Web aplikacija za rezerviranje termina u restoranu

Vidan, Deni

Sveučilište Jurja Dobrile u Puli

Fakultet informatike u Puli

Deni Vidan

Izrada web aplikacije za rezerviranje termina u restoranu

Završni rad

Pula, rujan, 2023.

Sveučilište Jurja Dobrile u Puli

Fakultet informatike u Puli

Deni Vidan

Izrada web aplikacije za rezerviranje termina u restoranu

Završni rad

JMBAG: 0303088294, redovni student

Studijski smjer: Informatika

Znanstveno područje: Društvene znanosti

Znanstveno polje: Informacijske i komunikacijske znanosti

Znanstvena grana: Informacijski sustavi i informatologija

Kolegij: Programiranje

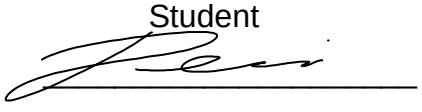
Mentor: izv. prof. dr. sc. Tihomir Orehovački

Pula, rujan, 2023.

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani Deni Vidan, kandidat za prvostupnika informatike ovime izjavljujem da je ovaj Završni rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Završnog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student



U Puli, 20.09.2023.

IZJAVA

o korištenju autorskog djela

Ja, Deni Vidan dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj završni rad pod nazivom “Izrada web aplikacije za rezerviranje termina u restoranu” koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.
Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

Potpis

U Puli, 20.09.2023.

Sažetak

Razvoj web aplikacija koje olakšavaju svakodnevne aktivnosti postaje sve popularniji. Jedna od uobičajenih aktivnosti je rezervacija stolova u restoranima. Tradicionalni načini rezervacije putem telefona ili e-pošte mogu biti sporiji i manje efikasni. Ovaj rad istražuje web aplikaciju "Cutlery" koja nudi inovativno rješenje za jednostavnu online rezervaciju stolova. Aplikacija koristi Vue.js, Express.js i SQLite tehnologije. Implementira sučelja za korisnike i vlasnike restorana radi lakšeg pregleda slobodnih termina, odabira željenih opcija i upravljanja rezervacijama. Korisnicima omogućuje informirani odabir restorana na temelju recenzija i fotografija. Vlasnicima restorana pruža alate za postavljanje rasporeda i vizualizaciju rezerviranih termina. Rad istražuje arhitekturu i implementaciju aplikacije te predlagane smjernice za daljnji razvoj. Cilj je pružiti uvid u korištenje modernih web tehnologija za unaprjeđenje procesa rezervacije stolova.

Ključne riječi: web aplikacija, rezervacija stolova, Vue.js, Express.js, SQLite

Abstract

The development of web applications that facilitate everyday activities is becoming increasingly popular. One common activity is making restaurant reservations. Traditional methods of booking by phone or email can be slower and less efficient. This paper explores the web application "Cutlery" which offers an innovative solution for easy online restaurant reservations. The application utilizes Vue.js, Express.js and SQLite technologies. It implements interfaces for users and restaurant owners for easier overview of available timeslots, selection of desired options and reservation management. It allows users to make informed restaurant choices based on reviews and photos. It provides restaurant owners with tools to schedule and visualize booked timeslots. The paper examines the architecture and implementation of the application and proposes guidelines for further development. The goal is to provide insight into leveraging modern web technologies to improve the restaurant reservation process.

Keywords: web application, restaurant reservation, Vue.js, Express.js, SQLite

Sadržaj

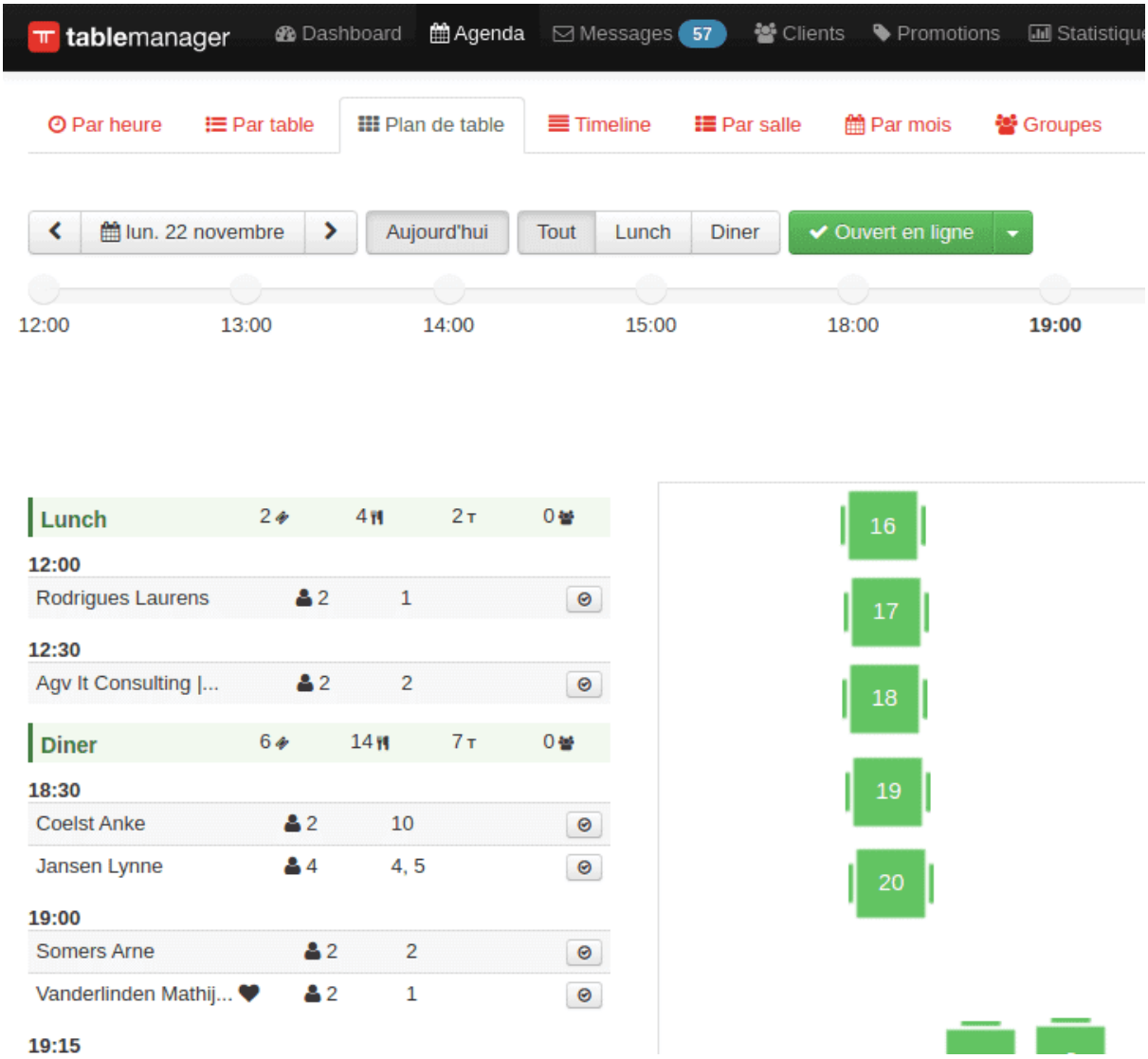
1. UVOD

U današnjem svijetu, rezervacija stolova u restoranima postala je neizbježna rutina za mnoge ljude. Međutim, tradicionalne metode rezervacija često mogu biti zamorne i neefikasne, uz dugotrajne telefonske pozive ili osobne posjete restoranima. Srećom, tehnologija je omogućila razvoj aplikacija koje olakšavaju proces rezervacije i pružaju korisnicima praktično rješenje za planiranje obroka. U ovom je radu predstavljena aplikaciju "Cutlery", inovativno rješenje koje rješava probleme tradicionalnih metoda rezervacija stolova. Cilj ove aplikacije je pružiti korisnicima jednostavan, brz i efikasan način rezervacije stolova u njihovim omiljenim restoranima. Prvenstveni je fokus na probleme s kojima se korisnici susreću pri rezervaciji stolova. Dugotrajno telefoniranje, često nedostupnost restorana i nedostatak informacija o restoranima samo su neki od problema koje "Cutlery" uspješno rješava."Cutlery" se ističe svojim jednostavnim i intuitivnim sučeljem koje omogućuje korisnicima brzo i jednostavno pregledavanje dostupnih restorana, odabir preferiranih termina i rezervaciju stolova. Aplikacija također pruža detaljne informacije o restoranima, uključujući ocjene korisnika, jelovnike i fotografije, čime korisnici mogu donijeti informiranu odluku pri rezervaciji. Osim toga, "Cutlery" pruža praktična rješenja i vlasnicima restorana. Omogućuje im upravljanje dostupnim stolovima, postavljanje termina za rezervacije i praćenje rasporeda rezervacija. Vlasnici restorana također mogu prihvatiti ili odbiti rezervacije te upravljati njima na učinkovit način. U ovom će se radu detaljnije prikazati funkcionalnosti i prednosti aplikacije "Cutlery" te njegov doprinos u poboljšanju iskustva rezervacije stolova u restoranima. Također izvršiti će se analizirati konkurentske aplikacije na tržištu zbog boljeg pozicioniranja "Cutlery"-ja i mogućnosti za daljnji razvoj. Uz to, rad istražuje i potencijalne izazove s kojima se "Cutlery" suočava te moguće prilike za daljnji rast i unaprjeđenje. Ovaj rad pruža uvid u važnost tehnoloških inovacija u optimizaciji rezervacija stolova te kako "Cutlery" pruža praktično rješenje za korisnike i vlasnike restorana. Cilj projekta je omogućiti odabir restorana na temelju recenzija, slika i komentara te na istom mjestu online putem mogućnost rezervacije stola sa željenim kapacitetom ljudi, te time administratorima restorana pojednostaviti te optimizirati uvid u raspored stolova i rezervacija.

2. KRITIČKA ANALIZA POSTOJEĆIH RIJEŠENJA

2.1 Table manager

Aplikacija "Cutlery" je jedinstvena platforma za rezervaciju stolova u restoranima, dok je "Table Manager" također popularna aplikacija koja pruža slične usluge. Ove dvije aplikacije imaju neke značajne razlike u pristupu i funkcionalnostima. "Cutlery" se ističe svojim jednostavnim sučeljem i intuitivnim korisničkim iskustvom. Ova aplikacija je posebno usmjerena na vlasnike restorana i korisnike koji žele brzo i jednostavno rezervirati stolove. Vlasnici restorana mogu dodati informacije o svom restoranu, uključujući opis, dostupne stolove i kapacitet, te postaviti termine za rezervacije. Također imaju mogućnost pratiti raspored rezervacija, prihvaćati ili odbijati rezervacije i upravljati njima. S druge strane, obični korisnici mogu pretraživati restorane na temelju lokacije i naziva, rezervirati stolove odabirom kapaciteta i preferiranog termina te ostavljati recenzije restorana. S druge strane, "Table Manager" je napredna aplikacija koja pruža sveobuhvatan set alata za upravljanje stolovima u restoranima. Ova aplikacija je posebno usmjerena na vlasnike restorana i menadžere koji žele učinkovito upravljati stolovima i rezervacijama kao što se vidi iz Slika 1. "Table Manager" nudi mogućnosti kao što su praćenje kapaciteta stolova, postavljanje rezervacijskih termina, upravljanje vremenom i rasporedom rezervacija te generiranje izvještaja o rezervacijama i popunjenosti stolova. Osim toga, aplikacija može pružiti dodatne značajke kao što su upravljanje osobljem, vođenje inventara i analitika poslovanja. U usporedbi s "Table Manager" aplikacijom, "Cutlery" se ističe svojom jednostavnošću i fokusom na osnovne funkcionalnosti rezervacije stolova. Dok "Table Manager" pruža napredne alate za upravljanje restoranom, "Cutlery" se usredotočuje na intuitivan proces rezervacije i recenziranja restorana. Konačni odabir između ove dvije aplikacije ovisit će o preferencijama vlasnika restorana i njihovim specifičnim potrebama u upravljanju stolovima i rezervacijama. Obje aplikacije se natječu na tržištu aplikacija za rezervaciju stolova u restoranima, a konačan izbor između njih ovisit će o tome koja aplikacija najbolje odgovara [9].

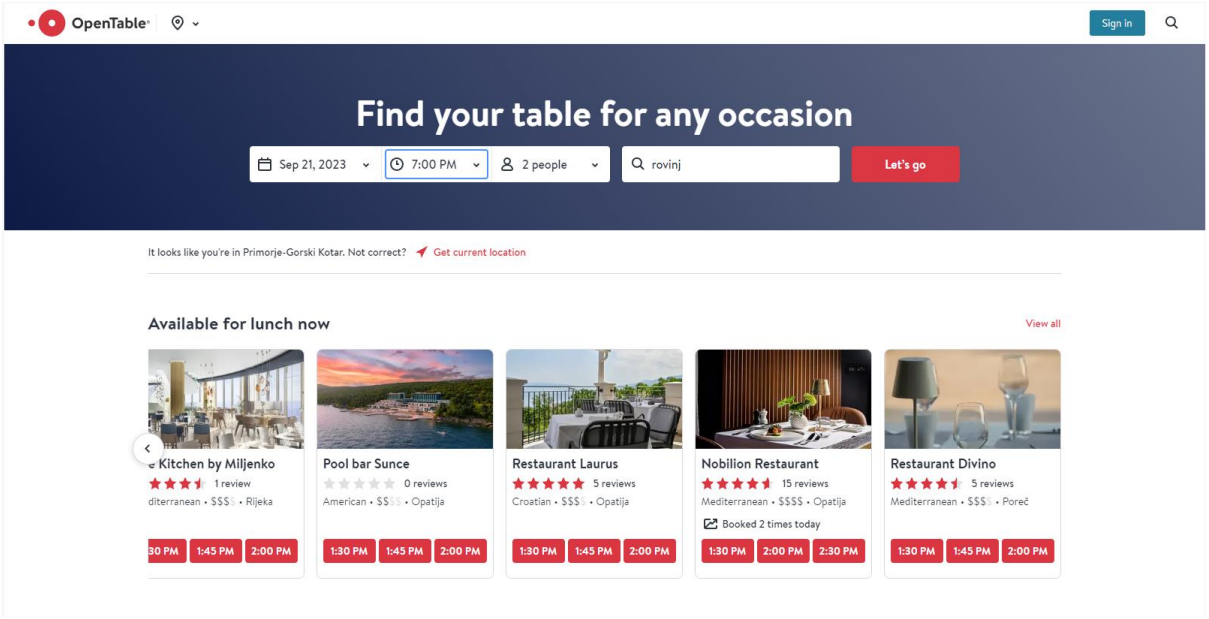


Slika 1. Izgled Table Manager aplikacije

2.2 Open Table

Aplikacija "Cutlery" i "Open Table" su dvije aplikacije koje pružaju usluge rezervacije stolova u restoranima. Iako imaju sličan koncept, postoje značajne razlike između njih koje ih čine jedinstvenim. "Cutlery" se ističe svojim jednostavnim i intuitivnim sučeljem koje omogućuje korisnicima brzo i jednostavno rezervirati stolove u restoranima. Ova aplikacija je usmjerena na vlasnike restorana i korisnike koji traže praktično rješenje za rezervaciju. Vlasnici restorana mogu dodati informacije o svom restoranu, uključujući opis, dostupne stolove i kapacitet, te postaviti termine za rezervacije. Također imaju mogućnost praćenja rasporeda rezervacija, prihvaćanja ili odbijanja

rezervacija i upravljanja njima. Obični korisnici mogu pretraživati restorane po lokaciji i nazivu, rezervirati stolove odabirom kapaciteta i preferiranog termina te ostavljati recenzije restorana. S druge strane, "Open Table" je veća i etablirana aplikacija koja pruža sveobuhvatan set alata za rezervaciju stolova u restoranima. Ova aplikacija se ističe po svojoj globalnoj dostupnosti i velikom broju restorana koji su dostupni za rezervaciju što prikazuje Slika 2. "Open Table" pruža dodatne značajke kao što su ocjene restorana, detaljni pregled jelovnika, fotografije i recenzije korisnika. Osim toga, aplikacija nudi pogodnosti poput nagrađivanja korisnika bodovima za svaku rezervaciju koja se može iskoristiti za popuste ili nagrade u restoranima. Uspoređujući "Cutlery" i "Open Table", "Cutlery" se ističe po svom jednostavnom pristupu i fokusu na osnovne funkcionalnosti rezervacije stolova. Dok je "Open Table" sveobuhvatna aplikacija koja pruža više informacija, veći izbor restorana i dodatne pogodnosti za korisnike. Vlasnici restorana mogu odabrati aplikaciju prema svojim potrebama i ciljevima. Ako žele jednostavnost i praktičnost, "Cutlery" je idealan izbor. S druge strane, ako traže veću globalnu prisutnost i dodatne značajke za svoje restorane, "Open Table" pruža sve što im treba. U tržišnom natjecanju, "Open Table" je već etabliran i poznat kao jedan od vodećih pružatelja usluga rezervacija stolova u restoranima. S druge strane, "Cutlery" ima novi pristup rezervaciji stolova u restoranima. Sa svojim jednostavnim sučeljem i osnovnim funkcionalnostima, nudi praktično rješenje za korisnike i vlasnike restorana. Iako još nije toliko prepoznatljiv kao "Open Table", "Cutlery" se ističe svojom jednostavnošću i intuitivnošću. Korisnici mogu brzo rezervirati stolove, ostavljati recenzije i pretraživati restorane. Za vlasnike restorana, aplikacija omogućuje upravljanje stolovima, postavljanje termina rezervacija i praćenje rasporeda. "Cutlery" ima potencijal za rast i privlačenje korisnika koji cijene jednostavnost i praktičnost [10].

