

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 5979

Sustav za dostavu hrane

Kristijan Vrbanc

Zagreb, lipanj 2019.

Umjesto ove stranice umetnite izvornik Vašeg rada.

Da bi ste uklonili ovu stranicu obrišite naredbu \izvornik.

SADRŽAJ

1.	Uvod	1
2.	Razrada teme	2
2.1.	Tipovi korisnika	2
2.1.1.	Neprijavljeni korisnik	2
2.1.2.	Prijavljeni korisnik sa ulogom kupca	3
2.1.3.	Prijavljeni korisnik sa ulogom vlasnika restorana	3
2.1.4.	Prijavljeni korisnik sa ulogom dostavljača hrane	3
2.1.5.	Dijagram obrazaca uporabe (engl. <i>Use Case Diagram</i>)	3
2.2.	Korištene tehnologije	4
2.2.1.	HTML	4
2.2.2.	CSS	5
2.2.3.	JavaScript	6
2.2.4.	Bootstrap	8
2.2.5.	Razvojni okvir Spring	9
2.2.6.	Thymeleaf	12
2.2.7.	SQL	13
2.3.	Baza podataka	14
2.3.1.	Tablica User	14
2.3.2.	Tablica Restaurant	15
2.3.3.	Tablica Item	16
2.3.4.	Tablica Ordered_item	16
2.3.5.	Tablica Placed_order	16
2.4.	Opis funkcionalnosti	17
2.4.1.	Početna stranica	17
2.4.2.	Registracija	18
2.4.3.	Prijava	19
2.4.4.	Pregled restorana sustava	20

2.4.5. Pregled vlastitih restorana	20
2.4.6. Pregled jelovnika	22
2.4.7. Košarica	24
2.4.8. Pregled narudžba	24
2.4.9. Aktivna dostava	25
3. Zaključak	27
Literatura	28

1. Uvod

Web aplikacije su programska rješenja kojima se pristupa putem internet preglednika. Vrtoglavni porast njihovog razvoja započeo je globalnim širenjem interneta. Danas gotovo svaka osoba ima pristup internetu, sa jednog ili više uređaja, stoga web aplikacije postaju dostupne sa bilo kojeg mesta.

Neovisnost o pristupu osnovna je karakteristika web aplikacija. Svi korisnici pristupaju aplikaciji na isti način, kao i ostalim web stranicama, putem internet preglednika. Također, web aplikacija je dinamična. Sadržaj se jednostavno dodaje i uklanja. Bilo koja promjena u aplikaciji odmah je vidljiva svim korisnicima. Pomoću web aplikacije moguće je ostvariti dvosmjernu komunikaciju sa korisnicima, stoga se mnoga poslovanja unaprijeđuju web aplikacijom.

Postoji velik raspon tehnologija za razvoj web aplikacija. Te tehnologije se uglavnom dijele u dva dijela, klijentske tehnologije (engl. *client-side technologies*) i poslužiteljske tehnologije (engl. *server-side technologies*). Klijentske tehnologije služe za definiranje sučelja prema korisniku, svega što korisnik vidi i sa čime može imati interakciju. Primjeri takvih tehnologija su HTML, CSS te JavaScript. Poslužiteljske tehnologije služe za obradu podataka generiranih od strane korisnika. Primjeri takvih tehnologija su Java, Ruby, PHP, .Net.

Izrada web aplikacija je zahtjevan proces, a služi za što bolju prezentaciju na internetu. Primjeri web aplikacija su pretraživači, aplikacije za kupnju, igranje, različiti portali, email aplikacije.

Sustav za dotavu hrane je internet trgovina gdje korisnici na lak i intuitivan način naručuju željenu hranu. Također, sustav omogućava dostavu hrane na odabranu lokaciju. Iako današnji ubrzani način života ostavlja vrlo malo vremena za pripremu kvalitetne i ukusne hrane, korištenjem sustava moguće je poboljšati kvalitetu života uz što veću uštedu slobodnog vremena te kvalitetniju prehranu.

2. Razrada teme

Sustav za dostavu hrane je web aplikacija arhitekture slične današnjim internet trgovinama. Prije svega, aplikacija služi za pregled jelovnika i narudžbu hrane. Od aplikacije se stoga zahtjevaju funkcionalnosti poput dodavanja restorana u sustav zajedno sa njihovim jelovnicima, kao i pregled restorana i jelovnika te mogućnost narudžbe. Također, aplikacija omogućava dostavu na željenu lokaciju, što proširuje popis traženih funkcionalnosti. Potrebno je korisnicima omogućiti unos željene lokacije. Također, potrebno je definiranje dostavljača hrane pojedinih restorana. Dostavljačima je potrebno pružiti zaseban pogled na sustav, gdje im se pruža uvid u narudžbe restorana te omogućuje preuzimanje dostava. Pritom im se pruža skup podataka potrebnih za dostavu hrane.

Sustav podržava četiri tipa korisnika. Pogledi korisnika se razlikuju ovisno o njihovoj ulozi u sustavu. Njihova uloga također određuje prava korisnika u sustavu.

Kako bi se ostvarila takva funkcija sustava, potrebno je definirati tipove korisnika, bazu podataka te funkcionalnost same aplikacije. Također, potrebno je odabratи tehnologije za razvoj same aplikacije.

2.1. Tipovi korisnika

Aplikacija podržava sljedeća četiri tipa korisnika: naprijavljeni korisnik, prijavljeni korisnik sa ulogom kupca, prijavljeni korisnik sa ulogom vlasnika restorana te prijavljeni korisnik sa ulogom dostavljača hrane. Svaki tip korisnika ima prava i ograničenja određena za njega.

2.1.1. Neprijavljeni korisnik

Neprijavljeni korisnik ima ograničena prava u sustavu. Omogućena mu je prijava u sustav ukoliko posjeduje korisnički račun ili izrada novog korisničkog računa ukoliko ga ne posjeduje. Također, neprijavljenom korisniku omogućen je pregled svih resto-

rana sustava kao i njihovih jelovnika, ali uz nemogućnost narudžbe hrane.

2.1.2. Prijavljeni korisnik sa ulogom kupca

Prijavljenom korisniku sa ulogom kupca, kao i neprijavljenom korisniku, omogućen je pregled svih restorana sustava i njihovih jelovnika. Glavna razika između ta dva korisnika sustava je mogućnost narudžbe. Kupcu je omogućena narudžba hrane. Prilikom narudžbe kupac ima mogućnost odabira lokacije dostave.

2.1.3. Prijavljeni korisnik sa ulogom vlasnika restorana

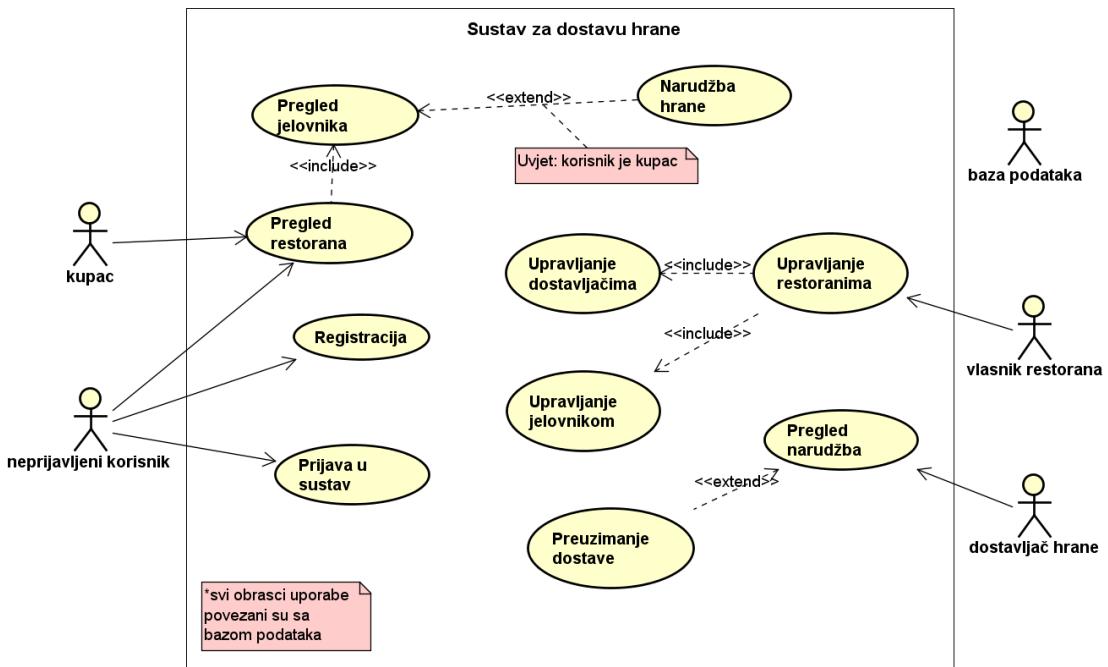
Prijavljenom korisniku sa ulogom vlasnika restorana omogućeno je dodavanje restorana u sustav. Prilikom dodavanja restorana u sustav pruža osnovne podatke o restoranu (naziv, radno vrijeme, opis...) kao i njegovu lokaciju. Također, omogućeno mu je definiranje jelovnika pojedinog restorana. Za pojedini restoran, omogućeno mu je dodavanje dostavljača hrane.

