Internet aplikacija za naručivanje hrane

Dželajlija, Boris

Undergraduate thesis / Završni rad

2018

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:215401

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-11-30**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORNACIJSKIH TEHNOLOGIJA OSIJEK

Stručni studij

INTERNET APLIKACIJA ZA NARUČIVANJE HRANE

Završni rad

Boris Dželajlija

Osijek, 2018.

SADRŽAJ

1. UVOD. ..... **Error! Bookmark not defined.**

1.1. Zadatak završnog rada. ..... **Error! Bookmark not defined.**

2. PREGLED KORIŠTENIH TEHNOLOGIJA. ..... **Error! Bookmark not defined.**

2.1. HTML5. ..... **Error! Bookmark not defined.**

2.2. CSS3. 3

2.3. JavaScript 4

2.4. jQuery. 5

2.5. Bootstrap. 5

2.6. NodeJS. 6

2.7. SQLite. 6

3. RAZVOJ APLIKACIJE ZA NARUČIVANJE HRANE. 7

4. ZAKLJUČAK 20

LITERATURA 21

SAŽETAK 22

ABSTRACT 22

ŽIVOTOPIS 23

1. UVOD

U današnje vrijeme internet pruža najveći izvor informacija. Velik se broj ljudi navikao u tolikoj mjeri oslanjati na informacije pronađene na internetu, da ih nigdje drugdje niti ne traži. Ako željenu informaciju ne uspije pronaći unutar pola minute, mnoge će se osobe ponašati kao da ta informacija niti ne postoji. Zbog toga svaki poduzetnik treba nastojati da njegove usluge budu što dostupnije. U svome radu fokusirat ću se na internet aplikaciju za naručivanje hrane. Cilj je napraviti internet stranicu kojoj mogu pristupiti svi potrošači, no također aplikaciju u kojoj će zaposlenici restorana moći zaprimati narudžbe i slati obavijest da je narudžba spremna. Napravljeno je i sučelje za dostavljača koji će sa pametnog telefona slati informaciju da je uspješno izvršio dostavu. Ovakva aplikacija vlasniku restorana daje puno precizniji uvid u rad zaposlenika, kao i u narudžbe potrošača. Smanjuje se vjerojatnost pogrešne dostave, a samim time i broj nezadovoljnih potrošača. Menadžeru restorana bit će korisni podaci o narudžbama koji su spremljeni u jednoj tablici u bazi podataka. Na osnovu prethodnih narudžbi, lakše će procijeniti koliko kojih namirnica nabaviti. U preostalom dijelu ovog rada, opisat ću tehnologije korištene pri izradi, te izgled i funkcionalnost ove aplikacije.

1.1. Zadatak završnog rada.

U teorijskom dijelu rada potrebno je proučiti i opisati tehnologije za izradu Internet aplikacija (HTML, CSS, Node.js, baze i sl.). U praktičnom dijelu rada potrebno je izraditi sučelje (*engl. FrontEnd*) u kojem će korisnik moći naručiti hranu i sastaviti vlastito jelo iz ponuđenih namirnica. Također, omogućiti neobavezno registriranje uz unos potrebnih kontakt podataka. Kroz pozadinski sustav (*engl. BackEnd*) potrebno je izraditi administratorski dio u kojem se zaprima narudžba i šalje pomoću elektroničke pošte ili SMS-a obavještava korisnika o statusu narudžbe.

2. PREGLED KORIŠTENIH TEHNOLOGIJA

2.1. HTML5

HTML je jezik za opisivanje strukture internet stranica. On nije programski jezik i zbog toga nije sposoban obavljati funkcije, pokretati programske petlje, izvršavati zadatke, pa ni obavljati računske operacije.

HTML korisniku omogućuje:

- Objavljivanje *online* dokumenata sa naslovima, tekstom, tablicama, listama, slikama, itd.
- Preuzimanje informacija sa interneta putem hipertekstualnih poveznica.
- Dizajniranje formulara za izvođenje transakcija sa udaljenim uslugama, za pretraživanje informacija, pravljenje rezervacija, naručivanje proizvoda, itd.
- Pridruživanje tablica, video zapisa, zvučnih zapisa, i raznih drugih aplikacija direktno u dokument.

Važno je napomenuti da postoje određene oznake (*engl. markup*) koje se koriste kako bi se obični tekst, kojeg želimo prikazati na stranici, razlikovao od opisnih riječi. Oznake govore pregledniku kako da rasporedi sadržaj na internet stranici i kako ona treba izgledati. Na primjer, oznaka * * pregledniku daje do znanja da u kodu slijedi jedan paragraf. Paragraf završava oznakom * *. Uloga HTML-a je da predstavlja osnovicu internet stranice. Kasnije se određenim alatima, npr. CSS i JavaScript, na osnovicu dodaju novi elementi i funkcionalnosti kako bi stranica na kraju imala cjelovitu strukturu. Budući da je HTML jezik označavanja, a ne programski, jedini alati potrebni za izradu internet stranice su uređivač teksta i internet preglednik. Nije potreban nikakav dodatan kompajler, internet preglednik interpretira ono što je napisano u tekstualnom dokumentu. Tekstualni HTML dokumenti se spremaju kao *.html* datoteke [1].

Pri izradi internet stranice, važno je stranicu testirati na više internet preglednika, jer nisu svi jednaki i neće svi jednako interpretirati kôd zapisan u tekstualnom dokumentu. Web dizajneru je cilj svoju stranicu učiniti funkcionalnom na što većem broju internet preglednika, ili barem na onima koji su najviše korišteni: Chrome, Mozilla, Safari, Internet Explorer, Edge i Opera.

HTML5 je najnovija verzija jezika. Objavljena je u listopadu 2014. godine od strane *World Wide Web Consortium*a (W3C) kako bi unaprijedila jezik podrškom za najnoviju multimediju. Cilj HTML5 je bio objediniti funkcionalnosti HTML 4, XHTML 1 i DOM Level 2 HTML jezika [2].