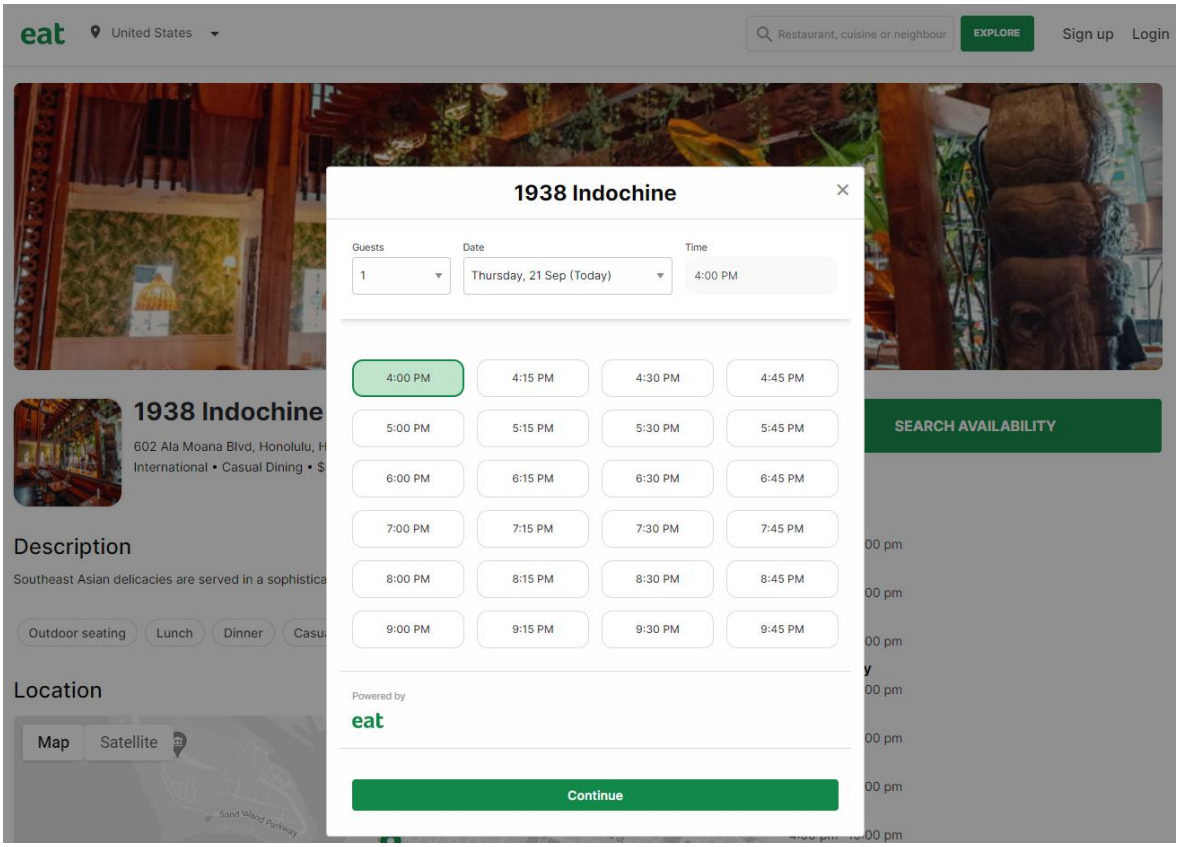


Slika 2. Izgled Open Table aplikacije

2.3 Eat App

Aplikacija "Cutlery" i "EatApp" su dvije različite aplikacije kojima je zajednički cilj olakšavanje procesa rezervacije stolova u restoranima. Iako oba alata pružaju sličnu funkcionalnost, postoje neke bitne razlike između njih. "Cutlery" je jednostavna aplikacija koja se ističe svojim intuitivnim sučeljem i osnovnim funkcionalnostima. Glavni korisnici aplikacije su vlasnici restorana i obični korisnici. Vlasnici restorana imaju mogućnost dodavanja informacija o svom restoranu, poput opisa, dostupnih stolova i njihovog kapaciteta te unosa termina za rezervacije. Također mogu pratiti raspored rezervacija, prihvaćati ili odbijati rezervacije i upravljati njima. S druge strane, obični korisnici mogu pretraživati restorane na temelju lokacije i imena, rezervirati stolove odabirom kapaciteta i željenog termina te ostavljati recenzije restorana. S druge strane, "EatApp" je naprednija aplikacija koja može sadržavati algoritme za pretraživanje restorana koji omogućuju korisnicima detaljnije filtriranje rezultata pretrage. Osim osnovnih funkcionalnosti rezervacije stolova, "EatApp" može pružati dodatne značajke kao što su ocjene korisnika, komentari, fotografije hrane, napredna pretraga temeljena na raznim kriterijima (kao što su vrsta kuhinje, cijene, ocjene) te mogućnosti ocjene različitih aspekata restorana kao što su hrana, usluga i atmosfera. Također, "EatApp" može pružiti napredne mogućnosti upravljanja restoranom, kao što

su vođenje inventara, upravljanje osobljem i analitika poslovanja. Konkurencija na tržištu aplikacija za rezervaciju restorana je jaka, s različitim aplikacijama koje nude različite funkcionalnosti i usluge. Konačni odabir između "Cutlery" i "EatApp" ovisit će o preferencijama korisnika. Ako korisnik preferira jednostavnost i osnovne funkcionalnosti, "Cutlery" može biti idealan izbor. S druge strane, ako korisnik traži naprednije mogućnosti pretrage, ocjene restorana i dodatne značajke za upravljanje restoranom, "EatApp" može zadovoljiti njihove potrebe. Slika 3 prikazuje sistem rezervacije [11].



Slika 3. Izgled Eat App aplikacije

3. APLIKACIJA „CUTLERY“

3.1 Kritička analiza

Tradicionalni načini rezervacije stolova u restoranima poput telefonskih poziva ili dolaska osobno u restoran mogu biti spori i neučinkoviti. Često su telefonske linije restorana zauzete, a osoblje nema dovoljno vremena posvetiti se svakom pozivu.

Gosti ponekad moraju dugo čekati da dođu na red ili ne uspijevaju dobiti željeni termin. Online rezervacijski sustavi poput aplikacije "Cutlery" nude rješenje ovih problema i unapređuju iskustvo rezerviranja stola. Putem intuitivnog sučelja, gosti mogu brzo pretražiti restorane, provjeriti recenzije i fotografije te odabrati slobodan termin prema svojim željama. Smanjuje se opterećenje osoblja restorana i eliminiraju duga čekanja telefonskih poziva. Međutim, postojeće aplikacije za rezervacije često imaju određena ograničenja. Većina pokriva samo veće restorane u velikim gradovima. Manji restorani teško se probijaju na takve platforme. Neki nude rezervacije bez mogućnosti otkazivanja, što frustrira korisnike. Sučelja mogu biti zbunjujuća i teška za korištenje. Aplikacija "Cutlery" razvijena je imajući na umu ove nedostatke. Jednostavno i intuitivno sučelje čini rezervacije pristupačnima svima. Mogućnost uključivanja manjih restorana otvara im nove mogućnosti poslovanja. Fleksibilnost u otkazivanju i promjenama termina zadovoljava potrebe korisnika. Međutim, još uvijek postoji prostor za napredak. Potrebno je proširiti bazu korisnika i restorana kako bi se ostvarila kritična masa. Performanse i stabilnost sustava zahtijevaju dodatnu evaluaciju pod opterećenjem. Sigurnosni aspekti poput zaštite podataka korisnika trebaju se razmotriti. Ukratko, "Cutlery" predstavlja inovativno rješenje koje modernizira proces rezerviranja stolova. Jednostavnost korištenja i fleksibilnost ističu se kao glavne prednosti. Daljnjim unapređenjem i razvojem ova aplikacija ima potencijal postati vodeće rješenje za rezervacije u restoranima.

3.2 SWOT analiza

3.2.1 Snage

Snage su unutarnji resursi, vještine i prednosti koje proizvod čine konkurentnim. Ova analiza identificira ono u čemu organizacija dobro funkcionira i kako može iskoristiti te prednosti. SWOT analiza identificirala je sljedeće prednosti i jake strane ovog projekta:

- Jednostavno sučelje: Aplikacija "Cutlery" se ističe po svojem jednostavnom sučelju koje omogućuje korisnicima intuitivno korištenje aplikacije bez potrebe za složenim uputama.

- Brza rezervacija: Korisnici mogu brzo pregledati dostupne restorane, rezervirati stolove i upravljati rezervacijama putem jednostavnih koraka, pružajući im praktično rješenje za rezervaciju stolova.
- Informacije o restoranima: Aplikacija pruža korisnicima detaljne informacije o restoranima, uključujući opise, ocjene korisnika, fotografije i jelovnike, što korisnicima pomaže u donošenju informirane odluke prilikom rezervacije.

3.2.2 Slabosti

Slabosti su unutarnji nedostaci ili ograničenja proizvoda odnosno organizacije. Ova analiza istražuje aspekte u kojima organizacija može biti ranjiva ili koje treba poboljšati kako bi bila konkurentnija. Određene slabosti i nedostaci projekta prepoznati SWOT analizom uključuju:

- Nedovoljna prepoznatljivost: Budući da je "Cutlery" nova aplikacija, može se suočiti s izazovom nedovoljne prepoznatljivosti na tržištu u usporedbi s etabliranim konkurentima.
- Ograničen broj restorana: Aplikacija može imati ograničen izbor restorana za rezervaciju u početnim fazama, što može ograničiti opcije korisnicima.

3.2.3 Prilike

Prilike predstavljaju vanjske okolnosti koji mogu koristiti proizvodu odnosno organizaciji. Ova analiza pomaže identificirati trenutne ili buduće mogućnosti na tržištu koje proizvod može iskoristiti za rast i uspjeh. Analiza je ukazala na sljedeće prilike za daljnji razvoj i unaprjeđenje projekta:

- Povećanje korisničke baze: "Cutlery" ima priliku privući više korisnika privlačnim marketinškim strategijama i poboljšanim korisničkim iskustvom, čime se povećava korisnička baza i širi tržišni udio.
- Partnerstva s restoranima: Suradnja s restoranima i uspostavljanje partnerstava omogućuje aplikaciji veći izbor restorana za rezervaciju, čime se privlače novi korisnici i pruža širi spektar opcija.

3.2.4 Prijetnje

Prijetnje su vanjske okolnosti koji mogu predstavljati rizik za proizvod odnosno organizaciju. Ova analiza pomaže identificirati potencijalne izazove i opasnosti koje organizacija treba uzeti u obzir i zaštititi se od njih. Kao moguće vanjske prijetnje i izazove prepoznate su:

- Konkurencija: Industrija rezervacija stolova u restoranima već ima etablirane konkurente poput "Open Table", koji imaju veću prepoznatljivost i veći broj restorana na svojoj platformi.
- Tehnički problemi: Tehničke poteškoće ili nestabilnost aplikacije mogu dovesti do nezadovoljstva korisnika i lošeg iskustva, što može negativno utjecati na reputaciju aplikacije.

3.3 Problem koji rješava

Aplikacija "Cutlery" je inovativno rješenje koje se bavi problemima rezervacije stolova u restoranima i pruža korisnicima jednostavan način za pronalaženje, rezerviranje i upravljanje svojim restoranskim iskustvom.

Jedan od ključnih problema s kojima se korisnici susreću prilikom rezervacije stolova u restoranima je neefikasnost i dugotrajnost procesa. Tradicionalno, rezervacije se vrše putem telefonskih poziva ili e-pošte, što može biti zamorno, posebno za restorane koji često imaju veliki broj rezervacija. Aplikacija "Cutlery" rješava taj problem omogućavajući korisnicima da brzo i jednostavno pregledaju dostupne restorane, rezerviraju stolove i upravljaju svojim rezervacijama putem intuitivnog sučelja. Drugi problem s kojim se korisnici susreću je nedostatak informacija o restoranima prilikom rezervacije. Često se korisnici osjećaju nesigurno ili neinformirano o restoranu koji žele posjetiti. Aplikacija "Cutlery" rješava taj problem pružajući detaljne informacije o restoranima, uključujući opise, ocjene korisnika te fotografije. Korisnici mogu proučiti te informacije prije nego što donesu odluku o rezervaciji, što im pruža veću sigurnost i povjerenje u odabir restorana. Također, "Cutlery" se bavi problemom otkazivanja rezervacija. Uobičajeno je da korisnici moraju kontaktirati restoran izravno kako bi otkazali ili izmijenili rezervaciju, što može biti nezgodno i za korisnike i za restorane. Aplikacija "Cutlery" omogućuje korisnicima da jednostavno odgode ili otkažu

rezervaciju putem sučelja aplikacije, čime se olakšava proces i smanjuje nepotrebna komunikacija. Vlasnici restorana također susreću svoje specifične izazove prilikom upravljanja rezervacijama i rasporedom stolova. Aplikacija "Cutlery" pruža vlasnicima restorana alate za jednostavno postavljanje dostupnih stolova, odabir termina rezervacija i praćenje rasporeda. Također im omogućuje da pregledaju i upravljaju rezervacijama na jednom mjestu, što olakšava njihovo poslovanje i povećava učinkovitost. Kroz svoje jednostavno sučelje, "Cutlery" rješava probleme rezervacije stolova u restoranima pružajući korisnicima brz i praktičan način za pronalaženje, rezerviranje i upravljanje svojim restoranskim iskustvom. Aplikacija osigurava informacije o restoranima, olakšava proces otkazivanja rezervacija i pruža alate za vlasnike restorana kako bi učinkovito upravljali svojim poslovanjem. "Cutlery" je inovacija koja donosi olakšanje i poboljšava korisničko iskustvo u industriji restorana.

4. KORIŠTENE TEHNOLOGIJE

Za izradu aplikacije "Cutlery" korištene su tri ključne tehnologije: Vue.js, Express.js i SQLite.

Vue.js je moderni JavaScript okvir za izgradnju korisničkih sučelja koji se ističe po svojoj jednostavnosti i iznimnoj fleksibilnosti. Razvijen od strane evanescentsa, Vue.js je stekao popularnost među programerima zbog svoje intuitivne sintakse i brzog učenja. Ovaj okvir koristi komponentnu arhitekturu, omogućujući programerima da razdvajaju korisničko sučelje na neovisne komponente koje se lako ponovno koriste što se jasno vidi sa Slika 4. To rezultira čistim i organiziranim kodom. Jedna od ključnih karakteristika Vue.js-a je reaktivnost, što znači da se korisničko sučelje automatski ažurira kad se podaci promijene. Ova značajka čini razvoj dinamičkih web aplikacija jednostavnim i efikasnim. Također, Vue.js pruža programerima mogućnost postupnog uvođenja u svoje projekte, što znači da se može koristiti i kao dio većih aplikacija. Vue.js također ima bogat izbor dodataka i biblioteka, što olakšava integraciju s drugim alatima i resursima. Također, zajednica Vue.js-a je aktivna i pruža mnogo resursa i podrške za programere. U "Cutlery"-ju, Vue.js je iskorišten za stvaranje dinamičkih i privlačnih korisničkih sučelja, omogućujući korisnicima jednostavno pretraživanje restorana, rezervaciju stolova i ostavljanje recenzija [2].

```vue
<div class="content">
  <div class="add-table">
    <add-table :tables="tables" />
  </div>
  <div class="add-termin">
    <add-termin />
  </div>
  <div class="get-table">
    <get-table />
  </div>
  <div class="get-termin">
    <get-termin />
  </div>
  <div class="description">
    <add-description />
  </div>
</div>
```

Slika 4. Primjer korištenja komponenti u Vue.js-u

Express.js je brzi i minimalistički web okvir za Node.js [4], razvijen kako bi olakšao izradu brzih, skalabilnih i pouzdanih web aplikacija. Ovaj okvir omogućuje programerima da efikasno upravljaju rutama, zahtjevima i odgovorima na web serveru, čime se pojednostavljuje proces razvoja kao što se vidi sa Slika 5. Jedna od ključnih karakteristika Express.js-a je njegova minimalistička priroda, što znači da pruža osnovne alate i funkcionalnosti za izradu web aplikacija, ali ostavlja dovoljno prostora za prilagodbu i dodavanje specifičnih funkcionalnosti prema potrebama projekta. Ovaj pristup čini Express.js vrlo fleksibilnim i prilagodljivim. U "Cutlery"-ju, Express.js je korišten za razvoj poslužiteljskog dijela aplikacije. To uključuje upravljanje rutama, obradu zahtjeva korisnika, komunikaciju s bazom podataka i provođenje sigurnosnih provjera. Express.js omogućuje efikasno upravljanje podacima i osigurava pouzdanost i skalabilnost aplikacije [3].

```js
const express = require("express");
const router = express.Router();

router.post("/auth/user", user.authUser());
```