2.1.4. Prijavljeni korisnik sa ulogom dostavljača hrane

Prijavljenom korisniku sa ulogom dostavljača hrane omogućen je pregled svih narudžba restorana za kojeg je nadležan. Omogućeno mu je preuzimanje dostave pojedine narudžbe pri čemu na uvid dobiva sve podatke potrebne za dostavu (popis naručenih jela i njihova količina, kontakt broj osobe koja je postavila narudžbu, smjernice...).

2.1.5. Dijagram obrazaca uporabe (engl. *Use Case Diagram*)

Obrasci uporabe predstavljaju tehniku preuzetu iz UML (engl. *Unified Modeling Language*) standarda. UML je normirani jezik opće namjene koji se koristi za modeliranje računalnih sustava temeljenih na objektno-orientiranoj paradigmi. Dijagram obrazaca uporabe koristi se kako bi se prikazalo ponašanje sustava, dijelova sustava ili konkretnog razreda na način vidljiv korisniku sustava [9]. Slika 2.1 prikazuje dijagram obrazaca uporabe za sva četiri tipa korisnika.



Slika 2.1: Dijagram obrazaca uporabe sustava

2.2. Korištene tehnologije

Pri izradi aplikacije korištene su razne klijentske i poslužiteljske tehnologije. Klijentske tehnologije uključuju HTML, CSS, JavaScript te Bootstrap. Poslužiteljske tehnologije uključuju Thymeleaf, radni okvir Spring te upitni jezik SQL.

2.2.1. HTML

Internet stranice sastoje se od različitih tipova sadržaja poput teksta, grafike, obrazaca, audio i video zapisa. Svaka internet stranica je različita, te njen izgled i funkcija ovise o tome kako je pisan kôd. Ono što internet stranice imaju zajedničko je sintaksa programiranja HTML.

HTML (engl. *HyperText Markup Language*) je standardni jezik za stvaranje online i offline hipertekstualnih dokumenata. HTML se kao jezik za označavanje (engl. *markup language*) koristi za stvaranje logičke strukture dokumenta. Označavanje se vrši korištenjem oznaka (engl. *tag*) kojima se stvaraju, povezuju i strukturiraju elementi HTML dokumenta. Oznake određuju način na koji će Internet preglednik prikazati tekst koji slijedi nakon oznake. HTML datoteke imaju ekstenziju *.html* ili *.htm*, a mogu se stvarati bilo kojim uređivačem teksta.

HTML se sastoji od elemenata i atributa. Elementi identificiraju različite dijelove

HTML stranice korištenjem oznaka, a atributi daju informacije o instanci elementa. Elementi se sastoje od oznaka i sadržaja koji se nalazi unutar tih oznaka. Oznake obično dolaze u paru gdje prva oznaka započinje element, a druga ga završava. Pišu se u obliku `<oznaka></oznaka>`. Sav sadržaj elementa se nalazi unutar tih oznaka, a dopušteno je i gniježđenje elemenata. Sav sadržaj koji se nalazi unutar oznaka će biti prikazan na stranici, dok se oznake koriste kao instrukcije internet pregledniku o oblikovanju tog sadržaja. Postoje elementi koji se sastoje samo od jedne oznake. Oznaka se piše u obliku `<oznaka/>`. Takvi elementi zovu se prazni elementi.

HTML dokumenti sastoje se od zaglavlja i tijela dokumenta. Prva oznaka dokumenta je oznaka `<html>` koja govori internet pregledniku da slijedi početak HTML dokumenta. Zadnja oznaka dokumenta je `</html>` koja označuje kraj. Zaglavljje dokumenta nalazi se unutar oznaka `<head></head>`. Sadržaj zaglavlja ne prikazuje se izravno unutar pretraživača, već služi za postavljanje elemenata koji se ne prikazuju izravno unutar pretraživača, ali mogu imati utjecaja na prikaz i ponašanje dokumenta. Tijelo dokumenta je sadržaj koji se nalazi između oznaka `<body></body>`. Sadržaj koji se nalazi u tijelu dokumenta bit će vidljiv pri prikazu stranice. Slika 2.2 prikazuje primjer jednostavnog HTML dokumenta.

```
<html lang="en">
  <head>
    <title>Naslov HTML dokumenta</title>
  </head>
  <body>
    <p>Element sa oznakom p označuje paragraf.</p>
    <p>Sve što se nalazi u tijelu dokumenata bit će vidljivo.</p>
  </body>
</html>
```

Slika 2.2: Primjer jednostavnog HTML koda

2.2.2. CSS

CSS (engl. *Cascading Style Sheets*), odnosno kaskadni stilovi, mehanizam su za oblikovanje izgleda dokumenta pisanih u jezicima za označavanje (HTML, XML...). CSS je dizajniran kako bi omogućio razdvajanje logičke strukture dokumenta od njegovog dizajna. Pomoću CSS-a definiraju se pravila u stilskom obrascu koji određuje izgled sadržaja opisanog određenim HTML kodom i povezuju se stilski pravila i HTML kod. HTML kôd postaje pregledniji i manji što znači da ga je puno lakše kontrolirati, a također je moguće jednostavnom promjenom parametara promijeniti izgled stranice.

Stilski obrasci sastoje se od stilskih pravila. Svako stilsko pravilo sastoje se od selektora i deklaracije. Selektor određuje element na koji se stilsko pravilo odnosi, a

deklaracija određuje izgled sadržaja opisan CSS-om. Stilsko pravilo piše se u obliku *selektor{deklaracija;}*. Deklaracija se sastoji od svojstva (engl. *property*) i vrijednosti (engl. *value*). Svojstva su aspekti prikaza sadržaja (boja, veličina fonta...), a vrijednosti određuju iznose tih svojstva. Svojstvo se od vrijednosti odvaja dvotočjem, a svaka deklaracija završava sa točkom-zarez: *selektor{svojstvo:vrijednost;}*.

CSS kôd se u HTML dokument može uključiti na više načina. Kôd je moguće postaviti unutar zaglavlja dokumenta između oznaka *<style></style>*. Tako postavljen kôd vrijedi na cijelom HTML dokumentu. Ukoliko se kôd zapiše unutar oznake HTML elementa, vrijedit će samo unutar tog elementa. CSS kôd također može biti zapisan u zasebnoj datoteci što omogućuje korištenje istih stilova na više različitih dokumenata. Slika 2.3 prikazuje primjer CSS koda koji postavlja boju, vrstu te veličinu fonta zaglavlja HTML dokumenta.

```
h1, h2, h3{
    color : teal;
    font-family: Arial;
    font-size: 36px;
}
```

Slika 2.3: Primjer CSS koda

2.2.3. JavaScript

JavaScript je dinamički programski jezik koji, ukoliko se ugradi u HTML dokument, može pružiti dinamičku interakciju na stranici. Dinamički programski jezici su programski jezici u kojima se operacije koje se inače izvrše u vrijeme prevođenja mogu izvršiti tijekom izvođenja[2]. Jezik JavaScript podržan je od strane svih internetskih preglednika.

Izvorni kod pisan jezikom JavaScript ugrađuje se u tijelo HTML dokumenta unutar oznaka *<script></script>*. JavaScript omogućava izradu interaktivnih web stranica zbog mogućnosti pristupa i izmjene sadržaja i oznaka koje se koriste u HTML dokumentu za vrijeme njegovog prikazivanja. Slika 2.4 prikazuje jednostavni JavaScript kôd ugrađen u HTML dokument. Prilikom učitavanja dokumenta, u njegovo tijelo upisuje se tekst.

```

<html>
  <head>
    <title>Naslov HTML dokumenta</title>
  </head>
  <body>
    <script type="text/javascript">
      document.write("Primjer JavaScript koda.");
    </script>
  </body>
</html>

```

Slika 2.4: Primjer JavaScript koda

Maps JavaScript API

JavaScript API (engl. *Application Programming Interface*) za Google Karte (engl. *Google Maps*) omogućuje prilagođavanje karata vlastitim sadržajem i slikama u svrhu prikaza na web stranicama i mobilnim uređajima. JavaScript API za Karte sadrži četiri osnovna tipa karata (cestovna karta (engl. *roadmap*), satelit (engl. *satellite*), hibrid (engl. *hybrid*) i teren (engl. *terrain*)) koje se mogu mijenjati korištenjem slojeva i stilova, kontrola i događaja te raznih usluga i knjižnica[6]. Slika 3.4 prikazuje kôd web stranice koja prikazuje kartu sa središtem u Zagrebu.

```

<html>
  <head>
    <title>Jednostavna karta sa središtem u Zagrebu</title>
    <meta name="viewport" content="initial-scale=1.0">
    <meta charset="utf-8">
    <style>
      #map {
        height: 100%;
      }
    </style>
  </head>
  <body>
    <div id="map"></div>
    <script>
      var map;
      function initMap() {
        map = new google.maps.Map(document.getElementById('map'), {
          center: {lat: 45.8150, lng: 15.9819},
          zoom: 12
        });
      }
    </script>
    <script src="https://maps.googleapis.com/maps/api/js?key=YOUR_API_KEY&callback=initMap"
      async defer></script>
  </body>
</html>

```

Slika 2.5: Kod web stranice koja prikazuje kartu sa središtem u Zagrebu

Geocoding API

Geocoding API je usluga koja pruža geokodiranje i obrnuto geokodiranje adresa. Geokodiranje je postupak pretvorbe adresa u zemljopisne koordinate (kao što su zemljopisni

pisna širina i dužina) koje se mogu koristiti za postavljanje oznaka na kartu ili postavljanje karte[3]. Obrnuto geokodiranje je postupak pretvorba zemljopisnih koordinata u adresu.