Osnovna struktura HTML dokumenta sastavljena je od zaglavlja (*engl. head*) i tijela (*engl. body*) što je vidljivo u kôdu 2.1. U opisu izrade internet stranice neke od ovih stvari bit će detaljnije pojašnjene.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Naslov internet stranice</title>
    <link rel = 'stylesheet' href = 'style.css'>
  </head>
  <body>
    <p>Tekst koji se prikazuje u prozoru internet stranice</p>
  </body>
</html>
```

Programski kôd 2.1. – Osnovna struktura HTML dokumenta

2.2. CSS3

CSS (*engl. Cascading Style Sheets*), što u prijevodu znači: kaskadni stilski predlošci, jednostavan je mehanizam za dodavanje stila (npr. font, boje, razmaci, pozadine...) internet dokumentu. On je također opisni jezik. Razvijen je kako bi se proširile mogućnosti i uveli novi standardi u internet dizajnu. CSS je neovisan o HTML-u i može ga se koristiti uz bilo koji opisni jezik koji je baziran na XML-u. (XML je jednostavniji i praktičniji podskup Standard Generalized Markup Languagea, standarda za stvaranje strukture dokumenta). Odvojenost CSS-a od HTML-a olakšava održavanje stranica, dijeljenje stilskih predložaka diljem stranice i prilagođavanje stranica različitim okolinama [1]. CSS kôd se sastoji od selektora, svojstava i vrijednosti. Selektorima se referenciraju određeni elementi HTML stranice pomoću *id* ili *class* karakteristike tog elementa. Svojstvima se opisuju pojedini atributi kao što su veličina fonta, margine, boja, sjene i sl. Vrijednostima definiramo veličine koju pojedino svojstvo može poprimiti.

CSS kôd može se povezati sa HTML dokumentom na više načina [3]:

- Umetanje u zaglavlje HTML dokumenta,
- Dodavanje unutar linije HTML koda,
- Povezivanje s vanjskim dokumentom.

U praksi se najviše koristi povezivanje s vanjskim dokumentom. To znači da se sav CSS kôd piše u zasebnu datoteku, a zatim se u HTML dokumentu upiše jedna linija kôda koja povezuje

dva dokumenta (*Programski kôd 1.0.*). Ova metoda omogućava najveću preglednost i najlakše snalaženje u kôdu [4].

2.3. JavaScript

JavaScript (JS) je skriptni programski jezik, koji se izvršava u internet 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 [5].

JavaScript sa HTML-om i CSS-om čini konstrukciju moderne internet stranice. HTML pribavlja strukturu, CSS dodaje stil, a JavaScript omogućava dodatne funkcionalnosti. JS programeru omogućava da detektira svaki događaj na internet stranici, poput klika na tipku, smanjenja prozora preglednika, ili unosa teksta u neko polje. JavaScript je skriptni jezik, što znači da ga nije potrebno prevesti prije izvođenja (*engl. compile*), nego će internet preglednik čitati skriptu liniju po liniju i prema tome ju izvršiti na stranici. S obzirom da je JS skriptni programski jezik, moguće je napisati kôd koji reagira na određene događaje. Može vršiti izračun, dinamički mijenjati slike na stranici, čak i obavljati potvrdu unesenih podataka [6].

JavaScript kôd je moguće implementirati na nekoliko načina [5]:

- Spremiti ga unutar samog HTML dokumenta,
- Upisati ga u polje za adresu u internet pregledniku,
- Spremiti ga u zasebnu JavaScript datoteku.

U praksi se najčešće koristi spremanje JS kôda u zasebnu datoteku, jer pruža bolju preglednost i snalaženje u okruženju. Unutar HTML dokumenta je važno pomoću <*script*> oznake internet pregledniku dati do znanja da slijedi skriptni jezik. Unutar oznaka <*script*> piše se JS kôd.

2.4. jQuery

Biblioteka jQuerry je biblioteka gotovih JavaScript funkcija koje se koriste u velikom broju komercijalnih internetskih sjedišta. Službeni jQuery slogan glasi: "Piši manje, učini više". Web programer ne bi trebao potrošiti sate na rješavanje hirova pojedinih internet preglednika i različitih postupaka u JavaScriptu za postizanje istih ciljeva. Tome služe gotove JavaScript biblioteke funkcija pa samim tim i jQuery.

Prije nego počnemo s jQuery razvojem potrebno je učitati posljednju inačicu jQuery biblioteke i uključiti je u vašu internet stranicu. Primjer uključivanja inačice biblioteke jQuery "jquery-1.11.0.js" u internet stranicu može se vidjeti u programskom kôdu 2.2.

```
<script type="text/javascript" src="jquery-1.11.0.js"></script>
```

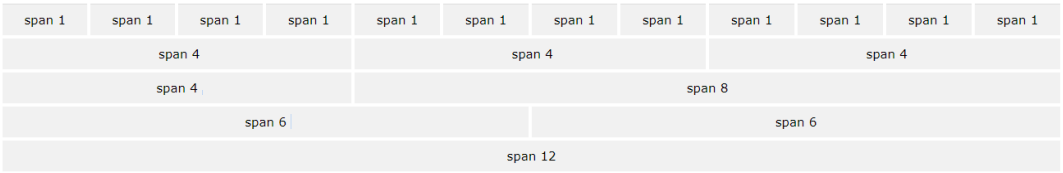
Programski kôd 2.2. – Uključivanje jQuery biblioteke u Internet stranicu

Gornji redak se mora umetnuti unutar *<head>* elementa a prije bilo kojeg jQuery kôda, u suprotnom će se dogoditi greška, odnosno kôd neće raditi. Uz to biblioteka funkcija "jquery-1.11.0.js" se mora nalaziti na zadanom mjestu u strukturi mapa [7].

2.5. Bootstrap

Bootstrap je najpopularniji HTML, CSS i JS *framework* za izradu responzivnih i mobilnih projekata na internetu. Bootstrap lako i efikasno skalira internet stranice i aplikacije s jednom bazom kôda, od mobitela preko tableta, do stolnih računala koristeći CSS-ove upite. S njime se dobiva opsežna i kvalitetna dokumentacija za uobičajene HTML elemente, deseci prilagođenih HTML i CSS komponenata, te odlični jQuery priključci [8].

Bootstrapov tablični sustav omogućava do 12 stupaca širom stranice. Ako ne želite koristiti svih 12 individualnih stupaca, moguće ih je grupirati zajedno stvarajući manje širih stupaca, kao što je vidljivo na slici 2.1.



Slika 2.1. – Bootstrapov tablični sustav

Bootstrapov tablični sustav ima 4 klase koje se mogu kombinirati kako bi se stvorilo fleksibilno i dinamičnije sučelje:

- *xs* (za mobitele)
- *sm* (za tablete)
- *md* (za stolna računala)
- *lg* (za stolna računala s većim ekranom)

2.6. NodeJS

Node.js je runtime JavaScript-a izrađen na Chromeovom V8 JavaScript pokretaču. Node.js koristi događajem vođen, neblokirajući I/O model koji ga čini vrlo efikasnim. Najjednostavnije rečeno, Node.js je JavaScript koji se kompajlira na poslužitelju (*engl. server*), a ne u internet pregledniku. Njegov ekosistem paketa zvan npm, najveći je ekosistem biblioteka otvorenog kôda u svijetu.

S obzirom da je Node asinkroni, događajem vođen runtime JavaScript-a, dizajniran je za izradu skalabilnih mrežnih aplikacija. On može rukovati sa više konekcija istovremeno. Prilikom svake konekcije, šalje se povratni poziv (*engl. callback*), ali ako nema nikakvog posla za obaviti, Node će mirovati.

To je u suprotnosti sa trenutno učestalijim modelom konkurencije u kojima se upošljavaju niti operacijskog sustava. Umrežavanje bazirano na nitima je relativno neefikasno i vrlo teško za korištenje. Korisnici Nodea se ne plaše zastoja (*engl. deadlock*) procesa, jer je onemogućeno da dođe do toga. Gotovo niti jedna funkcija u Nodeu ne izvršava direktno I/O, tako da proces nikada ne blokira. A pošto ništa ne blokira, skalabilni sustavi su odlična primjena za Node.js [9].

2.7. SQLite

Za potrebu izrade aplikacije, mora se napraviti baza podataka koja sadrži tablice koje sadrže informacije o narudžbama, korisnicima, djelatnicima, itd. Za izradu baze podataka korišten je SQLite. SQLite je unutarprocesna biblioteka koja izvršava SQL bazu podataka koja je samostalna i neovisna o serveru. Kôd SQLite-a je u javnoj domeni i kao takav, besplatan je i može se koristiti u bilo koju svrhu, komercijalnu ili privatnu. SQLite je najšire razvijena baza podataka u svijetu, korištena u nekoliko projekata visokog profila (Dropbox, iPhone, iTunes…).

SQLite je ugrađeni SQL sustav baza podataka. Za razliku od većine drugih SQL baza podataka, SQLite nema odvojeni proces za server. On čita i piše direktno u obične datoteke na disku. Potpuna SQL baza podataka sa višestrukim tablicama, indeksima, okidačima i pogledima, sadržana je u jednoj datoteci na disku. Format datoteke ove baze podataka radi na mnogim platformama – slobodno možete kopirati bazu sa 32-bitnog na 64-bitni sistem. Ove karakteristike čine SQLite popularnim izborom [10].

3. RAZVOJ APLIKACIJE ZA NARUČIVANJE HRANE

3.1. Korisnički zahtjevi

Korisnici ove internet aplikacije se mogu podijeliti u dvije grupe: djelatnici i potrošači. Potrebno je napraviti kvalitetno i intuitivno korisničko sučelje koje će potrošačima omogućiti da što prije dobiju informaciju koju traže. Bilo da se radi o proizvodima koji su u ponudi, radnom vremenu restorana, lokaciji ili kontaktu, cilj je da posjeta ovoj internet stranici bude pozitivno iskustvo. Baš zbog toga, velik naglasak je stavljen na dizajn stranice. Osim toga, potrebno je omogućiti potrošačima narudžbu putem interneta, i napraviti sustav koji će ih obaviještavati o trenutnom statusu njihove narudžbe; npr. narudžba je zaprimljena, dostavljač je krenuo/stigao. Druga stvar, potrebno je napraviti pozadinski sustav u kojem djelatnici kuhinje zaprimaju narudžbe. Korisničko sučelje i ovdje mora biti dobro napravljeno kako bi bilo jasno koja narudžba pripada kojem kupcu. Djelatnici moraju moći potvrditi da su naručeno jelo dovršili. Druga skupina djelatnika su dostavljači, oni također moraju imati vlastito sučelje. U njemu mora postojati mogućnost slanja obavijesti da je dostavljač dostavio narudžbu na adresu.

3.2. Zajednički dizajn stranice

Stranica je koncipirana tako da se ispod naslovne slike nalazi navigacijska traka sa poveznicama na naslovnu stranicu, menu, stranicu za kontakt i stranicu za narudžbu. Na svakoj kartici ove internet stranice položaj navigacijske trake je isti. Na slici ispod se može vidjeti kao to izgleda. Pritiskom na naziv restorana „Pizzeria“, otvara se naslovni prozor. Slika 3.1. prikazuje naslovnu sliku i navigacijsku traku koja je zajednička svim prozorima.

Slika 3.1. – Navigacijska traka

Kôd koji se nalazi ispod odnosi se na navigacijsku traku. Jedan dio kôda namijenjen je prikazu trake na stolnim računalima, a drugi na uređajima s manjim ekranima (mobitel, tablet). Prikaz HTML kôda navigacijske trake vidljiv je u programskom kôdu 3.1.