Slika 5. Primjer rute koristeći express.js

SQLite je jednostavna, brza i ugrađena relacijska baza podataka koja se često koristi u aplikacijama za pohranu i upravljanje podacim. Ova popularna baza podataka razvijena je kao besplatni i otvoreni izvor i nudi jednostavno rješenje za pohranu podataka na lokalnim uređajima, što ga čini popularnim izborom u različitim aplikacijama, uključujući mobilne aplikacije i aplikacije za stolna računala. Jedna od

ključnih prednosti SQLite-a je njegova sposobnost rada bez potrebe za zasebnim poslužiteljem ili konfiguracijom, jer su podaci pohranjeni u jednom lokalnom datotečnom sustavu. Ovo pojednostavljuje implementaciju i održavanje baze podataka u aplikacijama. U "Cutlery"-ju, SQLite je odabran kao baza podataka jer nudi lagano i jednostavno rješenje za pohranu informacija o restoranima, stolovima, rezervacijama i korisnicima. Kao lokalna baza podataka, SQLite omogućuje brzo dohvaćanje i manipulaciju podacima putem SQL upita vidljivo sa Slika 6. Također pruža visoku razinu pouzdanosti i sigurnosti podataka, uz podršku transakcija i ACID svojstava.

```
async function makeReview(req) {
  const { restaurant_id, user_id, review, rate, image, time } = req.body;
  console.log("restaurant_id, review: ", restaurant_id, user_id, review, rate);
  let sql = `INSERT INTO restaurant_rating (restaurant_id, user_id,
    rate, review, images, date_time) VALUES (?, ?, ?, ?, ?, ?)`;
  try {
    let restaurant_review = await new Promise((resolve, reject) => {
      let rows = db.run(
        sql,
        [restaurant_id, user_id, rate, review, image, time],
        function (err) {
          if (err) {
            reject(err);
          } else {
            resolve({
              rows,
            });
          }
        }
      );
    });
    return { restaurant_review };
  } catch (err) {
    console.log(err);
    throw new Error("Something went wrong!");
  }
}
```

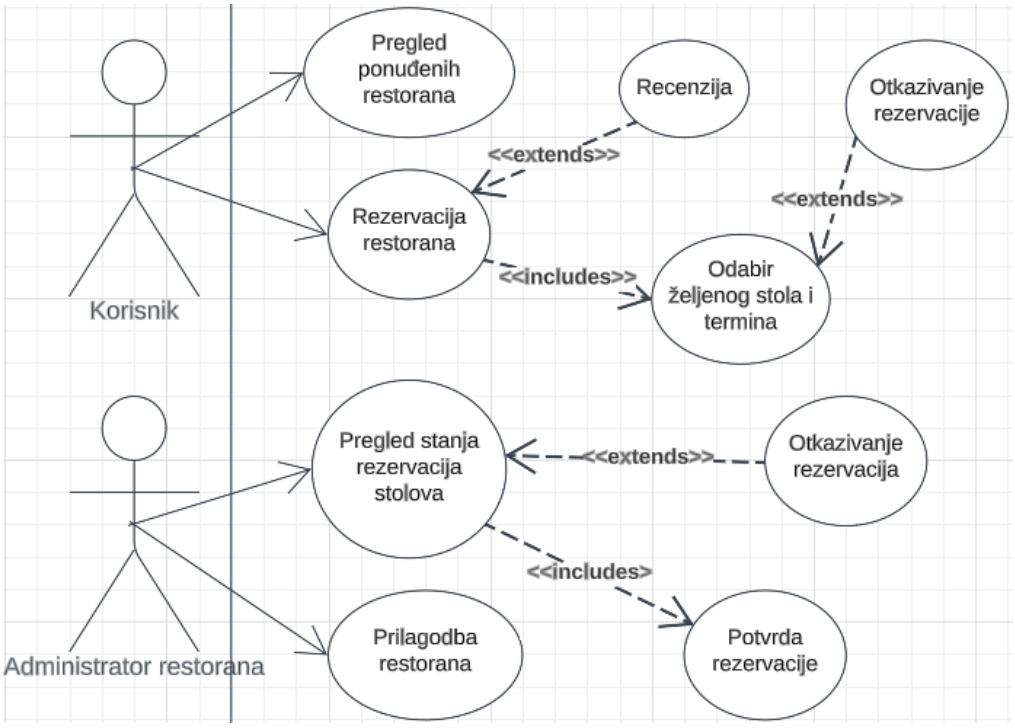
Slika 6. Primjer "insert" upita prema bazi podataka

Kombinacija Vue.js-a, Express.js-a i SQLite-a omogućuje "Cutlery"-ju da pruži korisnicima intuitivno i atraktivno korisničko iskustvo, učinkovito upravljanje podacima te sigurnu i pouzdanu platformu za rezervaciju stolova u restoranima [7].

5. MODELIRANJE FUNKCIONALNOSTI I PODATAKA

5.1 Dijagram slučajeva korištenja

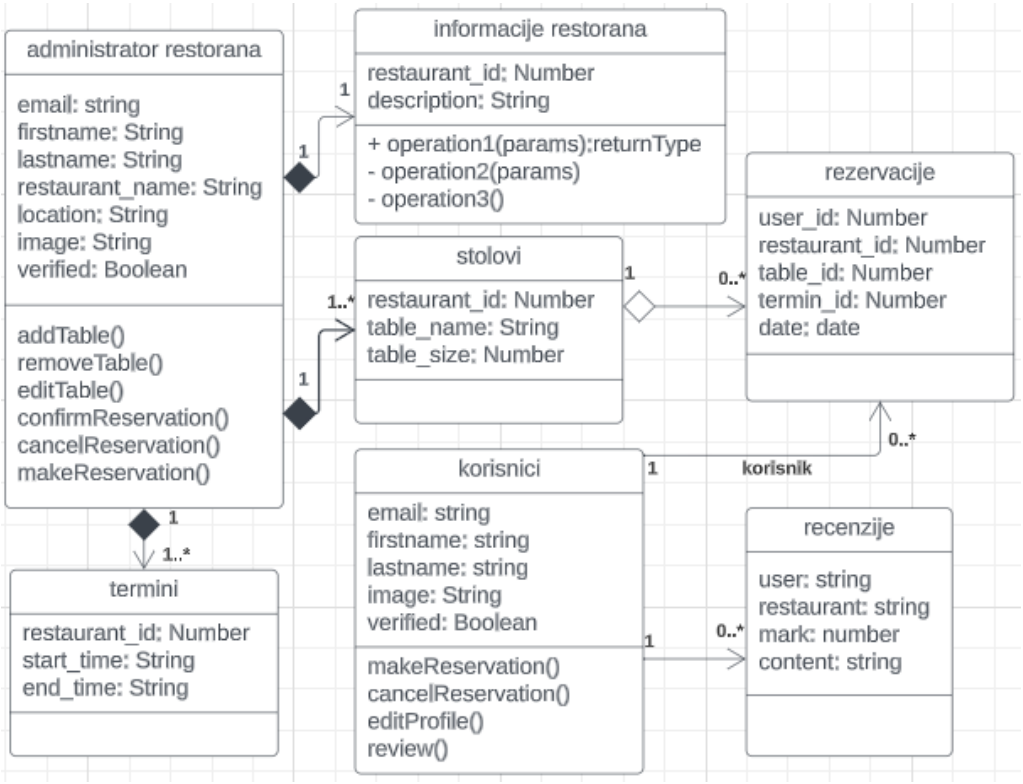
Dijagram slučajeva korištenja kao što se vidi sa Slika 7 koristan je alat za modeliranje funkcionalnosti aplikacije i odnosa između korisnika i sustava. Za ovaj projekt online rezervacija, dijagram ilustrira tko su glavni korisnici i koje akcije mogu poduzeti. Klijenti, kao krajnji korisnici, mogu pretraživati restorane, odabrati termine i stolove te iste rezervirati, otkazati rezervaciju, recenzirati restorane i pregledavati vlastite rezervacije. Sučelje im pruža sve potrebne informacije i alate za jednostavnu rezervaciju stola u restoranu. S druge strane, restoranu putem administratorskog sučelja imaju detaljan pregled rasporeda. Mogu upravljati stolovima i terminima te rezervacijama, bilo da su unesene online ili ručno. Dobivaju obavijesti o novim rezervacijama koje onda potvrđuju. Također mogu otkazati zakazane rezervacije. Nakon potvrde rezervacija, podaci o terminu se šalju klijentu. Restorani uvijek imaju uvid u sve nadolazeće rezervacije i pripadajuće klijente. Ovakav model oslikava tijek informacija i interakcija između korisnika i sustava. Reflektira ključne korake u online rezerviranju - odabir, rezervacija, potvrda. Dijagram uporabe validira funkcionalnosti koje rješavaju probleme i ispunjavaju zahtjeve korisnika.



Slika 7. Dijagram slučajeva korištenja

5.2 Klasni dijagram

Klasni dijagram prikazan na Slika 8 važan je alat za vizualni prikaz strukture i odnosa između različitih entiteta u sustavu. Za potrebe modeliranja aplikacije za rezervaciju stolova, izrađen je dijagram kako bi se jasno predočili ključni objekti i njihove veze. Dijagram se sastoji od nekoliko glavnih klasa koje predstavljaju bitne objekte u sustavu. Klasa „korisnici“ modelira osnovne podatke o korisnicima poput imena, emaila i slike. Iz nje je izvedena klasa „administrator restorana“ koja nasljeđuje njene atribute i dodaje podatke specifične administratorima restorana. Klasa „rezervacije“ sadrži podatke o rezervaciji kao što su ID korisnika i restorana, ID stola i termina, datum rezervacije. Klasa „stolovi“ modelira stolove s ID-em restorana, imenom i veličinom stola. Klasa „recenzije“ predstavlja recenzije s referencama na korisnika i restoran, ocjenom i tekstom. Ostale klase modeliraju opise restorana, termine i slične objekte sustava. Definirane su i metode specifične raznim ulogama poput rezervacije stola. Što se tiče relacija, primijenjene su kompozicija, agregacija i nasljeđivanje ovisno o prirodi veze objekata. Primjerice, rezervacija je u kompoziciji s korisnikom, dok termini nasljeđuju restoran. Ovakav klasni dijagram pruža pregledan uvid u strukturu i odnose ključnih objekata sustava što olakšava daljnji razvoj baze podataka i aplikacijske logike.



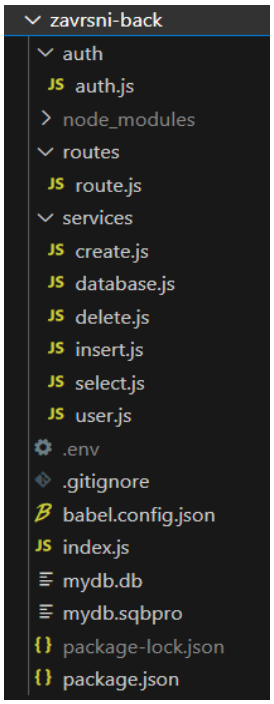
Slika 8. Klasni dijagram

6. IMPLEMENTACIJA WEB APLIKACIJE

6.1 Poslužiteljski servis

6.1.1 Struktura datoteka poslužitelja

Projekt se sastoji od dva glavna direktorija: "poslužitelj" i "klijent". Direktorij "poslužitelj" dalje je podijeljen na nekoliko poddirektorij što je prikazano na Slika 9, svaki od kojih obavlja specifičnu funkciju. Direktorij "auth" sadrži datoteku "auth.js" koji se brine za prijavu, registraciju i verifikaciju prilikom registracije, kao i pri provjeri pristupa svakoj stranici koja zahtijeva prijavu. U direktoriju "routes" nalazi se datoteka "route.js" koji sadrži definicije svih ruta. Svaka ruta poziva odgovarajuću funkciju. Sljedeći direktorij je "services", koji sadrži datoteke "create.js", "delete.js", "insert.js", "select.js" i "user.js". "create.js" kreira tablice u bazi podataka ako već ne postoje, "delete.js" sadrži funkcije za brisanje podataka iz baze podataka, "insert.js" sadrži funkcije za ubacivanje i ažuriranje podataka u bazu, "select.js" sadrži funkcije za dohvaćanje potrebnih podataka, dok "user.js" poziva sve navedene funkcije i vraća grešku ako je do nje došlo. Datoteka "index.js" služi za pokretanje servera. Također, baza podataka se nalazi u inicijalnom direktoriju. U datoteci ".env" nalaze se tajni ključevi kao što su JWT_SECRET, te e-mail i njegova pripadajuća zaporka koji služe za slanje potvrdne e-pošte.



Slika 9. Struktura datoteka poslužitelja

6.1.2 Struktura datoteka klijenta

U klijentskom dijelu projekta, koji je inicijativno generiran od strane Vue.js-a, nalazi se sljedeća struktura direktorija prikazana na Slika 10 koja pomaže u organizaciji i izgradnji korisničkog sučelja. Glavni izvor koda klijentske strane nalazi se u "src" direktoriju. Unutar tog direktorija nalaze se nekoliko ključnih direktorija i datoteka koje su odgovorne za različite aspekte aplikacije.

Direktorij "assets" je namijenjen za pohranjivanje statičnih datoteka kao što su slike, ikone ili stilovi koje se koriste u aplikaciji. Ovo omogućava jednostavan pristup i upotrebu tih resursa tijekom izgradnje sučelja.

"Components" direktorij sadrži Vue komponente koje se koriste u izgradnji korisničkog sučelja. Svaka komponenta je zasebna datoteka i može se koristiti na više mjesta u aplikaciji. Upravo to omogućava ponovno korištenje komponenti kako bismo olakšali razvoj i održavanje aplikacije.

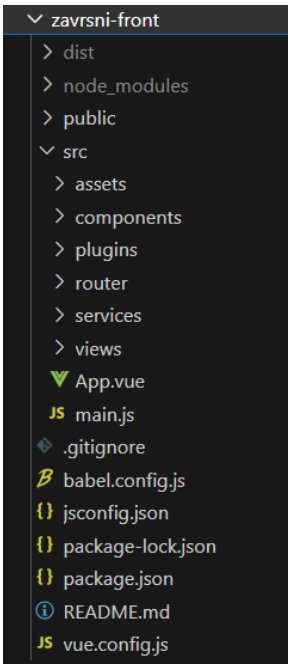
"Router" direktorij sadrži datoteku "index.js" koji definira i konfigurira rute za aplikaciju. Ovdje su definirane putanje te mogu se povezati s odgovarajućim komponentama. To omogućava navigaciju između različitih stranica u aplikaciji.

Direktorij "services" sadrži datoteku "services.js" koji obavlja važne zadatke na klijentskoj strani aplikacije, odgovoran je za verifikaciju kod svakog klijentskog zahtjeva te za postavljanje i dohvaćanje podataka iz lokalne pohrane preglednika. Kroz provjeru autentičnosti, omogućava se samo ovlaštenim korisnicima pristup određenim dijelovima aplikacije. Ova modularna struktura pruža organiziran pristup autentifikaciji i upravljanju lokalnim podacima, što rezultira sigurnijom i pouzdanijom komunikacijom s serverom.

"Views" direktorij sadrži Vue komponente koje predstavljaju stranice u aplikaciji. U pojedinom „view-u“ definira se logika i izgled svake stranice. To omogućava organiziranje i strukturiranje te maksimalnu prilagodljivost aplikacije na potrebe proizvođača.

Glavne datoteke kao što su "App.vue" i "main.js" su odgovorni za inicijalizaciju i montiranje Vue aplikacije. "App.vue" je glavna Vue komponenta koja djeluje kao korijen aplikacije i može sadržavati globalne postavke i komponente koje se primjenjuju na cijelu aplikaciju. "main.js" je glavna točka ulaska u klijentski dio

aplikacije i koristi se za inicijalizaciju Vue instance i montiranje aplikacije na HTML element u DOM-u.



Slika 10. Struktura datoteka klijenta

6.1.3 Struktura baze podataka

U okviru ove aplikacije, baza podataka igra ključnu ulogu u čuvanju svih relevantnih informacija o korisnicima, restoranima i njihovim rezervacijama. Baza podataka je organizirana u nekoliko tablica, svaka od kojih ima svoju specifičnu svrhu.

Tablica "Users" sadrži sve informacije o registriranim korisnicima aplikacije, uključujući osnovne podatke poput imena, e-mail adrese, zaporke te dodatne informacije kao što su uloga korisnika (role) i status provjere korisničkog računa (verified). Ukoliko je korisnik registriran kao vlasnik restorana, ime i lokacija restorana će također biti spremljeno u ovoj tablici.

