

SVEUČILIŠTE U SPLITU
SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

Preddiplomski stručni studij Informacijska tehnologija

MATEA ORDULJ

Z A V R Š N I R A D

**IZRADA WEB APLIKACIJE ZA
PRETRAŽIVANJE RESTORANA**

Split, veljača 2022.

SVEUČILIŠTE U SPLITU
SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

Preddiplomski stručni studij Informacijska tehnologija

Predmet: Oblikovanje web stranica

Z A V R Š N I R A D

Kandidat: Matea Ordulj

Naslov rada: Izrada web aplikacije za pretraživanje restorana

Mentor: Haidi Božiković, viši predavač

Split, veljača 2022.

SADRŽAJ

SAŽETAK.....	1
SUMMARY	2
1. UVOD	3
2. KORIŠTENE TEHNOLOGIJE	4
2.1 Hypertext Markup Language	4
2.2 Cascading Style Sheets	4
2.3 JavaScript	6
2.4 Node.js	6
2.5 Express.js	7
2.6 Bootstrap	7
2.7 MongoDB	7
3. OPIS PRAKTIČNOG RADA	9
3.1 Klijentski dio	9
3.1.1 Pretraga restorana	9
3.1.2. Gosti aplikacije	12
3.1.3. Prijavljeni korisnici	14
3.1.4 Dodavanje novog restorana	17
3.1.5. Dodavanje recenzije	19
3.1.6. Administratorski dio.....	20
3.2. Dizajn	21
3.3 Poslužiteljski dio aplikacije	23
3.3.1. Stvaranje poslužitelja	23
3.3.2. Model-View-Controller	23
3.3.3. Učitavanje slika	26
4. ZAKLJUČAK.....	28
LITERATURA.....	29
Popis slika	30
Popis ispisa	31

SAŽETAK

Cilj ovog završnog rada je izrada web aplikacije za dodavanje novih i pretraživanje postojećih restorana. Pretraga se temelji na lokaciji i recenziji restorana. Izrađen je korisnički i administratorski dio web aplikacije. Gostima aplikacije je omogućeno samo pregledavanje, pretraživanje i filtriranje restorana dok je registriranim korisnicima uz to, omogućeno i dodavanje novih restorana kao i ocjenjivanje postojećih. Pretraga objekata putem lokacije odrađena je preko karte, gdje se jedinice u neposrednoj blizini grupiraju u značku zbog jednostavnijeg prikaza na karti, a sama značka prikazuje broj objekata u skupini. Karta se može povećati da bi se vidjeli pojedinačni objekti odnosno udaljiti da bi se isti ponovno grupirali. Administrator ima mogućnost upravljati restoranima, korisnicima i recenzijama.

Prilikom izrade ovog završnog rada korištene su tehnologije HTML (*HyperText Markup Language*), CSS (*Cascading Style Sheets*), JavaScript, NodeJS, Bootstrap i MongoDB baza podataka.

Ključne riječi: *karta, recenzije, restorani*

SUMMARY

Development of web application for restaurant search

The aim of this final work is to create a web application for adding new and searching existing restaurants. The search is based on the location and reviews of the restaurant. The user and administrator part of the web application has been created. Guests of the application are only allowed to browse, search and filter restaurants, while registered users are also allowed to add new restaurants and rate existing ones. The search of objects by location was done via a cluster map, where the units in the immediate vicinity are grouped for easier display on the map, and the pin itself shows the number of objects in the cluster. The map can be enlarged to see individual objects or zoomed out to group them again. The administrator has the ability to manage restaurants, users and reviews.

Technologies used for this project are HTML (HyperText Markup Language), CSS (Cascading Style Sheets), JavaScript, NodeJS, Bootstrap and MongoDB database.

Keywords: *map, reviews, restaurants*

1. UVOD

U ovom radu opisana je izrada i funkcionalnost web aplikacije za pretraživanje restorana. Web aplikacija „Fine-Eats“ može uvelike olakšati pretragu restorana turistima, ali i domaćem stanovništvu. U današnje vrijeme ljudi su naviknuti da dođu do bilo koje informacije sa samo par klikova preko pametnog telefona i sve šire je razmišljanje “ako nema na internetu, nije se dogodilo“. S ovom aplikacijom nema više lutanja ulicom u potrazi za restoranom niti putovanja autom i čekanja idućeg restorana koji naiđe. Hrvatska, kao mala zemlja, velika je turistička destinacija stoga se treba još više okrenuti online marketingu s ciljem privlačenja mlađe populacije. Aplikacija omogućava malim obiteljskim restoranima, koji svoju ponudu zasnivaju na lokalnim, tradicionalnim jelima, da se približe široj masi ljudi i tako izjednače mogućnost zarade i posjećenosti sa velikim brendiranim lancima restorana koji ostvaruju veliki profit na temelju svoga imena.

Osim što im je ovim putem omogućena bolja promocija, imaju priliku poboljšati svoju kompletnu ponudu, a samim tim steći veću popularnost te proširiti posao, a sve to uz pomoć recenzija koje gosti mogu ostaviti putem aplikacije. Svojim pozitivnim komentarima gosti pružaju sigurnost svim budućim zainteresiranim posjetiteljima, a vlasnicima je u interesu da ih skupe što više, i na taj način dobiju na popularnosti. Također, omogućeno im je da na svaku kritiku pravovremeno reagiraju, odnosno da prepoznaju i isprave sve postojeće probleme i samim tim se što bolje pripreme za daljnje poslovanje.

2. KORIŠTENE TEHNOLOGIJE

2.1 Hypertext Markup Language

HTML je prevladavajući i trenutno najpopularniji jezik za razvoj web stranica. HTML je jezik za opisivanje strukture tekstualnih detalja u datoteci korištenjem specifičnih oznaka i korištenjem određenih tekstualnih veza, zaglavlja, paragrafa, tablica, popisa i naslova. HTML daje upute web pregledniku kako prikazati tekst i slike na web stranici.

Godine 1989. Tim Berners Lee razmatrao je probleme s kojima se susreću znanstvenici Europske organizacije za nuklearna istraživanja, CERN, u dijeljenju svojih istraživanja [1]. Kad bi znanstvenici zatražili dokument od svojeg kolege, trebali su ili naučiti koristiti drugo računalo, odnosno program za pisanje koji koristi njihov kolega ili su trebali promijeniti samu vrstu dokumenta kako bi odgovarala njihovom programu. Stoga je Berners-Lee započeo dizajniranje svog jednostavnog hipertekstualnog jezika temeljenog na SGML-u (engl. *Standard Generalized Markup Language*). Ideja Berners-Leeja bila je model u kojem se korisnik može kretati od jednog skupa informacija na jednom računalu do drugog skupa informacija na drugom računalu.

Tim Berners-Lee je stvorio HTML krajem 1991., ali te godine nije službeno objavljen. Izdan je tek 1995. kao HTML 2.0. [2] .

2.2 Cascading Style Sheets

Dok se HTML koristi za kostur web dokumenta, CSS određuje vizualni dio stranice. Odnosno, dok HTML u velikoj mjeri određuje tekstualni sadržaj, CSS određuje vizualnu strukturu, izgled i estetiku. HTML je jezik za označavanje, a CSS je jezik stilskih predložaka koji omogućava odvajanje stiliziranja dokumenta od sadržaja samog dokumenta što olakšava bolje snalaženje unutar dokumenta.

