana – administrator ima mogućnost pregleda svih rezervacija, svakog restorana
- Odjava iz sustava – administrator se može odjaviti iz sustava i postati gost aplikacije

3.3.2 Nefunkcionalni zahtjevi

Nefunkcionalni zahtjevi aplikacije zahtijevaju:

- Pristup mreži – kako bi se aplikacija mogla koristiti i komunicirati sa serverskom stranom, korisnici trebaju biti povezani na mrežu
- Kompatibilnost web preglednika – aplikacija treba funkcionirati u svakom pregledniku
- Jednostavnost korištenja – sustav treba biti dovoljno jednostavan za korištenje krajnjem korisniku kako bi se on snašao bez veće obuke
- Sigurnost – korisnički podatci trebaju biti zaštićeni od zloupotrebe
- Skalabilnost – sustav treba biti skalabilan kako bi po potrebi mogao dodavati nove informacije kao što su novi restorani, rezervacije, korisnički računi...
- Dostupnost – sustav bi trebao biti dostupan u svakom trenutku

3.4 Slojevi aplikacije

Najčešća podjela web aplikacije jest na strukturu od tri razine. Prva razina (prezentacijski sloj) predstavlja korisnička sučelja i prezentaciju podataka. Druga razina bila bi aplikacijska logika, odnosno definiranje načina komunikacije korisničkog sučelja s bazom podataka. Baza podataka predstavlja treću razinu, pohranu podataka. Na taj način realiziran je i sustav RezervirajStolHr. Takva podjela sustava, osim što omogućava veću organiziranost i modularnost, pruža jednostavnije održavanje sustava te njegovo ažuriranje.

3.4.1 Klijentski dio

U razvoju klijentskog dijela aplikacije korišten je Bootstrap *framework* kao glavni HTML i CSS *framework* uz dijelove koda posebno programiranih u JavaScriptu. Kako bi se

lakše komuniciralo s poslužiteljskom stranom, HTML kôd je pretvoren u pug datoteku. Pug je *view-engine* koji omogućava renderiranje stranica i zamjenu ili umetanje varijabla unutar HTML-a. Glavna zadaća ove razine je prezentiranje podataka i interakcija s korisnikom.

Organizacija klijentskog dijela vrši se po direktorijima. Glavni direktoriji su *assets* i *views*. *Assets* sadrži direktorije s css i javascript kodovima (stilovi i skripte) te direktorij *img* u kojoj su smještene slike korištene unutar aplikacije. *Views* datoteka, kako joj ime govori, sadrži pogleda. Pogledi su načini prikaza podataka. Unutar nje smještene su pug datoteke.

3.4.2 Serverski dio

Za razvoj serverske strane aplikacije korišten je NodeJS uz pomoć Express dodatka. Glavna zadaća ove razine je definiranje komunikacije prezentacijskog sloja s bazom podataka. Organizacija serverske strane podijeljena je na glavna dva dijela: rute i modeli.

Rute predstavljaju definirane putove pomoću kojih se izvršavaju funkcije na serverskoj strani kada klijentska strana aplikacije zatraži određeni zahtjev ili put. Realizirane su kroz HTTP metode, od kojih su najčešće korištene GET i POST metode. GET metode koriste se za dohvaćanje resursa (npr. HTML datoteka), a POST metode koriste se za slanje dijela poruke s informacijama serveru poput podataka unesenih u formu koje dalje dodajemo u bazu podataka dok se dohvaća određeni resurs.

Slika 3.2 prikazuje primjer poziva funkcija kroz definiranu rutu na serverskoj strani.

```
router.post('/login', function(req, res) {
  if (req.body.username && req.body.password) {
    User.authenticate(req.body.username, req.body.password, function (error, user) {
      if (!user) {
        res.redirect('/');
      } else {
        req.session.uniqueID = user.username;
        res.redirect('/user');
      }
    });
  }
});
```

Slika 3.2 Primjer rute

Kako bi se olakšao rad s bazom podataka, koriste se modeli. Modeli su strukture podataka (*Slika 3.3*) koji predstavljaju shemu zapisa unutar baze. Omogućavaju lakšu manipulaciju i rad s podacima, a osim sheme, sadrže i osnovne funkcije za rad s bazom.

```
var UserSchema = new mongoose.Schema({
  username: {
    type: String,
    required: true,
    unique: true,
    trim: true
  },
  password: {
    type: String,
    required: true
  }
  email:{
    type: String
    required: true,
    unique: true,
    trim: true
  }
});
```

