

UNIVERZITET U BIHAĆU

TEHNIČKI FAKULTET

Elektrotehnički odsjek

Smjer informatika

PROJEKTNI ZADATAK IZ PREDMETA

Napredne tehnike internet programiranja

TEMA:

Web sistem za personalizirani dnevnik recepata sa dodatkom REST servisa

Profesor: prof.dr.Ognjanović Ivana

Student: Melkić Nejra 958

Student: Ivušić Ivan 983

Bihać, septembar 2020.

SADRŽAJ

| | |
|---|-----|
| UVOD | 1. |
| 1. OPIS I CILJ SISTEMA | 2. |
| 2. BLOKOVSKA STRUKTURA SISTEMA | 3. |
| 2.1. Modeli | 3. |
| 2.2. Kontroleri | 5. |
| 2.3. Pogledi | 5. |
| 3. CILJEVI I ZADACI SOFTVERSKOG INŽENJERSTVA | 6. |
| 3.1. Metodologija razvoja sistema | 6. |
| 3.2. Pregled korištenih alata | 7. |
| 3.2.1. Sublime Text | 8. |
| 3.2.2. Visual Studio Code | 8. |
| 3.2.3. Node JS | 8. |
| 3.2.4. MongoDB | 10. |
| 4. OPIS SLUČAJEVA KORIŠTENJA PODSISTEMA | 12. |
| 5. ANALIZA PODSISTEMA | 18. |
| 5.1. Sekvencijalni dijagrami | 18. |
| 6. IMPLEMENTACIJA PODSISTEMA | 23. |
| 6.1. Neregistrirani korisnici | 23. |
| 6.1.1. Početna stranica | 24. |
| 6.1.2. Stranica „O nama“ | 25. |
| 6.1.3. Stranica „Kontakt“ | 26. |
| 6.1.4. Galerija recepata | 28. |
| 6.1.5. Pretraga recepata | 29. |
| 6.1.6. Registracija | 32. |
| 6.1.7. Prijava | 34. |
| 6.2. Registrirani korisnici | 36. |
| 6.2.1. Izmjena korisničkih informacija | 36. |
| 6.2.2. Zaboravljena šifra | 37. |
| 6.2.3. Prikaz jednog recepta | 39. |
| 6.2.4. Spremanje recepta | 43. |
| 6.2.5. Pregled i izmjena spremljenih recepata | 45. |
| 6.2.6. Brisanje recepata | 48. |
| 6.2.7. Pregled planova ishrane | 49. |
| 6.2.8. Spremanje planova ishrane | 50. |
| 6.2.9. Brisanje plana ishrane | 57. |
| 6.2.10. Odjava | 59. |
| ZAKLJUČAK | 60. |
| LITERATURA | 61. |

UVOD

Pismeni dio ispita iz predmeta Napredne tehnike internet programiranja se sastoji od projektovanja infomacionog sistema, te dokumentovanja specifikacije istog. U sklopu dokumentacije opisan je informacioni sistem “Recipely”, koji omogućuje uvid u različite recepte, dostupne u bazi podataka. Recipely nije stvarna organizacija, već je osmišljena radi prezentovanja podataka sa servisa Spoonacular. Dakle, svi podaci o receptima i planovima ishrane, koji se prikazuju na stranici Recipely, su posuđeni od strane web REST servisa *Spoonacular*, (<https://spoonacular.com/>) koji ima veliku bazu podataka, koje se preuzimaju putem oficijalne API stranice ove organizacije (<https://spoonacular.com/food-api/>).

Korištena programska okruženja u toku razvijanja projekta su *Sublime Text*, kao pomoćno okruženje, *Visual Studio Code*, kao okruženje za implementaciju kôda, *Node JS*, kao okruženje za kreiranje kôda, te *MongoDB*, kao okruženje za bazu podataka. Za izradu UML dijagrama (dijagrami korištenja, slijeda) korišten je program StarUML.

1. OPIS I CILJ SISTEMA

Cilj ovog projektnog zadatka je implementacija informacionog sistema, koji će omogućiti uvid u različite recepte koji su dostupni na stranici. Svrha je omogućiti korisniku stvaranje vlastitog dnevnika recepata, kao i planova ishrane. Svi podaci o receptima i planovima ishrane se prikupljaju sa servisa: <https://spoonacular.com/food-api/> i spremaju u online bazu.