Postoje tri načina uključivanja CSS-a u HTML dokument [3]. Unutarnji ili ugrađeni CSS zahtijeva dodavanje oznake `<style>` u zaglavlje HTML dokumenta, koje je označeno oznakom `<head>`. Linijski CSS se koristi za stiliziranje određenog HTML elementa, gdje se svakoj HTML oznaci bez korištenja selektora, može dodati atribut `style` i na taj način definirati stil za taj element. Kod vanjskog načina CSS-a je zasebna .css datoteka i vidljiva je na ispisu 1.

```
body {  
    background-color: darkslategrey;  
    color: azure;  
    font-size: 1.1em;  
}  
h1 {  
    color: coral;  
}
```

Ispis 1: Primjer vanjske css datoteke

Na ispisu 2 prikazani su elementarni CSS selektori koji se konkretno odnose na izgled naslova i tijela HTML dokumenta. Dodavanjem sintakse `<link rel="stylesheet" type="text/css" href="style.css"/>` u `<head>` odjeljak HTML dokumenta, link element HTML dokumentu daje do znanja gdje se nalazi CSS datoteka pomoću `href` atributa.

```
<!DOCTYPE html>  
<html>  
    <head>  
        <title>My Example</title>  
        <link rel="stylesheet" href="styles.css">  
    </head>  
    <body>  
        <h1>External Styles</h1>  
        <p> Allows defining styles for the website.</p>  
    </body>  
</html>
```

Ispis 2: Uključivanje vanjskog CSS-a u HTML datoteku

2.3 JavaScript

JavaScript ili skraćeno JS je dinamički programski jezik koji se koristi za web razvoj, u web aplikacijama, za razvoj igara i još mnogo toga. JavaScript je prvenstveno jezik na strani klijenta, što znači da se izvodi na računalu unutar preglednika. No, nedavno uvođenje Node.js je omogućilo JavaScriptu da izvršava kôd i na poslužiteljima. Dok su HTML i CSS jezici koji web stranicama daju strukturu i stil, JavaScript web stranicama daje interaktivne elemente koji privlače korisnike.

Krajem 1995., programer Brandan Eich u samo 10 dana razvio je novi skriptni jezik. Originalno ime mu je bilo Mocha, ali se ubrzo populariziralo kao LiveScript, a kasnije i JavaScript kako ga i dan danas svi znaju [4]. S obzirom da je skriptni jezik, JavaScript se ne može samostalno izvoditi već je preglednik odgovoran za pokretanje JS koda. Kada korisnik zatraži HTML stranicu s JavaScriptom, skripta se šalje pregledniku koji ju potom izvršava. Velika prednost JavaScripta je što je podržan od strane svih današnjih web preglednika. Također, JS radi na bilo kojem operacijskom sustavu uključujući Windows, Linux ili Mac.

2.4 Node.js

Node.js je platforma na strani poslužitelja izgrađena u JavaScriptu Google Chrome (V8 pokretač). U početku se mogao pokrenuti samo u pregledniku, dok je kasnije omogućeno da se može pokrenuti kao program na osobnom računalu. Razvio ga je Ryan Dahl 2009. godine.

Node.js platforma je izgrađena na Chromeovom JavaScript vremenu izvođenja (engl. *runtime*) radi jednostavne izrade brzih i skalabilnih web aplikacija. Node.js je vođen događajima, asinkronim I/O (engl. *Input-Output*) modelom koji ga čini savršenim za aplikacije u stvarnom vremenu (engl. *real-time app*) koje intenzivno koriste podatke i koje se pokreću na distribuiranim uređajima. Node.js jezik je otvorenog kôda i koristi se na više platformi. Aplikacije Node.js-a napisane su u JavaScriptu i mogu se izvoditi unutar Node.js vremena izvođenja na macOS-u, Microsoft Windowsu i Linuxu. Node.js omogućuje programerima da koriste JavaScript i na poslužitelju i na klijentu, čime se olakšava dostavljanje podataka. Node.js također nudi opširnu biblioteku različitih JavaScript modula što u velikoj mjeri pojednostavljuje razvoj web aplikacija koje koriste Node.js.

2.5 Express.js

Express.js je najpopularniji Node web okvir, a i temeljna biblioteka za niz drugih popularnih Node web okvira kao što su: *Feathers*, *ItemsApi*, *KeystoneJS* i slično [5]. Specijalno je dizajniran za izradu aplikacija s jednom stranicom, više stranica i hibridnih web aplikacija. Prvotno je objavljen krajem 2010. godine, a zadnja izdana verzija je 4.17.1.

Kod klasičnih web stranica koje se temelje na podacima, web aplikacija čeka HTTP zahtjeve (iz web preglednika ili nekog drugog klijenta). Kod primitka zahtjeva, aplikacija procijenjuje koja je akcija potrebna i tada može čitati ili pisati informacije iz baze podataka ili obavljati druge zadatke potrebne za izvršavanje zahtjeva. Aplikacija potom vraća odgovor web pregledniku, pri čemu najčešće kreira dinamičku HTML stranicu koju web preglednik prikazuje umetanjem dohvaćenih podataka u rezervirana mjesta u HTML predlošku.

Express.js pruža metode za određivanje funkcija koje se pozivaju za određeni HTTP zahtjev (GET, POST, SET, DELETE). Instaliranje i korištenje Expressa je vrlo jednostavno. Express se instalira putem Node paket upravitelja. To se postiže izvršenjem sljedećeg retka u terminalu: 'npm install express'. Spomenuta naredba zahtijeva od upravitelja paketa Node.js preuzimanje tražene *express* module i u skladu s tim ih instalira.

2.6 Bootstrap

Bootstrap je izvorno stvoren od strane dizajnera i programera na Twitteru i brzo je postao najpopularniji HTML, CSS i JavaScript okvir za razvoj responzivnih web stranica usmjerenih na mobilne uređaje. Izvorno je objavljen u petak, 19. kolovoza 2011. godine, a do danas, objavljeno je više od dvadeset izdanja.

Bootstrap je temelj i kostur brojnih modernih i responzivnih web stranica. Služi za izradu korisničkog sučelja (engl. *front-end*) web stranica, te je kompatibilan sa svim modernim preglednicima (Chrome, Firefox, Internet Explorer, Safari, i Opera). Najveća prednost Bootstrapa je ta što u sebi ima ugrađen set alata i biblioteka za kreiranje fleksibilnih i responzivnih web formi sa svim pripadajućim elementima.

2.7 MongoDB

MongoDB je baza podataka otvorenog kôda, stvorena od strane Dwighta Merrimana i Eliota Horwitza. Dizajnirana je za pohranu velike količine podataka i uz to omogućuje vrlo učinkovit rad s tim podacima.

MongoDB pripada u kategoriju NoSQL (engl. *Not only Structured Query Language*) baza podataka. Takve baze podataka podatke ne pohranjuju tabličnim relacijama već rade s dokumentima umjesto sa definiranim tablicama. NoSQL baze podataka dolaze u različitim vrstama na temelju njihovog modela podataka. Glavne vrste su dokument, ključ/vrijednost, široki stupac i graf [6]. MongoDB koristi model podataka temljen na dokumentu jer je prirodniji način za opisivanje podataka. Dokumenti predstavljaju odvojenu strukturu s pripadajućim podacima ugrađenim kao pod dokumenti i nizovi. Tako je dokumente moguće usporediti s objektima u objektno-orijentiranom programiranju, što uvelike olakšava rad programerima.

MongoDB baza podataka sadrži zbirke kao što MySQL baza podataka sadrži tablice. Omogućeno je stvaranje više baza podataka i više zbirki. Unutar zbirke se nalaze dokumenti koji sadrže podatke za pohranu u bazu podataka. Dokumenti se kreiraju pomoću polja, a polja su parovi ključ/vrijednost. Podaci pohranjeni u MongoDB su u formatu BSON (binarni JSON (engl. *Javascript Object Notation*) prikaz) dokumenata.