Slika 3.3 Primjer modela korisnika

3.4.3 Baza podataka

Kako se radi o NoSQL bazi podataka, podaci unutar baze spremljeni su u JSON obliku kao što je prikazano pomoću modela mongoose sheme. Razlikujemo 3 kolekcije u bazi s kojima radi aplikacija, a to su users, reservations i restaurants. Users kolekcija sadrži informacije o korisnicima (korisničko ime, lozinku i e-mail adresu), reservations o rezervacijama (korisničko ime, datum, vrijeme i rezervirani stol), a restaurants općenite informacije o pojedinom restoranu, kao i popis stolova koji se u njemu nalaze.

3.4.3.1 Sigurnost podataka

Sigurnost podataka korisnika, jedan je od nefunkcionalnih zahtjeva aplikacije. Korisniku je potrebno omogućiti zaštitu njegovih osobnih podataka unutar baze da ne bi došlo do neovlaštenog korištenja korisničkih profila.

Zaštita podataka korisnika vrši se kroz kriptiranje lozinki pri registraciji korisnika prije samog spremanja podataka u bazu (Slika 3.4).

```
// hash password before saving to database
UserSchema.pre('save', function(next) {
  var user = this;
  bcrypt.hash(user.password, 10, function(err, hash) {
    if (err) {
      return next(err);
    }
    user.password = hash;
    next();
  })
});
```

Slika 3.4 Kriptiranje lozinke

Kriptiranje lozinki omogućava bcrypt modul. Bcrypt je funkcija šifriranja koja ima ugrađenu vrijednost nazvanu sol (slučajni fragment koji se koristi za generiranje *hash*-a povezanog s lozinkom koji je spremljen s njom u bazi podataka). To sprječava da dvije identične lozinke generiraju isti *hash* i probleme koji to podrazumijevaju (poput napada na lozinke). Sa soli se dodaje stupanj složenosti koji sprječava da *hash* povezan s lozinkom ne bude jedinstven. Stupanj složenosti se naziva *saltRounds* i njegova se vrijednosti može proizvoljno odrediti. Što je taj broj veći, potrebno je više vremena da uređaj izračuna hash povezan s lozinkom. Pri odabiru ove vrijednosti, važno je izabrati dovoljno visok broj da bi dešifriranje lozinke za provalnika bilo što teže i dugotrajnije, a s druge strane, mora biti dovoljno malen da ne prekine korisnikovo strpljenje prilikom registracije i prijave. Prema zadanim postavkama, vrijednost *saltRounds* je 10. [13]

3.5 Korisnička sučelja

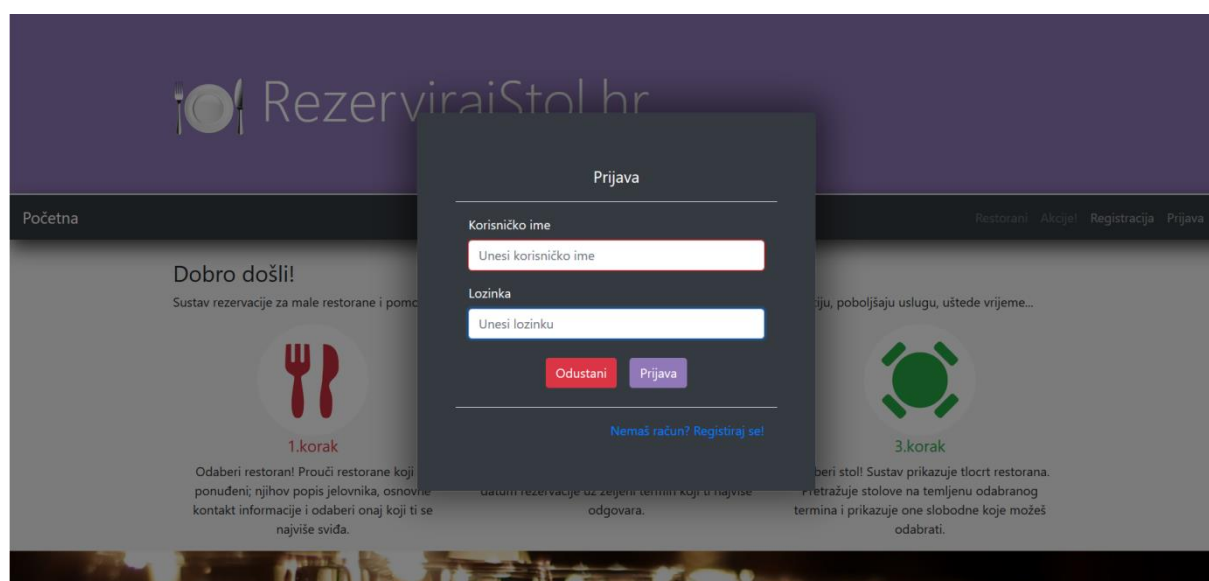
Korisnička sučelja trebaju biti jednostavna za uporabu te se korisnici trebaju znati služiti njima bez veće obuke. RezervirajStol.hr sustav sadrži sučelja za javne korisnike (goste), korisnike aplikacije i administratora. Sučelja za korisnike aplikacije i administratora su privatna te im se može pristupiti samo prijavom u sustav.