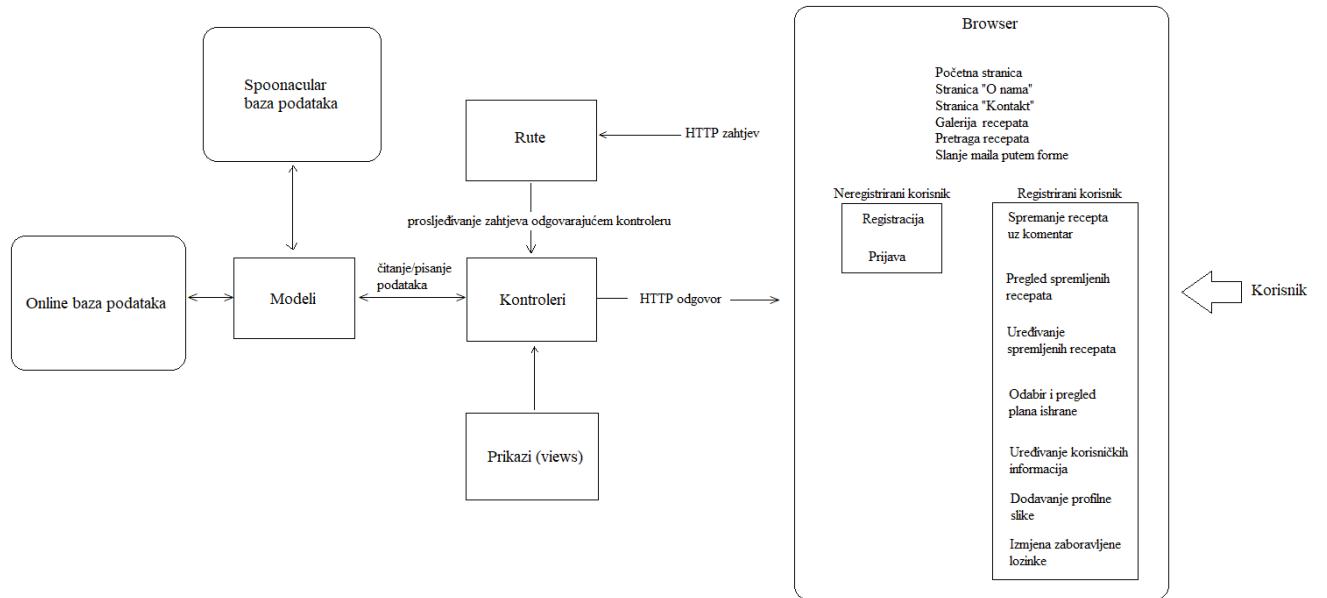
Postoje dvije vrste korisnika: *neregistrirani* i *registrirani* korisnici. Stranica ne zahtijeva administratorsku ulogu, budući da registrirani korisnici samostalno spremaju, uređuju i brišu recepte (uz dodatak spremanja planova ishrane).

Neregistrirani korisnici mogu pregledati sve recepte na stranici. Osim toga, korisnici mogu pretražiti bazu svih recepata prema željenoj riječi, kao i ograničiti pretragu prema vrsti kuhinje i prema tipu ishrane. Ovi podaci su dostupni zahvaljujući korištenju API-a Spoonacular. Svi neregistrirani (kao i registrirani) korisnici imaju mogućnost da organizaciji „Recipely“ šalju mail u određenoj formi koji se nalazi na „Kontakt“ stranici. Unutar mail forme, korisnik unosi ime, e-mail, broj telefona i poruku koja će se poslati na mail organizacije. Ove opcije su omogućene korištenjem API-a SendGrid.

Da bi mogli rukovati naprednjim opcijama (kao što su spremanje recepta, pregled i uređivanje spremljenih recepata, prikaz i odabir planova ishrane i sl) korisnici se moraju registrirati. Pri registraciji su obavezna polja kao što su: korisničko ime, lozinka, ime, prezime, e-mail adresa. Kada se registruje, korisnik može dodati i korisničku sliku, ali i promijeniti svoje korisničke podatke. Ukoliko korisnik zaboravi vlastitu šifru, moguće je da se šifra ponovo postavi, korištenjem API-a SendGrid.

2. BLOKOVSKA STRUKTURA SISTEMA

Aplikacija radi na osnovu *MVC (Models-Views-Controllers)* strukture, što znači da se sastoji od ove tri osnovne karakteristike, koje se povezuju *rutama*. Modeli sadrže osnovne osobine pojedinih objekata, kontroleri sadrže osnovne funkcionalnosti tog objekta, dok pogledi/views sadrže *html+css(+js)* kôd, koji će ujediniti osobine i funkcionalnosti objekta. Blokovska MVC struktura sistema prikazana je na slici 3.1.



Slika 2.1. Blokovska struktura sistema

Na slici 2.1 se vidi da se stalno vrši komunikacija među svim komponentama MVC strukture. Svaki put kada se na određenu rutu pošalje HTTP zahtjev, on se proslijeđuje odgovarajućem kontroleru. Na primjer, ako se na rutu /me pošalje HTTP zahtjev GET, proslijeđuje se kontroleru userController. Tada kontroler čita podatke iz odgovarajućeg modela (entiteta u bazi podataka). Kontroler čita podatke iz User.js i šalje ih kroz prikaz (view) na Browser kao odgovor. Preko Modela se vrši komunikacija između kreirane online baze i API baze podataka. Korisnici na Browseru mogu otvoriti početnu stranicu, „o nama“, „kontakt“, galeriju recepata i pretražiti recepte. Ukoliko je korisnik neregistriran, može se prijaviti ili registrirati. Ukoliko je korisnik već registriran, omogućena mu je lista opcija koja je prikazana na slici 2.1. Detaljnija funkcionalnost modela, kontrolera i pogleda je objašnjena u ostatku poglavlja.

2.1. Modeli

U sklopu MVC-a, **model** se kreira za svaki entitet u bazi. Entitet unutar baze podrazumijeva objekat, odnosno događaj. Svaki entitet ima atribute, tj. svojstva ili obilježja entiteta. Dakle, jedan od modela (entiteta) u projektu će biti *User* i on će imati atribute *ime*, *prezime*, *e-mail* i slično. Svaki entitet ima više instanci [1]. Jedna od mnogobrojnih instanci pomenutog modela/entiteta može biti npr. *User.ime = Nejra*, *User.prezime = Melkić*, *User.e-mail = nejramelkic@hotmail.com*.

Unutar svake baze podataka postoje veze/relacije između entiteta. Veze mogu biti 1:1, 1:N, N:M. Da bi se objasnile veze, uzet' će se nekoliko primjera entiteta nevezanih za projekat.