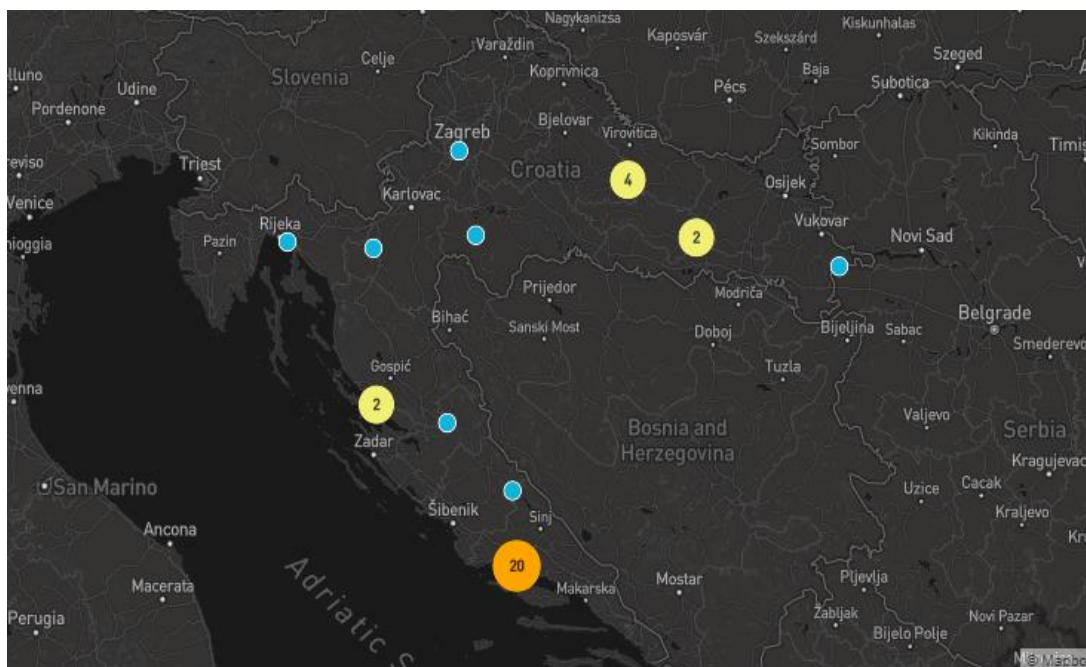
3. OPIS PRAKTIČNOG RADA

3.1 Klijentski dio

Svrha ovoga rada je izrada praktične web aplikacije za pronalaženje restorana. Praktični dio ovoga rada sastoji se od klijentske i poslužiteljske strane web aplikacije koje će u nastavku rada biti detaljnije razrađene.

3.1.1 Pretraga restorana

Kao što je već ranije spomenuto, pregled svih restorana izveden je korištenjem karte. Za prikaz karte korišten je Mapbox, američki distributer mrežnih karata za web stranice i aplikacije [7], gdje se pojedinačni objekt na karti prikazuje plavim krugom, žutom je prikazana grupacija do dvadeset objekata, a narančasti krug označava grupu do trideset objekata. Grupa koja ima više od trideset objekata prikazuje se crvenim krugom. Klikom na krug, karta se povećava te je tako omogućena detaljnija lokacija određenih objekata. Klikom na sami objekt, prikazuje se značka koja sadrži ime restorana, a koja je također i link koji vodi na profil tog restorana. Izgled karte prikazan je na slici 1, a ispis 3 prikazuje na koji način karta dobije podatke o restoranima.



Slika 1: Prikaz cluster karte

Prilikom učitavanja dohvaćaju se podaci o svim restoranima iz baze podataka. Nadalje se njihove GPS koordinate mapiraju i prikazuju na karti.

```
map.on('load', function () {  
    map.addSource('restaurants', {  
        type: 'geojson',  
        data: all,  
        cluster: true,  
        clusterMaxZoom: 14,  
        clusterRadius: 50  
    });  
});
```

Ispis 3: Prikaz učitavanja restorana na kartu

Osim karte korisnicima je na naslovnoj stranici omogućeno filtriranje restorana po kategoriji, njihovom imenu i prosječnoj ocjeni. Pretraga po kategoriji je zapravo pretraga po vrsti jela koja se poslužuju u restoranu, a filter je odrađen padajućim izbornikom koji sadrži sve moguće opcije. Pretraga imena restorana izvedena je uz pomoć trake za pretraživanje koja nije osjetljiva na velika i mala slova te funkcionira za bilo koji upit koji se dijelom ili u cijelosti nalazi u imenu restorana. Posljednji filter koji korisnik može koristiti je sortiranje restorana po njihovoj srednjoj ocjeni. Na ispisu 4 prikazan je kôd u kojem je izvedeno filtriranje. Radi na način da se uzimaju vrijednosti iz prethodno spomenuta tri polja po kojima korisnik može filtrirati restorane i spremaju se u varijable. Varijabla `nameCondition` predstavlja filter za ime restorana, `restaurantRating` za prosječnu ocjenu restorana a `categoryCondition` je varijabla za kategoriju restorana. Zatim se provjerava koji od tih filtera ima vrijednost (jer u jednom trenutku samo jedan može imati vrijednost) i potom se njegova vrijednost sprema u varijablu `filter`. Zatim se šalje upit u bazu podataka da se vrate svi rezultati koji zadovoljavaju filter.

```

module.exports.index = async (req, res) => {
    const {category, name, rating, page} = req.query;
    var restaurantsAll= await Restaurant.find({});
    var categoryCondition = category ? { category } :
    {};

    var restaurantRating = rating ? { averageRating:
    {$gte: rating} } : {};

    var nameCondition = name ? {title: { $regex: new
    RegExp(name), $options: "i" }} : {};

    var filter = !_.isEmpty(categoryCondition) ?
    categoryCondition : !_.isEmpty ( nameCondition) ?
    nameCondition : !_.isEmpty( restaurantRating) ?
    restaurantRating : {};

    if(!req.query.page){
        const restaurants = await
    Restaurant.paginate(filter, {
        populate: {
            path: 'popupText'
        }
    });

        const categoryList = await Category.find({});

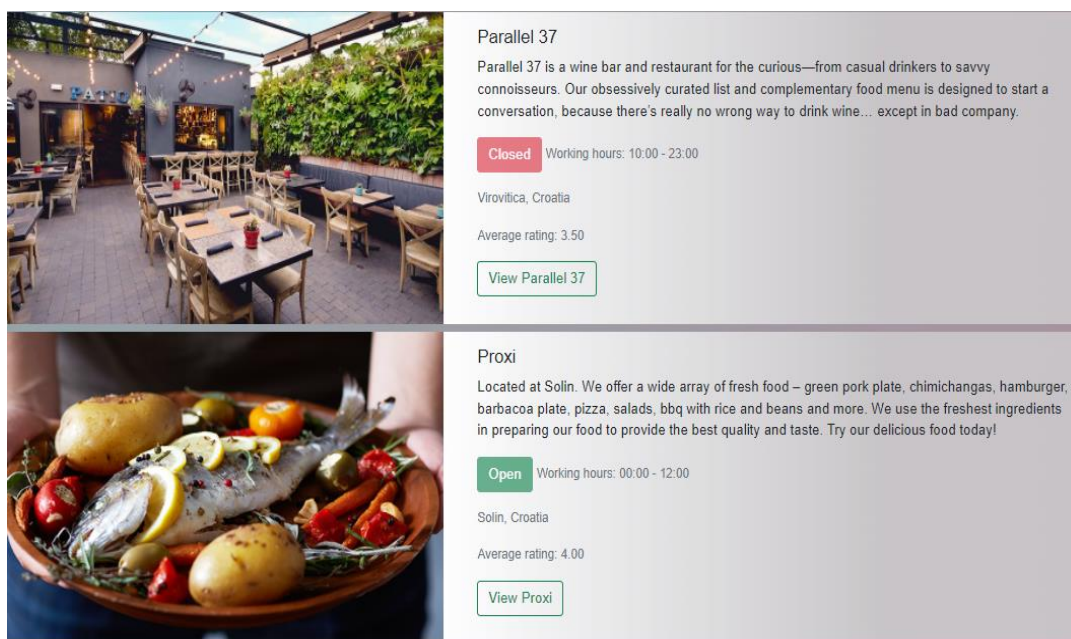
        res.render('restaurants/index',
    {restaurants,categoryList,restaurantsAll});
    } else {
        const {page}= req.query;
        const restaurants = await
    Restaurant.paginate(filter, {
            page,
            populate: {
                path: 'popupText'
            }
        });

        res.status(200).json(restaurants);
    }
}

```

Ispis 4: Filteri na naslovnoj stranici

Ispod filtera, izlistani su svi restorani koje aplikacija sadrži. Prikazano je ime i kratak opis restorana, radno vrijeme sa oznakom “*Open/Closed*“, lokacija i prosječna ocjena. Prilikom pregleda restorana uspoređuje se radno vrijeme objekta sa trenutnim vremenom pa ovisno o njihovoj podudarnosti korisnik vidi “*Open*“ ili “*Closed*“. Izgled kartica dva restorana na naslovnoj stranici prikazan je na slici 2.