```
<nav class="navbar navbar-inverse">
    <div class="container-fluid">
        <!-- HAMBURGER ICON -->
        <div class="navbar-header">
           <!-- navbar-header se odnosi na navigacijsku traku pri nižim rezolucijama -->
            <button type="button" class="navbar-toggle" data-toggle="collapse"
               data-target="#mainNavBar">
                  <span class="icon-bar"></span>
                  <span class="icon-bar"></span>
                  <span class="icon-bar"></span>
            </button>
        </div>
        <!-- Menu Items -->
        <div class="collapse navbar-collapse" id="mainNavBar">
            <ul class="nav navbar-nav">
                <li class="popup"><a href="#">Naslovna</a></li>
                <li><a href="../menu">Menu</a></li>
                <li><a href="../kontakt">Kontakt</a></li>
            </ul>
            <ul class="nav navbar-nav navbar-right">
                <li><a href="kupac-login">Prijava</a></li>
            </ul>
        </div>
    </div>
</nav>
```

Programski kôd 3.1. - HTML kôd navigacijske trake

Za izradu navigacijske krate koristio sam predložak iz biblioteke Bootstrap. Koristio sam Bootstrap jer on nudi jako dobru funkcionalnost, omogućava automatsko skaliranje trake u odnosu na veličinu ekrana. No, ipak sam promijenio dizajn trake, jer mi njihov standardni nije odgovarao. Programski kôd 3.2. prikazuje CSS kôd navigacijske trake.

```
.navbar{
    background: #ffdb4d;
    margin: 0px;
    border: none;
    -webkit-border-radius:0px;
    box-shadow:#333333 0 0 10px;
}
```

Programski kôd 3.2. - CSS kôd navigacijske trake

Korištenjem sjene i svijetlijom nijansom žute boje postigao sam da tipka „Naslovna“ djeluje ispupčeno. To korisniku daje do znanja da se trenutno nalazi na naslovnoj stranici. Kada je na stranici za kontakt, tada će tipka *Kontakt* biti ispupčena. To sam učinio tako što sam <*li*> elementu dao klasu *popup*. U CSS dokumentu je definirano kakve karakteristike sadrži klasa *popup*, što se može vidjeti u kôdu 3.3.

```
.popup{
    background: #ffe066; /* ovaj kod definira boju pozadine */
    box-shadow: #333333 0 0 10px; /* ovaj kod definira izgled sjene oko tipke */
}
```

Programski kôd 3.3. - Prikaz klase *popup*

Na svakoj se kartici (Naslovna, Menu, Kontakt i Prijava) ispod navigacijske trake nalazi sadržaj sukladan nazivu te kartice. Sadržaj svakog prozora ću naknadno prikazati. Sada se fokusiram na zajedničke elemente. Na dnu svake kartice nalazi se podnožje sa informacijom o radnom vremenu restorana. Slika 3.2. prikazuje izgled podnožja stranice (*engl. footer*).

Slika 3.2. – Prikaz podnožja stranice

Također, nezaobilazan bio stranice je element <*body*> unutar kojeg se nalaze svi ostali elementi koji su vidljivi na stranici. On je tamno ljubičaste boje i raširen je preko cijele površine ekrana. Njegov CSS kôd prikazan je u programskom kôdu 3.4. Na mobilnoj verziji stranice nije vidljiv jer je u potpunosti zakriven elementom <*div*> klase *big-wrapper*.