- Veza 1:1 podrazumijeva jednostruku vezu između dva entiteta. Uzet' će se primjer entiteta *Student* i *Indeks*. Jedan Student mora imati samo jedan Indeks, i jedan Indeks pripada samo jednom Studentu. Dakle, radi se o obostranoj 1:1 vezi.
- Veza 1:N podrazumijeva jednostruku vezu jednog entiteta prema drugom, te mnogostruku vezu drugog entiteta prema prvom. Uzet' će se primjer entiteta *Čovjek* i *Automobil*. Jedan Čovjek može imati više Automobila, dok jedan Automobil pripada samo jednom Čovjeku. Dakle: Čovjek-Automobil=1:N
- Veza M:N podrazumijeva mnogostruku obostranu vezu dva entiteta, te se ovaj oblik najčešće izbjegava, ubacivanjem trećeg, *međuentiteta*, kojim će se ova veza pojednostaviti na dvije 1:N veze. Uzet' će se primjer entiteta *Student* i *Knjiga*. Jedan Student može pročitati više Knjiga, dok jedna Knjiga može biti pročitana od strane više Studenata.

Unutar projekta postoji model *userModel*, čiji su atributi prikazani u userShemi. Model sadrži *username* i *email*, koji moraju biti unikatni, zatim *firstname* i *lastname*.

Atribut *photo* definira putanju ka slici profila tog korisnika. Dodavanje i mijenjanje slike je implementirano putem modula *multer*. Ovaj modul sliku iz forme pretvara u heksadecimalne vrijednosti, koje smješta u buffer. Nadalje, softverski, putem definisanih metoda koje dolaze u sklopu modula *sharp*, se podešavaju atributi slike, kao što su veličina, format, kvalitet formata i krajnja putanja.

Osim toga, model sadrži i atribute *password* i *passwordConfirm*. Šifra se, prije spremanja u bazu podataka, hashuje, kako bi sistem bio sigurniji u slučaju provale u bazu podataka. Model sadrži atribute *passwordChangedAt*, *passwordResetToken* i *passwordResetExpires*, koji se koriste u procesu zamjene šifre.

Korisnički podaci iz modula userModel se šalju API-u, gdje ih API sprema, te kao odgovor aplikaciji šalje novo API korisničko ime i korisničku šifru, koji će se koristiti pri komunikaciji između ova dva sistema.

Ovaj model sadrži i atriput *recepti*, koji se pojavljuje u obliku niza, čiji su elementi atributi *id* i *comment*. Unutar id atributa se pohranjuje identifikacija spremlijenog recepta, dok comment predstavlja zabilješku korisnika na taj recept. Samim time, jedan korisnik može spremiti niz recepata. Ovakva veza se naziva *umetnuta veza*. Pored ovih, model sadrži i atribut *datumiObroka*, koji se također javlja u obliku niza. Tako je ostvarena još jedna umetnuta veza sa objektima vezanim uz planove ishrane. Jedan korisnik može imati jedan ili više plan ishrane.

2.2. Kontroleri

Kontroleri u sklopu MVC strukture služe da grupišu sve funkcionalnosti odgovarajućeg modela. Dakle, sve funkcije vezane za korisnike, iz modela User, smještene su u kontroleru *UserController*. Analogno tome, odgovarajuće funkcije su smještene u odgovarajuće preostale kontrolere.. Osim njih, kreiran je i *AuthController*, unutar kojeg su smještene funkcije za login, signup, logout, protect i restrict. Može se reći da je ovaj kontroler najvažniji, jer se unutar njega vrši autentifikacija – login, signup, protect i autorizacija – restrict. Ono što autentifikacija omogućuje jeste prijavu klijenata u bazu podataka, i zaštitu pojedinih funkcija od strane neregistrovanih korisnika. Funkcija protect je nužna, budući da vrši zaštitu pojedinih dijelova aplikacije, te prosljeđuje status i podatke korisnika koji je trenutno logovan, tako da bi se oni mogli koristiti u nekim drugim funkcijama [1]. Do kontrolera se dolazi putem još jedne veoma bitne komponente MVC strukture koja se naziva *route* ili ruta. Ruta se postavlja u glavnoj aplikaciji, i kada klijent dospije na odgovarajuću rutu, poziva se GET, POST, PATCH ili DELETE HTTP metoda, ovisno o tome da li će se čitati podaci iz baze, unijeti podaci u bazu, doraditi podatke u bazi ili izbrisati podatke u bazi, respektivno.

2.3. Pogledi

Budući da se u svakom kontroleru nalazi odgovarajuća funkcija, kada se izvrši ta funkcija, ona mora vratiti određenu vrijednost. U sklopu web aplikacija, funkcije ne vraćaju vrijednosti nego poglедe, i to se naziva *renderovanje*. Pogled predstavlja web stranicu s kompletним html, css i možda js kôdom, koji će se vratiti iz pojedine funkcije i dati neku obavijest, neki rezultat iz baze, upisati nešto u bazu, i slično, ovisno o tipu funkcije [1]. Ukoliko se želi izvršiti pregled svih recepata, odlazi se na rutu /receipts. Ona poziva GET metodu, unutar koje se nalazi precizirana funkcija unutar receptControllera, koja se naziva getAllReceipts, i koja kupi 12 nasumičnih recepata iz baze. Kao return vrijednost ove funkcije, renderuje se *view*, s odgovarajućim html+css kôdom, koji omogućuje da se pojave kartice s podacima.