Slika 2: Prikaz dva restorana na naslovnoj stranici

3.1.2. Gosti aplikacije

Neprijavljeni, odnosno, neregistrirani korisnici aplikacije mogu koristiti prethodno spomenute filtere, a dostupne su im i detaljnije informacije o svakom restoranu koje se dobiju klikom na gumb “Pregledaj“ (engl. *View*), a osim toga mogu pristupiti profilima onih korisnika koji su dodali barem jedan restoran.

Prijava, odnosno, registracija korisnika omogućena je navigacijskom trakom koja sadrži botune “*Log in*“ i “*Register*“. Navigacijska traka mijenja svoj izgled ovisno o trenutnom korisniku aplikacije te kod neprijavljenih korisnika sadrži ranije spomenute botune. Ukoliko gost aplikacije nema napravljen račun za istu, registraciju obavlja klikom na gumb “*Register*“ koji ga usmjerava na obrazac prikazan na slici 3.



Create an account.

Enter your username

Enter your e-mail

Enter your first name

Enter your last name

Enter your password

Choose File No file chosen


Sign Up

You already have an account? [Login here.](#)

Slika 3: Obrazac za registraciju

Forma za registraciju sadrži sljedeća polja: korisničko ime, e-mail, ime, prezime, lozinku i polje za unos slike profila. Potrebno je ispuniti sva polja da bi registracija bila uspješna.

Neprijavljeni korisnici, koji imaju prethodno kreiran račun, prijavu mogu odraditi preko botuna “*Log in*” koji ih preusmjerava na obrazac prikazan na slici 4.



Welcome back.

Username

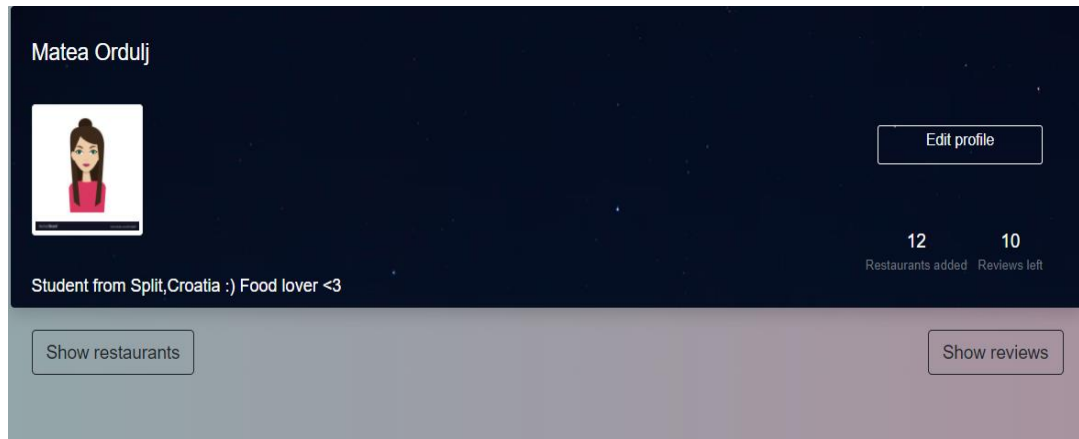
Password

Log in

Slika 4: Obrazac za prijavu

3.1.3. Prijavljeni korisnici

Uz sva prava koja imaju gosti aplikacije, prijavljeni korisnici imaju mogućnost dodavanja novog restorana, ostavljanja recenzija na restorane te uvid na svoj profil. Korisnik na svom profilu ima prikazanu sliku profila, ime, prezime, kratak opis o sebi te ukupan broj trenutno dodanih restorana i recenzija, kao što je prikazano na slici 5.



Slika 5: Izgled korisničkog profila

Slika 6 prikazuje strukturu jednog objekta (u ovom slučaju restorana) u bazi. Korišten je MongoDB Compass, grafičko korisničko sučelje (engl. *graphical user interface*) za upite, agregaciju i analizu MongoDB podataka u vizualnom okruženju. Vidljivo je da sami restorani kao objekt sadrži referencu na korisnika koji ga je kreirao, odnosno njegov identifikator, a tako isto sadrži i identifikator svake recenzije.

```
_id: ObjectId("61ca125dfc90253da09dcebb")
> geometry: Object
✓ reviews: Array
  0: ObjectId("61ca12ed94de8f05748326b1")
  1: ObjectId("61ca12f394de8f05748326be")
  2: ObjectId("61d735e0cce4a439e069d0db")
  3: ObjectId("61d73700cf749c11b056bb2f")
> category: Array
averageRating: 4.5
author: ObjectId("60f4388e90987d34f8aceef4")
location: "Rovinj, Croatia"
title: "Lazy Bear"
description: "Lorem ipsum dolor sit amet consectetur adipisicing elit. Cum totam duc..."
price: "4"
> images: Array
__v: 7
```