Pristup usluzi Geocoding vrši se preko HTTP sučelja. Slanjem HTTP zahtjeva standardiziranog oblika dobiva se odgovor od poslužitelja. U kontekstu geokodiranja, odgovor sadrži zemljopisnu širinu i dužinu adrese za koju je podnijet zahtjev. U kontekstu obrnutog geokodiranja, odgovor sadrži formatiranu adresu koja se nalazi na određenoj zemljopisnoj širini i dužini. Odgovor je formata JSON (engl. *JavaScript Object Notation*), stoga se vrlo lako obrađuje korištenjem jezika JavaScript. Slika 3.5 prikazuje primjer HTTP zahtjeva za geokodiranje adrese *Unska 3, Zagreb*.

https://maps.googleapis.com/maps/api/geocode/json?address=Unska+3,+Zagreb&key=YOUR_API_KEY

Slika 2.6: Prikaz HTTP zahtjeva za geokodiranjem adrese

Directions API

Directions API je usluga koja računa smjernice između lokacija. Smjernice se mogu tražiti za nekoliko načina prijevoza, uključujući javni prijevoz, vožnju automobilom, hodanje ili vožnju biciklom[4].

Pristup usluzi Directions vrši se preko HTTP sučelja. Zahtjev je standardiziranog oblika, konstruiran kao URL koristeći adrese ili zemljopisne širine i dužine za identifikaciju lokacija. Korištenjem jezika JavaScript može se postići dinamička promjena zahtjeva. Slika 3.6 prikazuje primjer HTTP zahtjeva za smjernice između lokacija *Unska 3* i *Ulica grada Vukovara 114*.

https://maps.googleapis.com/maps/api/directions/json?origin=Unska+3&destination=Ulica+grada+Vukovara+114&key=YOUR_API_KEY

Slika 2.7: Prikaz HTTP zahtjeva za smjernice

2.2.4. Bootstrap

Bootstrap je jedan od najpopularnijih HTML, CSS i JavaScript razvojnih okvira razvijen od strane Twittera, čija je svrha razvoj prilagođljivih web aplikacija. Sastoji se od niza predloška (engl. *templates*) razvijenih tehnologijama HTML i CSS, za uobičajene HTML elemente korisničkog sučelja poput tablica, formulara, navigacijskih

traka, padajućih izbornika... Bootstrap je moguće koristiti uz bilo koju poslužiteljsku tehnologiju.

Korištenje razvojnog okvira Bootstrap nosi mnoge prednosti. Web stranice koje koriste Bootstrap automatski se prilagođuju različitim veličinama ekrana, stoga mijenjaju svoj dizajn ovisno o uređaju na kojem se prikazuju. Korištenje Bootstrapa u razvoju aplikacije može uštedjeti mnogo vremena. Bootstrap nudi gotove blokove HTML, CSS i JavaScript koda koji se mogu koristiti i prilagoditi potrebama aplikacije.

Kako bi se koristio Bootstrap unutar web stanice potrebno je povezati stranicu sa CSS i JavaScript datotekama razvojnog okvira. Povezivanje sa CSS datotekama izvodi se u zaglavlju HTML dokumenta unutar praznog elementa `<link>` dok se povezivanje sa JavaScript datotekama izvodi u tijelu HTML dokumenta unutar oznaka `<script></script>`. Slika 2.8 prikazuje kôd jednostavne web stranice koja koristi razvojni okvir Bootstrap.

```
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet"
          href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
          integrity="sha384-ggOyR0iXcbMqv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
          crossorigin="anonymous">
    <title>Jednostavna web stranica</title>
  </head>
  <body>
    <h1>Stranica koristi razvojni okvir Bootstrap.</h1>

    <!-- Optional JavaScript -->
    <!-- jQuery first, then Popper.js, then Bootstrap JS -->
    <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
           integrity="sha384-q8i/X+965Dz00rT7abK41JSstQIAqVgRVzbzo5smXKp4YfRvH+8abTE1Pi6jizo"
           crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.14.7/dist/umd/popper.min.js"
           integrity="sha384-UO2eT0CpHqdsJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86diHNDz0W1"
           crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"
           integrity="sha384-JJSmVgyd0p3pXB1rRibZUAYoIIy6orQ6VrjIEaFF/nJGzIxFDsf4x0xIM+B07jRM"
           crossorigin="anonymous"></script>
  </body>
</html>
```

Slika 2.8: Jednostavna web stranica koja koristi razvojni okvir Bootstrap[5]

2.2.5. Razvojni okvir Spring

Razvojni okvir Spring olakšava stvaranje aplikacija koristeći platformu Java EE (engl. *Java Enterprise Edition*). Platforma Java EE pruža API i okruženje izvođenja potrebnih za razvoj velikih, višeslojnih, skalabilnih, pouzdanih i sigurnih mrežnih aplikacija[1]. Uz

programski jezik Java, Spring sadrži podršku za jezike Groovy i Kotlin kao alternativne jezike na JVM (engl. *Java virtual machine*).

Spring je razvojni okvir otvorenog koda. Ima veliku aktivnu zajednicu koja osigurava kontinuiranu povratnu informaciju temeljenu na raznolikim slučajevim korištenja u stvarnom svijetu[8].

Razvojni okvir Spring podijeljen je u module. Aplikacije mogu odabrati koji moduli su im potrebni. U središtu su moduli jezgrenog spremnika, uključujući i mehanizam za ubrizgavanje zavisnosti (engl. *dependency injection mechanism*). Ubrizgavanje zavisnosti je naziv za prijenos odgovornosti stvaranja objekta na neki drugi objekt i izravno korištenje te stvorene ovisnosti. Također, Spring osigurava temeljnu podršku za različite arhitekture aplikacija, uključujući slanje poruka, prijenos podataka te web.

Spring Boot

Spring Boot pruža dobru platformu za razvoj samostalnih i proizvodno spremnih aplikacija koje se mogu odmah pokrenuti. Također, omogućuje početak sa minimalnim konfiguracijama bez potrebe za čitavom konfiguracijom razvojnog okvira Spring.

Spring Boot konfigurira aplikaciju na temelju okoline, kao i onih informacija koje razvojni programer pruži. Aplikacija koja koristi Spring Boot potpuno je samostalna. Aplikaciju nije potrebno postaviti na web poslužitelj ili neko posebno okruženje kako bi se pokrenula.

Spring Web MVC

Spring MVC (engl. *Model-View-Controller*) okvir pruža Model-Pogled-Upravljač arhitekturu i spremne komponente koje se mogu koristiti za razvoj fleksibilnih web aplikacija. Uzorak MVC rezlutira razdvajanjem različitih aspekata aplikacije poput ulazne logike, poslovne logike i korisničkog sučelja, dok istovremeno pruža vezu između tih elemenata.