Tablica "Verify" ima ključnu ulogu u procesu verifikacije korisničkih računa. Kada se korisnik registrira na aplikaciju, generira se jedinstveni kod za verifikaciju koji se pohranjuje u tablicu "Verify". Taj generirani kod je povezan s e-mail adresom korisnika koji se treba verificirati. Ova tablica služi kao privremeno skladište za korisnike koji su se registrirali, ali još nisu potvrdili svoj račun putem e-mail verifikacije. Kada korisnik unese verifikacijiski kod poslan na njegov e-mail, aplikacija provjerava da li postoji

odgovarajući zapis u tablici "Verify". Ako postoji, korisnikov račun se verificira, a zapis u tablici "Verify" se briše. Tablica "Termins" služi za pohranu informacija o unesenim terminima restorana. Svakom terminu je dodijeljen početak i završetak, što omogućava korisnicima da pregledaju i rezerviraju dostupne termine. Tablica "Tables" čuva informacije o stolovima u restoranima. Svaki unos u ovoj tablici sadrži detalje o stolu, uključujući kapacitet i ime. Tablica "Pending" koristi se za pohranu rezervacija koje su napravljene od strane korisnika, ali još nisu potvrđene od strane restorana. Ove rezervacije ostaju u "pending" stanju dok restoran ne potvrdi ili odbije rezervaciju. Tablica "Reservations" služi za čuvanje potvrđenih rezervacija od strane restorana. Kada restoran potvrdi rezervaciju, podaci o njoj se premještaju iz "Pending" tablice u "Reservations" tablicu. Tablica "Restaurant_rating" sadrži podatke o recenzijama restorana, uključujući ocjenu, tekstualni komentar i slike koje su korisnici podijelili. Ova tablica omogućava korisnicima pregled ocjena i recenzija restorana prije nego što naprave rezervaciju. Tablica "Restaurant_info" sadrži uvodne tekstualne opise restorana koji se prikazuju na početnoj stranici restorana. Ovo omogućava vlasnicima restorana da predstave svoj restoran na informativan način. Tablica "Restaurant_gallery" služi za pohranu slika koje je restoran dodao kako bi predstavio svoj prostor i ponudu. Ove slike se prikazuju kao galerija na profilu restorana. Uz navedene tablice, implementirani su i različiti pogledi (Views) koji služe za spajanje podataka iz više tablica kako bi se prikazali korisnicima na jednostavan način. Pogled "Pending_users" koristi se za prikazivanje imena korisnika koji su na "pending" listi, a koristi se „inner join“ kako bi se dobili podaci korisnika uz pripadajuće "pending" podatke. Slično tome, pogledi "Rates" i "Reservations_users" koriste se za povezivanje podataka korisnika s ocjenama restorana i potvrđenim rezervacijama. Struktura ove baze podataka omogućava učinkovito pohranjivanje i upravljanje informacijama o korisnicima, restoranima i rezervacijama, što osigurava glatko funkcioniranje aplikacije "Cutlery".

6.1.4 Autentifikacija

6.1.4.1 Registracija

U kôdu na Slika 11 definirana je asinkrona funkcija za registraciju koja prima zahtjev kao parametar „req“. Iz tijela zahtjeva izdvajaju se podaci o imenu, prezimenu, emailu, ulozi i zaporci korisnika. Zatim se zaporka „hashira“ koristeći bcrypt biblioteku [5] kako

bi bila pohranjena u sigurnom obliku u bazi podataka. Definiran je SQL upit koji će umetnuti podatke o korisniku u tablicu "user" baze podataka. Koriste se „Promisi“ kako bi se izvršili asinkroni upit prema bazi podataka i dobili natrag rezultat tog upita. Ako dođe do greške, „Promisi“ se odbacuju s porukom o grešci. Inače, generira se JSON Web Token za tu email adresu koristeći tajni ključ. Taj token koristit će se za autentikaciju tog korisnika u budućnosti. U „resolved“ objekt dodaju se generirani ID korisnika, token i ostali podaci o korisniku. Te na posljetku ispisuje se uspješna registracija i vraćaju se podaci o novom korisniku. Ako dođe do greške hvata se iznimka i javlja se da je email već u upotrebi. Ova funkcija na ovaj način omogućava registraciju novog korisnika u bazu podataka koristeći moderne asinkrone tehnike i sigurno spremanje podataka.

```
async function register(req) {
  const { firstname, lastname, email, role, password } = req.body;
  const password_hash = await bcrypt.hash(password, 8);
  const sql =
    `INSERT INTO user (firstname, lastname, email,
     role, password) VALUES (?, ?, ?, ?, ?)`;
  try {
    let user = await new Promise((resolve, reject) => {
      db.run(
        sql,
        [firstname, lastname, email, role, password_hash],
        function (err) {
          if (err) {
            reject(err);
          } else {
            const userId = this.lastID; // Retrieve the auto-incremented
            const token = jwt.sign({ email }, process.env.JWT_SECRET, {
              algorithm: "HS512",
              expiresIn: "1 week",
            });
            resolve({
              id: userId, // Add the user ID to the resolved object
              token,
              firstname: firstname,
              lastname: lastname,
              email: email,
              role: role,
              verified: 0,
            });
          }
        }
      );
    });
    console.log("User registered successfully: ", email);
    return { user };
  } catch (error) {
    console.error(error.message);
    throw new Error("Email already in use");
  }
}
```

Slika 11. Funkcija za registraciju korisnika

6.1.4.2 Prijava

U kôdu na Slika 12 definira se asinkrona funkcija login koja prima zahtjev kao parametar „req“. Iz zahtjeva izdvajaju se email i zaporka korisnika. Koriste se „Promisi“ kako bi se izvršio upit prema bazi podataka tražeći korisnika s unesenim emailom. Ako dođe do greške, odbacuje se „Promise“ s porukom o grešci, u suprotnom, vraćaju se podaci o tom korisniku. Ako postoji jedan podatak, uspoređuje se unesena zaporka s pohranjenom zaporkom u bazi koristeći bcrypt. Ako se podudaraju, generira se JSON web token za autentikaciju korisnika. Ako je zaporka točna prijava korisnika je uspješna te se vraćaju potrebni podaci. Inače, javlja se greška o ne važećem emailu ili zaporci. Ako u bazi ne postoji korisnik s unesenim emailom, također se javlja greška. U slučaju bilo kakve druge greške, javlja se generička greška. Ovom funkcijom omogućuje se prijava postojećeg korisnika uz provjeru zaporke i generiranje tokena za autentikaciju.

```
async function login(req) {
  const { email, password } = req.body;
  try {
    const rows = await new Promise((resolve, reject) => {
      db.all(`SELECT * FROM user WHERE email = ?`, [email], (err, rows) => {
        if (err) reject(err);
        else resolve(rows);
      });
    });
    if (rows.length > 0) {
      const passwordMatch = await bcrypt.compare(password, rows[0].password);
      if (passwordMatch) {
        const token = jwt.sign({ email }, process.env.JWT_SECRET, {
          algorithm: "HS512",
          expiresIn: "1 week",
        });
        return {
          token,
          id: rows[0].id,
          firstname: rows[0].firstname,
          lastname: rows[0].lastname,
          email: rows[0].email,
          restaurant_name: rows[0].restaurant_name,
          location: rows[0].location,
          role: rows[0].role,
          verified: rows[0].verified,
        };
      } else {
        throw new Error("Invalid email or password!");
      }
    } else {
      throw new Error("Invalid email or password!");
    }
  } catch (err) {
    throw new Error("Something went wrong!");
  }
}
```

Slika 12. Funkcija za prijavu korisnika

6.1.4.3 Autentifikacija

Funkcije sa Slika 13 iz zaglavlja zahtjeva izdvaja se autorizacijski token. Token je tipa Bearer te se sastoji od dvije komponente - tip autorizacije i sam token. Provjerava se je li tip autorizacije Bearer. Ako nije, vraća se status 401 i poruka o grešci. Ako je tip Bearer, pokušava se verificirati token koristeći tajni ključ. Ako je token valjan, prosljeđuje se zahtjev dalje middleware funkciji next(). Ako dođe do bilo kakve greške prilikom provjere tokena, vraća se status 401 i poruka o grešci. Ovom funkcijom provjerava se da li je korisnik autenticiran prije nego mu se dopusti pristup zaštićenim rutama. Samo korisnici s valjanim JWT tokenom mogu pristupiti tim rutama.

```
async function verify(req, res, next) {
  try {
    let authorization = req.headers.authorization.split(" ");
    let type = authorization[0];
    let token = authorization[1];
    //console.log(token);
    if (type !== "Bearer") {
      return res.status(401).send({ error: "Cant authorize" });
    } else {
      req.jwt = jwt.verify(token, process.env.JWT_SECRET);
      return next();
    }
  } catch (error) {
    return res.status(401).send({ error: "Cant authorize" });
  }
}
```

Slika 13. Funkcija za verifikaciju tokena korisnika

6.1.4.4 Registracija administratora restorana

U funkciji za registraciju administratora restorana sa Slika 14, za razliku od prethodne funkcije za registraciju koja služi za registraciju običnih korisnika. Kao i u prethodnoj funkciji, iz tijela zahtjeva izdvajaju se podaci koje je korisnik poslao - ime, prezime, email, naziv restorana, lokacija, uloga i zaporka. Zaporka se hashira kako bi se sigurno pohranila u bazu podataka. Definira se SQL upit za umetanje podataka u tablicu "user". Ovaj upit sadrži dodatna polja za naziv restorana i lokaciju koja su specifična samo za administratorske korisnike. Upit se asinkrono izvršava korištenjem Promise-a. U slučaju uspjeha, generira se token za autentikaciju i administratoru se postavlja „verified“ polje na 1 kako bi odmah bio verificiran. Funkcija vraća podatke o administratoru. U slučaju greške javlja da je email već u upotrebi. Dakle, iako je načelno slična, ova funkcija sadrži dodatnu logiku specifičnu za administratorske korisnike, poput dodatnih polja u bazi i automatskog verificiranja računa.

```
async function registerAdmin(req) {
  const {
    firstname,lastname, email, restaurant_name, location, role, password,
  } = req.body;
  const password_hash = await bcrypt.hash(password, 8);
  const sql =
    `INSERT INTO user (firstname, lastname, email, restaurant_name,
      location, role, password, verified) VALUES (?, ?, ?, ?, ?, ?, ?, ?)`;
  try {
    let user = await new Promise((resolve, reject) => {
      db.run(
        sql,
        [
          firstname, lastname, email, restaurant_name, location, role, password_hash, 1,
        ],
        function (err) {
          if (err) {
            reject(err);
          } else {
            const userId = this.lastID;
            const token = jwt.sign({ email }, process.env.JWT_SECRET, {
              algorithm: "HS512",
              expiresIn: "1 week",
            });
            resolve({
              id: userId,
              token,
              firstname: firstname,
              lastname: lastname,
              email: email,
              role: role,
              verified: 1,
            });
          }
        }
      );
    });
    return { user };
  } catch (error) {
    console.error(error.message);
    throw new Error("Email already in use");
  }
}
```

Slika 14. Funkcija za registraciju administratora restorana

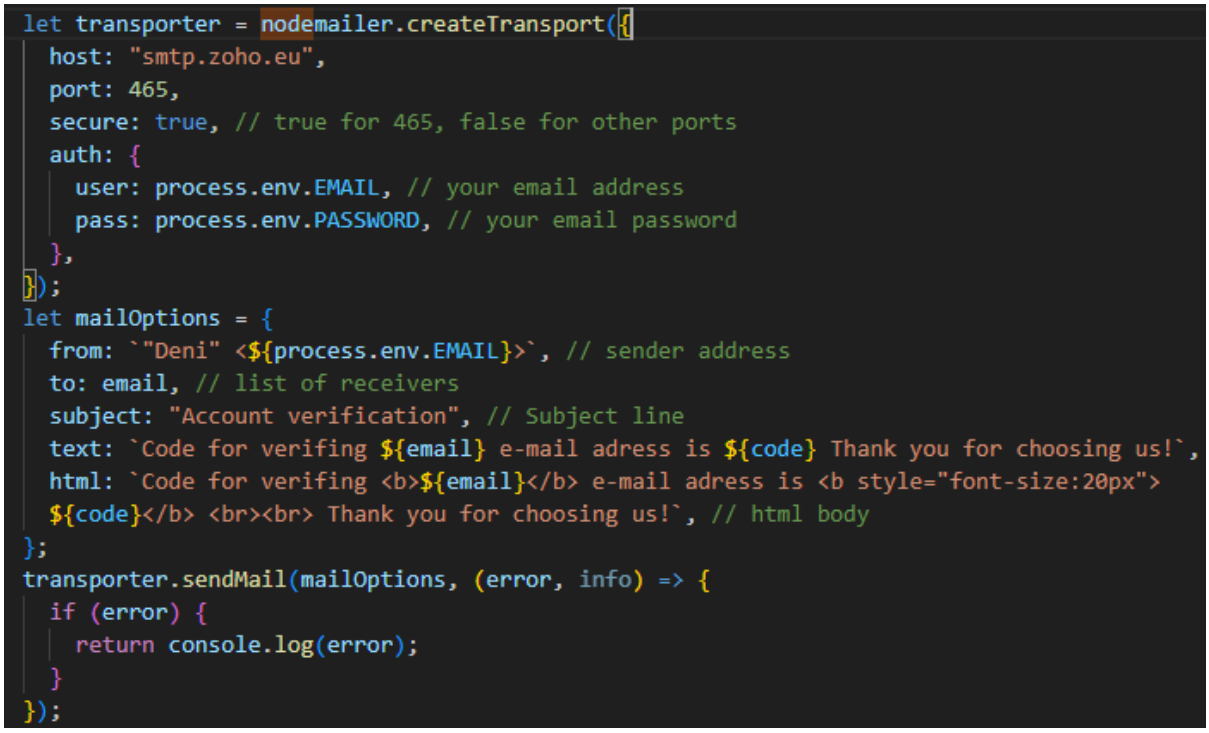
6.1.4.5 Potvrdi račun

Funkcija makeVerifyCode() služi za generiranje privremene verifikacijskog koda i slanje iste korisniku putem email-a što prikazuje Slika 15. Prvo se generira nasumični string određene duljine koji će predstavljati verifikacijski kôd. Taj kôd se hashira kako bi se pohranio u bazu. Zatim se dohvaća korisnik iz baze pomoću email adrese. Generirani hash kôda i email se spremaju u tablicu "verify" baze podataka. Šalje se email korisniku koji sadrži generirani verifikacijski kôd u tekstualnom i HTML formatu

koristeći nodemailer [1], biblioteku za automatizirano slanje e-mailova koja je prikazana na Slika 16. Ako je sve uspješno, vraćaju se podaci o verifikaciji. U slučaju greške, javlja se da email već postoji u bazi. Dakle, funkcija omogućava slanje jednokratnog verifikacijskog kôda korisniku emailom i spremanje istog u bazu. Time se postiže verifikacija email adrese pri registraciji.

```
async function makeVerifyCode(req, email) {
  async function generateRandomString(length) {
    let result = "";
    const characters =
      "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";
    const charactersLength = characters.length;
    for (let i = 0; i < length; i++) {
      result += characters.charAt(Math.floor(Math.random() * charactersLength));
    } return result;
  }
  const code = await generateRandomString(7);
  const result_hash = await bcrypt.hash(code, 8)
  const getUserSQL = "SELECT * FROM user WHERE user.email = ?";
  try {
    let getEmail = await new Promise((resolve, reject) => {
      db.run(getUserSQL, [email], function (err) {
        if (err) {
          reject(err);
        } else {
          resolve(getEmail);
        }
      });
    });
  } catch (error) {
    console.error(error.message);
    throw new Error("Email cant get email");
  }
  const sql = "INSERT INTO verify (email, code) VALUES (?, ?)";
  try {
    let verify = await new Promise((resolve, reject) => {
      db.run(sql, [email, result_hash], function (err) {
        if (err) {
          reject(err);
        } else {
          resolve({
            email: email,
            code: code,
          });
        }
      });
    });
    console.log("EMAIL: ", email);
```

Slika 15. Funkcija za kreiranje verifikacijskog koda za potvrdu profila



```
let transporter = nodemailer.createTransport({
  host: "smtp.zoho.eu",
  port: 465,
  secure: true, // true for 465, false for other ports
  auth: {
    user: process.env.EMAIL, // your email address
    pass: process.env.PASSWORD, // your email password
  },
});
let mailOptions = {
  from: `"Deni" <${process.env.EMAIL}>`, // sender address
  to: email, // list of receivers
  subject: "Account verification", // Subject line
  text: `Code for verifing ${email} e-mail adress is ${code} Thank you for choosing us!`,
  html: `Code for verifing <b>${email}</b> e-mail adress is <b style="font-size:20px">
  ${code}</b> <br><br> Thank you for choosing us!`, // html body
};
transporter.sendMail(mailOptions, (error, info) => {
  if (error) {
    return console.log(error);
  }
});
```

Slika 16. Nodemailer modul za automatizirano slanje verifikacijskih kodova putem e-maila

Funkcija verifyCode() sa Slika 17 služi za provjeru unesenog verifikacijskog kôda i potvrdu korisničkog računa. Iz tijela zahtjeva dohvaćaju se email i uneseni kôd. Iz tablice "verify" dohvaća se spremljeni hash kôda za taj email. Uneseni kôd i spremljeni hash uspoređuju se koristeći bcrypt. Ako se podudaraju, u tablici "user" postavlja se polje "verified" na 1 za tog korisnika kako bi mu se označilo da je verificirao račun. Funkcija vraća poruku o uspješnoj verifikaciji i status 200. Ako se kôdovi ne podudaraju, vraća se poruka o neuspješnoj verifikaciji i status 500. U slučaju greške javlja se da se email ne može verificirati. Dakle, ova funkcija omogućuje potvrdu korisničkog računa usporedbom jednokratnog verifikacijskog kôda poslanog korisniku ranije putem email-a. Time se postiže dodatna provjera vlasništva nad emailom.

```
async function verifyCode(req) {
  const { email, input_code } = req.body;
  console.log("user_id, code: ", email, input_code)
  try {
    const rows = await new Promise((resolve, reject) => {
      db.all("SELECT * FROM verify WHERE verify.email = ?", [email], (err, rows) => {
        if (err) reject(err);
        else resolve(rows);
      });
    });
    const codeMatch = await bcrypt.compare(input_code, rows[0].code);
    if(codeMatch){
      const updateVerify = await new Promise((resolve, reject) => {
        db.all("UPDATE user SET verified = 1 WHERE user.email = ?", [email], (err, rows) => {
          if (err) reject(err);
          else resolve(rows);
        });
      });
      return {
        msg: "successfully verified",
        status: 200
      }
    } else{
      console.log("Not matching!")
      return {
        msg: "Code doesn't match",
        status: 500
      }
    }
  } catch (err) {
    console.error(err.message);
    throw new Error("Cannot verify email");
  }
}
```

Slika 17. Funkcija za verifikaciju unesenog verifikacijskog koda

6.1.5 Organizacija API ruta

Datoteka koja sadrži definiranje ruta na poslužiteljskom servisu je pojednostavljeni način za postavljanje REST API krajnjih točka koji pozivaju odgovarajuće funkcije za obradu zahtjeva. U toj datoteci se koristi objekt „router“ iz Express.js biblioteke za definiranje ruta s određenim HTTP metodama poput GET, POST, PUT, DELETE. Za svaku rutu se poziva odgovarajuća funkcija koja će obraditi taj zahtjev. Te funkcije su definirane u zasebnim datotekama, što olakšava preglednost i održavanje kôda. Prednost ovog pristupa je jednostavnost - sve rute su definirane na jednom mjestu. No mana je potencijalno velika datoteka ruta te gniježđenje funkcija za obradu u zasebne datoteke. Za veće aplikacije bolja je praksa koristiti princip ruta, kontrolera i

servisa. Rute i dalje definiraju krajnje točke i metodu, ali pozivaju kontrolere. Kontroleri zatim komuniciraju sa servisima koji izvode stvarnu poslovnu logiku. No za jednostavnije API-je direktno pozivanje funkcija za obradu unutar ruta je prihvatljiv pristup. Svakako, valja organizirati funkcije u smislene cjeline i odvojiti složenu logiku u servise po potrebi.