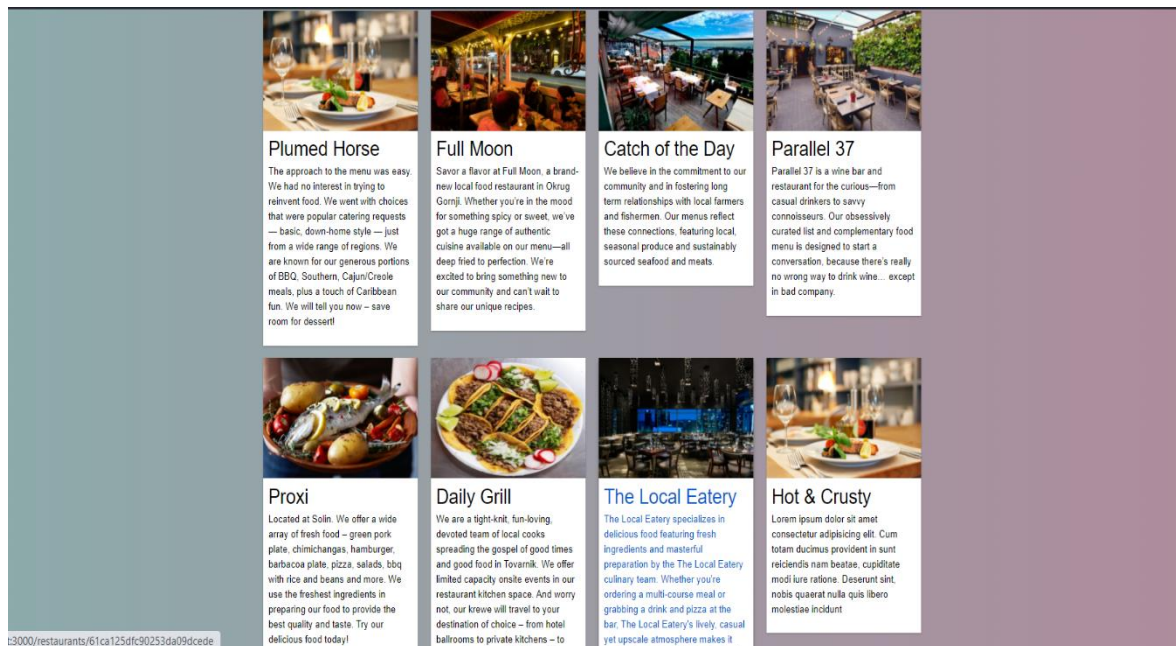
Slika 6: Prikaz objekta u bazi

Ispis 5 sadrži kôd koji prikazuje način na koji je dohvaćen podatak o ukupnom broju restorana i recenzija na korisničkom profilu. Iz parametara GET zahtjeva dohvaćen je korisnik pa se preko njegovog identifikatora dohvaćaju svi restorani koji kao identifikator autora sadrže korisnikov identifikator. Isti proces se ponavlja za pronalaženje recenzija korisnika.

```
module.exports.ShowProfile= async(req, res) => {
  User.findById(req.params.id, function(err, foundUser){
    if(err) {
      req.flash("error","Something went wrong.");
      return res.redirect („/“);
    }
    Restaurant.find().where('author').equals(foundUser._id).
    exec(function(err, restaurants) {
      if(err) {
        req.flash("error","Something went wrong.");
        return res.redirect („/“);
      }
      Review.find().where('author').equals(foundUser._id).popu
      late('restaurant').exec(function(err, reviews) {
        if(err) {
          req.flash("error","Something went wrong.");
          return res.redirect („/“);
        }
        res.render("users/show", { user: foundUser,
        restaurants: restaurants, restaurantCount:
        restaurants.length, reviews: reviews, reviewsCount:
        reviews.length});
      });
    });
  });
}
```

Ispis 5: Zbrajanje dodanih restorana i recenzija pojedinačnog korisnika

Na profilu korisnika postoje dva botuna “*Show restaurants*“ i “*Show reviews*“ koji na klik otvaraju ili restorane dodane od tog korisnika ili recenzije koje je taj korisnik ostavio. Lista dodanih restorana od jednog korisnika prikazana je na slici 7.



Slika 7: Prikaz otvorene liste dodanih restorana

Korisnici mogu uređivati informacije o sebi klikom na botun “*Edir*“. Polja koja su dostupna za ažuriranje su: ime, prezime, slika profila i opis. Osim toga postoji i botun “*Delete*“, koji briše njihov profil. Ukoliko korisnik odluči izbrisati svoj profil, svi restorani koje je on prethodno dodao se ne brišu već se njihov autor prebacuje sa tog korisnika na prvog nađenog administratora (Ispis 6).

```
const admin=await User.findOne({isAdmin:'true'});

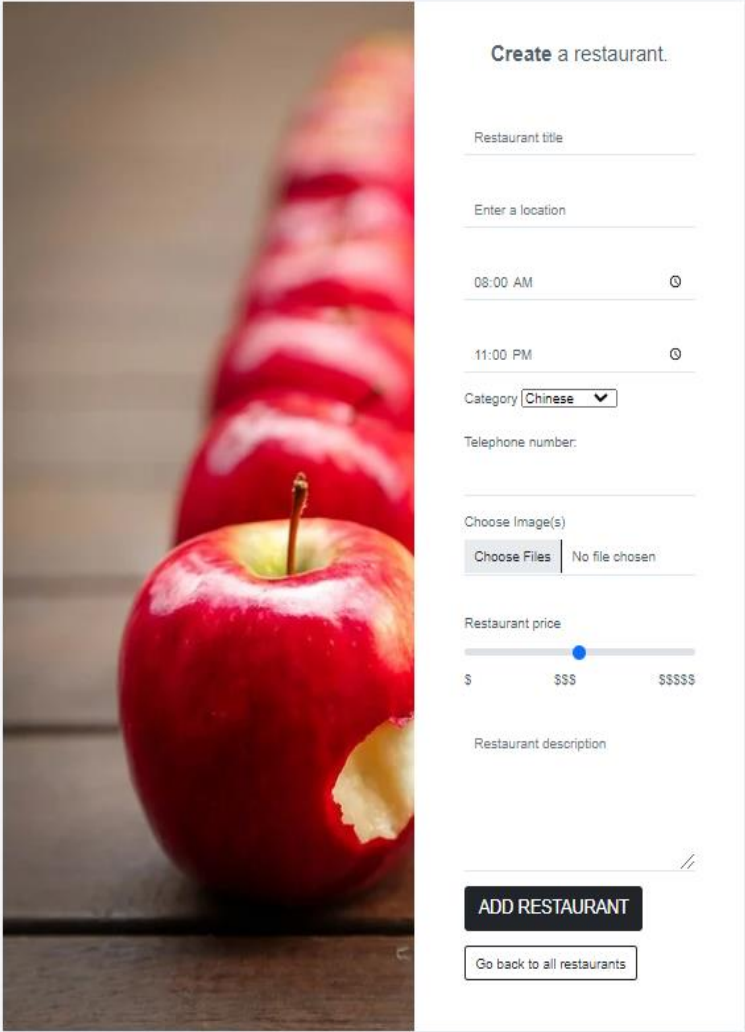
let restaurants= await
Restaurant.find().where('author').equals(user.id);

restaurants.forEach ( async restaurant => {
  restaurant.author= admin;
  await restaurant.save();
});
```

Ispis 6: Prebacivanje autora restorana na administratora

3.1.4 Dodavanje novog restorana

Prijavljeni korisnici u navigacijskoj traci imaju vidljiv botun “*New restaurant*“ koji ih usmjerava na formu za dodavanje restorana, a koja je prikazana na slici 8. Obrazac za kreiranje restorana sadrži sljedeća polja: naziv restorana, lokaciju, početak radnog vremena, kraj radnog vremena, vrstu restorana, telefonski broj, polje za dodavanje slika, klizač (engl. *slider*) za cjenovni rang restorana i na kraju polje za kratki opis restorana. Unos lokacije je izveden preko Googleovog API-ja (engl. *application programming interface*) Places te je samodovršavajuća (engl. *autocomplete*) i nudi korisniku par prijedloga lokacija ovisno o početnom upisanom slovu. Polje za slike omogućava dodavanje jedne ili više slika te ukoliko korisnik sam ne doda sliku pri izradi restorana, aplikacija postavi prethodno zadanu sliku.