Model objedinjuje podatke aplikacije i općenito se sastoji od objekata POJO (engl. *Plain Old Java Object*). POJO je običan Java objekt nevezan nikakvim posebnim ograničenjima osim onih koja zahtijeva programski jezik Java. *Pogled* je odgovoran za prikazivanje podataka modela i općenito stvara HTML izlaz koji klijentski preglednik može interpretirati. *Upravljač* obrađuje korisničke zatjeve i odgovoran je za gradnju modela kojeg proslijeđuje pogledu za stvaranje prikaza. Slika 2.9 prikazuje upravljač zahtjeva potvrde korisničkog računa.

```

@Controller
@RequestMapping(value={"/validation"})
public class ValidationController {

    @Autowired
    private UserService userService;

    @GetMapping
    public String validateUser(Model model, @RequestParam("usr")String usr){
        User user=userService.findUserByEmailBcrypt(usr);
        if(user==null){
            model.addAttribute( "validation", "An error has occurred with account verification");
            return "home";
        }
        if(user.isVerified()){
            model.addAttribute( "validation", "Your account has already been verified.");
            return "home";
        }
        userService.updateUserVerifiedTrue(user.getId());
        model.addAttribute( "validation", "You have successfully verified your account. Now you can log in.");
        return "home";
    }
}

```

Slika 2.9: Upravljač zahtjeva potvrde korisničkog računa

Spring Security

Spring Security je snažan i vrlo prilagodljiv okvir za provjeru autentičnosti i pristupa. To je modul ravnojnog okvira Spring usmjeren na pružanje i provjeru autentičnosti i autorizacije za Java aplikacije. Snaga Spring Security modula nalazi se u lakoći proširenja u svrhu zadovoljenja određenih zahtjeva. Slika 2.10 prikazuje konfiguraciju modula Spring Security kako bi se svim korisnicima omogućili zahtjevi poput pregleda restorana sustava i njihovih jelovnika, a ograničili zahtjevi poput pregleda košarice ili pregleda narudžba samo na korisnike sa određenom ulogom u sustavu.

```

@Override
protected void configure(HttpSecurity http) throws Exception {
    http
        .authorizeRequests() ExpressionInterceptUrlRegistry
        .antMatchers( ...antPatterns: "/", "/home","/index","/register",
                     "/validation","/validation**","/restaurant-list","/restaurant-list**","/setRestaurantMenu",
                     "/setRestaurantMenu**","/js/**","/css/**","/img/**","/webjars/**").permitAll() ExpressionUrlAuth
        .antMatchers( ...antPatterns: "/my-restaurants*","/my-restaurants**").access( attribute: "hasRole('ROLE_OWNER')")
        .antMatchers( ...antPatterns: "/shopping-cart*","/shopping-cart**").access( attribute: "hasRole('ROLE_USER')")
        .antMatchers( ...antPatterns: "/order-list*","/order-list**","/take-order*","/take-order**",
                     "/active-order*","/active-order**","/complete-order*",
                     "/complete-order**").access( attribute: "hasRole('ROLE_WORKER')") ExpressionUrlAuthorizationConfigurer<
        .anyRequest().authenticated() ExpressionUrlAuthorizationConfigurer<HttpSecurity>.ExpressionInterceptUrlRegistry
        .and() HttpSecurity
        .formLogin() FormLoginConfigurer<HttpSecurity>
        .loginPage("/login") FormLoginConfigurer<HttpSecurity>
        .permitAll() FormLoginConfigurer<HttpSecurity>
        .and() HttpSecurity
        .logout() LogoutConfigurer<HttpSecurity>
        .permitAll();
}

```

Slika 2.10: Konfiguracija modula Spring Security

Spring Data JPA

Spring Data JPA (engl. *Java persistance API*), dio veće Spring Data obitelji, olakšava implementiranje spremišta (engl. *repositories*) baziranih na JPA. Ovaj se modul bavi podrškom JPA baziranom sloju za pristup podacima.

Implementacija sloja za pristup podacima je već dulje vrijeme neefikasna, popraćena sa puno programskog koda. Veliku količinu koda potrebno je napisati kako bi se izvršili jednostavni upiti nad bazom podataka. Spring Data JPA značajno poboljšava implementaciju slojeva za pristup podacima, smanjenjem potrebnog koda. Sve što je potrebno je definirati sučelja spremišta, uključujući prilagođene metode pretrage podataka, a Spring omogućava automatsku implementaciju. Slika 2.11 prikazuje implementaciju spremišta za pohranu korisnika sustava. Šest nevedenih metoda pretrage prilagođene su potrebama aplikacije.

```
public interface UserRepository extends JpaRepository<User, Long> {
    User findByEmail(String email);

    User findByBcrypt(String bcrypt);

    User findById(Long id);

    @Modifying(clearAutomatically = true)
    @Transactional
    @Query("UPDATE User user SET user.validated=true WHERE user.id=:userId")
    void updateUserValidatedTrue(@Param("userId") Long id);

    @Modifying(clearAutomatically = true)
    @Transactional
    @Query("UPDATE User user SET user.password=:pass WHERE user.id=:userId")
    void updateUserPassword(@Param("userId") Long id, @Param("pass") String password);

    List<User> findByRestaurantId(Long id);
}
```

Slika 2.11: Implementacija spremišta za pohranu korisnika sustava

2.2.6. Thymeleaf

Thymeleaf je Java knjižnica. To je XML / XHTML / HTML5 obrađivač predloška (engl. *template engine*) koji može primjeniti skup transformacija na datoteke predloška kako bi se prikazali podaci stvoreni aplikacijom. Thymeleaf je najprikladniji za obradu XHTML / HTML5 datoteka u web aplikacijama, ali može obraditi i bilo koju XML datoteku na webu ili samostalno.

Primarni cilj tehnologije Thymeleaf je pružiti elegantan i strukturiran način izrade predložaka. Kako bi se to postiglo, Thymeleaf je temeljen na XML oznakama

i atributima koji određuju izvođenje unaprijed definirane logike nad modelom DOM (engl. *Document Object Model*). Arhitektura tehnologije Thymeleaf omogućuje brzu obradu predloška, oslanjajući se na predmemoriranje obrađenih datoteka kako bi se koristila što manja količina ulazno izlaznih operacija.

Razvojni uzorak MVC je metoda za odvajanje problema unutar web aplikacije. Logika aplikacije odvojena je od tehnologija za prikaz informacija korisniku. Model prenosi podatke između logike aplikacije i sloja prikaza. Unutar aplikacije, sloj pregleda može koristiti jednu ili više različitih tehnologija za obradu pogleda. Web aplikacije razvijene u razvojnog okviru Spring podržavaju različite opcije prikaza koji se često nazivaju prikazni predlošci (engl. *view templates*). Te se tehnologije nazivaju predlošci jer pružaju jezik za označavanje koji omogućava izlaganje atributa modela unutar prikaza tijekom prikazivanja na strani klienta.

Slika 2.12 prikazuje dio HTML koda obogaćenog XML oznakama tehnologije Thymeleaf. Oznake sadrže prefiks *th* i dvotočje. Prikazani kod služi za prikaz podataka o korisničkim računima dostavljača pojedinog restorana. Prikaz se ostvaruje prolaskom po elementima atributa modela naziva *workerList*, koji je zapravo lista korisnika. Prikazuju se njihove email adrese sa pripadnim lozinkama.

```
<div class="row" th:unless="${#lists.isEmpty(workerList)}" th:each="worker : ${workerList}">
    <div class="col-sm-7">
        <pre><strong><i>Email:</i></strong></pre>
        <p th:text="${worker.email}"></p>
    </div>
    <div class="col-sm-3">
        <pre><strong><i>Pass:</i></strong></pre>
        <p th:text="${worker.id}"></p>
    </div>
    <div class="col-sm-2">
        <a class="btn btn-danger" href="#" th:href="@{/removeWorker(id=${worker.id},restaurant=${restaurant.id})}"
           sec:authorize="hasRole('ROLE_OWNER')">X</a>
    </div>
</div>
```

Slika 2.12: Prikaz podataka o korisničkim računima dostavljača

2.2.7. SQL

SQL (engl. *Structured Query Language*) je standardni računalni jezik za upravljanje relacijskim bazama podataka i manipulaciju podacima. Korištenjem jezika SQL nad bazom podataka mogu se provesti razne operacije poput umetanja podataka u postojeće tablice baze podataka i brisana podataka. Uz to, može se mijenjati sama struktura baze podataka stvaranjem, mijenjanjem i brisanjem tablica i drugih objekata baze podataka.

SQL kod podijeljen je u četiri glavne kategorije: upiti, dio jezika za manipulaciju podacima naziva DML (engl. *Data Manipulation Language*), dio jezika za definiranje podataka naziva DDL (engl. *Data Definition Language*) te dio jezika za upravljanje podacima naziva DCL (engl. *Data Control Language*).

Upiti se izvode pomoću SELECT izraza, koji je nadalje podijeljena na klauzule, uključujući SELECT, FROM, WHERE i ORDER BY.

DML se koristi za dodavanje, promjenu ili brisanje podataka. Zapravo je podskup izraza SELECT i sastoji se od INSERT, DELETE i UPDATE izraza, kao i kontrolnih izraza poput BEGIN TRANSACTION, SAVEPOINT, COMMIT i ROLLBACK.

DDL se koristi za upravljanje strukturama baze podataka. Sastoje se od izraza poput CREATE, ALTER, TRUNCATE i DROP.

DCL se koristi za dodjeljivanje i opoziv prava i dozvola nad bazama podataka. Glavni izrazi tog skupa su GRANT i REVOKE.