```
body{         /* ovdje unosimo karakteristike koje se primjenjuju u cijelom body-u */
    background: #50394c; /* ovime se definira boja pozadine */
    width: 100%;          /* ovime definiramo koliku širinu ekrana će zauzeti element
body */
    display: -webkit-box; /* ovime aktiviramo funkcionalnost -webkit-box */
    -webkit-box-pack: center; /* ovime centriramo sve podelemente */
}
```

Programski kôd 3.4. – CSS kôd elementa <*body*>

Unutar elementa <*body*> nalazi se <*div*> element klase *big-wrapper* kao njegov podskup. Sav ostali sadržaj stranice (slike, navigacijska traka, tekst, itd.) nalazi se u njemu. Sav taj sadržaj se podređuje karakteristikama ovoga <*div*> elementa. Drugim riječima, navigacijska traka može biti široka maksimalno onoliko koliko je ovaj <*div*> element širok. Njegov CSS kôd prikazan je u programskom kôdu 3.5.

```
#big-wrapper{      /* ovo je glavni div element koji se nalazi unutar body elementa */
    border: solid 2px #b2b2b2;
    background: #ffeead;
    max-width: 1200px; /* ovime se definira maksimalna širina ovog div elementa */
    margin: 0px 0px;   /* ovime se daje do znanja da nema margina */
```

```
    padding: 0px 0px 20px 0px;
    display: -webkit-box;
    -webkit-box-orient: vertical;
    -webkit-box-flex: 1; /* ovime se uključuje mogućnost prilagodbe ekranu */
    box-shadow:black 0 0 15px;
}
```

Programski kôd 3.5. - CSS kôd elementa <*div*> klase *big-wrapper*

S obzirom da danas ljudi češće idu na internet preko svojih mobilnih uređaja nego preko stolnih računala, izuzetno je bitno internet stranicu prilagoditi mobilnim uređajima. To i je jedan od razloga zašto sam se koristio Bootstrap tehnologijom. Kao što je već navedeno u opisu Bootstrap tehnologije, on ima klase kojima prilagođava izgled sučelja veličini ekrana. U kôdu 3.6. moguće je vidjeti kako je definirana veličina elementa u galeriji za stolna računala i za mobilne uređaje. Za računala je korištena oznaka *md* (*engl. medium*). Iz kôda je vidljivo da jedan element zauzima širinu 5 od ukupnih 12 stupaca. To znači da u jedan redak stanu 2 takva elementa jedan pored drugoga. Za mobitele (slika 3.3.) je korištena oznaka *xs* (*engl. extra small*). Iz kôda je vidljivo da jedan element zauzima širinu svih 12 stupaca. To znači da u jedan redak stane jedan element.

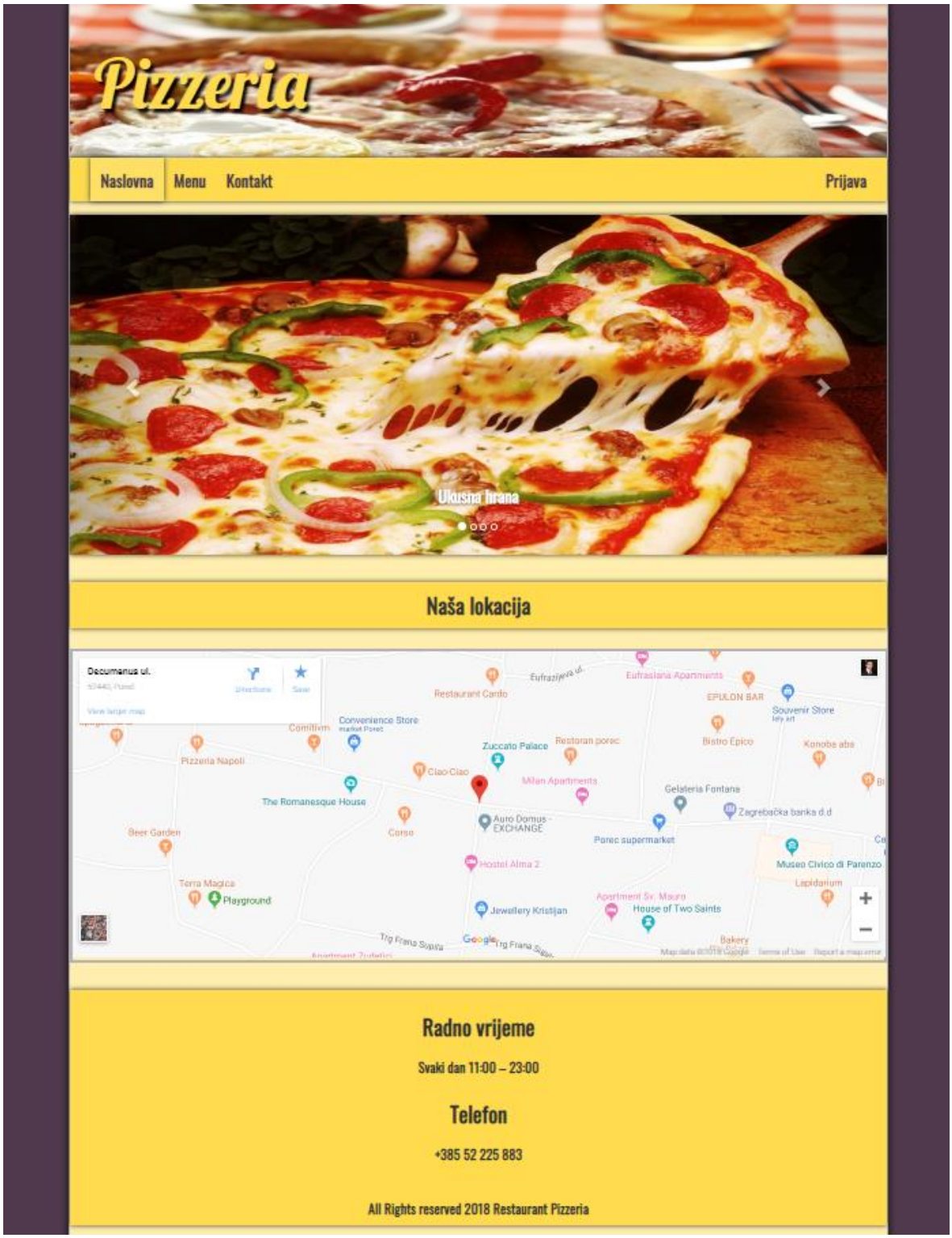
```
<div class="col-md-5 col-md-offset-1 col-xs-12 col-xs-offset-0">
```

Programski kôd 3.6. – Bootstrapove klase za definiranje širine elemenata

Slika 3.3. – Prikaz mobilne verzije naslovne stranice i navigacijske trake

3.3. Prikaz zasebnih prozora

Ovdje ću se fokusirati na svaku od zasebnih prozora i pokazati što je za svaku karakterističnо, te objasniti kôd. Prva je u nizu naslovna stranica, a vidljiva je na slici 3.4.



Slika 3.4. – Prikaz prozora „Naslovna“

Na naslovnoj stranici, ispod navigacijske trake nalazi se galerija od 4 slike koje se automatski izmjenjuju svakih 6 sekundi. Ovdje sam također koristio predložak iz biblioteke Bootstrap. Ime ovog predloška je *Carousel*. U HTML dokument unosim slijedeći kôd kako bih aktivirao ovu funkcionalnost:

```
<div id="myCarousel" class="carousel slide" data-ride="carousel"></div>
```

Programski kôd 3.7. – Unošenje predloška *Carousel*

Kako bi ovaj kôd bio funkcionalan, u zaglavlju HTML dokumenta mora omogućiti Bootstrap. To je prikazano kôdom 3.8. Iz ovog kôda je vidljivo da se Bootstrap koristi JavaScriptom, jQuery-em i CSS-om.