Slika 8: Obrazac za kreiranje restorana

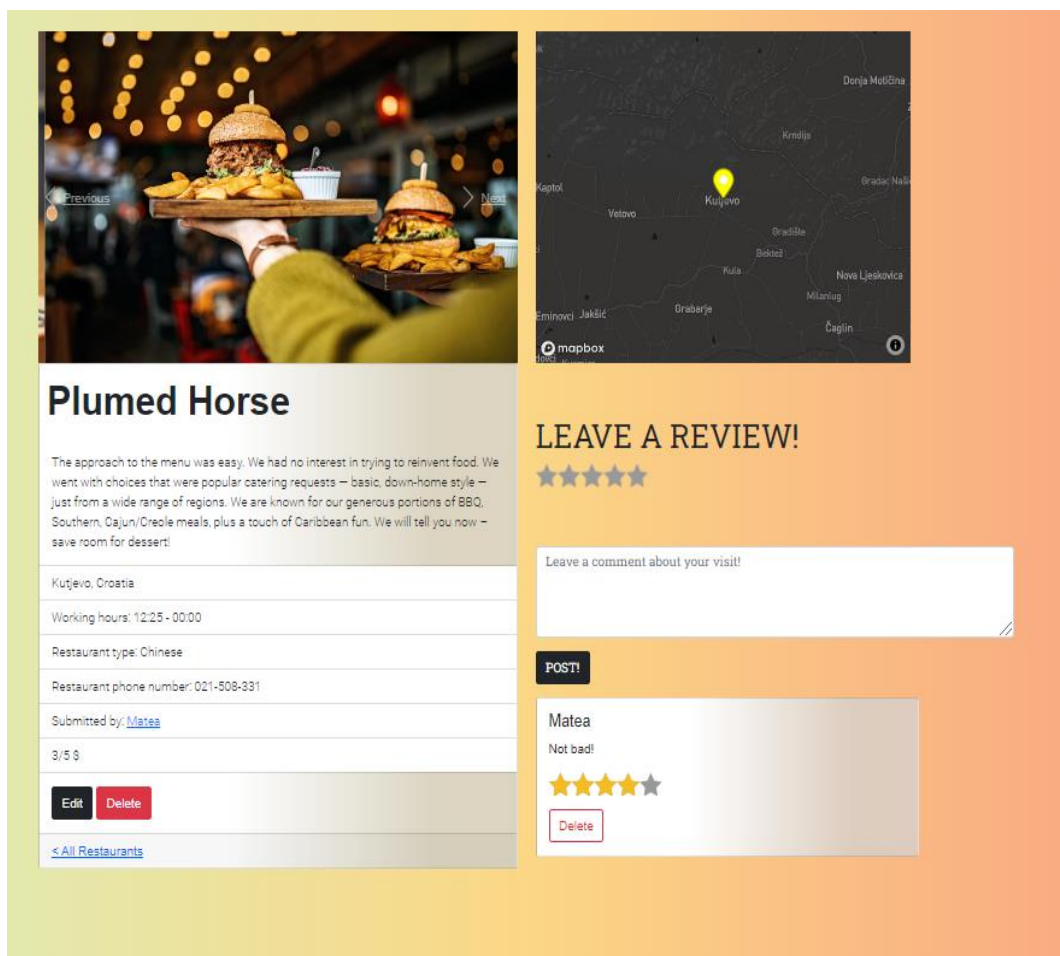
Ispis 7 prikazuje dio kôda gdje se provjerava niz slika koje je korisnik dodao pa u slučaju da je taj niz prazan, u njega se postavlja zadana slika.

```
restaurant.images=req.files.map(f => ({url: f.path,
filename: f.filename}));

    if (restaurant.images.length===0) {
        var nophoto = [
            {
                fieldname: 'image',
                originalname: 'no-foto.jpg',
                encoding: '7bit',
                mimetype: 'image/jpeg',
                path:
                'https://res.cloudinary.com/dx4xgystc/image/upload/v1641505
212/FineEats/depositphotos_227725020-stock-illustration-
image-available-icon-flat-vector_hkz8rc.webp',
                size: 103978,
                filename: 'FineEats/depositphotos_227725020-
stock-illustration-image-available-icon-flat-
vector_hkz8rc.webp'
            }
        ];
        restaurant.images=nophoto.map(f => ({url: f.path,
filename: f.filename}));
    }
```

Ispis 7: Zadano dodavanje slike

Polja koja korisnik mora ispuniti kod kreiranja restorana su: ime, lokacija, početak i kraj radnog vremena te kategorija, a sva ostala polja su izborna i kasnije se mogu ažurirati. Nakon uspješno dodanog restorana korisnik se preusmjerava na stranicu koja sadrži sve informacije o restoranu koje uključuju: jednu ili više slika restorana, njegovo ime, lokaciju, radno vrijeme, kategoriju, broj telefona restorana i informaciju tko je autor restorana, kartu koja je centrirana na točnoj lokaciji restorana te kod restorana koji su ih dobili, recenzije za taj restoran. Primjer stranice jednog restorana prikazan je na slici 9.

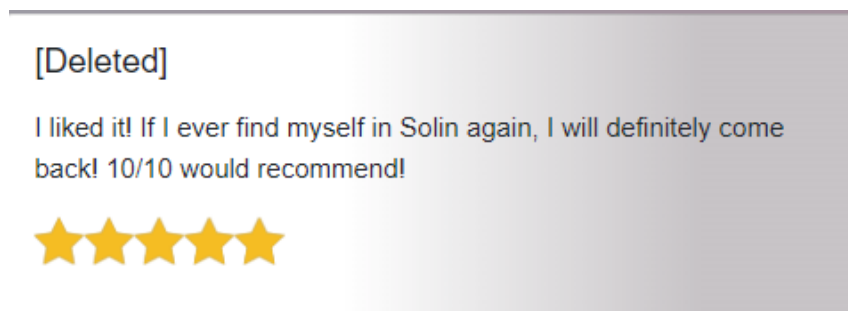


Slika 9: Show-page restorana

Vlasniku restorana i administratorima dostupna su dva botuna, jedan za ažuriranje restorana, a drugi za brisanje. Forma za uređivanje je vrlo slična formi za dodavanje restorana, a jedina razlika je nemogućnost mijenjanja lokacije objekta. Korisniku je uz dodavanje novih, omogućen pregled i brisanje dosad dodanih slika. Botun “Delete” radi upravo ono što mu to samo ime govori, briše taj restoran.

3.1.5. Dodavanje recenzije

Korisnici imaju mogućnost ostavljanja recenzije na restoran. Ona se sastoji od imena korisnika, ocjene i komentara. Ocjene su moguće od jedan do pet, a prikazane su kao zvjezdice. Nije moguće ostaviti samo ocjenu bez komentara ili obratno, odnosno obje stavke moraju biti ispunjene da bi se recenzija objavila. Jednom dodana recenzija se ne može uređivati, a brisati je može samo vlasnik ili administrator. Ukoliko netko izbriše svoj korisnički račun, njegove objavljene recenzije će i dalje postojati, no ime korisnika neće se prikazati. Recenziju dodanu s obrisanoj profila prikazuje slika 10.



Slika 10: Prikaz recenzije izbrisanog korisnika

3.1.6. Administratorski dio

Administratorski dio ove aplikacije je izrađen tako da postoji više administratora koji zajedno upravljaju. Administratori, uz sve korisničke mogućnosti imaju i sljedeće ovlasti:

- postavljanje ili ukidanje administracije drugom korisniku
- mogućnost brisanja svih korisnika, restorana i recenzija
- mogućnost ažuriranja profila svakog korisnika i restorana

Administratori bi trebali brisati recenziju u slučaju da je ona uvredljivog sadržaja, ili je besmislena, odnosno, ne govori o iskustvu u restoranu. Administrator bi trebao brisati restorane koji su se trajno zatvorili, sadrže lažne informacije (npr. slike jela preuzete sa interneta) ili koji uopće ne postoje. Korisnike koji postavljaju uvredljive recenzije i objavljuju nepostojeće restorane bi administratori također trebali izbrisati.

Administratori u navigaciji također imaju link “*Users*” odnosno administratorsko sučelje (slika 11). U tablici su prikazani svi korisnici te je preko toga administratorima omogućeno lakše upravljanje korisnicima budući da na jednom mjestu mogu brisati korisnike, pregledavati profil korisnika te dodjeljivati (kao i oduzimati) korisnicima rolu administratora.

Username	E-Mail	Is admin?		
Matea	matea@gmail.com	<input checked="" type="checkbox"/>	View profile	DELETE USER
bob	bob@gmail.com	<input checked="" type="checkbox"/>	View profile	DELETE USER
Stipe	stipe@gmail.com	<input type="checkbox"/>	View profile	DELETE USER
Anita	anita@gmail.com	<input type="checkbox"/>	View profile	DELETE USER
Damjan	damjan@yahoo.com	<input type="checkbox"/>	View profile	DELETE USER
Ana	anam01@gmail.com	<input type="checkbox"/>	View profile	DELETE USER

Slika 11: Administratorska tablica

Administratorsko sučelje također sadržava i tražilicu koja omogućava administratorima bržu i jednostavniju pretragu korisnika. Administratorima je omogućeno pretraživanje korisnika po e-mail adresi koju su upisali prilikom registracije.

3.2. Dizajn

U posljednje vrijeme, sve je veća posjećenost web stranica preko prijenosnih uređaja različitih veličina. Ukoliko web stranica nije prilagođena za sve vrste zaslona, korisnici se neće dugo zadržati. Responzivnost je danas obavezna stavka kod kreiranja web aplikacije, a responzivni web dizajn je dizajn koji se automatski prilagođava veličini zaslona uređaja što omogućava korisnicima da neometano pregledavaju sadržaj stranice što uvelike poboljšava korisničko iskustvo (engl. *User Experience*).