2.3. Baza podataka

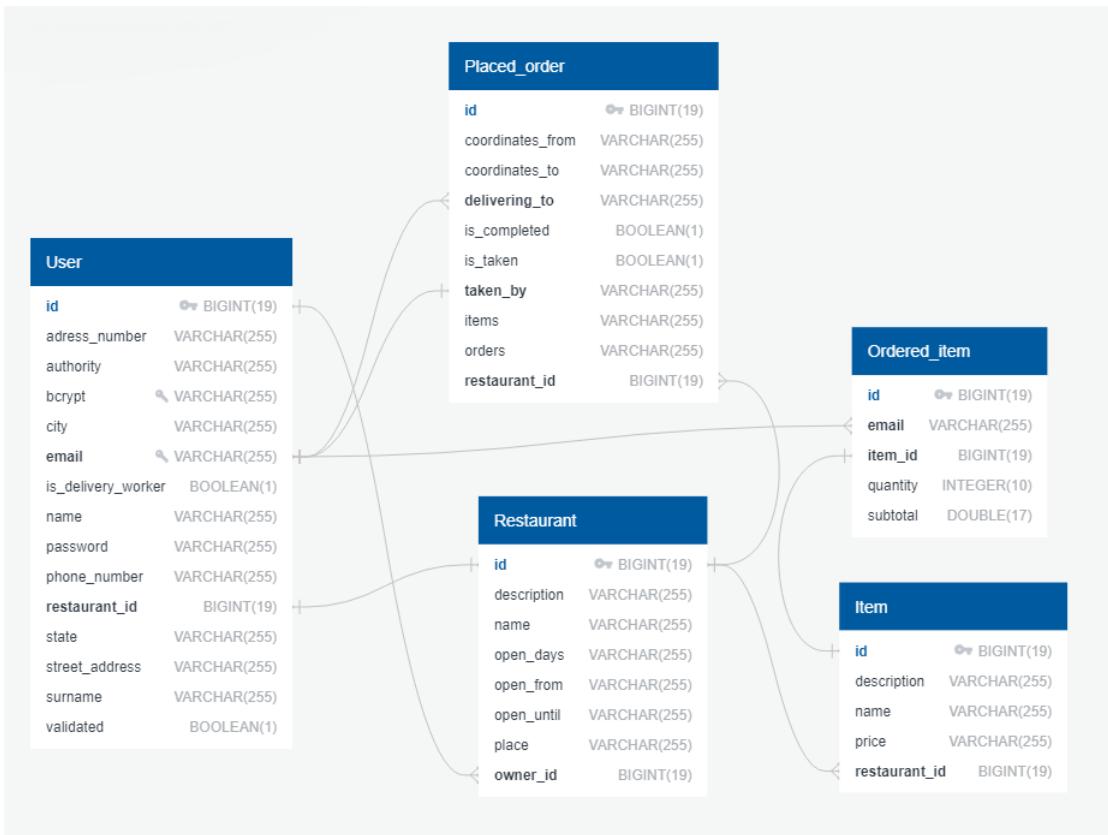
Baza podataka je skup zapisa pohranjenih u računalu na sustavan način. Svaki zapis uobičajeno se sastoji od skupa elemenata podataka. Računalni program korišten za upravljanje i ispitivanje baze podataka naziva se sustav upravljanja bazom podataka.

Aplikacija koristi PostgreSQL sustav upravljanja bazom podataka. PostgreSQL je objektno-relacijski sustav upravljanja bazom podataka koji koristi i proširuje sintaksu SQL jezika. PostgreSQL radi na svim većim operacijskim sustavima, a od 2001. godine je kompatibilan sa svojstvima ACID (atomarnost (engl. *atomicity*), dosljednost (engl. *consistency*), izoliranost (engl. *isolation*), trajnost (engl. *durability*))[7].

Slika 2.13 prikazuje dijagram baze podataka, a u nastavku slijedi kratki opis pojedine tablice.

2.3.1. Tablica User

Tablica User služi za pohranu svih registriranih korisnika. Sadrži podatke koje korisnik pruža prilikom registracije, kao i neke dodatne podatke bitne za rad aplikacije. Tablica User je centralna tablica u bazi podataka jer se na nju veže većina ostalih tablica. Primarni ključ tablice je *id* koji je cjelobrojnog tipa i postavljen na automatsko povećavanje svakim uzastopnim korisnikom dodanim u bazu. Polje *email* postavljeno je kao jedinstveni indeks kako bi se spriječilo registriranje više korisničkih računa putem jedne email adrese. Razlog tomu je prijava u sustav koja se vrši preko email



Slika 2.13: Dijagram baze podataka

adrese i pripadne lozinke. Polje *authority* služi za definiranje uloge korisnika u sustavu. Može poprimiti vrijednosti ROLE_OWNER (definira korisnika kao vlasnika restorana), ROLE_USER (definira korisnika kao kupca) te ROLE_WORKER (definira korisnika kao dostavljača hrane). Polja *bcrypt* i *validated* služe za validaciju korisnika putem elektroničke pošte. Polje *is_delivery_worker* i *restaurant_id* služe za definiranje korisnika kao dostavljača i vezu sa restoranom za kojeg je nadležan. Ostala polja sadrže osobne podatke korisnika.

2.3.2. Tablica Restaurant

Tablica Restaurant služi za pohranu svih restorana sustava. Sadrži podatke koje korisnik sustava sa ulogom vlasnika restorana pruža prilikom dodavanja restorana u sustav. Ti podaci sadrže ime i opis restorana, kao i dane u tjednu tijekom kojih je restoran otvoren sa pripadnim radnim vremenom. Primarni ključ tablice je *id* koji je cjelobrojnog tipa i postavljen na automatsko povećavanje svakim uzastopnim restoranom dodanim u bazu. Polje *place* sadrži geografsku lokaciju (u obliku geografske širine i dužine) koju je korisnik postavio za svoj restoran. Polje *owner_id* služi kao veza sa tablicom

User koja određuje vlasnika tog restorana.

2.3.3. Tablica Item

Tablica Item služi za pohranu prehrambenih proizvoda. Sadrži podatke koje korisnik sustava sa ulogom vlasnika restorana pruža prilikom dodavanja proizvoda u sustav. Ti podaci sadrže ime i opis proizvoda kao i njegovu jediničnu cijenu. Primarni ključ tablice je *id* koji je cjelobrojnog tipa i postavljen na automatsko povećavanje svakim uzastopnim proizvodom dodanim u bazu. Element *restaurant_id* služi kao veza sa tablicom Restaurant te određuje pripadnost tom restoranu.

2.3.4. Tablica Ordered_item

Tablica Ordered_item služi za pohranu prehrambenih proizvoda koji se nalaze u košarici pojedinog korisnika. Primarni ključ tablice je *id* koji je cjelobrojnog tipa i postavljen na automatsko povećavanje svakim uzastopnim dodavanjem proizvoda u košaricu. Polje *email* sadrži email adresu korisnika koji je stavio proizvod u košaricu te služi kao veza prema tablici User. Polje *item_id* služi za povezivanje sa tablicom Item te određuje naručeni proizvod. Polja *quantity* i *subtotal* određuju brojnost i ukupnu cijenu naručenog proizvoda.

2.3.5. Tablica Placed_order

Tablica Placed_order služi za pohranu narudžba sustava. Sadrži podatke koji se proslijedu dostavljačima hrane. Svrha ove tablice je omogućavanje naručivanja prehrambenih proizvoda iz više različitih restorana. Primarni ključ tablice je *id* koji je cjelobrojnog tipa i postavljen na automatsko povećavanje svakim uzastopnim dodavanjem narudžbe u bazu. Polja *coordinates_from* i *coordinates_to* definiraju geografsku lokaciju restorana te geografsku lokaciju koju je pružao korisnik prilikom narudžbe. Te se lokacije koriste prilikom prikaza uputa dostavljaču putem usluge Directions. Polje *delivering_to* služi kao poveznica sa tablicom User i određuje korisnika koji je podnio narudžbu. Polja *is_taken* i *taken_by* ažuriraju se prilikom preuzimanja naružbe. Svrha tih polja je onemogućavanje preuzimanja narudžbe od strane više dostavljača. Ukoliko je dostava izvršena, ažurira se vrijednost polja *is_completed*. Polje *items* sadrži ključeve proizvoda, dok *quantities* sadrži brojnost proizvoda narudžbe. Polje *restaurant_id* služi kao poveznica sa tablicom Restaurant te određuje pripadnost pojedinom restoranu.