```
    <!-- Latest compiled and minified CSS -->
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

    <!-- jQuery library -->
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>

    <!-- Latest compiled JavaScript -->
    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
```

Programski kôd 3.8. – Skripte potrebne za uključivanje Bootstrapa u stranicu

Ispod ove galerije slika nalazi se karta sa lokacijom restorana. Korištene su Google karte. Kôd 3.9. korišten je za implementaciju karte u HTML. Za implementaciju se koriste JavaScript funkcije.

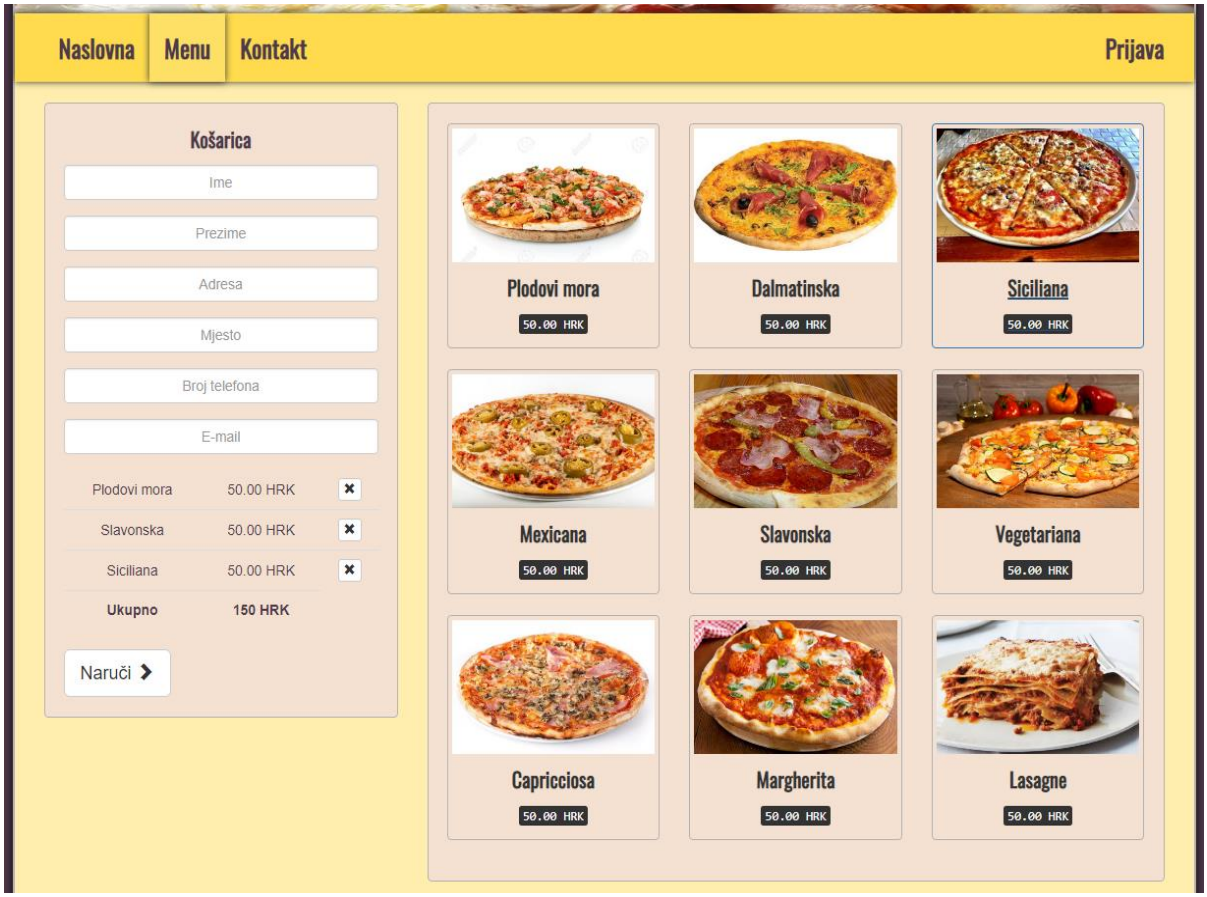
```
<div id="map">
    <iframe
src="https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d322.09506649685864!2d13.59202
2505730307!3d45.228028340309834!2m3!1f0!2f0!3f0!3m2!1i1024!2i768!4f13.1!3m3!1m2!1s0x477c
977424f440b3%3A0x9bd2c18b9a5dd949!2sDecumanus+ul.%2C+52440%2C+Pore%C4%8D!5e0!3m2!1shr!2s
hr!4v1535801819753" width="600" height="450" frameborder="0" style="border:0"
allowfullscreen>
    </iframe>
</div>
```

Programski kôd 3.9. – Uključivanje Google karte u stranicu

Sljedeći prozor je *Menu*. Ovaj se prozor sastoji od galerije jela koje restoran ima u ponudi. I jednostavne „košarice“ u kojoj se vide trenutno naručena jela. Slika 3.5. daje prikaz prozora *Menu.*

Klikom na sliku jela, ono se dodaje u košaricu. Svakom jelu je pridružena cijena koja se zbraja sa cijenama ostalih naručenih jela. U primjeru na slici 3.5. narudžbi su dodana tri jela i njihove cijene su se zbrojile u ukupnu cijenu narudžbe. Ta funkcionalnost omogućena je kôdom 3.10. Nadalje, tu je također forma gdje kupac unosi svoje kontakt podatke. Podataka za unos ima dosta, ali

stranica nudi mogućnost registriranja korisnika. Prednost registriranja je u tome što se tada korisnik može samo prijaviti i svi njegovi podaci će se popuniti automatski. Potrebno je samo da odabere jelo.



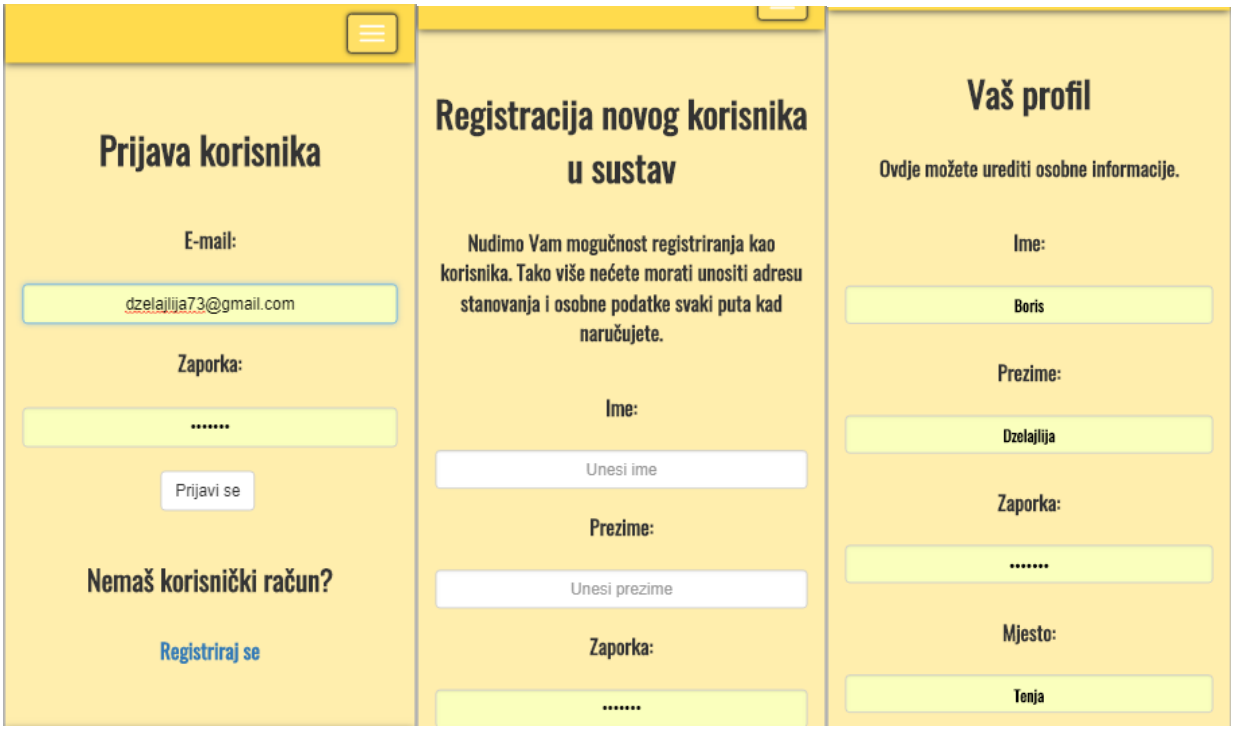
Slika 3.5. – Prikaz prozora „Menu“