3.5.1 Javno sučelje

Javnom sučelju (*Slika 3.5*) imaju pristup svi javni korisnici (gosti). To je početna pozdravna stranica koja se otvara kada se uđe u aplikaciju. Iz javnog sučelja moguće je obaviti prijavu (*Slika 3.6*) ili registraciju kako bi se pristupilo privatnim sučeljima.

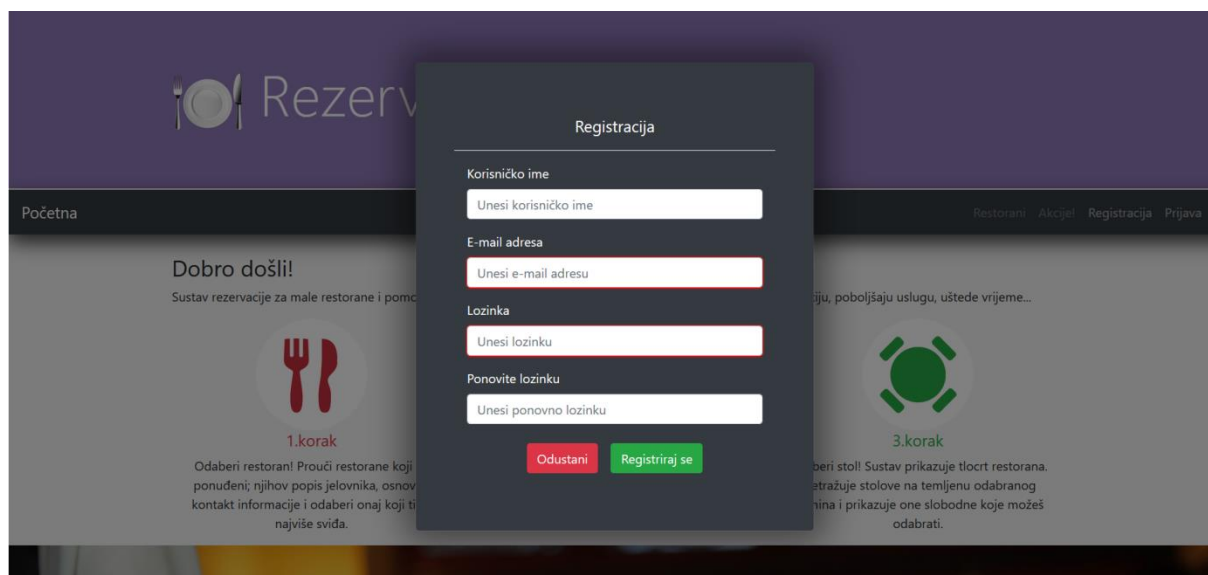


Slika 3.5 Sučelje za gosta



Slika 3.6 Sučelje za prijavu

Registracija korisnika (*Slika 3.7*) obavlja se unošenjem osnovnih informacija koje sustav zahtjeva. Sustav provjerava jesu li sva polja unesena te da li je lozinka pravilno potvrđena. Ukoliko je sve ispravno odrađeno, stvara se novi korisnik koji se upisuje u bazu podataka i ima ovlašten pristup privatnim sučeljima koje može koristiti.