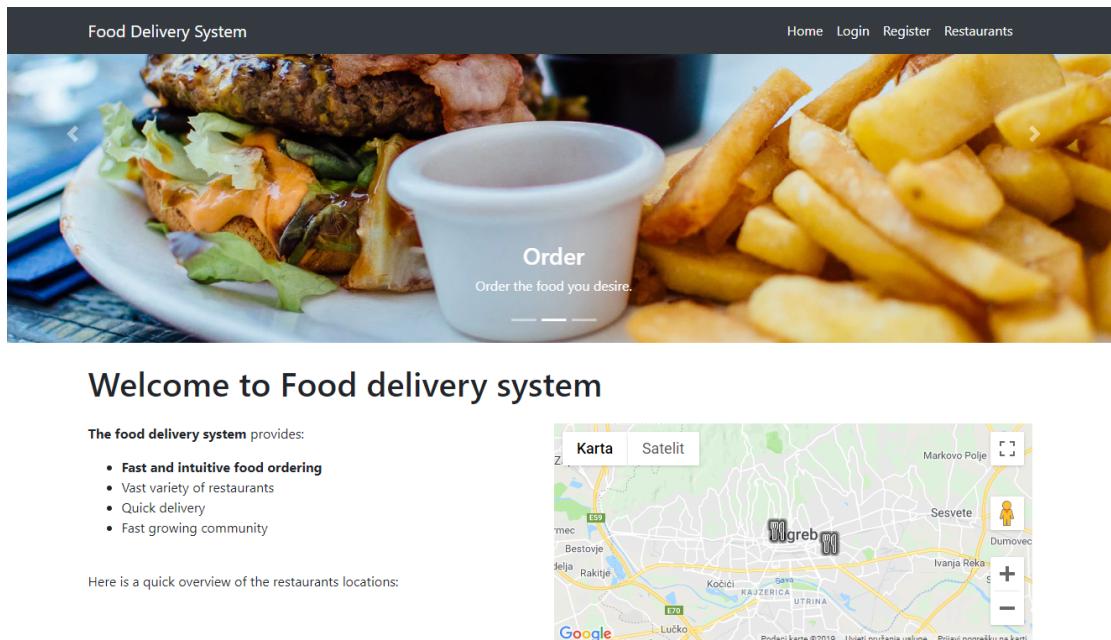
2.4. Opis funkcionalnosti

Sustav se sastoji od više stranica koje nude različite funkcionalnosti. Slijedi opis pojedinih stranica.

2.4.1. Početna stranica

Početna stranica korisnicima prikazuje kratki opis usluga koje aplikacija pruža. Također, pružan je pregled lokacija svih restorana sustava korištenjem usluge Google Maps. Sadržaj navigacijske trake razlikuje se ovisno o ulozi pojedninog korisnika u sustavu.

Slika 2.14 prikazuje početnu stranicu neprijavljenog korisnika. Navigacijska traka sastoje se od poveznica *Home*, *Login*, *Register* i *Restaurants*. Tim poveznicama korisnik ostvaruje tražene funkcionalnosti sustava. Pristup stranici za prijavu u sustav ostvaruje se poveznicom *Login*, a pristup stranici za izradu novog korisničkog računa poveznicom *Register*. Korisnik pristupa stanici za pregled svih restorana sustava poveznicom *Restaurants*. Poveznica *Home* korisnika prosljeđuje ponovno na početnu stranicu.



Slika 2.14: Početna stranica naprijavljenog korisnika

Navigacijska traka svih prijavljenih korisnika sadrži prikaz korisničkog imena korištenog za prijavu u sustav te mogućnost odjave iz sustava poveznicom *Sign out*.

Slika 2.15 prikazuje navigacijsku traku prijavljenog korisnika sa ulogom kupca. Kao i kod neprijavljenog korisnika, traka sadrži poveznice *Home* i *Restaurants*. Ta-

kođer, kupcu je omogućena narudžba hrane stoga mu je prikazana poveznica u obliku košarice za kupnju. Putem nje, korisnik pristupa svojoj virtualnoj košarici.



Slika 2.15: Navigacijska traka prijavljenog korisnika sa ulogom kupca

Slika 2.16 prikazuje navigacijsku traku prijavljenog korisnika sa ulogom vlasnika restorana. Uz poveznicu *Home*, traka vlasnika restorana sadrži i poveznicu *My restaurants*. Tom poveznicom korisnik pristupa stranici sa popisom njegovih restorana te opcijom dodavanja novog.



Slika 2.16: Navigacijska traka prijavljenog korisnika sa ulogom vlasnika restorana

Navigacijska traka prijavljenog korisnika sa ulogom dostavljača hrane nalazi se na slici 2.17. Kao i kod svih korisnika, traka sadrži poveznicu *Home*. Također, traka sadrži poveznice *Active order* i *Orders*. Prisup popisu svih narudžbi restorana za kojeg je nadležan korisnik ostvaruje poveznicom *Orders*. Poveznicom *Active order* korisnik pristupa stranici svoje trenutno aktivne dostave.

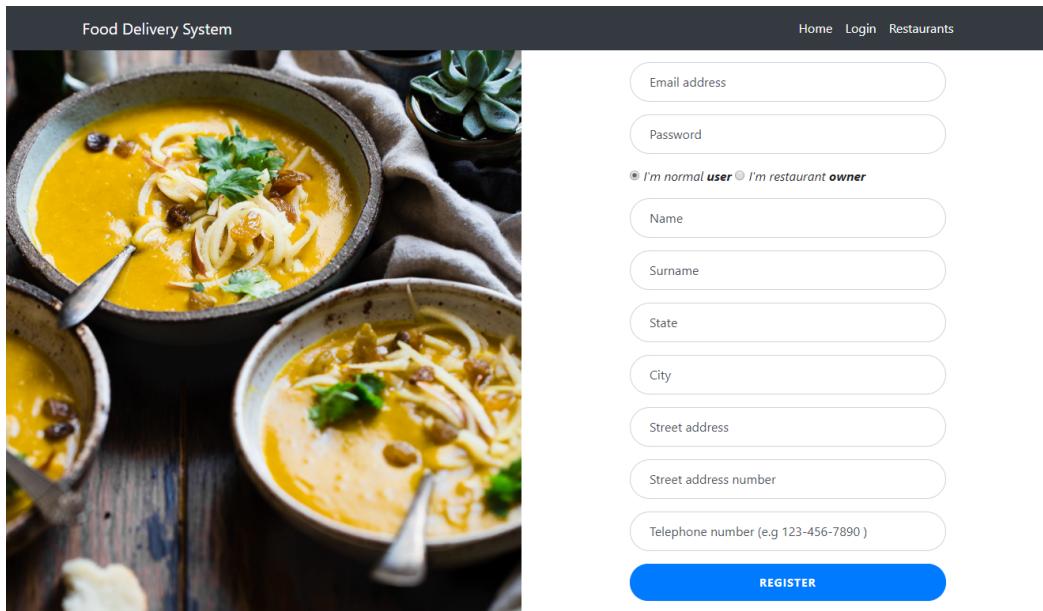


Slika 2.17: Navigacijska traka prijavljenog korisnika sa ulogom dostavljača hrane

2.4.2. Registracija

Korisnik izrađuje novi korisnički račun upisujući potrebne podatke u obrazac na stranici registracije. Stranica registracije prikazana je na slici 2.18. Sva polja obrasca su obavezna. Sadrže podatke poput email adrese i lozinke koje će korisnik kasnije koristiti prilikom prijave, kao i osobne podatke poput imena i prezimena, kontakt broja te adrese prebivališta. Prilikom registracije korisnik izjavljuje koju ulogu želi poprimiti u sustavu. Izabire između uloge kupca i uloge vlasnika restorana.

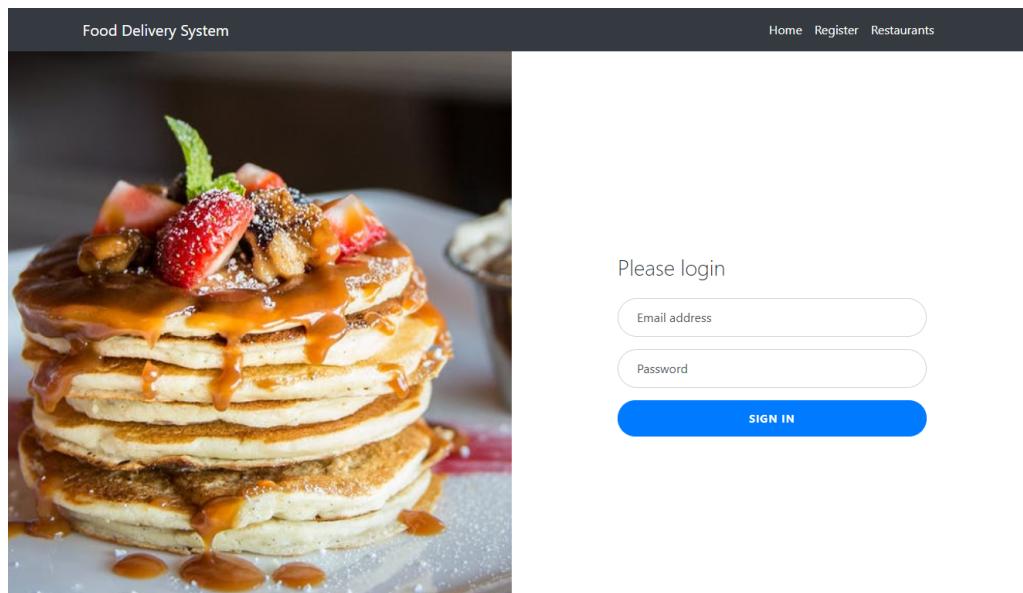
Nakon uspešne registracije koja iziskuje email adresu koja je jedinstvena u sustavu, na tu adresu šalje se verifikacijska poveznica koja potvrđuje izradu korisničkog računa. Ukoliko se račun ne potvrdi, korisniku je onemogućena prijava u sustav.