Zbog boljeg korisničkog iskustva i ova web aplikacija je izrađena responzivno, koristeći CSS i Bootstrap. CSS pravilo „@media“ koristi se u medijskim upitima (engl. *media query*) jer omogućava prilagođavanje sadržaja web stranice različitim uvjetima, poput razlučivosti zaslona. Na primjeru prikazanom u Ispisu 8 vidljivo je da se za zaslone manje od 800px primjenjuje navedeni raspored i veličina elemenata.

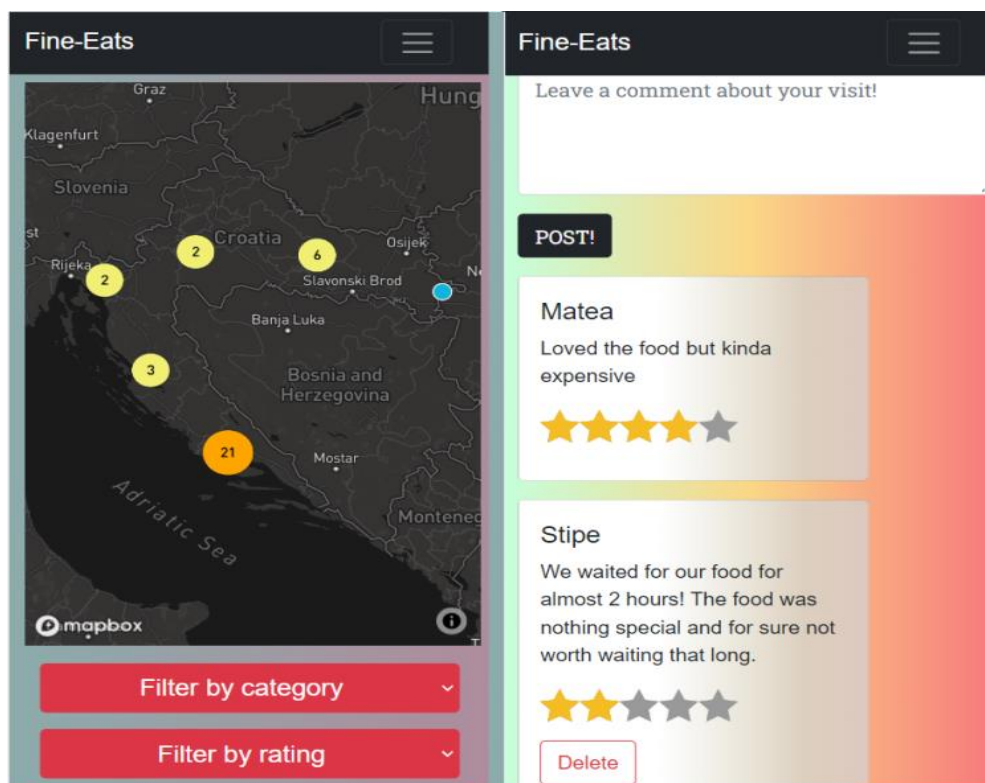

```

@media screen and (max-width: 800px) {
    .parent{
        display:grid;
    }
    .karta,.carousel,.details,.rating,.comments{
        width:95vw;
    }
}

```

Ispis 8: Prikaz kôda za responzivnost elemenata

Slika 12 prikazuje izgled navigacije, karte i filtera za restorane te objavljene recenzije na zaslonu manje veličine.



Slika 12: Prikaz aplikacije na manjem zaslonu

3.3 Poslužiteljski dio aplikacije

3.3.1. Stvaranje poslužitelja

Stvaranje poslužitelja u Node.js-u je dosta olakšano korištenjem paketa Express. Primjer Express poslužitelja prikazan je na Ispisu 9.

```
const express = require ('express');
const path = require ('path');
const mongoose = require('mongoose');
const app = express();

app.engine('ejs', ejsMate);
app.set('view engine', 'ejs');
app.set('views',path.join(__dirname, 'views'))
app.listen(3000, () => {
    console.log ('Serving on port 3000')
});
```

Ispis 9: Pokretanje servera

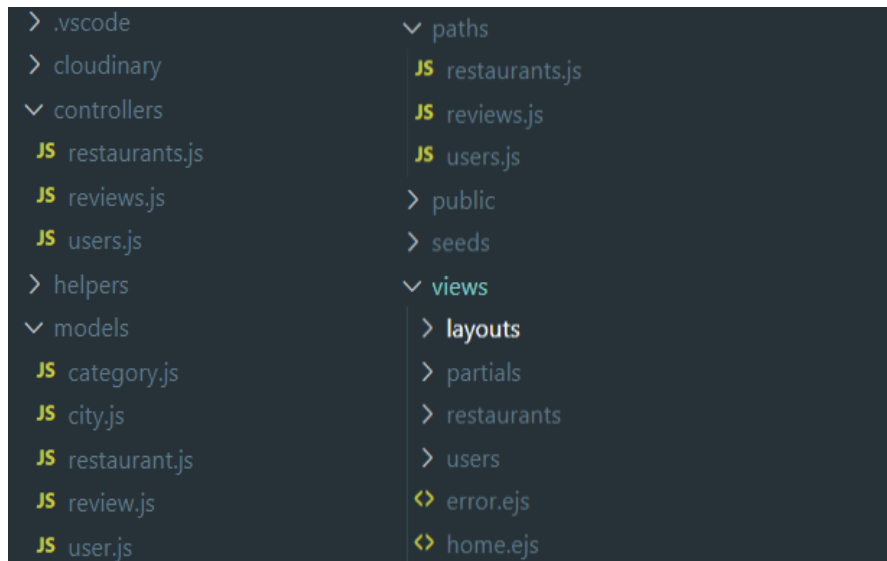
Pozivom funkcije `express()` stvara se Express aplikacija koja se dodjeljuje varijabli `app` koja se u ovom slučaju koristi za osluškivanje porta 3000.

3.3.2. Model-View-Controller

Model-View-Controller ili skraćeno MVC je način izrade aplikacije koji se temelji na podijeli aplikacije na tri glavne komponente: model, pogled i kontroler. Programeri koji rade po ovoj metodi, svoj kôd organiziraju u direktorije ovisno o dijelu aplikacije kojoj pripadaju.

Kao što mu samo ime sugerira, model predstavlja strukturu podataka i kontroler pristupa bazi podataka preko njega. Prikaz se sastoji od kôda koji sadrži sve funkcije s kojima korisnik ima izravnu interakciju. Jednostavnije rečeno, to su sve datoteke HTML predložaka. Kontroler funkcionira kao veza između modela i prikaza, koji stupa u komunikaciju s modelom i služi za odgovor i funkcionalnost na prikazu. Kada korisnik uputi zahtjev, on se šalje kontroleru koji zatim stupa u interakciju s bazom podataka. Kod izrade ovog projekta,

kontolori su razdvojeni u dva dijela: rute koje preusmjeravaju zahtjev kontrolerima i sami kontroleri koji upravljaju podacima. Slika 13 pokazuje spomenute direktorije u uređivaču.