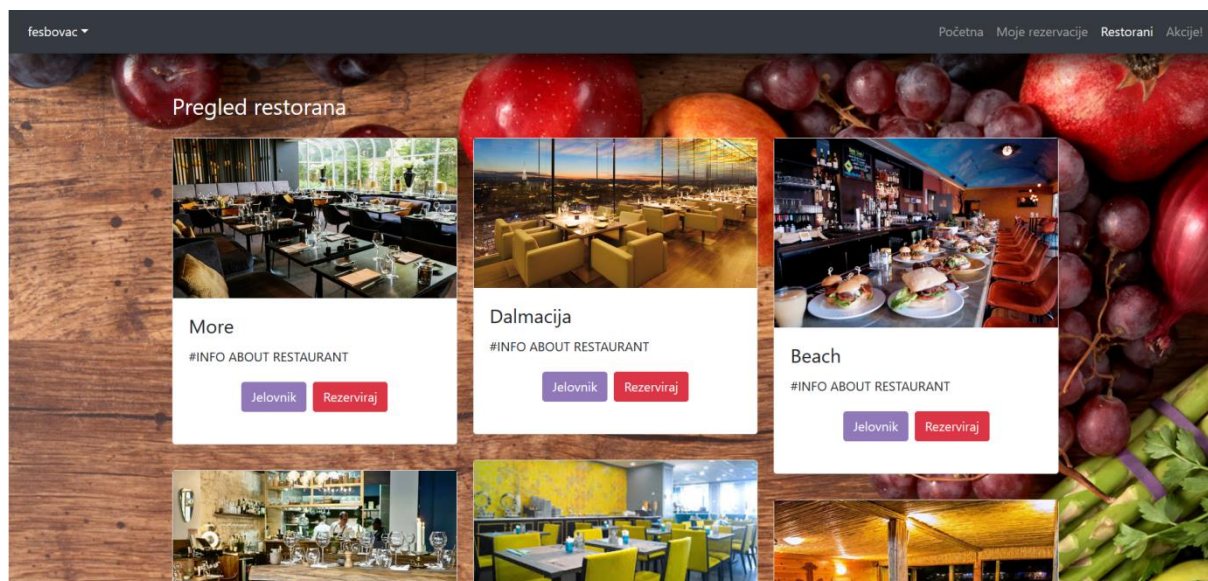
The image shows a web browser window with a dark purple header. The word 'Rezervacija' is partially visible in the top left. A dark grey modal box titled 'Registracija' is centered on the screen. It contains four input fields: 'Korisničko ime' (User name), 'E-mail adresa' (Email address), 'Lozinka' (Password), and 'Ponovite lozinku' (Repeat password). Each field has a placeholder text 'Unesi...' (Enter...). Below the fields are two buttons: a red 'Odustani' (Cancel) button and a green 'Registriraj se' (Register) button. The background of the website is visible through the modal, showing a 'Dobro došli!' (Welcome!) message and a '1.korak' (Step 1) section with a fork and knife icon.