Slika 2.18: Registracijska stranica

2.4.3. Prijava

Korisnik se prijavljuje koristeći svoju email adresu i lozinku. Slika 2.19 prikazuje stranicu prijave. Preduvjet uspješnoj prijavi korisnika je posjedovanje korisničkog računa koji je verificiran putem email adrese. Ukoliko korisnik unese podatke koji se ne nalaze u sustavu ispisuju mu se odgovarajuće poruke upozorenja.



Slika 2.19: Stranica za prijavu u sustav

2.4.4. Pregled restorana sustava

Neprijavljeni korisnici te korisnici sa ulogom kupca u sustavu mogu pristupiti stranici za pregled svih restorana sustava koja je prikazana na slici 2.20. Stranica sadrži popis restorana sa njihovim kratkim opisima. Kratki opis sadrži naziv restorana, opis, radno vrijeme te dane u tjednu tijekom kojih je restoran otvoren za poslovanje. Također, korisnicima se prikazuje slika restorana učitana od strane vlasnika. Poveznica *View menu* vodi korisnika na stranicu za pregled jelovnika restorana.

The screenshot shows a dark-themed web interface for a food delivery system. At the top, there's a navigation bar with the text "Food Delivery System" on the left and "Home Login Register" on the right. Below the navigation bar, the main content area has a heading "Restaurant list Find a restaurant that best suits your taste".
Red Lobster
Seafood restaurant serving only the freshest ingredients.
open days : M T W
open hours : 08:30 - 20:00
[View Menu](#)

Batak
A nice Croatian restaurant serving local food.
open days : M W THU SAT
open hours : 08:00 - 17:00

Slika 2.20: Stranica za pregled svih restorana sustava

2.4.5. Pregled vlastitih restorana

Korisnicima sa ulogom vlasnika u sustavu omogućen je pregled vlastitih restorana. Stranica prikazana na slici 2.21 pruža korisniku pregled svih restorana koje je pret-hodno dodao u sustav. Kao i kod pregleda restorana sustava, svaki restoran prikazan je svojim kratkim opisom i slikom. Vlasniku je također omogućeno uklanjanje restorana iz sustava korištenjem poveznice *Remove* te odlazak na stranicu za pregled jelovnika putem povezince *Manage restaurant*.

Također, stranica omogućava dodavanje novog restorana u sustav korištenjem obrasca prikazanog na slici 2.22. Sva polja obrasca su obavezna. Potrebno je unijeti naziv restorana, označiti dane u tjednu tijekom kojih je restoran otvoren, kratko opisati restoran, zadati njegovu lokaciju te učitati sliku koja će predstavljati taj restoran. Lokaciju res-

The screenshot shows a dark-themed web application. At the top, there's a header bar with the text "Food Delivery System" on the left, "Home restaurant.owner@gmail.com" in the middle, and a "Sign out" button on the right. Below the header, the main content area has a title "Restaurant list Here you will find the list of your restaurants". To the left is a photograph of a restaurant interior with tables and chairs. To the right of the photo is the restaurant's name "Red Lobster" and its description: "Seafood restaurant serving only the freshest ingredients." Below the description are two buttons: "Manage restaurant" (blue) and "Remove" (red). At the bottom of the page, there are two buttons: "Add a new restaurant:" and "Add a restaurant".

Copyright © Kristijan Vrbanc, 2019

Slika 2.21: Stranica za pregled vlastitih restorana

torana moguće je unijeti na dva načina. Prvi način je unosom adrese u polje *location* i pritiskom na gumb *Check the address* koji postavlja oznaku na kartu. Drugi način je postavljanjem oznake na kartu, čime se automatski postavlja pripadajuća adresa u polje *location* što služi kao dodatna provjera prilikom unosa.

The screenshot shows a form for adding a new restaurant. The fields include:

- Restaurant name:** Red Lobster
- Open days:** M T W T F S S
- Open hours:** From: 08:30 Until: 20:00
- Description:** e.g. A nice family restaurant serving chinese food..
- Location:** A map of Zagreb with several location markers. Below the map is a text input field labeled "Location(address)" and a yellow "Check the address" button.
- Restaurant image:** A note stating "Odaberite datoteku" and "Nije odabrana niti jedna datoteka." Below this is a green "Add a restaurant" button and a "Hide" button.

Slika 2.22: Obrazac za dodavanje restorana u sustav

2.4.6. Pregled jelovnika

Stranica za pregled jelovnika dostupna je svim korisnicima osim korisicima sa ulogom dostavljača. Pogled na stranicu ovisi o ulozi korisnika u sustavu.

Slika 2.23 prikazuje pogled neprijavljenog korisnika te korisnika sa ulogom kupca. Prikazan je kratki opis restorana koji sadži njegovo ime, radno vijeme te dani u tjednu tijekom kojih je restoran otvoren. Također, prikazana je slika restorana te njegova lokacija korištenjem usluge Google Maps. Korisniku je prikazan popis prehrabnenih proizvoda restorana. Svaki prehrabeni proizvod prikazan je slikom, nazivom, opisom te jedniničnom cijenom u hrvatskim kunama. Korisniku sa ulogom kupca omogućeno je dodavanje proizvoda u košaricu uz naznaku količine korištenjem poveznice *Add to cart*. Ukoliko se proizvod već nalazi u košarici, ispisuje se pripadna poruka. Neprijavljenom korisniku onemogućeno je dodavanje proizvoda u košaricu uz ispis pripadne poruke.

The screenshot shows the Food Delivery System interface. At the top, it says "Food Delivery System" and "Red Lobster". It displays the restaurant's opening days (M T W), hours (08:30 - 20:00), and a map showing its location in Zagreb, Croatia. A large image of the restaurant's interior is shown. Below the map, there are two food items: "Pizza" (15 kn, A yummy Italian pizza) and "Sushi" (30 kn, Japanese salmon sushi). Each item has a quantity input field (Qty: 0) and an "Add to cart" button.

Slika 2.23: Pogled neprijavljenog korisnika te kupca na stranicu za pregled jelovnika

Slika 2.24 prikazuje pogled korisnika sa ulogom vlasnika restorana. Pogled dijeli elemente pogleda neprijavljenog korisnika i kupca, uz neke razlike. Korisniku nije prikazana lokacija već popis dostavljača koje je pridjelio tom restoranu. Također, omogućeno mu je dodavanje novih dostavljača u sustav putem obrasca. Dodavanje novog dostavljača iziskuje unos jedinstvene email adrese i odabir poveznice *Add worker*. Ukoliko je dostavljač uspješno dodan, vlasniku se prikazuje email adresa te pripadna lozinka dostavljača koja je automatski generirana. Pomoću tih podataka dostavljač ima

mogućnost prijave u sustav. Ukoliko dostavljač nije uspješno dodan, vlasniku se ispisuje odgovarajuća poruka. Također, vlasnik ima mogućnost uklanjanja dostavljača iz sustava.

Slika 2.24: Pogled vlasnika restorana na stranicu za pregled jelovnika

Vlasnik također ima uvid u sve prehrambene proizvode prethodno dodane u sustav. za svaki proizvod, omogućeno mu je uklanjanje proizvoda iz sustava putem poveznice *Remove*. Također, vlasnik je u mogućnosti dodavati nove proizvode u sustav putem obrasca na slici 2.25. Sva polja obrasca su obavezna. Obrazac zahtjeva unos imena proizvoda, opisa, jedinične cijene kao i slike koja će predstavljati proizvod.

Add a menu item

Item name <input type="text" value="e.q Pizza"/>	Unit price (in kn) <input type="text" value="e.q 10"/>
Description <div style="border: 1px solid #ccc; height: 40px; margin-top: 5px;"></div>	
Item image <input type="button" value="Odaberi datoteku"/> Nije odabrana niti jedna datoteka.	<input type="button" value="Add item"/> <input type="button" value="Hide"/>

Slika 2.25: Obrazac za dodavanje prehrambenog proizvoda u sustav

2.4.7. Košarica

Prijavljenom korisniku sustava sa ulogom kupca omogućen je pregled virtualne košarice. U košarici se nalazi popis svih jela pripravnih za postavljanje narudžbe. Za svaki proizvod prikazana je slika tog proizvoda, naziv, jedinična cijena, postavljena količina te ukupna cijena proizvoda. Korisniku je omogućena promjena količine proizvoda postavljanjem polja količine te odabirom poveznice *Change Qty.*. Također, kupcu je omogućeno uklanjanje proizvoda iz košarice.