```
  for (var i = 0; i < pizzas.length; ++i) {
    pizza = pizzas[i];
    total += pizza.price;
    order_html += '<tr id="' + i + '"><td id="vrsta">' + pizza.name + '</td><td>'   +
pizza.price.toFixed(2) + ' HRK</td>';
    console.log(show_delete)
    if (show_delete) {
      order_html += '<td><button type="button" class="btn btn-default btn-xs"> <span
class="glyphicon glyphicon-remove"></span></button></td>';
    }
    order_html += '</tr>';
  }
```

Programski kôd 3.10. – Računanje ukupne cijene narudžbe

Izrađeni su zasebni prozori za prijavu, registraciju i profil korisnika. Svi prozori ove internet stranice imaju zajednički dizajn kako bi iskustvo korisnika bilo što ugodnije. Prozor „Prijava korisnika“ služi za prijavu već registriranih korisnika. Ukoliko nemaju korisnički račun, kupci mogu klikom na link *Registriraj se* doći na stranicu za registraciju i registrirati se. U slučaju da

žele promijeniti neke svoje informacije, korisnici mogu otići na svoj profil i tamo izvršiti izmjene. Na slici 3.6. mogu se vidjeti ova tri prozora.



Slika 3.6. – Prikaz prozora „Prijava“, „Registracija“ i „Profil“

Svi se ovi unosi zapisuju u bazu podataka pomoću pozadinskog sustava (*engl. BackEnd*). Tu se koriste funkcije jezika Node.js i SQLite. Podaci koje je korisnik unio u formu za registraciju se uspoređuju sa podacima u tablici *buyer*. Ako korisnik sa tim imenom već ne postoji, novi korisnik se unosi u bazu. U tablicu *buyer* tada bivaju upisani svi podaci koje je unio u formu za registraciju: ime, prezime, lozinka, mjesto, adresa, broj telefona i adresa elektroničke pošte. Kôd 3.11. izvodi se prilikom registracije korisnika.

Prilikom prijave korisnika se adresa elektroničke pošte i lozinka koje je korisnik unio u formu za prijavu uspoređuju sa podacima u tablici *buyer*. Ukoliko takav korisnik postoji i lozinka je ispravna, korisnika se automatski preusmjerava na prozor „Menu“. Svi korisnikovi podaci u formi su automatski popunjeni te nema potrebe da ih korisnik ručno unosi. To je prednost registriranja u sustav. Kôd 3.12. izvodi se prilikom prijave korisnika.

```
var createUser = function (user, callback) {
    if (user.email.lenght > 1) {
```

```
        console.log(user.email + ' user is not in db, he will be inserted');
    } else {
        console.log(user + ' user is not in db, he will be inserted');
    }
    console.log("Inserting user to db");
    db.serialize(function () {
        var stmt = db.prepare("INSERT INTO buyer (first_name, last_name, password, city,
address, telephone, email) VALUES (?,?,?,?,?,?,?)");
        stmt.run(user.first_name, user.last_name, user.password, user.city,
user.address, user.telephone, user.email);
        stmt.finalize();

        getUserByEmail(user.email, function (err, row) {
            if (err) {
                return callback(err);
            }
            return callback(null, row);
        });
    });
};
```

Programski kôd 3.11. – Registracija korisnika

```
function buyerLogin(email, password, callback) {
    var loginResult = "Not_OK";
    db.serialize(function () {
        let queryString = "SELECT * FROM buyer WHERE email LIKE '" + email + "' AND
password LIKE '" + password + "'";
        console.log(queryString);
        db.each(queryString, function (err, row) {
            console.log("Buyer : " + row);
            if (row) {
                loginResult = "OK";
            } else {
                loginResult = "Not_OK";
            }
        }, function () {
            console.log("I have the buyer result: " + loginResult);
            callback(loginResult);
        });
    });
}
```

Programski kôd 3.12. – Prijava korisnika

Sljedeći prozor je „Kontakt“. Ovdje korisnici mogu vidjeti adresu i broj telefona restorana. Također postoji opcija slanja elektroničke pošte. Pritiskom na tipku *E-mail* otvara se, inače skriveni, formular za slanje elektroničke pošte. Slika 3.7. je prikaz neuspjelog slanja elektroničke pošte.

Od korisnika se zahtijeva da unese tražene podatke. Ako nije unio neki od traženih podataka, ne može poslati poruku i javlja se poruka kod one trake za unos teksta koja je ostala nepopunjena. To je postignuto kôdom 3.13.

Slika 3.7. – Kontakt forma za slanje elektroničke pošte

```
<form>
            <div id="kontakt" class="form-group">
                <label for="usr">Ime:</label>
                <input type="text" class="form-control" id="ime" required>
            …
```

Programski kôd 3.13. – Jedan element kontakt forme

Opisna riječ *required* ovaj element formulara čini neizostavnim. Ta funkcionalnost je uvedena sa HTML5 verzijom, prije se za to koristio JavaScript, a sada je ugrađeno.

Otkrivanje skrivenog formulara za kontakt je postignuto jQuery funkcijom koja se može vidjeti u kodu 3.15. Funkcija osluškuje tipku *button* i aktivira se pritiskom na nju. Elementu <*div*> unutar kojeg se nalazi formular pridodan je identifikator: *id="skriven"*, što je vidljivo u kôdu 3.14.) Funkcija se koristi identifikatorom <*div*> elementa kako bi precizirala na kojem elementu treba obaviti radnju. Funkcija mijenja CSS svojstvo *display* iz *none* u *inline*.