Slika 3.7 Sučelje za registraciju

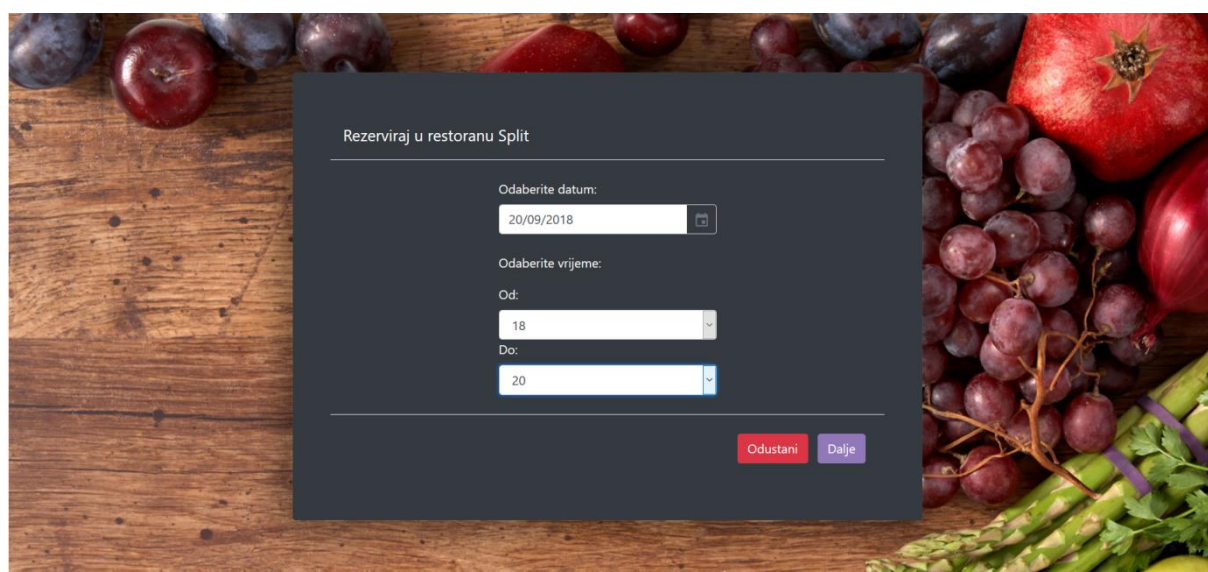
3.5.2 Privatna sučelja

Nakon prijave u sustav, omogućen je pristup privatnim sučeljima. Razlikuju se sučelja za korisnike i administratora. Administratorska sučelja sadrže dodatne funkcionalnosti koje nemaju sučelja prijavljenih korisnika.

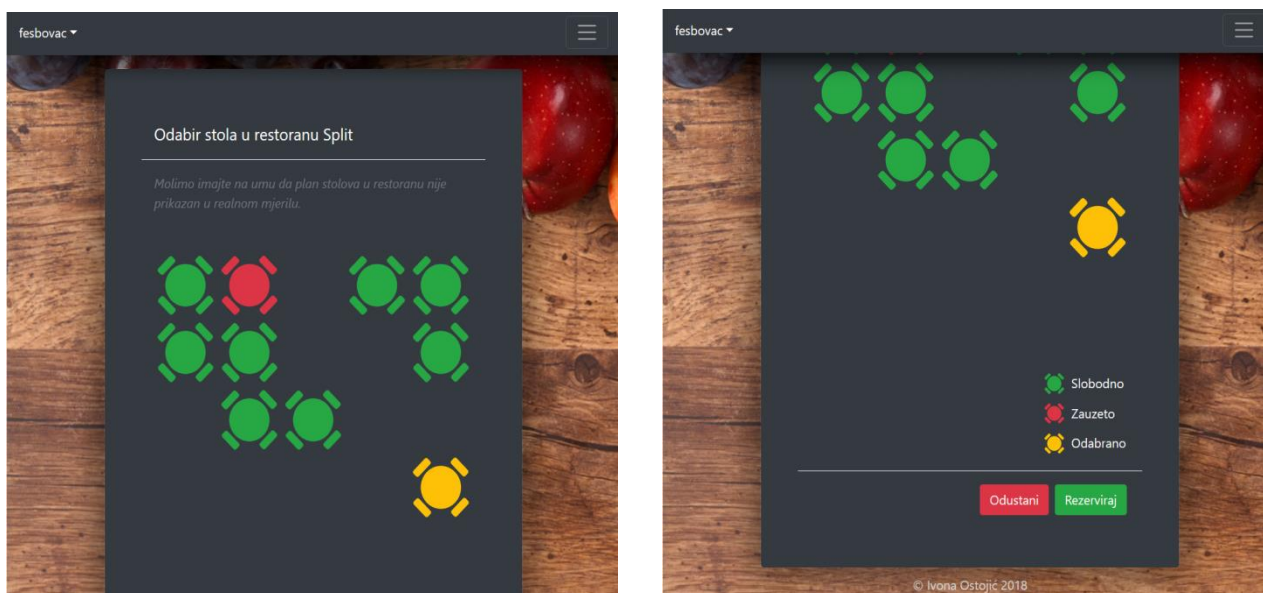
Postupak obavljanja rezervacije odvija se kroz nekoliko sučelja. Prvo korisničko sučelje je ono u kojem korisnik odabire restoran (*Slika 3.8*). Nakon njega ulazi u sučelje za odabir termina rezervacije (*Slika 3.9*) i zatim u sučelje za odabir stola (*Slika 3.10*).