6.1.6 Servisi

Struktura direktorija „services“ s datotekama create.js, delete.js, insert.js i select.js predstavlja uobičajeni pristup odvajanju poslovne logike prema operacijama nad bazom podataka. Svaka datoteka sadrži funkcije koje izvode određenu vrstu akcije:

- create.js - funkcije koje kreiraju/dodaju nove tablice u bazu podataka(ako već ne postoji)
- delete.js - funkcije koje brišu postojeće podatke iz baze podataka
- insert.js - funkcije koje umeću/ažuriraju podatke u bazi podataka
- select.js - funkcije koje čitaju/dohvaćaju podatke iz baze podataka

Ovakva podjela olakšava održavanje koda i razumijevanje gdje se nalazi određena vrsta logike. Prednost je izbjegavanje jedne velike datoteke s miješanom logikom. Nedostatak može biti potreba za koordinacijom logike između više datoteka. Idealno bi bilo izdvojiti zajedničku/ponovljenu logiku u zasebne klase. Ukratko, radi se o čistom pristupu grupiranju srodnih funkcija, iako bi daljnja apstrakcija i modularizacija bila preporučljiva.

6.1.7 Index.js

Datoteka index.js predstavlja glavnu datoteku Node.js/Express aplikacije koja pokreće i konfigurira sam server. Prvo se uvoze potrebni moduli:

- express - za pokretanje Express servera
- cors - za omogućavanje CORS zahtjeva
- baza podatka - za povezivanje s bazom podataka
- route - za usmjeravanje upita prema odgovarajućim funkcijama

Zatim se inicijalizira Express aplikacija i konfigurira middleware:

- urlencoded - za parsiranje URL kodiranih podataka
- cors - za CORS podršku

- json - za parsiranje JSON podataka
- zaglavlje middleware - za postavljanje CORS zaglavlja

Definira se glavna ruta "/api" koja će sadržavati sve REST krajnje točke aplikacije. Postavlja se port na koji će server slušati te se pokreće poslužitelj pozivom app.listen. Dakle, ova datoteka vrši osnovnu inicijalizaciju i konfiguraciju Express aplikacije, povezuje usmjeravanje i bazu podataka te pokreće poslužiteljski servis.

6.2 Klijentski servis

6.2.1 Main.js

Datoteka main.js služi kao središnja točka koja povezuje sve dijelove Vue aplikacije u jednu cjelinu. Prvo se uvoze svi potrebni vanjski moduli i biblioteke, kao što su Vue, router, vuetify [8], itd. Ovi moduli sadrže funkcionalnosti koje će se koristiti u aplikaciji. Zatim se učitavaju fontovi i ikone kako bi bili dostupni. Nakon toga kreira se glavna Vue instanca pozivom createApp() funkcije iz uvezenog Vue modula. Ovoj funkciji se predaje glavna App komponenta koja će sadržavati predložak cijele aplikacije. Sljedeće se registriraju router i vuetify biblioteka sa Vue instancom, kako bi njihove funkcije i komponente bile dostupne unutar cijele aplikacije. Na kraju, .mount() metoda pokreće i renderira Vue aplikaciju te je "montira" na element #app u HTML dokumentu. Dakle glavna datoteka spaja sve dijelove u jednu cjelinu, dok se komponente i rute definiraju odvojeno u drugim datotekama i mapama. Te odvojene komponente i rute onda glavna datoteka uvozi i koristi za pokretanje aplikacije.

6.2.2 App.vue

App.vue datoteka u Vue.js programskom okviru je korijenska komponenta i predložak cijele aplikacije. Ona definira osnovnu strukturu i raspored svih dijelova aplikacije.
Neka ključna svojstva app.vue datoteke:

- Sadrži <template> koji definira HTML kod koji će se renderirati
- <script> dio koji sadrži logiku i podatke komponente
- <style> za CSS stilove komponente
- Definira se kao .vue komponenta
- Referencira se u glavnoj datoteci pri pokretanju aplikacije (npr. main.js)
- Obično sadrži <router-view> za prikazivanje pogleda iz rutiranja

- Sve ostale komponente su potomci ove korijenske komponente

Dakle, App.vue postavlja temeljnu strukturu i raspored aplikacije te je polazna točka iz koje se granaju sve ostale komponente i rute.

Pritiskom na gumb za odjavu poziva se metoda za odjavu sa Slika 18 koja se nalazi u zasebnoj datoteci „services.js“ koja vodi brigu o prijavi, odjavi, dohvaćanju podataka korisnika.

```
logout() {
  Auth.logout();
  console.log("User logged out!");
  this.$router.go();
},
```

Slika 18. Funkcija za odjavu korisnika

Funkcija getImage() prikazana na Slika 19 dohvaća sliku trenutnog korisnika.

```
async getImage() {
  try {
    let res = await Service.get("/get/profile/image", {
      params: {
        user_id: this.currentUser.id,
      },
    });


    this.currentImage = res.data.result[0].image;
  } catch (error) {
    console.log(error);
  }
},
```

Slika 19. Funkcija za dohvaćanje slike korisnika

Slika 20 prikazuje metode koje provjeravaju je li korisnik običan korisnik ili administrator restorana te shodno s time prikazuju potrebne elemente jednostavnim korištenjem v-if-a.

```
isUser() {
  if (Auth.getUserRole() == "user") {
    this.role = true;
  } else if (Auth.getUserRole() == "admin") {
    this.role = false;
  } else {
    this.$router.push({ path: "/login" });
  }
},

isProfilePath() {
  if (this.$route.path == "/profile") {
    return false;
  } else return true;
},
isRestaurantPath() {
  if (this.$route.name == "restaurantopen" || this.$route.name == "restaurantdetail") {
    return false;
  } else return true;
},
```

Slika 20. Funkcije za provjeru vrste korisnika te provjeru rute

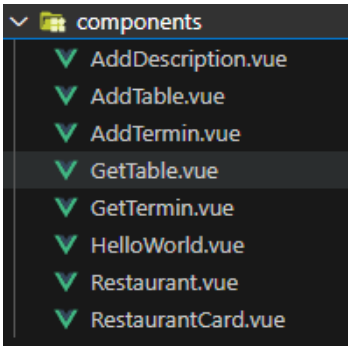
6.2.3 Assets

Direktorij "assets" ima važnu ulogu u Vue.js projektima jer omogućava centralizirano spremanje statičnih datoteka poput slika na jednom mjestu. To donosi veću preglednost i bolju organizaciju koda, pojednostavljuje referenciranje statičkih resursa iz raznih dijelova aplikacije i olakšava dijeljenje resursa među komponentama.

6.2.4 Komponente

Direktorij komponenti ima veoma važnu ulogu u strukturiranju Vue.js projekata. On služi kao mjesto za organiziranje i grupiranje svih manjih Vue komponenti od kojih se sastoji aplikacija. Vue komponente predstavljaju male modularne dijelove korisničkog sučelja poput gumba, navigacije, modalnih prozora i slično vidljivo na Slika 21. One sadrže vlastiti HTML, CSS i JavaScript kod za funkcionalnost i stilove. Prednost je što se komponente mogu višekratno koristiti na raznim mjestima unutar aplikacije. Držanje svih komponenti u zasebnom direktoriju donosi nekoliko benefita. Prvo, struktura koda postaje transparentnija jer se odmah može vidjeti koje su komponente dostupne. Drugo, olakšano je njihovo ponovno korištenje jer su sve na jednom mjestu. Treće,

srodne komponente se mogu grupirati zajedno što dodatno povećava preglednost. Također, održavanje i razvoj aplikacije je jednostavnije kada je jasno odvojena logika komponenti od ostalih dijelova. Sve u svemu direktorij komponenti je ključan za organizaciju Vue.js koda u manje dijelove i kritičan je za skalabilnost i održivost projekta.



Slika 21. Direktorij komponenta

6.2.5 Ruter

Ruter ima ključnu ulogu u Vue.js aplikacijama jer omogućava navigaciju između različitih pogleda ili stranica aplikacije bez ponovnog učitavanja cijele stranice. Ruter funkcionira tako da svakoj ruti pridružuje određeni komponentu koja predstavlja pogled za tu rutu. Na primjer, ruta "/about" može prikazivati About komponentu, dok ruta sa Slika 22 "/" prikazuje Home komponentu. Kad korisnik unese neku rutu u pregledniku, ruter "hvata" tu rutu i dinamički renderira odgovarajuću komponentu bez osvježavanja cijele stranice. Time se postiže fluidno kretanje između pogleda. Dodatno, ruter omogućava prosljeđivanje parametara kroz rutu do komponente. Tako se mogu prikazivati dinamičke komponente ovisno o parametrima u samoj ruti. Još jedna važna značajka je navigacija između ruta direktno iz JavaScript koda, što dozvoljava programiranje tokova aplikacije.

```
const routes = [
  {
    path: "/",
    name: "homeview",
    component: HomeView,
  },
```

Slika 22. Primjer rute („/“)

Osim osnovnog rutiranja, ruter u Vue.js sadrži i kontrolu navigacije (eng. navigation guard) koji omogućavaju provjeru prije ulaska ili izlaska iz rute, na primjer za provjeru autentikacije korisnika prikazano na Slika 23. Upravo on u Vue ruteru provjerava ulogu i status korisnika prije ulaska na svaku rutu kako bi osigurao da korisnik ima pravo pristupa. Prvo provjerava je li ruta javna ili zahtijeva prijavu. Zatim provjerava je li korisnik verificirao račun. Nakon toga provjerava ulogu korisnika i usmjerava na odgovarajuće rute ako korisnik „admin“ pokuša pristupiti korisničkom dijelu za koji nema ovlasti.

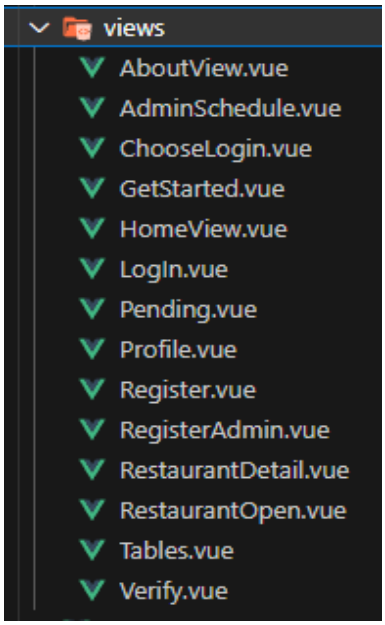
```
router.beforeEach(async (to, from, next) => {
  const publicSite = ["/getstarted", "/login", "/register", "/register/admin", "/choose"];
  const adminSite = ["/schedule", "/tables", "/pending"];
  const userSite = ["/", "/restaurant", "/verify"];
  const userRole = Auth.getUserRole();
  const userActivated = Auth.getUser().verified;
  console.log("verified: ", userActivated);
  console.log("user role: ", userRole)
  let isAuthenticated = Auth.authenticated();
  const loginRequired = !publicSite.includes(to.path);
  const adminPath = adminSite.includes(to.path);
  const userPath = userSite.includes(to.path);

  if (!isAuthenticated && loginRequired) {
    next("/login");
  } else if (userActivated === 0 && to.path !== "/verify") {
    next("/verify");
  } else if (userActivated === 1 && to.path === "/verify") {
    next("/"); // Redirect to home or other appropriate route
  } else if (isAuthenticated && !loginRequired && userActivated !== 0) {
    next("/");
  } else if (isAuthenticated && userRole !== "admin" && adminPath) {
    console.log("User access denied to admin site");
    next("/"); // redirect to user site
  } else if (isAuthenticated && userRole !== "user" && userPath) {
    // Only redirect to admin site if the current path is in adminSite and doesn't contain
    console.log("Admin access denied to user site");
    next("/schedule"); // redirect to admin site
  } else {
    next();
  }
});
```

Slika 23. Kontrola navigacije

6.2.6 Pogledi

Direktorij pogleda (eng. Views) ima važnu ulogu u organizaciji koda Vue.js aplikacije jer odvaja komponente koje predstavljaju cjelovite poglede ili stranice od sitnijih,

općenitijih komponenti prikazano na Slika 24. Pogledi su zapravo Vue komponente koje sadrže predloške, logiku i podatke za cijelu jednu rutu definiranu u ruteru. Na primjer, „HomePage.vue“ komponenta može biti pogled koji predstavlja početnu stranicu aplikacije i renderira se kada korisnik ode na glavnu rutu. Grupiranje svih pogleda zajedno u „views“ direktorij povećava preglednost i organiziranost koda. Održavanje i proširivanje aplikacije je jednostavnije kada je jasno koje datoteke predstavljaju poglede, a koje sitnije komponente. Svaki pogled je samostalan i ne ovisi toliko o ostatku aplikacije.



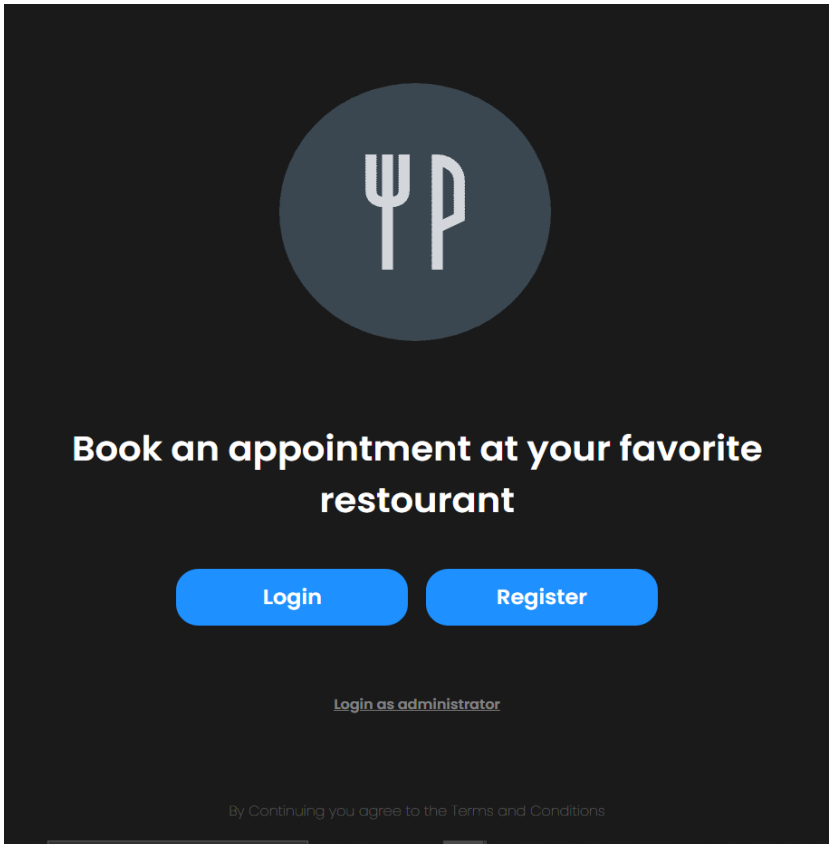
Slika 24. Direktorij pogleda (eng. Views)

6.2.7 Services.js

Servisi.js datoteka u ovom projektu služi za centralizirano rukovanje autentikacijom korisnika i povezanim funkcionalnostima. Definira se Auth objekt koji sadrži metode za registraciju, prijavu, odjavu korisnika i dohvaćanje podataka o prijavljenom korisniku iz lokalne pohrane. Također postoje helperi za provjeru autentikacije, dohvaćanje tokena i uloge korisnika. Koriste se dvije Axios [6] instance, jedna za javne rute a druga za zaštićene koja postavlja autentikacijski token u zahtjeve. Interceptori hvataju 401 i 403 greške za automatsku odjavu. State dio drži informaciju o autentikaciji. Dakle, ova datoteka kapsulira svu logiku povezanu s autentikacijom korisnika na jednom mjestu - registracija, prijava, tokeni, uloge, odjava itd. To olakšava centralizirano upravljanje tim aspektima aplikacije.