3. POJAM SOFTVERSKOG INŽENJERSTVA

Softversko inženjerstvo je sistematska primjena inženjerskih pristupa razvoju softvera i smatra se dijelom cjelokupnog sistemskog inženjerstva. Softversko inženjerstvo je inženjerska disciplina koja se bavi svim aspektima proizvodnje softvera i ono obuhvata znanje, alate, rukovanje te održavanje i praćenje rada softvera. Softver koji se izučava u sklopu ove dokumentacije je informacioni sistem Recipely. Informacioni sistem je kolekcija podataka i informacionih tehnologija koje su povezane radi prikupljanja, obrade, skladištenja i pružanja informacija potrebnih za organizaciju koja se njima predstavlja [2].

Strukturu softverskog inženjerstva sačinjavaju tri osnovne komponente: metode, alati i postupci (procedure). Njihovo jedinstvo definira kvalitetu razvoja, pa je zbog toga značajno da se odaberu one komponente koje se postavljenim zadacima i problemima u razvoju najlakše prilagođavaju. Donošenje prave odluke nije jednostavan zadatak, zbog individualnog i kreativnog karaktera postupka razvoja, zbog različitosti pojedinih sustava za koji se razvija softver i različitog okruženja sustava, ali i zbog nepostojanja „recepata“ za izbor.

Tokom godina razvoja softvera, planska izrada sa svom potrebnom dokumentacijom koja uključuje punu specifikaciju i izradu modela se pokazala kao vrlo pouzdana i dobra. Međutim, ubrzani razvoj tehnologije, sve veća potreba tržišta i u novije vrijeme ogromna potreba za brzinom izbacivanja novog i nadograđenog softvera čine ovu staru i pouzdanu metodu neučinkovitom. Zbog toga je došlo do promjene pogleda na razvoj softvera. Upravo potreba za brzinom i fleksibilnošću dovela je do zamjene starog planskog razvoja sa novim metodama razvoja softvera [2].

3.1. Metodologija razvoja sistema

Metodologija razvoja softvera zapravo podrazumijeva pristup ili način, tok, specifičnosti i ostalo u vezi procesa, načina osmišljavanja i komercijalizacije softvera. U današnjici su razvijene različite metode, odnosno pristupi razvoja softvera, a pri njihovu razmatranju riječ je o analizi metodologija izrade softvera. Može se tvrditi kako su u današnjici najprimjenjivanije i najraširenije one metodologije koje se ističu kvalitetom i prilagodljivošću, a optimalno zadovoljavaju potrebe sudionika [2].

Razvoj softvera složen je posao koji se sastoji od velikog broja koraka, počevši od početnog zahtjeva korisnika, preko analize i izgradnje, pa do implementacije i korištenja. Razvoj softvera podrazumijeva korištenje većeg broja različitih tehnologija i tehnika. Metodologija razvoja sistema je standardizovani proces koji definiše skup metoda i aktivnosti koje se koriste za razvoj i usavršavanje informacionog sistema [2].

Metodologija koja je korištena u ovom radu ima tri koraka:

1. *Opis slučajeva korištenja*
2. *Analiza*
3. *Implementacija*

Metodologija razvoja sistema podrazumijeva modelovanje sistema. Modelovanje sistema se koristi zbog složenosti stvarnih sistema. Model je pojednostavljeni prikaz sistema, koji se može postaviti u određenu računarsku simulaciju, što košta znatno manje od eksperimentisanja sa stvarnim sistemima. Da bi se model kreirao potrebno je najprije projektirati željeni model sistema. Prilikom projektiranja treba odlučiti o komponentama sistema, načinu njihovog vezivanja i funkcionalnoći. Postupci projektiranja mogu biti neformalni ili formalni [3].

Neformalni postupci ne ulaze u finalnu dokumentaciju, ali služe projektantu da nabrinu nacrta odgovarajuću skicu, napiše tekstualni opis ili skicu dijagrama stanja. Neformalni postupci omogućuju jednoj osobi brži pristup do rješenja korištenjem (samo) njoj prikladne metode. Skica koju je projektant sebi na brzinu nacrtao (npr. rukom na papiru) možda će mu omogućiti da bolje vizualizira problem i riješi ga brže nego da primjerice za isto koristi UML sekvensijalni dijagram za koji će mu trebati puno više vremena da ga nacrtava [3].

Problemi neformalnih postupaka su upravo u njihovoј neformalnoј strukturi koja se na različite načine može interpretirati i dovesti do problema. Ipak, ako se koriste samo kao pomoćno sredstvo pri rješavanju dijela problema mogu značajno olakšati razumijevanje problema ili predloženog rješenja. Formalni postupci projektiranja mogu biti različiti, ali se najčešće koriste UML dijagrami. Osim UML dijagrama, za analizu i ispitivanje mogućih rješenja prikladne mogu biti Petrijeve mreže i vremenske Petrijeve mreže. Pri pojedinačnoj analizi (početku analize) mogu se koristiti i neformalne skice, ali bi one prije ulaska u službenu dokumentaciju (i dijeljenja s ostatom tima) ipak trebale biti prevedene u neki formalni oblik, a da se izbjegne mogućnost drugačije interpretacije [3].

Formalni postupci koji će biti priloženi u nastavku su UML dijagrami. UML dijagram je skraćenica iz engleskog jezika koja se sastoji od riječi Unified (ujedinjuje sve postojeće notacije), Modelling (koristi se za modelovanje softverskih elemenata) i Language (sredstvo komunikacije). UML je grafički jezik za vizuelizaciju, specifikaciju, konstruisanje i dokumentovanje sistema programske podrške [3].