```
<div class="alert fade in col-md-6 col-md-offset-3 col-xs-10 col-xs-offset-1"
style="display:none" id="skriven">
```

Programski kôd 3.14. – Identificiranje elementa <*div*>

```
    <!-- sakrij/pokaži -->
    <script>
        $(document).ready(function(){
            $("#tipka").click(function(){
                $('#skriven').css('display', 'inline');
            });
        });
    </script>
```

Programski kôd 3.15. – Funkcija koja mijenja svojstvo vidljivosti elementa

Slanje upita mailom omogućeno je Node.js dodatkom zvanim Nodemailer. Funkcija za slanje je preuzeta sa njihove internet stranice [11], a prikazana je u kôdu 3.16.

```
function sendMailForContact(ime, prezime, email, comment, callback) {

    let transporter = nodemailer.createTransport({
        host: 'smtp.gmail.com',
        port: 465,
        secure: true,
        auth: {
            user: 'pizzeria.zavrsni@gmail.com',
            pass: 'naocale73'
        }
    });


    let mailOptions = {
        from: email, // sender address
        to: 'pizzeria.zavrsni@gmail.com', // list of receivers
        subject: 'Upit', // Subject line
        html: '<p>Pošiljatelj: </p><b>' + ime + ' ' + prezime + '</b><p>Email: </p><b> '
+ email + '</b><p>' + comment + '</p>'  // html body
    };