Prilikom potvrde narudžbe, korisnik pruža lokaciju dostave. Lokacija se zadaje u polju iznad karte te pritiskom na gumb *Check address* koja postavlja oznaku na kartu, ili postavljanjem oznake na kartu koja odgovarajuću adresu postavlja u polje adrese. Slika 2.26 prikazuje stranicu virtualne košarice.

Product	Price	Quantity	Subtotal
Pizza	15 kn	2	30.0 kn
Sushi	30 kn	1	30.0 kn

Total: 60.0 kn

Provide the location for your delivery:
(provide the address in the input field below or place a marker on the map)

Check address

Karta Satelit

Zagreb

Checkout >

Slika 2.26: Stranica virtualne košarice

2.4.8. Pregled narudžba

Pregled narudžba restorana omogućen je korisniku sa ulogom dostavljača hrane. Stranica koja se nalazi na slici 2.27 prikazuje popis svih nepreuzetih narudžba restorana za kojeg je dostavljač nadležan. Za svaku narudžbu, prikazan je broj narudžbe, korisničko ime osobe koja je podnijela narudžbu, količina naručenih proizvoda te ukupna cijena narudžbe. Korisniku je omogućeno preuzimanje dostave poveznicom *Take order* nakon čega se narudžba uklanja sa popisa.

Food Delivery System

Home Active order worker1@gmail.com Sign out

Order num. :	User:	Num. of items:	Total	
1	buyer@gmail.com	2	60.0 kn	Take order
2	buyer@gmail.com	1	90.0 kn	Take order
3	buyer@gmail.com	2	75.0 kn	Take order

Orders Pick an order you wish to deliver:

Copyright © Kristijan Vrbanc, 2019

Slika 2.27: Stranica za pregled narudžba

2.4.9. Aktivna dostava

Stranica aktivne dostave, prikazana na slici 2.28, omogućena je korisniku sa ulogom dostavljača hrane. Ukoliko je korisnik preuzeo dostavu neke narudžbe, prikazuju mu se svi podaci bitni za dostavu.

Food Delivery System

Home Orders worker1@gmail.com Sign out

Product	Price	Quantity
Pizza	15 kn	2
Sushi	30 kn	1

Total: 60.0 kn

User's email: buyer@gmail.com Contact num.: 123-456-7890

Link to Google Maps directions: [Directions](#)

Complete delivery: [Delivery completed](#)

A map shows the delivery route from point A (Hrvatsko narodno kazalište u Zagrebu) to point B (Cetina dvorana - Petar Preradović). The route is highlighted in blue on a green map of the city center.

Slika 2.28: Stranica aktivne dostave

Prikazan je popis proizvoda koje je potrebno dotaviti, njihova jedinična cijena, količina i ukupna cijena. Također, dostavljaču je prikazano korisničko ime korisnika

koji je podnio narudžbu, kao i korisnikov kontakt broj. Dostavljač na uvid dobiva kartu sa ucrtanim putem od restorana do lokacije koje je korisnik postavio za dostavu. Ukoliko dostavljač pristupa stranici preko pametnog telefona, poveznica *Directions* otvara aplikaciju usluge Google Maps sa prikazom smjernica od restorana do tražene lokacije.

Korištenjem poveznice *Delivery completed* korisnik završava dostavu.

3. Zaključak

Izrada web aplikacije zahtjevan je zadatak. Prije samog početka izrade potrebno je njeni dobro planiranje. Potrebno je definirati sve funkcionalne zahtjeve koje bi korisnici mogli imati na aplikaciju jer se tako mogu otkriti potencijalni problemi izrade. Također, na temelju funkcionalnih zahtjeva moguće je sagraditi rani koncept aplikacije koji se koristi prilikom izbora tehnologija za razvoj.

Izbor razvojnih tehnologija također je vrlo važan proces. Dobar izbor tehnologija može uvelike olakšati izradu aplikacije. Potrebno je izabrati tehnologije koje pružaju mnogo funkcionalnosti. U kontekstu poslužiteljskih tehnologija to su funkcionalnosti poput sigurnosti aplikacije, pristupa bazi podataka, jednostavnog načina posluživanja korisničkih zahtjeva... Kod klijentskih tehnologija potrebno je izabrati tehnologije koje pružaju veliku količinu gotovog koda, ali i jednostavan i intuitivan način za prilagođavanje tih gotovih dizajna i stvaranja svojih. Također, bitno je da tehnologije imaju što veću aktivnu zajednicu korisnika koji bi mogli pomoći pri učenju tehnologija te samoj izradi aplikacije.

Organizacija koda također je jako bitna. Kod je potrebno organizirati u razrede i tematske pakete jer se time dobiva na modularnosti koda te lakše dodavaju novi dijelovi aplikacije i prepravljuju postojeći.

Naposljetu, kao zadnju fazu izrade aplikacije potrebo je provesti testiranje. Prolaze se sve zahtjevane funkcionalnosti korisnika te se isprobavaju svi mogući scenariji korištenja aplikacije od strane korisnika. Testiranje služi kako bi se ispravili nedostaci u kodu, a samim time i povećalo zadovoljstvo korisnika.

Imajući sve to u vidu, možemo ustanoviti da izrada web aplikacije nije lak zadatak. Izrada se sastoji od mnoga faza koje iziskuju puno proučavanja, učenja, prilagođavanja ali i maštovitosti.

LITERATURA

- [1] An introduction to the java ee platform. <https://docs.oracle.com/javaee/6/firstcup/doc/gkhoy.html>. (01/06/2019).
- [2] Dynamic programming language. https://developer.mozilla.org/en-US/docs/Glossary/Dynamic_programming_language. (26/05/2019).
- [3] Geocoding api. <https://developers.google.com/maps/documentation/geocoding/start,.> (27/05/2019).
- [4] Directions api. <https://developers.google.com/maps/documentation/directions/start,.> (27/05/2019).
- [5] Bootstrap: Introduction. <https://getbootstrap.com/docs/4.3/getting-started/introduction/>. (27/05/2019).
- [6] Maps javascript api. <https://developers.google.com/maps/documentation/javascript/tutorial>. (27/05/2019).
- [7] Postgresql: About. <https://www.postgresql.org/about/>. (04/06/2019).
- [8] Spring framework overview. <https://docs.spring.io/spring/docs/current/spring-framework-reference/overview.html>. (01/06/2019).
- [9] Uml-dijagrami: zbirka riješenih zadataka. http://www.graphis.hr/news/uml-dijagrami/uml_dijagrami_za_web.pdf. (10/05/2019).

Sustav za dostavu hrane

Sažetak

Cilj ovoga rada je izrada web aplikacije. Sustav za dostavu hrane internet je trgovina gdje korisnici na lak i intuitivan način vrše pregled jelovnika i postavljaju narudžbe. Rad opisuje odabrane tehnologije upotrijebljene pri razvoju aplikacije, kao i samu strukturu aplikacije te njenu funkcionalnost .

Odabранe razvojne tehnologije dijele se na klijentske i poslužiteljske. Klijentske tehnologije služe za definiranje sučelja prema korisniku, svega što korisnik vidi i sa čime može imati interakciju. Korištene klijentske tehnologije su HTML, CSS, JavaScript te razvojni okvir Bootstrap. Poslužiteljske tehnologije služe za obradu podataka generiranih od strane korisnika. Korištene poslužiteljske tehnologije su Thymeleaf te razvojni okvir Spring. Konkretno, korištena su četiri modula razvojnog okvira Spring: Spring Boot, Spring Security, Spring MVC te Spring Data JPA. Za komunikaciju sa bazom podataka korišten je jezik SQL.

Aplikacija podržava četiri tipa korisnika. Pogled korisnika na aplikaciju ovisi o njegovojoj ulozi u sustavu.

Ključne riječi: web aplikacija, restoran, hrana, narudžba, dostava, smjernice, razvojni okvir Spring, Bootstrap

Abstract

The aim of this paper is to create a web application. The food delivery system is a web shop that delivers an easier and more intuitive way to browse menus and place orders. The paper describes the selected technologies used in the development of the application, as well as the very structure of the application and its functionality.

Selected development technologies are divided into client and server technologies. Client technologies are used to define user interface, all that user sees and can interact. The client technologies used are HTML, CSS, JavaScript and Bootstrap framework. Server technologies are used to process data generated by users. The server technologies used are Thymeleaf and the Spring framework. Specifically, four of the Spring framework modules used are: Spring Boot, Spring Security, Spring MVC and Spring Data JPA. The SQL language was used to communicate with the database.

The application supports four types of users. The user's view of the application depends on his role in the system.

Keywords: web application, restaurant, food, order, delivery, directions, Spring framework, Bootstrap