Složenom sistemu je najbolje pristupiti putem skupa gotovo neovisnih pogleda na model, koji pokazuju posebne aspekte sistema. Pogledi su aprstrakcije koje se sastoje od dijagrama. Svaki dijagram prikazuje određeni pogled na određeni dio sistema [3]. U projektnoj dokumentaciji će se iscrtavati sljedeći pogledi: pogled na primjenu (dijagram korištenja), pogledi na interakciju : dijagram slijeda (sekvensijalni) i pogled na ponašanje (dijagram akcije).

3.2. Pregled korištenih alata

Korištena programska okruženja u toku razvijanja projekta su *Sublime Text*, kao pomoćno okruženje, *Visual Studio Code*, kao okruženje za implementaciju kôda, *Node JS*, kao okruženje za kreiranje kôda, te *MongoDB*, kao okruženje za samostalno kreiranu online bazu podataka. U nastavku će se dati opis i funkcionalnost svakog od pomenutih okruženja.

3.2.1. Sublime Text

Za razvoj aplikacije korišteno je nekoliko programskih okruženja. Najprije su se osnovni elementi stranice (početna stranica, kontakt stranica, „o nama“ stranica, galerija i sl.) dizajnirali u programu Sublime Text, koji omogućuje razvoj *html* i *css* kôda. Ovaj program je preuzet sa web stranice <https://www.sublimetext.com/3> i instaliran na Windows 64bit verziji. U ovom programu su se kreirale stranice kao što je *Početna*, *Kontakt*, *O nama*, koje su se koristile u priloženoj specifikaciji. Tokom kreiranja stranice *Galerija*, *Profil*, *Login* i slično, također je korišten ovaj program, te bi se *html* i *css* kôd prekopirao u Visual Studio Code. Ovim je olakšano dizajniranje interfejsa informacionog sistema Recipely.

3.2.2. Visual Studio Code

Za detaljniji razvoj aplikacije korišteno je okruženje Visual Studio u kombinaciji s Node JS, zajedno s nerelacijskim okruženjem MongoDB za razvoj baze podataka. Aplikacija Visual Studio je korištena kao okruženje za razvoj projekta. Unutar aplikacije Visual Studio može se dobiti uredan pregled svih foldera koji se nalaze u projektu, pročitati i urediti cijelokupan kôd, te pokretati projekat putem terminala. Dakle, za cijelokupan razvoj i implementaciju projekta, korišteno je okruženje Visual Studio Code, koje podržava rad MVC strukture, koja će biti detaljnije opisana u narednom podnaslovu.

3.2.3. Node JS

Za pisanje kôda u sklopu projekta korištene su *node* skripte. Node JS je okruženje koje omogućuje da se *javascript* fajlovi pokreću unutar specifičnog *Node servera* zahvaljujući virtualnoj mašini V8, koja konvertuje javascript kôd u mašinski, koji kompjajler razumije. Dosad je javascript kôd importovan u *html* fajl da bi se izvršio u sklopu Browsera (Mozilla, Chrome, Explorer,...), ali okruženje Node JS omogućuje da se javascript kôd pokreće i izvan samog browsera. Skripte Node JS-a su kreirane pomoću *.js* ekstenzije, tako da svaki fajl u projektu (osim *view fajlova*) ima ekstenziju *.js*. Node skripte se u terminalu pokreću korištenjem naredbe *node* [1].

U nastavku će ukratko biti opisane pojedine karakteristike node skripte. Ove karakteristike je bitno poznavati da bi se uspješno moglo koristiti node skripte.

Moduli

Cjelokupni *node* sistem radi na osnovu tzv. *modula*. Postoje ugrađeni i korisnički definirani moduli. Ugrađeni moduli omogućuju uključivanje raznovrsnih vanjskih biblioteka u projekat, kao što su biblioteke za povezivanje s bazom podataka, kreiranje web servera, i slično. Da bi se moduli koristili u node skriptama, moraju se na početku svake i *importovati* korištenjem naredbe *require*. Također, svaki *js fajl* u sklopu projekta se može posmatrati kao korisnički definiran modul. Da bi se on mogao *importovati* u drugom fajlu, mora se i *exportovati*, što se radi naredbom *module.exports*. Najbolji paket modula je *npm* paket, koji se instalira unutar terminala. Najprije se inicijalizira npm naredbom *npm init*, koja se upisuje u terminal, i ona generiše fajl *package.json*, koji će sadržavati sve informacije o modulima koji se koriste u

projektu. [1] Nakon što je obavljena inicijalizacija, mogu se instalirati svi npm moduli koji su potrebni, naredbom *npm install ime-paketa*. Na taj način su instalirani mnogi moduli koji su korišteni u sklopu projekta, kao naprimjer *cookie-parser*, koji je korišten da bi se dobili podaci iz kolačića, ili naprimjer *nodemailer*, koji je korišten za slanje i primanje mailova.

Node server

Da bi se kreirao *node server*, potrebno je *importovati* modul *express*. Ovaj modul omogućuje prihvaćanje zahtjeva (*requests*), slanje odgovora (*responses*) i upravljanje s *GET*, *PATCH*, *DELETE* i *POST* html zahtjevima. Za osluškivanje *servera* se koristi naredba *listen*, tako da se aplikacija na odgovarajućem portu (korišteni port za projekat je 3000) može pokrenuti [1]. Ono što je bitno naglasiti jeste da je instaliran npm modul *nodemon* koji omogućuje da se server sam resetuje prilikom bilo koje nove promjene kôda u projektu. U sklopu projekta, kreiran je glavni fajl pod nazivom *index.js*, unutar kojeg se importuju svi važniji moduli koji se koriste tokom izvršavanja kôda i pokreće server na portu 3000. Kada se pokreće projekt, zapravo se pokreće ovaj glavni fajl (node skripta), koji će pozvati sve druge node skripte na odgovarajućim rutama koje su definisane unutar njega. Osim ostalih bitnih modula, unutar ovog fajla se importuje i modul *Mongoose*, za upravljanje bazom podataka, o kojem će detaljnije biti govora u sljedećem podnaslovu [1].