    console.log("Trying to send mail!");
    transporter.sendMail(mailOptions, (error, info) => {
        if (error) {
            console.log(error);
            callback();
        }
        console.log('Message %s sent: %s', info.messageId, info.response);
        callback();
    });
}
```

Programski kôd 3.16. – Dodatak Nodemailer

U uvodnom dijelu spomenuto je da je ova internet aplikacija namijenjena kupcima, ali i radnicima restorana. Sada će biti nešto više rečeno o dijelu aplikacije namijenjenom zaprimanju i dostavi narudžbi.

U već spomenutom prozoru „Prijava“, prijavljuju se također i zaposlenici. Postoje dva odjela: restoran i dostava. U kôdu 3.17. vidi se programska logika kojom se, ovisno o unesenoj adresi elektroničke pošte i lozinci, otvara odgovarajući prozor. Za pristup odjelu „restoran“, koristi se adresa elektroničke pošte *restoran@pizzeria.com* a za pristup odjelu „dostava“, *dostava@pizzeria.com*. U svakom drugom slučaju, smatra se da se prijavljuje kupac i podaci iz forme se uspoređuju s podacima u tablici *buyer* u bazi podataka. Tada ga se automatski preusmjerava na prozor „Menu.“

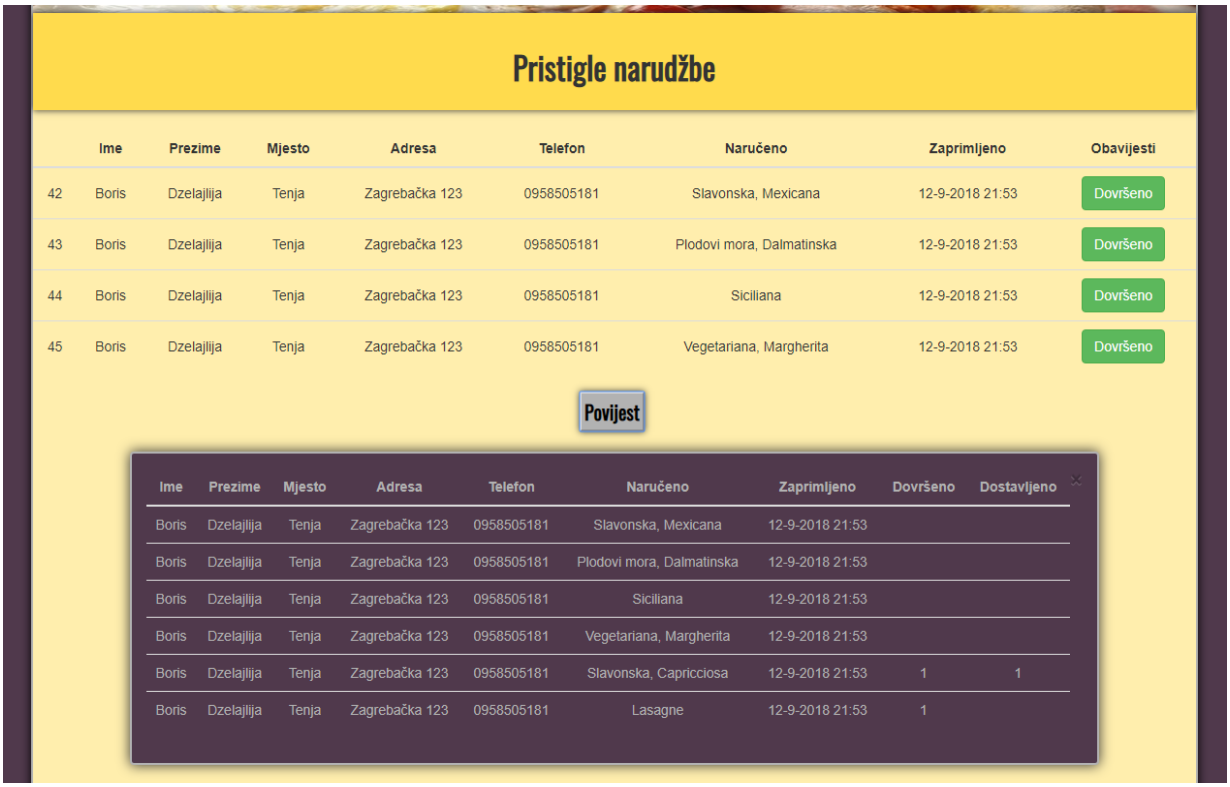
```
app.get('/kupac-profil', function (req, res) {
        switch (req.user.email) {
            case 'dostava@pizzeria.com':
                res.redirect('/pizzeria-dostava');
                break;
            case 'restoran@pizzeria.com':
                res.redirect('/pizzeria-restoran2');
                break;
            default:
                res.render('pages/kupac-profil', {
                    customer_name: req.user.first_name || '',
                    customer_surname: req.user.last_name || '',
                    customer_address: req.user.address || '',
                    customer_city: req.user.city || '',
                    customer_phone: req.user.telephone || '',
                    customer_email: req.user.email || '',
                    customer_id: req.user.id || '',
                });
            });
```

Programski kôd 3.17. – Funkcija za preusmjeravanje korisnika na odgovarajući prozor

Prozor „Menu“ je već ranije detaljno opisan, sada slijedi kratak opis prozora „Restoran“ i „Dostava.“ U prozoru „Restoran“, prikazanom na slici 3.8., centralno mjesto zauzima tablica sa podacima o pristiglim narudžbama. Pritiskom na tipku *Dovršeno*, šalje se elektronička pošta kupcu da je hrana gotova, i da će uskoro biti dostavljena, a šalje se obavijest u tablicu *pizza_orders* da je hrana dovršena. Tada ta narudžba nestaje iz glavne tablice jer više nije aktualna. Vidljiva je samo u povijesti narudžbi kojoj se može pristupiti pritiskom na tipku *Povijest*. Tipka *Povijest* aktivira istu funkciju kao u prozoru „Kontakt“. Ona mijenja CSS svojstvo *display* iz *none* u *inline*.

U prozoru „Dostava“, prikazanom na slici 3.9., centralno mjesto zauzima tablica sa podacima o zgotovljenim narudžbama. Pritiskom na tipku *Dostavljeno*, šalje se obavijest u tablicu *pizza_orders* da je hrana dostavljena, radi lakše evidencije. Tada ta narudžba nestaje iz glavne tablice jer više nije aktualna. Vidljiva je samo u povijesti narudžbi kojoj se može pristupiti pritiskom na tipku *Povijest*. Tipka *Povijest* aktivira funkciju koja mijenja CSS svojstvo *display* iz *none* u *inline*.

Uvid u povijest narudžbi je vrlo korisna u slučaju da dostavljač slučajno označi neko jelo kao dostavljeno prije reda. Pošto je aplikacija podešena tako da se dostavljeni artikli više ne prikazuju u tablici, važno je da u slučaju zabune može pogledati u povijest, inače neće znati gdje dostaviti hranu. Ljudi lako pogriješe i mora se sa time računati kako bi šteta bila što manja.



Slika 3.8. – Prikaz prozora „Restoran“

Slika 3.9. – Prokaz prozora „Dostava“

4. ZAKLJUČAK

U današnje, informatičko doba, neophodno je biti prisutan na internetu. Zadatak ova internet aplikacije je pomoći kupcima da što lakše naruče hranu i da se požele vratiti na nju. Osim toga, potrebno je da radnici koji se koriste aplikacijom imaju jasan uvid u pristigle narudžbe kako bi mogli osigurati što bolju uslugu kupcima. To je u ovoj aplikaciji postignuto privlačnim dizajnom koji je dobro prilagođen kako osobnim računalima, tako i mobilnim uređajima. Za izradu dizajna su korištene sljedeće tehnologije za razvoj internet stranica: HTML5, CSS, JavaScript i jQuery. HTML5 i CSS su opisni jezici i kao takvi ne mogu izvršavati nikakve matematičke i logičke operacije. Koriste se za unos i pozicioniranje teksta, te pridodavanje raznih svojstava elementima na internet stranici. Kako bi stranici dali dodatne mogućnosti, koristi se programski jezik JavaScript i njena izvedenica jQuery te *framework* Bootstrap. Korišteni su za skaliranje elemenata na stranici ovisno o veličini ekrana, prikaz i skrivanje određenih elemenata pritiskom na tipku, računanje ukupne cijene narudžbe i slično. Programski jezici Node.js i SQLite su korišteni za izradu pozadinskog sustava. Njihova uloga je komunicirati između elemenata na stranici (kao što su forme za unos podataka i tablice) i baze podataka. Rezultat je aplikacija koja je kupcima privlačna i jednostavna za korištenje, a ujedno olakšava vođenje restorana.

LITERATURA

[1] J. Duckett, HTML&CSS Design and build websites, John Wiley & Sons, Indianapolis, 2011.

[2] Wikipedija, HTML, https://hr.wikipedia.org/wiki/HTML, rujan 2018.

[3] TheServerSide, CSS, https://www.theserverside.com/definition/cascading-style-sheet-CSS, rujan 2018.

[4] W3Schools, https://www.w3schools.com, rujan 2018.

[5] Wikipedia, JavaScript, https://en.wikipedia.org/wiki/JavaScript, lipanj 2017.

[6] M. Morrison, Head First JavaScript, O'Reilly Media, Sebastopol, 2008.

[7] Moj web dizajn, jQuery, https://www.mojwebdizajn.net/, lipanj 2017.

[8] Bootstrap, Home, https://getbootstrap.com/, rujan 2018.

[9] Nodejs, About Node.js, https://nodejs.org/en/about/, rujan 2018.

[10] SQLite, About SQLite, https://www.sqlite.org/about.html, rujan 2018.

[11] Nodemailer, About Nodemailer, https://nodemailer.com/about/, rujan 2018.

SAŽETAK

U ovom je radu razvijena internet aplikacija za upravljanje restoranom. Aplikacija je namijenjena i kupcima i zaposlenicima restorana. Kupci se pomoću aplikacije informiraju o ponudi restorana i mogu izvršiti narudžbu. Zaposlenici imaju uvid u pristigle narudžbe koje su aktualne za njihov dio posla. Zaposlenici također imaju i uvid u povijest narudžbi pa s lakoćom mogu provjeriti je li došlo do greške u pripremanju ili dostavi hrane. U teorijskom dijelu rada su opisane korištene tehnologije. Opisani su postupci kojima su postignute gore navedene funkcionalnosti aplikacije. Praktični dio rada se sastoji od dizajniranja i izrade internet aplikacije i baze podataka, te njihovog međusobnog povezivanja. Korišteni su opisni jezici HTML5 i CSS, programski jezici JavaScript, Node.js i SQLite te *framework* Bootstrap.

ABSTRACT

In this thesis Internet application for managing the restaurant was developed. Application is intended for both customers and empoyees of the restaurant. Using the application, customers are informed about the services of the restaurant and are able to order a food. Employees have insight in incoming orders that are relevant for their job. Also, employees have insight into a history of the orders so they can easily check if there was a mistake in preparing or shipping the food. The theoretical part describes the technologies that were used. It describes the processes which led to the accomlishment of earlier described functionalities of the application. Practical part of the work consists of designing and developing the internet application and database, as well as their mutual connection. The coding languages that were used are HTML5, CSS, JavaScript, Node.js and SQLite together with the Bootstrap framework.

ŽIVOTOPIS

Boris Dželajlija rođen je 7. ožujka 1994. godine u Osijeku. Od 2001. do 2009. godine pohađa Osnovnu školu „Retfala“ u Osijeku. Godine 2009. upisuje Elektrotehničku i prometnu školu Osijek na smjer elektrotehničar koju završava 2013. godine polaganjem državne mature. Godine 2013. upisuje Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek na Sveučilištu Josipa Juraja Strossmayera u Osijeku na preddiplomski studij elektrotehnike. Godine 2014. prebacuje se na stručni smjer informatike na istom fakultetu. Godine 2016. pohađa program usavršavanja za poslove programera Web aplikacija u Pučkom otvorenom učilištu Obris. Od 2017. godine zaposlen je u firmi Alex Controls d.o.o. koja se bavi automatizacijom u industriji.

Boris Dželajlija