6.3 Prikaz aplikacije

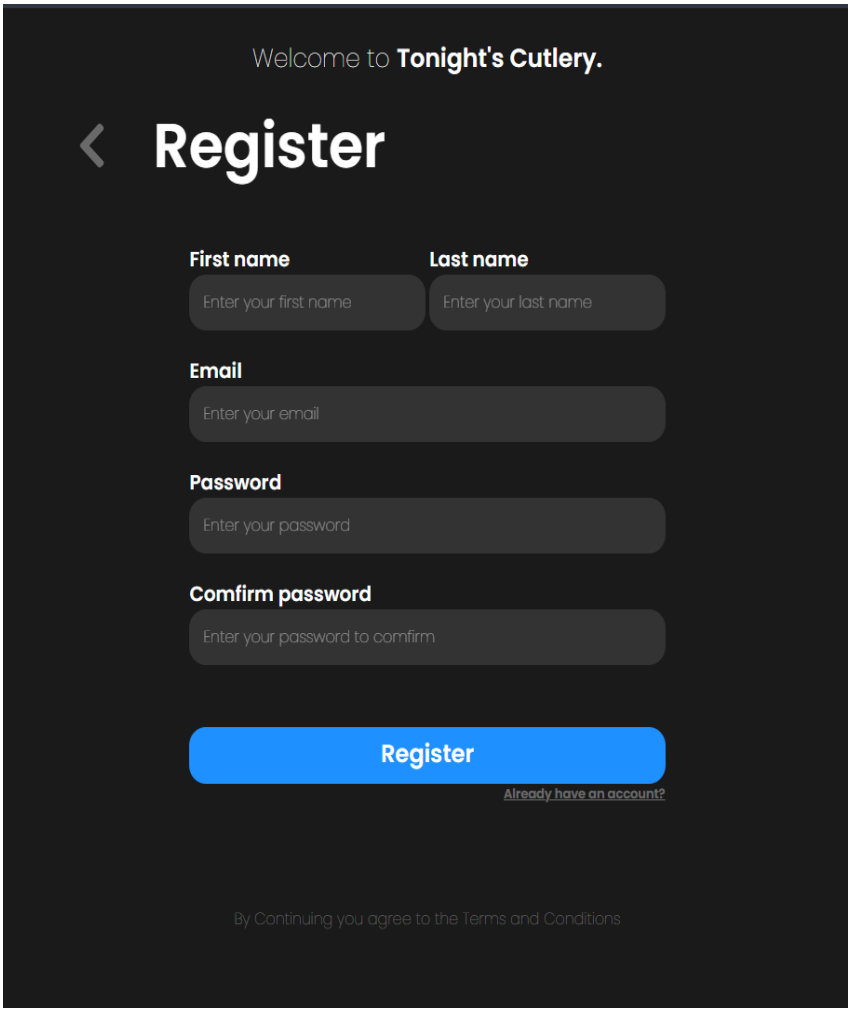
Web aplikacija “Cutlery” pruža intuitivno i jednostavno korisničko iskustvo za pregled sadržaja i navigaciju. Dizajnirana je s modernom tamnom temom koja dobro izgleda i lagano se koristi. Na početnoj stranici korisniku je istaknut naslov aplikacije te gumbi za prijavu i registraciju u slučaju da nije prijavljen što prikazuje Slika 25.



Slika 25. Početna stranica

6.3.1 Registracija korisnika

Stranica za registraciju korisnika vrlo je jednostavna i intuitivna zbog što uspješnijeg korisničkog iskustva. Sastoji se od nekoliko polja koja čine osnovne korisničke podatke vidljivo sa Slika 26, tek kada je korisnik unio ispravne podatke o novom korisniku vrši se registracija.



Slika 26. Obrazac za registraciju korisnika

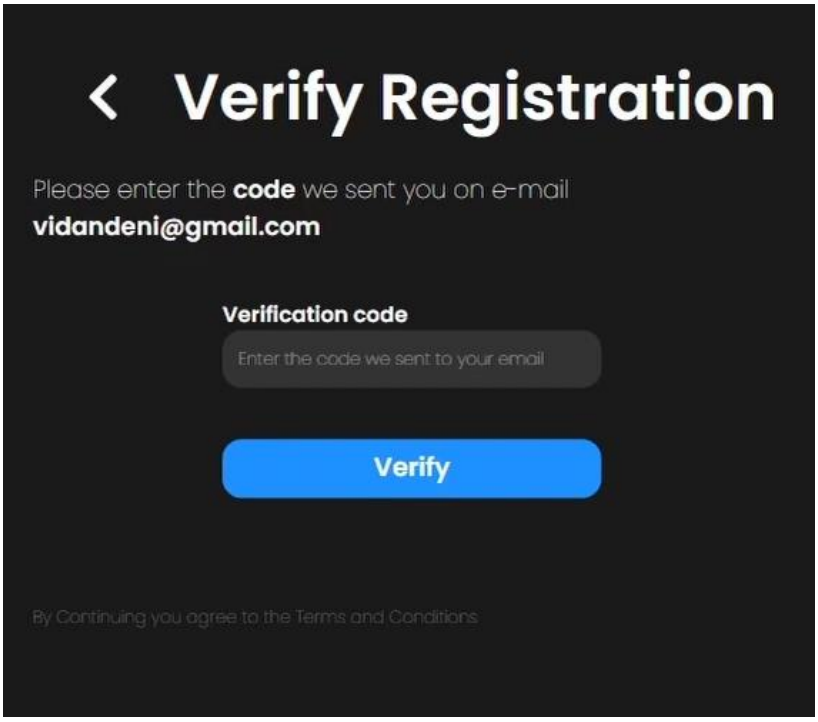
Proces registracije korisnika je implementiran u datoteci services.js, koja sadrži srodne funkcije za rad s korisnicima. Funkcija „register()“ sa Slika 27 prima parametre za ime, prezime, email, zaporku i ponovljenu zaporku novog korisnika. Prvo se provjerava podudaranje unesenih zaporki te se vraća greška ako se ne podudaraju. Nakon toga slijedi provjera jesu li sva obavezna polja popunjena, da se izbjegne slanje nepotpunog zahtjeva. Sljedeća provjera koristi regularni izraz za validaciju formata unesenog emaila. Regularni izrazi omogućuju deklarativno definiranje uzoraka za traženje i podudaranje stringova. Konačno se provjerava zadovoljava li unesena zaporka minimalnu duljinu za sigurnost. Tek nakon što podaci uspješno prođu sve ove provjere, funkcija nastavlja i izvršava registraciju novog korisničkog računa slanjem zahtjeva na poslužiteljski servis. Ovakvim postupnim validiranjem ulaznih podataka osigurava se da registracija bude uspješna i preveniraju uobičajene greške.

```
async register(firstname, lastname, email, password, passwordRepeat) {
  if (password != passwordRepeat) {
    return {
      status: 403,
      error: "Passwords do not match",
    };
  } else {
    if (firstname && lastname && email && password && passwordRepeat) {
      var mailformat = /^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/;
      if (email.match(mailformat)) {
        if (password.length >= 8) {
          try {
            let res = await ServiceAuth.post("/add/user", {
              firstname: firstname,
              lastname: lastname,
              email: email,
              role: "user",
              password: password,
              passwordRepeat: passwordRepeat,
            });
            if (res.status == 200) {
              const user = res.data.result.user;
              console.log("user: ", user);
              localStorage.setItem("user", JSON.stringify(user));
            }
            return {
              status: res.status,
              msg: res.data.msg,
            };
          } catch (err) {
            console.error(err.response.data.err);
            return {
              status: err.response.status,
              error: err.response.data.err,
            };
          }
        } else {
          return {
            status: 801,
            error: "Password must be at least 8 characters",
          };
        }
      } else {
        return {
          status: 800,
          error: "Not valid email form",
        };
      }
    } else {
      return {
        status: 401,
        error: "Please enter all the fields!",
      };
    }
  }
},
```

Slika 27. Upit za registraciju od strane klijenta

6.3.2 Verifikacija

Nakon uspješne registracije korisničkog računa, korisnik je preusmjeren na korak verifikacije email adrese. Sa poslužiteljskog servisa se šalje generirani verifikacijski kod na unesenu email adresu prilikom registracije. Korisnik zatim unosi primljeni kod u za to predviđeno polje prikazano na Slika 28.



Slika 28. Obrazac za verifikaciju korisničkog računa

Funkcija „verify()“ prikazana na Slika 29 prima parametre email i uneseni verifikacijski kod te ih šalje poslužitelju na provjeru. Poslužitelj provjerava odgovara li uneseni kod email adresi iz baze korisnika. Ako se podudaraju, poslužiteljski servis javlja uspješnu verifikaciju. Tada se u lokalnu pohranu spremaju token za autentikaciju i podaci o korisniku potrebni za normalan rad. Ako uneseni kod ne odgovara email adresi, poslužitelj vraća grešku da verifikacija nije uspjela.

```
async verify() {
  try {
    let res = await Service.post("/verify/code", {
      email: this.currentUser.email,
      input_code: this.code,
    });
    if (res.data.result.status == 200) {
      const userLocalStorage = JSON.parse(localStorage.getItem("user"));
      userLocalStorage.verified = 1;
      localStorage.setItem("user", JSON.stringify(userLocalStorage));
      this.$router.go();
      try {
        let del = await Service.delete("/delete/verify/code", {
          params: {
            email: this.currentUser.email,
          },
        });
      } catch (error) {
        console.log(error);
      }
    } else {
      this.error = "Wrong verification code";
    }
  } catch (error) {}
},
```

Slika 29. Funkcija za verifikaciju korisničkog računa od strane klijenta

6.3.3 Prijava korisnika

Proces prijave korisnika implementiran je funkcijom „login()“ prikazana na Slika 30 koja prima parametre email i zaporku korisnika koji se želi prijaviti. Prvi korak je provjera jesu li oba obavezna polja popunjena kako bi se izbjeglo slanje nepotpunog zahtjeva. Ako su email i zaporka popunjeni, šalju se uneseni podaci na provjeru poslužitelju. Tamo se uspoređuju s podacima u bazi podataka. Ako se podudaraju, poslužiteljski servis vraća statusni kod 200 što znači da je prijava uspješna. Taj statusni kôd 200 omogućava da klijentski servis pohrani autorizacijski token i podatke o prijavljenom korisniku u lokalnu pohranu preglednika. Ti podaci u lokalnoj pohrani su ključni za normalan rad aplikacije jer sadrže identifikaciju korisnika potrebnu za sve daljnje zahtjeve prema poslužitelju.

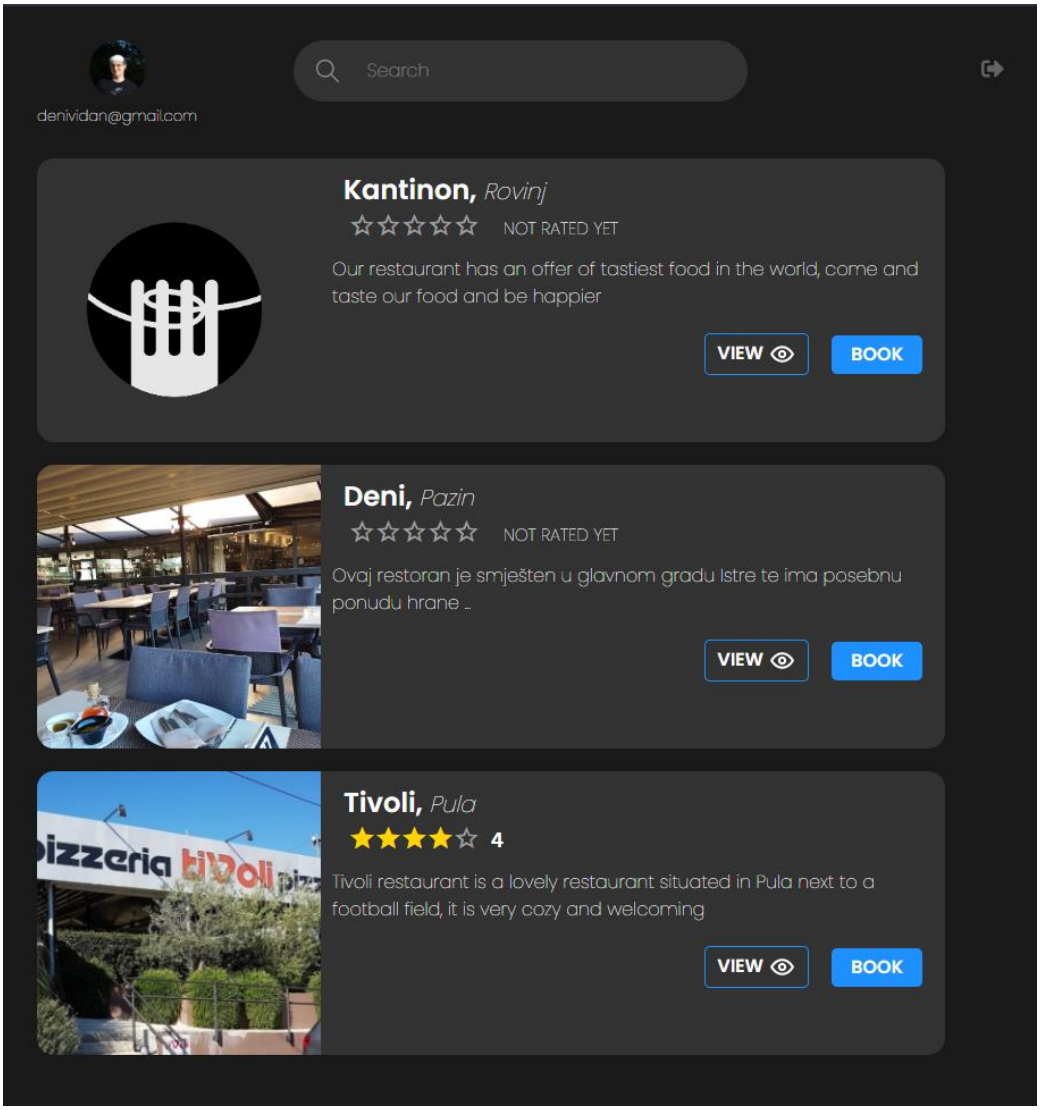
```
async login(email, password) {
  if (email && password) {
    try {
      let res = await ServiceAuth.post("/auth/user", {
        email: email,
        password: password,
      });
      if (res.status == 200) {
        const user = res.data.result;
        console.log("user: ", user);
        localStorage.setItem("user", JSON.stringify(user));
      }
      return {
        status: res.status,
        msg: res.data.msg,
        role: res.data.result.role,
      };
    } catch (err) {
      return {
        status: err.response.status,
        error: err.response.data.err,
      };
    }
  } else {
    return {
      status: 401,
      error: "Please enter all the fields!",
    };
  }
},
```

Slika 30. Funkcija za prijavu od strane klijenta

6.3.4 Stranica za pregled restorana

Stranica za pregled restorana predstavlja primarni pogled koji korisnik vidi nakon uspješne prijave ili registracije na aplikaciju. Dizajn ove stranice je minimalistički kako bi se korisniku pružilo jednostavno i intuitivno sučelje. Umjesto kompleksne navigacije, na početnoj stranici se nalaze samo osnovni linkovi za pristup korisničkom profilu, odjavi te pretraživanju restorana. Ovi elementi su jasno istaknuti na samom vrhu stranice te omogućuju brz pristup najvažnijim funkcionalnostima. Izostanak dodatnih elemenata čini ovu stranicu preglednom i lako konzumirajućom. Korisnik se može brzo orijentirati i odabrati željenu akciju. Fokus je stavljen na one mogućnosti koje će korisnik najvjerojatnije trebati na početku interakcije sa aplikacijom. Jednostavnost i minimalizam početne stranice reflektira temeljni pristup cijele aplikacije usmjeren na

korisnika. Intuitivno iskustvo vodi korisnika kroz tijek korištenja bez nepotrebnih komplikacija u sučelju vidljivo sa Slika 31.



Slika 31. Stranica za pregled restorana

Funkcija za odjavu sa Slika 32 aktivira se s početne stranice, vrlo je kratka funkcija koja jednostavno briše korisnika iz lokalne pohrane te osvježivanjem stranice korisnika preusmjerava na prijavu.

```
logout() {
  localStorage.removeItem("user");
},
```

Slika 32. Funkcija za odjavu trenutnog korisnika

Za filtraciju restorana koristi se funkcija sa Slika 33 koja prije svega provjerava ima li unesene ključne riječi, ako nema, funkcija vraća sve restorane. U slučaju unosa vrijednosti u tražilicu, vrši se provjera podudaranja ključne riječi i raspoloživih restorana. Filtracija je moguća istovremeno po imenu restorana kao i po lokaciji što omogućuje korisniku raznoliko pretraživanje.