Sintaksa

Standardna sintaksa koja je korištena je ES6, ili *ECMAScript 2015* koja je omogućila da sve varijable se definišu pomoću *const* (nepromjenjive vrijednosti) ili *let* (promjenjive vrijednosti). Naravno, sintaksa je drugačija od standardne *javascript* sintakse i u drugim aspektima, kao što su definisanje funkcije i njenih argumenata. [4]

Middleware

Cjelokupan rad *Node js* logike je zasnovan na tzv *middleware* funkcijama. Ovo su funkcije koje se izvršavaju između *requesta* (zahtjeva) i *response-a* (odgovora). U biti, svaka metoda (funkcija) koja je korisnički definirana u projektu funkcioniše kao *middleware*. Zahvaljujući ovim funkcijama, mogu se prenositi podaci tekućeg korisnika kroz aplikaciju, kao što su status, token i username, o čemu će detaljnije biti govora u nastavku dokumentacije. Prednost *middleware* logike leži i u tome što ona omogućuje *autentifikaciju i autorizaciju* korisnika pomoću funkcija *protect* i *restrict*, o kojima će, također, detaljnije biti govora u nastavku. Svaki *middleware* se izvršava *jedan-po-jedan*, po redu prema kojem su *definisani u kôdu*. Nakon što izvrši svoju funkciju, poziva se karakteristični argument *middleware-a* koji se naziva *next*, i koji, jednostavno, poziva idući *middleware*. Na taj način je omogućeno lančano izvršavanje istih [1].

Callback

Da bi se bolje razumio rad node skripti, u nastavku je objašnjena organizacija procesa u istima. Naime, postoji *top-level* i *non-top-level* kôd. Top level kôd je onaj koji se izvršava pod svaku cijenu, bez nekih uslova, stoga se on najčešće nalazi izvan funkcija, na početku

kôda, definisan *globalno*. Non-top-level kôd je onaj koji će biti uslovljen određenim radnjama, dakle, ne mora nužno značiti da će se uvijek izvršiti. Ovakav kôd se pohranjuje u tzv *callback funkcije*, koje se većinom obavljaju u pozadini [1].

Sinhroni i asinhroni procesi

U *Node JS* logici, postoje *sinhroni* i *asinhroni* procesi. Sinhroni procesi su takvi da, prilikom svog izvršavanja, ne dopuštaju idućem procesu da bude izvršen, budući da oni podrazumijevaju radnje koje su komplikirani. Asinhroni procesi se, nasuprot, izvode u pozadini i oni najčešće obuhvataju *callback* funkcije. Rad *Node JS* se razdvaja na *thread loop* i *event loop*. Thread loop obuhvaća teške zadatke, kao što je pristup fajlovima, uređivanje istih, i slično, dok event loop obuhvaća lagane zadatke, kao što su *callback funkcije*, pristup mreži, itd. Dakle, unutar *thread loopa* se pohranjuju sve sinhronne radnje, dok se unutar *event loopa* obavljaju sve asinhronne radnje [1].

Promise

Prilikom pisanja kôda, potrebno je obratiti pozornost na raspored asinhronih radnji, da ne bi došlo do pojave tzv. *callback pakla*. Ova pojava podrazumijeva pozivanje više *callback* funkcija unutar drugih *callback funkcija*. Da bi se ova pojava izbjegla, koriste se ***promises***. Promise obećava da će se dobiti neki podaci u budućnosti, i sprečava pozivanje tih podataka pomoću *callback funkcija*. Dakle, *promise* implementira koncept buduće vrijednosti – koja se očekuje negdje u bliskoj budućnosti. Svaki promise je u početku *pending* – sve dok ne poprimi neku vrijednost. Nakon što *promise* poprimi obećanu vrijednost, postaje *resolved* – i on ne mora nužno biti uspješan. Ukoliko nije uspješan, *resolved promise* će sadržavati odgovarajući *error* i bit će *odbijen*. U suprotnom, *resolved promise* će biti *prihvaćen*. [2] Kao što se mogu praviti lanci *callback funkcija*, mogu se i *promises* povezivati u lance pomoću oznake *.then*. Budući da su *callback funkcije* asinhronne, a *promise* je *callback funkcija*, znači da će *promise* biti asinhronog karaktera. Zbog toga, sve funkcije, koje vraćaju promise, moraju biti označene kao asinhronne pomoću naredbe *async*. Unutar jedne *async* funkcije može se pozvati onoliko *promises* koliko je potrebno, pomoću naredbe *await*. *Await* će stopirati kôd koji se trenutno izvršava sve dok *promise* ne bude *resolved*. Ovom logikom vođena je struktura svih asinhronih funkcija u sklopu priloženog projekta [1].

3.2.4. MongoDB

Okruženje koje se u projektu koristi za pohranjivanje podataka u bazu je ***MongoDB***, koje se povezuje s projektom zahvaljujući karakterističnom *npm modulu* koji se naziva *Mongoose*. Ovaj modul importovan u projekat omogućuje, kako povezivanje s bazom, tako i sve *CRUD* (*Create-Read-Update-Delete*) metode koje su nužne za upravljanje istom. Modul je importovan u glavni fajl (node skriptu) i sve što je potrebno za korištenje *CRUD* metoda u toku projekta jeste pozivanje istih u bilo kojoj drugoj *node skripti* [5].