Slika 3.8 Sučelje za prikaz restorana

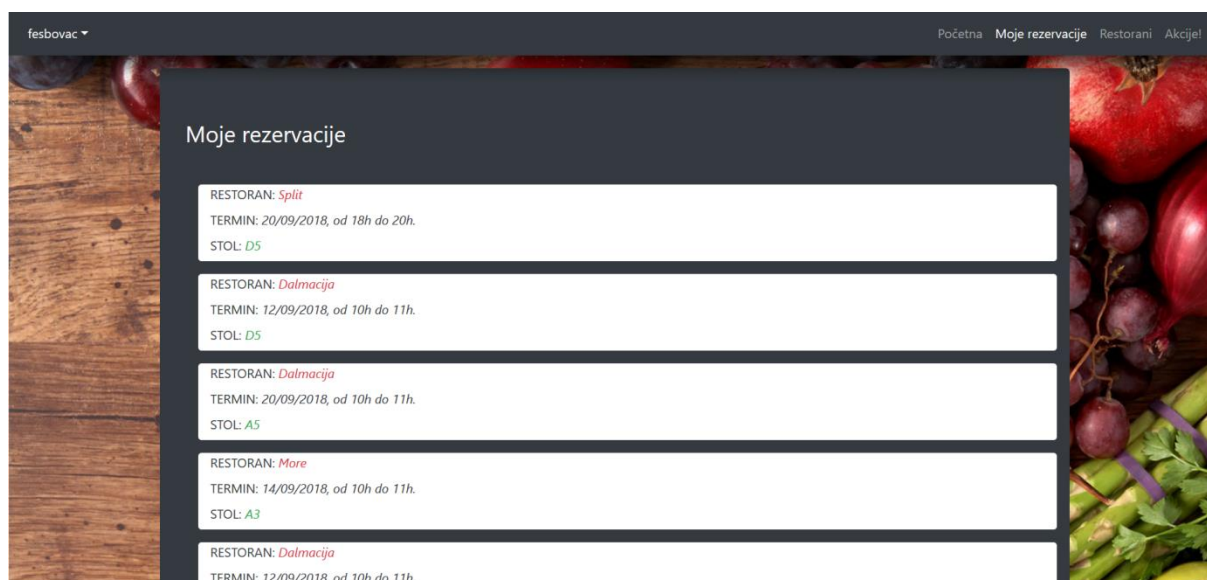


Slika 3.9 Sučelje za odabir termina



Slika 3.10 Sučelje za odabir stola

Kada je rezervacija uspješno obavljena, dodaje se na popis rezervacija. Popis osobnih rezervacija korisnik može pregledati u sučelju „Moje rezervacije“ gdje se ona posljednja nalazi na vrhu liste (Slika 3.11).



Slika 3.11 Sučelje za prikaz osobnih rezervacija

4 ZAKLJUČAK

Napretkom Interneta i web tehnologija, od stvaranja statičkih robusnih web stranica pomoću samog HTML-a, do razvoja CSS-a koji je omogućio uređivanje stranica, pa sve do JavaScripta koji je olakšao prikazivanje dinamičkog sadržaja, došlo je i do razvoja raznih web tehnologija koje su omogućile veći razvoj web aplikacija. Većinom sve tehnologije koje se koriste su *open-source* tipa i dostupne su svima na Internetu. Pomoću njih programeri diljem svijeta s lakoćom kreiraju nove aplikacije sve boljeg dizajna i novih funkcionalnosti te one postaju brže i jednostavnije za korištenje kako bi privukle sve veći broj korisnika.