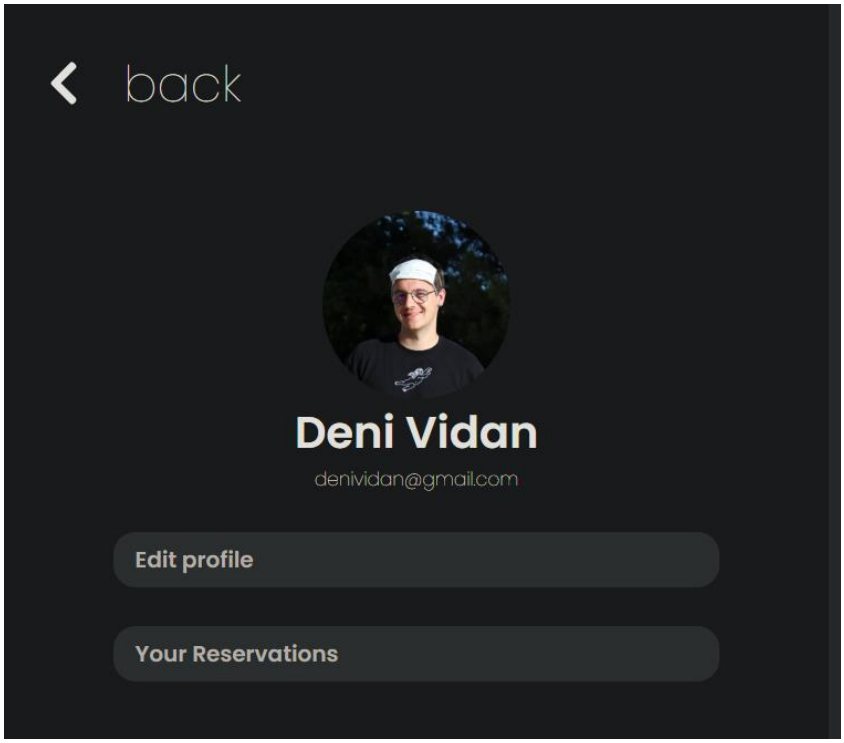
```
filteredRestaurants() {
  if (!this.searchItem) {
    return this.restaurants;
  }

  const lowercaseSearchWord = this.searchItem.toLowerCase();
  let temp = lowercaseSearchWord.split(" ");

  return this.restaurants.filter((restaurant) => {
    if (temp.length == 1) {
      return (
        restaurant.restaurant_name
          .toLowerCase()
          .includes(lowercaseSearchWord) ||
        restaurant.location.toLowerCase().includes(lowercaseSearchWord)
      );
    } else if (temp.length == 2) {
      let firstWord = temp[0];
      let secondWord = temp[1];

      return (
        restaurant.restaurant_name.toLowerCase().includes(firstWord) &&
        restaurant.location.toLowerCase().includes(secondWord)
      );
    }
  });
},
```

Slika 33. Funkcija za filtraciju restorana

6.3.5 Profil

Stranica korisničkog profila slijedi minimalistički dizajn vidljiv kroz čitavu aplikaciju, s fokusom na bitne elemente kao što prikazuje Slika 34. U središtu stranice ističe se profilna slika i osnovni podaci o korisniku, pregledno prikazani. Ispod osnovnih podataka nalazi se padajući izbornik koji daje brz pristup pregledu trenutnih rezervacija korisnika i uređivanju samog profila. Ove opcije su odmah dostupne bez potrebe za dodatnom navigacijom. Funkcija za izmjenu podataka vrlo je jednostavna, pri promjeni zaporke provjerava je li ispravno unesena prijašnja zaporka te je li novo

postavljena zaporka različita od trenutne te je li duljinom prihvatljiva. Kod promjene imena, provjerava je li uneseno ime različito od trenutnog te ako je uspješno promijenjeno ime, podaci u lokalnoj pohrani se izmjenjuju. Funkcija za odgađanje rezervacije pritiskom na gumb briše podatke o rezervaciji pomoću ID-ja rezervacije.

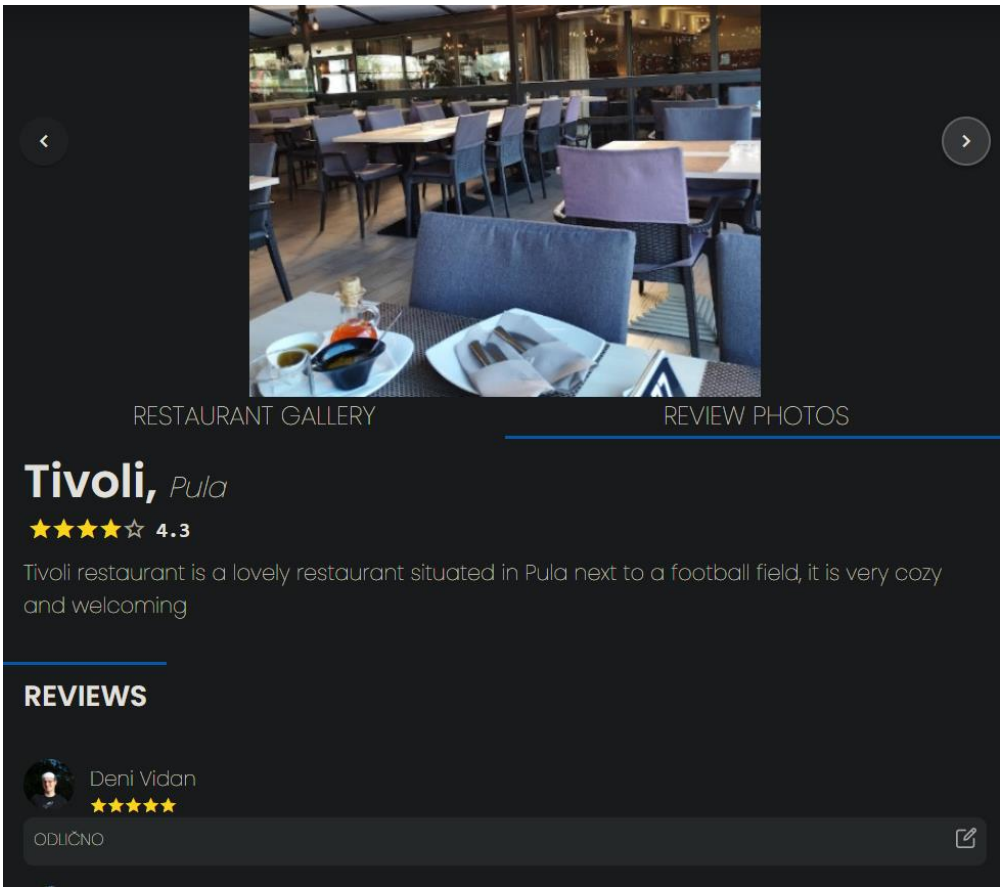


Slika 34. Stranica korisničkog profila

6.3.6 Stranica restorana

Kako bi korisnici imali što bolji uvid u atmosferu i iskustvo pojedinog restorana, na stranici svakog restorana prikazuje se galerija fotografija. Restoran može sam odabrati i postaviti set slika koje predstavljaju njihov prostor, hranu i osoblje u najboljem svjetlu. Ove fotografije daju dojam samog restorana i njegove ponude. Osim tih profesionalnih fotografija, korisnik jednostavnim klikom može pregledati i sve fotografije korisničkih recenzija. Ove autentične fotografije drugih gostiju pružaju uvid u stvarnu atmosferu i kvalitetu. Ispod galerije prikazane su recenzije koje sadrže tekstualnu recenziju te recenziju u obliku ocjene. Kombiniranjem profesionalnih i korisničkih fotografija, korisnicima se pruža najbolji vizualni dojam o restoranu i mogućnost da stvore realnu sliku o čemu mogu očekivati. Galerija fotografija korisnicima ovako pomaže u informiranoj odluci odabira restorana prikazano na Slika 35. Korisnik na jasno označeni gumb može ostaviti recenziju, da bi se spriječila zlouporaba objave

neželjenih recenzija implementirana je funkcija prikazana na Slika 37 i Slika 38 koja provjerava je li recenzija ostavljena unutar posljednjih 30 minuta, ako je ta tvrdnja točna, moguće je samo urediti i izbrisati recenziju, a ako ta tvrdnja nije točna, korisnik može ostaviti novu recenziju. Također postoji funkcija na Slika 36 koja provjerava je li korisnik ocijenio restoran, ako je ocijenio, recenzije za tog korisnika su onemogućene za taj dan.



Slika 35. Stranica odabranog restorana

```
async checkDate() {
  let currentTime = new Date().toISOString().split("T")[1].split(".")[0];
  let currentDate = new Date().toISOString().split("T")[0];
  if (this.review_date != currentDate) {
    this.isReviewable = true;
  } else {
    this.isReviewable = false;
  }
},
```

Slika 36. Funkcija za provjeru podudarnosti dana

```
passedTime(startDate, endDate) {
  const start = new Date(startDate);
  const end = new Date(endDate);
  const differenceInMilliseconds = Math.abs(end - start);
  const minutesPassed = Math.floor(differenceInMilliseconds / (1000 * 60));
  return minutesPassed;
},
```

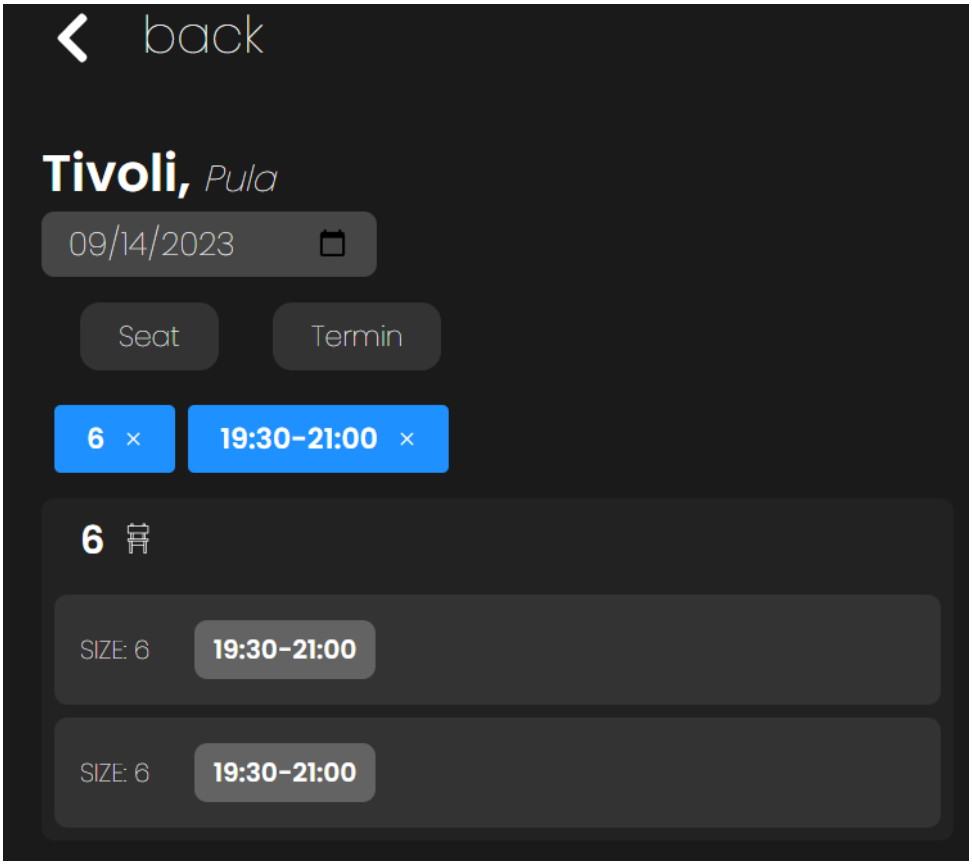
Slika 37. Funkcija za provjeru prolaska 30 minuta od zadnjeg uređivanja

```
async checkTime() {
  let currentTime = new Date().toISOString().split("T")[1].split(".")[0];
  let currentDate = new Date().toISOString().split("T")[0];
  let result = this.passedTime(
    new Date().toISOString(),
    await this.res.slice(-1)[0].date_time
  );
```

Slika 38. Funkcija koja poziva „passedTime()“ funkciju

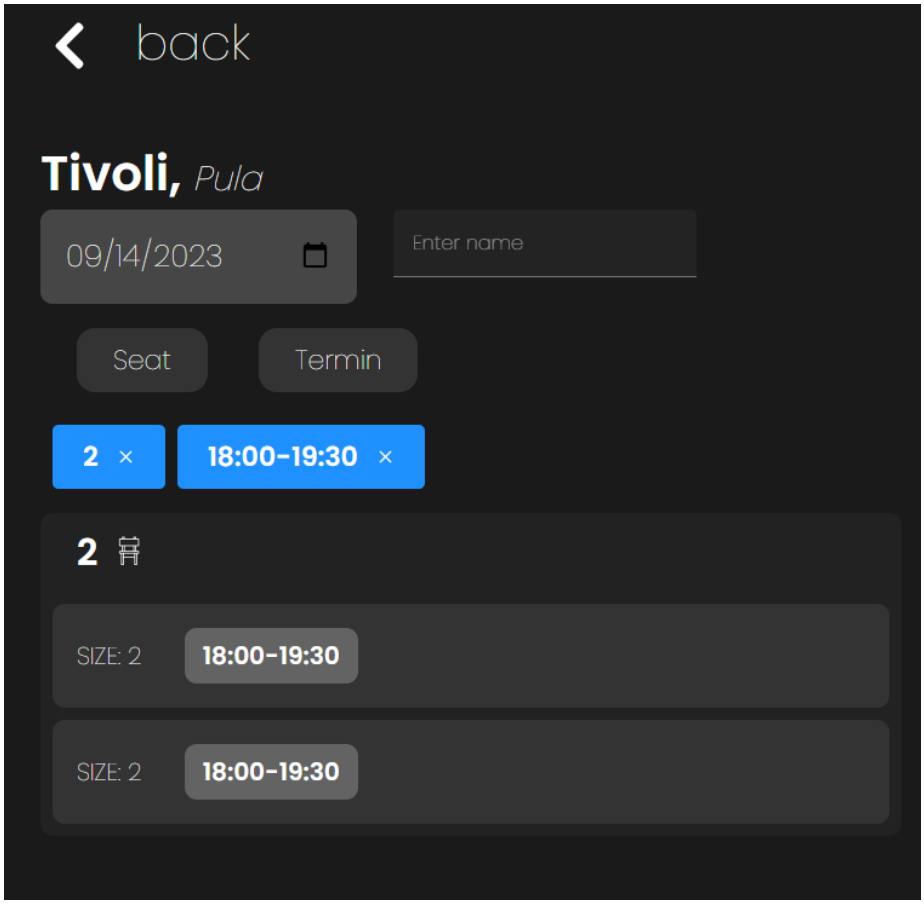
6.3.7 Stranica za rezervaciju

Proces rezervacije stola u restoranu putem aplikacije odvija se u nekoliko koraka. Prvi korak je da korisnik odabere filtere prema svojim preferencama - definira veličinu stola te željeni datum i vrijeme dolaska na večeru jasno vidljivo sa Slika 39. Ovi filteri služe da se suzi izbor i prikažu samo oni termini koji odgovaraju korisnikovim potrebama. Nakon što korisnik postavi filtere, dobiva pregled svih slobodnih termina koji zadovoljavaju unesena ograničenja. Iz tih ponuđenih opcija korisnik bira onaj termin koji njemu najviše odgovara. Sljedeći korak je slanje zahtjeva za rezervaciju odabranog termina restoranu putem aplikacije. Restoran zatim mora potvrditi taj zahtjev kako bi rezervacija bila finalizirana. Kada restoran prihvati zahtjev i time osigura rezervaciju, korisnik dobiva obavijest putem emaila kao službenu potvrdu da mu je stol rezerviran u traženo vrijeme.