- **Create:** Za kreiranje novih objekata u bazi potrebno je koristiti Mongoose metodu `create`, koja uzima podatke iz *resolved promise-a* i pohranjuje ih u novu instancu, koja se može kasnije pročitati iz baze.
- **Read:** Ove metode funkcionišu kao *query-i*. Na primjer, za pronađazak svih recepata u bazi se koristi metoda `Recipe.find`, gdje je `Recipe` objekat eksportovan iz modela `Recipe.js`. Može se koristiti i metoda `findById`, unutar koje se prilaže i ID objekta koji se želi pronaći. Također, mogu se koristiti i standardni *query-i*.
- **Update:** Uređivanje podataka u bazi je omogućeno pomoću metode `findByIdAndUpdate`, ukoliko je dostupan ID željenog objekta. Ukoliko se *query* ne radi pomoću ID-a, moguće je koristiti metodu `updateOne` ili `updateMany`, s odgovarajućim kriterijem kojeg objekat mora ispuniti da bi se *update-ovao*.
- **Delete:** Metoda za brisanje instance iz baze koja se najčešće koristi je `findByIdAndDelete`, kojoj se prosljeđuje ID objekta koji se treba izbrisati.

Potrebno je naglasiti da *MongoDB* spada u tzv. *NoSQL baze podataka*, što znači da standardna relaciona pravila ne vrijede u ovom okruženju.

Kod standardnih relacionih baza javljaju se pojmovi *Primarni* i *Strani ključ*. Primarni ključ je jedan od atributa (najčešće ID broj) koji ima unikatnu vrijednost, i po kojem se neka instanca entiteta može razlikovati od svih ostalih. Strani ključ se pojavljuje u instanci objekta, kada se on poveže drugom instancom i predstavlja primarni ključ druge instance. Ono što je karakteristično za *MongoDB* bazu podataka, jeste to da ona nije u potpunosti relaciona baza, odnosno, povezivanje entiteta ne funkcioniše na prethodno objašnjen način. Kao prvo, unutar *MongoDB-a* pojam instance objekta je zamijenjen pojmom **dokumenta**. Svaki skup više instanci (dokumenata) se naziva **kolekcija**. Dakle, u sklopu aplikacije, za svaki *dokument* se kreira jedan *model*, čije će karakteristike biti smještene u **kolekciji** [4].

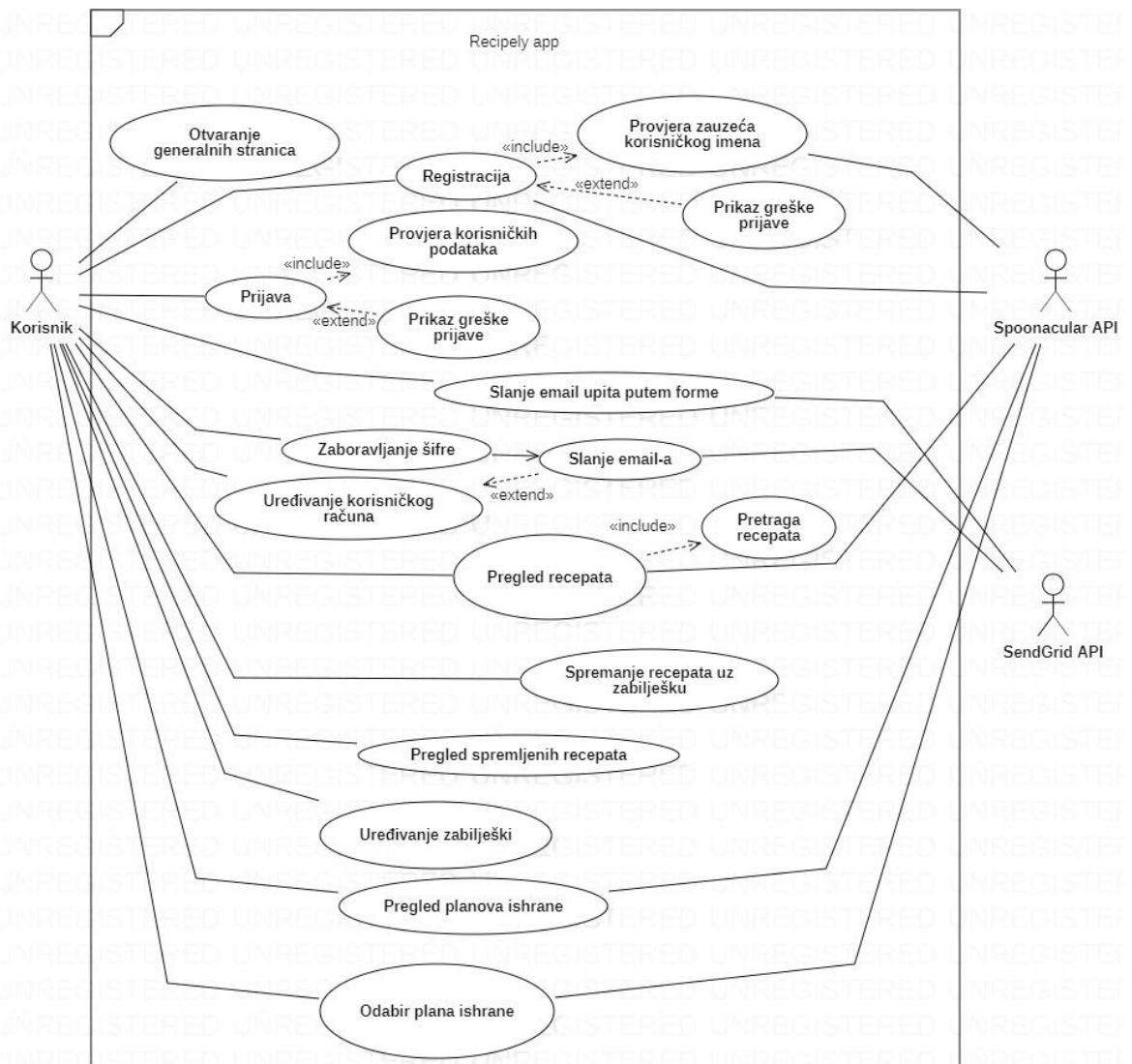
Povezivanje dokumenata unutar *MongoDB* baze može funkcionišati na dva načina: **referenciranjem** ili **umetanjem**. Referenciranje može biti *parent-referenciranje* ili *child-referenciranje*. Referenciranje općenito podrazumijeva postavljanje reference na drugu kolekciju. Ukoliko postoji kolekcija User-Recipe = 1:N, unutar definiranja modela User, može se referencirati niz recepata i to se naziva *child-reference*. Također, unutar modela Recipe može se referencirati User kojem pripada, i to se naziva *parent-reference*. Osim referenciranja, postoji i umetanje. Funkcioniše isto kao i referenciranje, samo što se umjesto reference na model po primarnom ključu, u model ubacuje cjelokupni drugi model [4].