Slika 13: Popis MVC direktorija

Ispis 10 prikazuje model za recenziju. Ona, kao objekt, se sastoji od tekstualnog dijela odnosno tijela, ocjene, identifikatora autora i restorana.

```
const reviewSchema = new Schema({
  body:String,
  rating: Number,
  author: {
    type: Schema.Types.ObjectId,
    ref: 'User'
  },
  restaurant: {
    type: Schema.Types.ObjectId,
    ref: 'Restaurant'
  })
module.exports = mongoose.model("Review", reviewSchema);
```

Ispis 10: Model za recenziju

Ispis 11 prikazuje kôd kontrolera za dodavanje recenzije i računanje prosječne ocjene restorana. Prvo je potrebno pronaći restoran preko svog identifikatora u parametrima zahtjeva, nakon čega se kreira recenzija sa podacima iz forme (ocjena i komentar). Za potrebe srednje ocjene restorana, prvo se pronalaze sve recenzije koje taj restoran ima, nakon toga se zajedno sa novom ocjenom, računa prosjek. Potom se restoran ažurira, a onda se i ažurirani podaci restorana spremaju u bazu podataka, kao i nova recenzija. Na kraju se korisnik preusmjerava na restoran gdje je ostavio recenziju.

```
module.exports.addReview=async(req, res) => {  
    const restaurant = await  
    Restaurant.findById(req.params.id);  
  
    const review = new Review(req.body.review);  
  
    const reviews= await  
    Review.find().where('restaurant').equals(restaurant._id);  
  
    let average = 0;  
    reviews.forEach ( async rev => {  
        average += rev.rating;  
    });  
  
    average += review.rating;  
    average = average / (reviews.length + 1);  
    review.restaurant=restaurant;  
    review.author=req.user._id;  
    restaurant.averageRating = average;  
    restaurant.reviews.push(review);  
  
    await review.save();  
    await restaurant.save();  
  
    req.flash('success', 'You successfully added a new  
review!')  
    res.redirect(`/restaurants/${restaurant._id}`);  
}
```

Ispis 11: Kontroler za dodavanje recenzije

3.3.3. Učitavanje slika

S obzirom da nije dobra praksa učitavanje slika u bazu podataka, u ovom projektu slike se spremaju na Cloudinary, SaaS (engl. *Software as a Service*) tvrtku koja pruža usluge upravljanja slika i videozapisa koje su temeljene na oblaku. Za samo učitavanje slika korišten je Multer, posrednički (engl. *middleware*) softver za obradu podataka. Kod slanja datoteka preko poslužitelja, zahtjev za `req.body` je prazan, a upotrebom `multer` stvara se objekt `req.file` koji omogućuje učitavanje datoteke. Učitane slike spremaju se na Cloudinary. Kako bi se uspostavila veza s oblakom, potrebno je konfigurirati cloudinary parametre (direktorij u koji će se slike učitati, dozvoljeni format slika i sl.). Na ispisu 12 prikazano je kako to izgleda u kôdu.

```
const cloudinary = require('cloudinary').v2;
const {CloudinaryStorage}= require ('multer-storage-cloudinary');
cloudinary.config({
  cloud_name:process.env.CLOUDINARY_CLOUD_NAME,
  api_key: process.env.CLOUDINARY_KEY,
  api_secret: process.env.CLOUDINARY_SECRET
});
const storage = new CloudinaryStorage({
  cloudinary,
  params: {
    folder: 'FineEats',
    allowedFormats: ['jpeg', 'png', 'jpg']
  }
});
module.exports = {
  cloudinary,
  storage,
}
```

Ispis 12: Konfiguracija cloudinERYja

Nakon učitavanja slike na oblak, Multer omogućava `req.files` koji, između ostalog, prikazuje “*path*” i “*filename*”. Ti podaci spremljeni su u bazi na sljedeći način: sve slike pojedinačnog restorana spremljene su u niz, a svaka slika predstavlja jedan objekt (slika 14) koji se sastoji od identifikatora, imena i URL-a odnosno putanje.

```
_id: ObjectId("61ca125dfc90253da09dced4")
> geometry: Object
> reviews: Array
> category: Array
averageRating: 3.3333333333333335
author: ObjectId("61e08fb69b439f3258675536")
location: "Solin, Croatia"
title: "Proxi"
description: "Located at Solin. We offer a wide array of fresh food – green pork pla..."
price: "3"
✓ images: Array
  ✓ 0: Object
    _id: ObjectId("61ddf7a36d51dc34fc0171e7")
    url: "https://res.cloudinary.com/dx4xgystc/image/upload/v1641936802/FineEats..."
    filename: "FineEats/ezxvyg8i0o7jptpfthi2"
  ✓ 1: Object
    _id: ObjectId("61ddf7a36d51dc34fc0171e8")
    url: "https://res.cloudinary.com/dx4xgystc/image/upload/v1641936803/FineEats..."
    filename: "FineEats/yi2j4riiabzt2bnezpo3"
__v: 5
```

Slika 14: Prikaz spremanja slika u bazi

4. ZAKLJUČAK

Osnovni cilj ovog završnog rada bio je izraditi aplikaciju koja bi korisnicima omogućila bržu i jednostavniju pretragu restorana, a vlasnicima restorana osigurati platformu gdje nesmetano mogu oglašavati svoje objekte. S obzirom da se ljudi svih generacija mogu pronaći u korištenju ove aplikacije, nužan faktor je bio kreirati aplikaciju koja je u isto vrijeme i jednostavna za korištenje i funkcionalna. Brz odabir JavaScripta kao programskog jezika ispostavio se dobrim te korištenje Node.JS-a omogućilo je korištenje JS-a i na *back-endu*.

Kako bi web aplikacija „Fine-Eats“ bila poboljšana, s vremenom se očekuje mogućnost dodavanja slike kod kreiranja recenzije budući da vizualni pogled na hranu dodatno privlači korisnika u taj restoran. Također, zbog sve češćeg korištenja pametnih uređaja u svakodnevnom životu, bilo bi dobro omogućiti online rezervaciju stolova, koja bi posebno bila dobro prihvaćena kod mlađe generacije. Također, informacija o trenutnoj posjećenosti restorana bi pridonijela ne samo populariziranju aplikacije već i samog restorana. Uz to, trebalo bi omogućiti komunikaciju administratora i korisnika u slučaju da korisnik krši pravila zajednice te ga je potrebno upozoriti.

LITERATURA

- [1] Berners-Lee T., The birth of the Web, CERN,
<https://home.cern/science/computing/birth-web> (10. 11. 2021.)
- [2] Berners-Lee T., HTML History, W3schools
<https://www.w3schools.in/html-tutorial/history/> (10. 11. 2021.)
- [3] Morris S., Tech 101: The Ultimate Guide To Css, Skillcrush
<https://skillcrush.com/blog/css/> (13.11.2021.)
- [4] DeGroat T.J., The History of JavaScript: Everything You Need to Know
<https://www.springboard.com/blog/data-science/history-of-javascript> (13.11.2021.)
- [5] Frameworks built on Express, Express
<https://expressjs.com/en/resources/frameworks.html> (13. 11. 2021.)
- [6] What is NoSQL : Types of NoSQL databases, MongoDB
<https://www.mongodb.com/nosql-explained> (13. 11. 2021.)
- [7] Our roots, Mapbox
<https://www.mapbox.com/about/company/#roots> (17. 12. 2021.)

Popis slika

Slika 1: Prikaz cluster karte	9
Slika 2: Prikaz dva restorana na naslovnoj stranici	12
Slika 3: Obrazac za registraciju	13
Slika 4: Obrazac za prijavu	13
Slika 5: Izgled korisničkog profila	14
Slika 6: Prikaz objekta u bazi	14
Slika 7: Prikaz otvorene liste dodanih restorana.....	16
Slika 8: Obrazac za kreiranje restorana	17
Slika 9: Show-page restorana	19
Slika 10: Prikaz recenzije izbrisanog korisnika	20
Slika 11: Administratorska tablica.....	21
Slika 12: Prikaz aplikacije na manjem zaslonu	22
Slika 13: Popis MVC direktorija	24
Slika 14: Prikaz spremanja slika u bazi	27

Popis ispisa

Ispis 1: Primjer vanjske css datoteke	5
Ispis 2: Uključivanje vanjskog CSS-a u HTML datoteku	5
Ispis 3: Prikaz učitavanja restorana na kartu	10
Ispis 4: Filteri na naslovnoj stranici	11
Ispis 5: Zbrajanje dodanih restorana i recenzija pojedinačnog korisnika	15
Ispis 6: Prebacivanje autora restorana na administratora	16
Ispis 7: Zadano dodavanje slike	18
Ispis 8: Prikaz kôda za responzivnost elemenata	22
Ispis 9: Pokretanje servera	23
Ispis 10: Model za recenziju	24
Ispis 11: Kontroler za dodavanje recenzije	25
Ispis 12: Konfiguracija cloudinaryja	26