Ovaj rad namijenjen je upoznavanju s nekim od osnovnih web tehnologija. HTML i CSS su osnova na kojoj je temeljen Bootstrap *framework* korišten na prezentacijskoj strani aplikacije. Gotove komponente Bootstrapa, ubrzale su stvaranje kostura aplikacije. Također, JavaScript, s kojim je realizirano prikazivanje dinamičkog sadržaja *front-enda*, osnova je NodeJS-a pomoću kojeg je stvorena serverska strana sustava. Korištenje NodeJS-a olakšao je i Express npm dodatak. Osim upoznavanja s navedenim tehnologijama, tu je i MongoDB *NoSQL* baza podataka unutar koje su podaci spremljeni u obliku vrlo sličnom JSON objektima.

Kroz izradu aplikacije RezervirajStol.hr prikazano je kako se kombinacijom Bootstrapa, JavaScripta, NodeJS-a i MongoDB baze podataka može stvoriti jednostavan i praktičan sustav.

LITERATURA

- [1] „HTML“, s Interneta, <https://hr.wikipedia.org/wiki/HTML>
- [2] Štula, M: „Klijentske web tehnologije – CSS“, Split, 2018.
- [3] „Bootstrap(front-end framework)“, s Interneta,
[https://en.wikipedia.org/wiki/Bootstrap_\(front-end_framework\)](https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework))
- [4] „About Bootstrap“, s Interneta, <https://getbootstrap.com/docs/4.0/about/overview/>
- [5] Štula, M: „Klijentske web tehnologije – JavaScript“, Split, 2018.
- [6] „Node.js“, s Interneta, https://en.wikipedia.org/wiki/Node.js#Platform_architecture
- [7] „What exactly is Node.js?“, s Interneta, <https://medium.freecodecamp.org/what-exactly-is-node-js-ae36e97449f5>
- [8] „Chrome V8“, <https://developers.google.com/v8/>, s Interneta
- [9] „Node.js Event Loop“, s Interneta, <https://nodejs.org/en/docs/guides/event-loop-timers-and-nexttick/>
- [10] „npm“, s Interneta, <https://docs.npmjs.com/>
- [11] „Top 30 NPM packages for Node.js Developers 2016“, s Interneta,
<https://colorlib.com/wp/npm-packages-node-js/>
- [12] Seguin, K. „The little MongoDB book“, 2015.
- [13] „Hashing passwords with NodeJS and MongoDB: bcrypt“,
<https://solidgeargroup.com/hashing-passwords-nodejs-mongodb-bcrypt>, s Interneta

SLIKE

- [14] https://www.w3schools.com/bootstrap/bootstrap_buttons.asp, s Interneta
- [15] https://www.w3schools.com/js/js_popup.asp, s Interneta
- [16] <https://codeburst.io/all-about-node-js-you-wanted-to-know-25f3374e0be7>, s Interneta
- [17] <https://docs.npmjs.com/>, s Interneta
- [18] <https://docs.mongodb.com/manual/core/document/>, s Interneta

SAŽETAK

Tema ovog završnog rada je razvoj web aplikacije za rezervaciju stolova u restoranima. Sve primijenjene tehnologije pri izradi sustava su detaljno definirane i objašnjene u prvom dijelu rada, dok se u drugom nalazi opis realizacije praktičnog dijela uz primjenu naučenih tehnologija. Za izradu klijentske strane aplikacije korišten je Bootstrap (HTML i CSS *framework*) u kombinaciji s JavaScriptom, a na serverskoj strani NodeJS platforma pomoću Express dodatka. Za pohranjivanje podataka zadužena je MongoDB *NoSQL* baza. Stvoreni sustav prilagodljiv je svim web preglednicima te je jednostavan i praktičan za korištenje.

KLJUČNE RIJEČI

Web aplikacija, Bootstrap, JavaScript, NodeJS, MongoDB

SUMMARY

The subject of this paper is the development of web application used for booking tables in restaurants. All the applied technologies in the design of the system are defined and explained in detail in the first part of the paper, while the rest is the description of the realization of the practical part accomplished by the application of the learned technologies. For creating the client side of the application, Bootstrap (HTML and CSS framework) was used in combination with JavaScript, while the server side was made by NodeJS platform with the Express Plugin. The MongoDB NoSQL database is used for storing the data. The system is designed to be customizable to all web browsers and is simple and convenient to use.

TITLE

BookTable.hr

KEY WORDS

Web application, Bootstrap, JavaScript, NodeJS, MongoDB