Slika 39. Stranica za rezervaciju restorana od strane krajnjeg korisnika

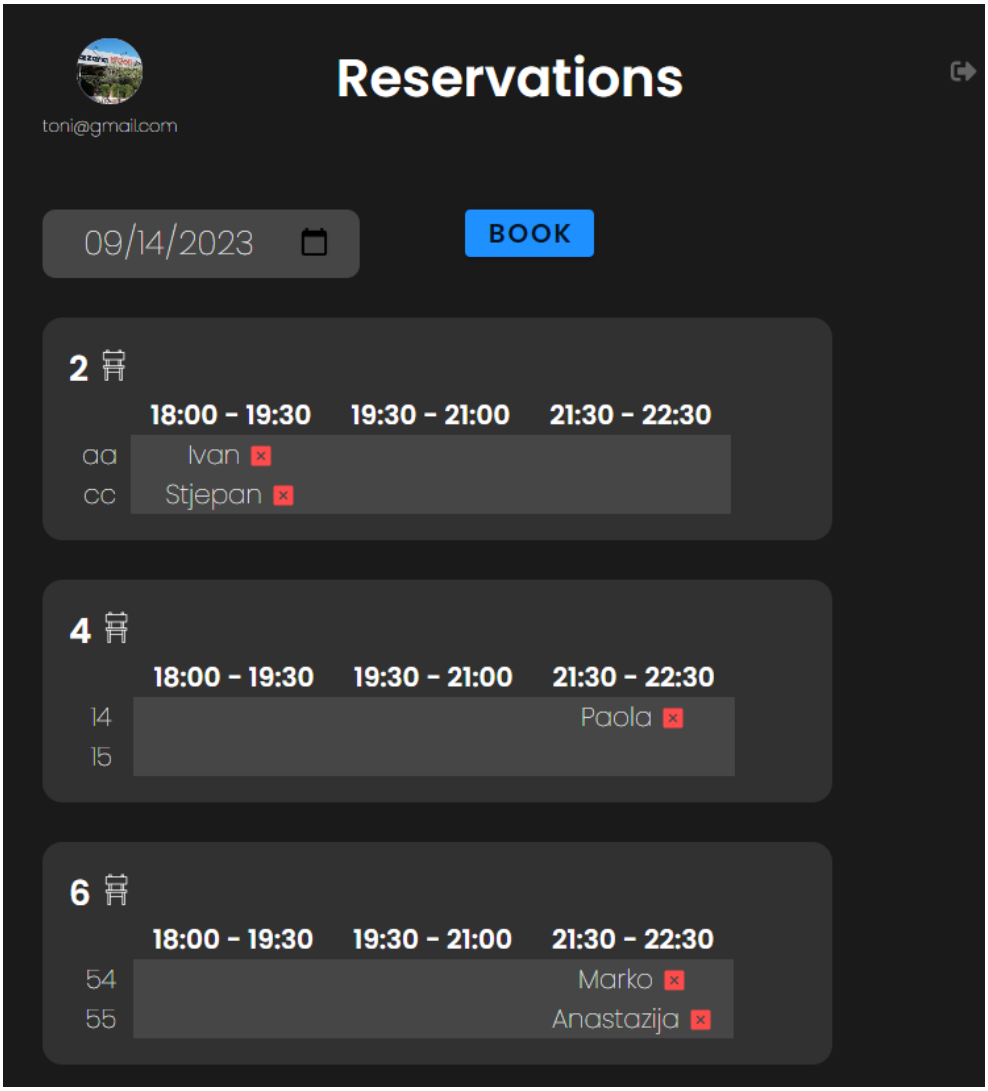
Osim online rezervacija od strane krajnjih korisnika, aplikacija omogućava i restoranima da samostalno unose rezervacije za goste koji su rezervirali osobno/telefonom. U tom slučaju, restoran ima opciju ručnog unosa nove rezervacije preko sučelja za administratore. Prilikom unosa, osim odabira broja osoba i željenog termina, unosi se i ime gosta za kojeg se radi rezervacija, prikazano na Slika 40. Time se omogućava jednostavno evidentiranje svih rezervacija na jednom mjestu u aplikaciji, bez obzira jesu li napravljene online ili osobnim dolaskom gostiju u restoran.



Slika 40. Stranica za rezervaciju restorana od strane administratora restorana

6.3.8 Raspored rezervacija

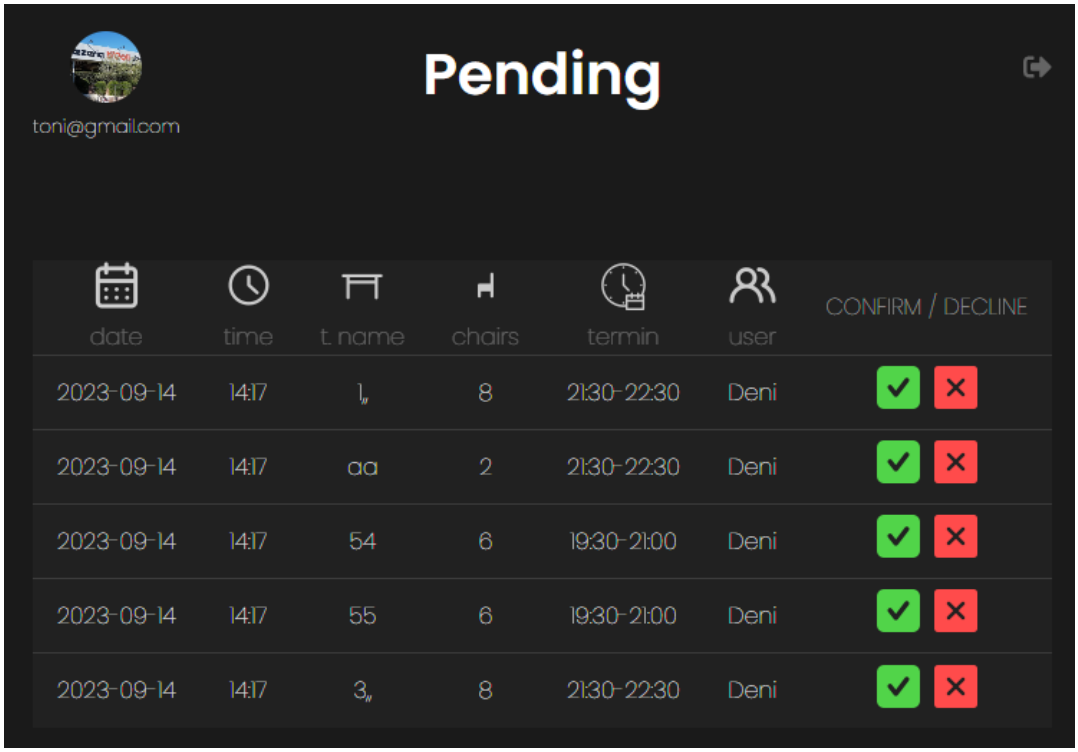
Raspored i pregled rezervacija u aplikaciji je implementiran na pregledan i intuitivan način. Rezervacije su grupirane u tablice po veličini stolova, posebno su prikazani stolovi za 2 osobe, 4 osobe, itd. Unutar svake tabele, svaki termin u danu i svaki stol imaju vlastito rezervacijsko polje. Ako je polje prazno, to znači da je taj termin slobodan i stol je dostupan za rezervaciju. Kada se napravi rezervacija, ime gosta se ispisuje u odgovarajuće polje termina i stola vidljivo sa Slika 41, čime se vizualno označava da je taj termin zauzet. Administrator ima mogućnost otkazivanja rezervacije tako da pored imena gosta klikne na ikonu "X". Time se to polje ponovno oslobađa i stol postaje dostupan. Ovakav jednostavan prikaz omogućava brz i lagan pregled zauzetosti svih stolova u svakom trenutku. Administrator uvijek ima potpun uvid u slobodne termine i može brzo otkazati rezervaciju ako je potrebno.



Slika 41. Stranica za upravljanje rezervacijama

6.3.9 Rezervacije na čekanju

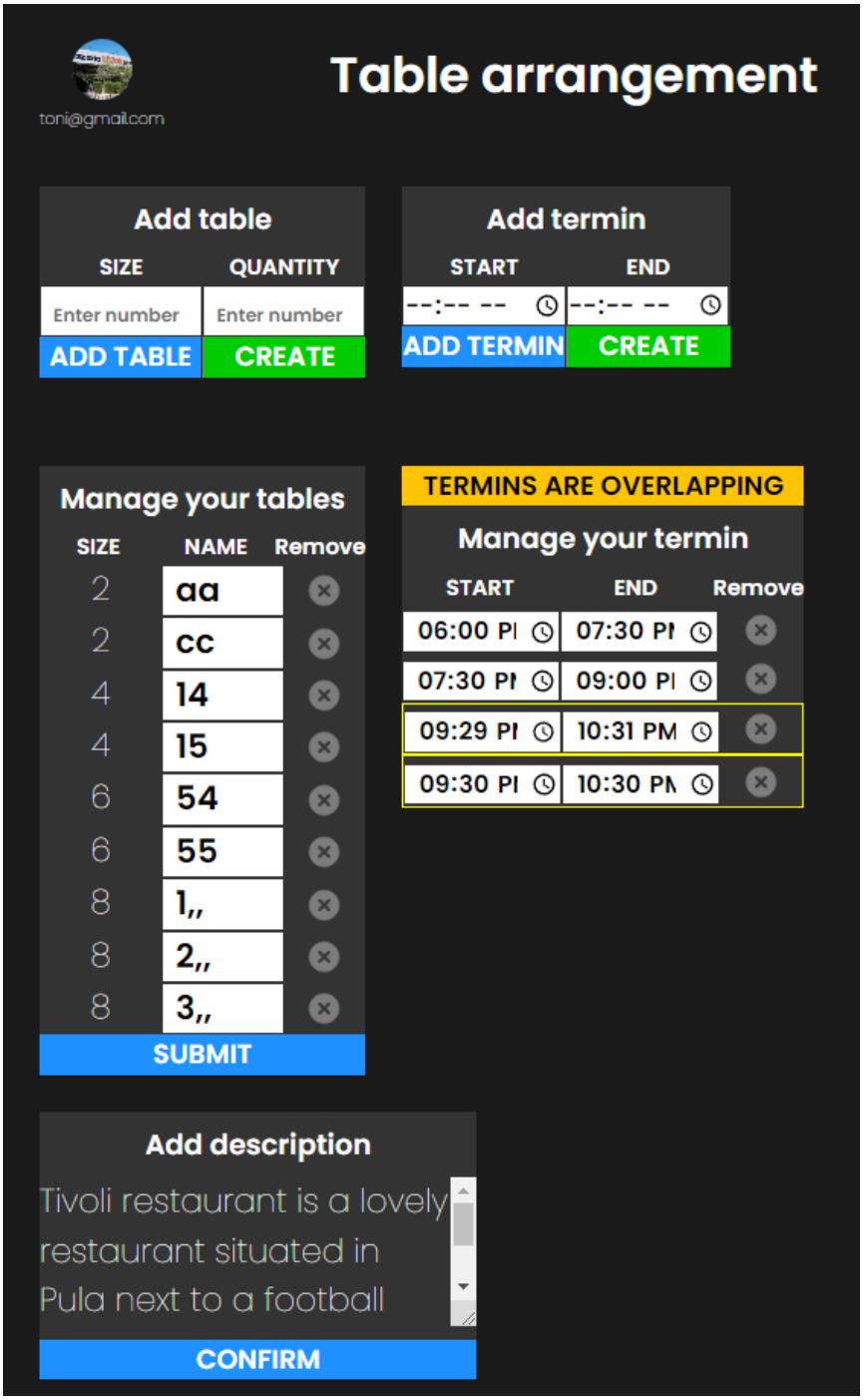
Stranica za pregled i potvrđivanje rezervacija implementirana je kao Vuetify tablica, sortirana kronološki od najstarijih prema najnovijim zahtjevima što je prikazano na Slika 42. U tablici su prikazani svi ključni podaci o svakoj rezervaciji, datum, naziv i veličina stola, odabrani termin te ime korisnika. Ova pregledna stranica optimizirana je za brzo i jednostavno prihvaćanje ili odbijanje zahtjeva za rezervacijom. Čim administrator potvrdi ili odbije rezervaciju, promjena statusa se trenutno ažurira bez kašnjenja. Fokus je stavljen na preglednost i efikasnost kako bi se olakšao tijek prihvaćanja rezervacija.



Slika 42. Stranica za prihvaćanje ili odbijanje rezervacija u tijeku

6.3.10 Postavljanje stolova, termina te opisa restorana

Restoran ima zasebnu administracijsku stranicu, prikazanu na Slika 43, za unos podataka o stolovima i dostupnim terminima. Za unos stolova postoje dva polja, jedno za veličinu stola, drugo za količinu stolova te veličine. Nakon unosa veličine i broja stolova, klikom na "Add table" ti podaci se spremaju i dodaje se novo unosno polje za sljedeći stol. Kad su svi stolovi uneseni, klikom na "Create" šalje se zahtjev za unos svih stolova u bazu. U tablici ispod prikazuju se uneseni stolovi gdje administrator dodjeljuje naziv svakom stolu. Slično se unose i termini, bira se vrijeme početka i završetka termina. Ako se unese preklapajući termin, dobije se upozorenje i termini se vizualno označavaju.



Slika 43. Stranica za upravljanje stolovima, terminima i opisom restorana

Funkcija checkOverlap() sa Slika 44 služi za provjeru preklapaju li se uneseni termini restorana. Prvo se prolazi petljom kroz sve termine i postavlja varijablu isOverlapping na vrijednost „false“ za svaki termin. Zatim se radi ugniježdena petlja kroz sve termine, za svaki „termin1“ uspoređuje se sa svakim drugim „terminom2“. Vrijeme početka i

završetka za oba termina se pretvara u minute radi lakše usporedbe. Provjerava se preklapaju li se vremena dvaju termina:

- Ako je početak termina1 prije završetka termina2 i završetak termina1 je poslije početka termina2, znači da se termini preklapaju.
- U tom slučaju, varijabla „isOverlapping“ se postavlja na vrijednost „true„ za oba termina.

Nakon petlje, provjerava se postoji li barem jedan preklapajući termin. Ako postoji, ispisuje se poruka o preklapanju i funkcija vraća vrijednost „true“. Inače vraća vrijednost „false“.

Ovime se omogućuje detekcija preklapajućih termina prilikom unosa u administracijskom sučelju.

```
checkOverlap(termins) {
  termins.forEach((termin) => {
    termin.isOverlapping = false;
  });
  termins.forEach((termin1, index1) => {
    termins.forEach((termin2, index2) => {
      if (index1 !== index2) {
        const startTime1 = this.getTimeInMinutes(termin1.start_time);
        const endTime1 = this.getTimeInMinutes(termin1.end_time);
        const startTime2 = this.getTimeInMinutes(termin2.start_time);
        const endTime2 = this.getTimeInMinutes(termin2.end_time);
        if (startTime1 < endTime2 && endTime1 > startTime2) {
          termin1.isOverlapping = true;
          termin2.isOverlapping = true;
        }
      }
    });
  });
  const overlapDetected = termins.some((termin) => termin.isOverlapping);
  if (overlapDetected) {
    this.message = "TERMINS ARE OVERLAPPING";
    return true;
  } else {
    this.message = "";
    return false;
  }
},
```

Slika 44. Funkcija za provjeru preklapanja termina pojedinog restorana

7. Zaključak

Razvoj web aplikacija za olakšavanje svakodnevnih aktivnosti postao je iznimno tražen u moderno doba. Jedna od uobičajenih aktivnosti koja se seli na web je rezervacija stolova u restoranima. Tradicionalni načini rezerviranja telefonom ili e-mailom mogu biti spori i neefikasni za korisnike i vlasnike restorana. Ovaj rad je istražio inovativnu web aplikaciju "Cutlery" koja nudi rješenje problema rezerviranja stolova kroz intuitivno online sučelje. Aplikacija koristi moderni Vue.js programski okvir i Express.js poslužitelj te SQLite bazu podataka. Implementirano je sučelje za korisnike koje omogućuje lako pretraživanje i rezerviranje stolova uz informacije poput recenzija i fotografija restorana. Vlasnicima restorana je omogućeno jednostavno postavljanje rasporeda i upravljanje rezervacijama. Sučelje za korisnike ističe se preglednošću i brzinom rezervacije u nekoliko koraka. Vlasnici restorana dobivaju alate za učinkovito upravljanje stolovima i lakšu koordinaciju rezervacija. Iako postoji prostor za daljnja poboljšanja, ova aplikacija predstavlja korak naprijed u modernizaciji procesa rezervacije stolova. Njena glavna prednost je jednostavnost korištenja i brzina rezerviranja za obje strane. Intuitivnost sučelja i optimizacija tijeka rezervacije čine ovo rješenje privlačnim. Daljnjim razvojem i širenjem baze korisnika i restorana, ovaj koncept ima potencijal postati vodeće rješenje za rezervacije. Investicija u digitalizaciju ovog procesa isplativa je za sve uključene strane. Restorani dobivaju alat za efikasnije poslovanje i veću popunjenost kapaciteta. Korisnici ostvaruju uštedu vremena i eliminiraju frustracije tradicionalnih načina rezerviranja. Aplikacija "Cutlery" dobar je primjer kako se modernom tehnologijom može unaprijediti uobičajena aktivnost na dobrobit korisnika i biznisa. Ovakva rješenja pokazuju snagu digitalizacije u pojednostavljivanju svakodnevnih zadataka i poboljšanju korisničkog iskustva u raznim sferama života.

LITERATURA

[1] Nodemailer usage. Preuzeto 16.4.2023. sa https://nodemailer.com/usage/

[2] Introduction — Vue.js. Preuzeto 10.2 2023. sa https://vuejs.org/guide/introduction.html

[3] Express.js home page. Preuzeto 10.2 2023. sa https://expressjs.com

[4] Node.js — About Documentation. Preuzeto 10.2.2023. sa https://nodejs.org/en/docs

[5] Node.bcrypt.js — Bcrypt. Preuzeto 25.3.2023. sa https://www.npmjs.com/package/bcrypt

[6] Getting Started — Axios. Preuzeto 10.2.2023. sa https://axios-http.com/docs/intro

[7] Getting Started — Sqlite. Preuzeto 11.2.2023. sa https://www.sqlite.org/quickstart.html

[8] Get started with Vuetify 3 — Vuetify. Preuzeto 10.2.2023. sa https://vuetifyjs.com/en/getting-started/installation/

[9] Table Manager — The best online reservation system for Restaurants. Preuzeto 20.6.2023. sa https://www.tablemanager.be/en/

[10] Open Table — Find your table for any occasion. Preuzeto 20.6.2023. sa https://www.opentable.com/

[11] Eat App — The all-in-one reservation platform. Preuzeto 20.6.2023. sa https://eatapp.co/

POPIS SLIKA