Pomoću servisa Mongo Atlas je na internetu kreirana baza podataka, koja je dostupna za sve IP adrese koje mogu priložiti odgovarajući *username* i *password*.

4. OPIS SLUČAJEVA KORIŠTENJA PODSISTEMA

Na osnovu ideje projektnog zadatka i cilja servisa, uočeni su sljedeći slučajevi korištenja:

1. Otvaranje generalnih stranica
2. Registracija
3. Prijava
4. Slanje mail-a putem forme
5. Pregled svih recepata
6. Spremanje recepata uz zabilješku
7. Pregled spremeljenih recepata
8. Uređivanje zabilješki na spremljenom receptu
9. Pregled svih planova ishrane
10. Odabir plana ishrane
11. Uređivanje korisničkog računa
12. Izmjena zaboravljene šifre



Slika 4.1. Dijagram korištenja sistema

SK1: Otvaranje generalnih stranica (Početna stranica, About ili Contact)

1. Aktori: Korisnik
2. Preduslov: Sistem je aktivan, mogućnost izlaza na internet

Osnovni scenario:

1. Korisnik odabire stranicu (Početna, About ili Contact)
2. Browzer prikazuje HTML predan od strane Recipely aplikacije

Alternativni scenario:

1. Ako se u sistemu desi greška, prikazuje se prikladna Error poruka i nemoguće je pristupiti generalnoj stranici

SK2: Registracija korisnika

1. Aktori: Korisnik, Spoonacular API
2. Preduslov: Sistem je aktivan, mogućnost izlaza na internet

Osnovni scenario:

1. Korisnik odabire dugme za Registraciju
2. Recipely prikazuje Signup formu na Browseru
3. Korisnik popunjava formu svojim podacima
4. Pritisom na dugme za Slanje, šalju se podaci aplikaciji Recipely
5. Recipely šalje podatke Spoonacular API-u
6. Spoonacular API provjerava duplicitiranje korisničkog imena
7. Ako ne postoji korisnik s tim korisničkim imenom, korisnik je uspješno registriran

Alternativni scenario:

1. Ako postoji korisnik s takvim korisničkim imenom, API obavještava aplikaciju da nije moguće registrirati korisnika s tim korisničkim imenom
2. Recipely putem Browsera korisniku ispisuje grešku registracije, i zahtijeva da se unese drugo korisničko ime

SK3: Prijava korisnika

1. Aktori: Korisnik
2. Preduslov: Sistem je aktivan, mogućnost izlaza na internet

Osnovni scenario:

1. Korisnik odabire dugme za Prijavu
2. Recipely prikazuje Login formu na Browseru
3. Korisnik popunjava formu sa korisničkim imenom i šifrom

4. Pritiskom na dugme za Slanje, šalju se podaci aplikaciji Recipely
5. Recipely app provjerava korisničke podatke unutar svoje baze
6. Ako su podaci tačni, korisnik je uspješno prijavljen

Alternativni scenario:

1. Ako korisnički podaci nisu tačni, Recipely putem Browsera korisniku ispisuje grešku o prijavi, i putem alerta zahtijeva da se unesu ispravni podaci

SK4: Slanje mail-a putem forme

1. Aktori: Korisnik, SendGrid API
2. Preduslov: Sistem je aktivan, mogućnost izlaza na internet

Osnovni scenario:

1. Korisnik se nalazi na stranici „Kontakt“
2. Korisnik popunjava mail formu (ime, e-mail, broj mobitela i poruka)
3. Pritiskom na dugme Posalji, mail se salje putem SendGrid API-a na e-mail koji je definiran kodom (ivymaster95@gmail.com)

Alternativni scenario:

1. Ako se u sistemu desi greška, prikazuje se prikladna Error poruka i nemoguće je poslati e-mail

SK5: Pregled svih recepata

1. Aktori: Korisnik, Spoonacular API
2. Preduslov: Sistem je aktivan, mogućnost izlaza na internet

Osnovni scenario:

1. Korisnik odabire dugme za prikaz svih recepata
2. Recipely app šalje zahtjev Spoonacular API-u za dohvatanje nasumičnih 12 recepata iz baze
3. Nakon što API pošalje recepte, Recipely app putem Browsera prikazuje iste

Alternativni scenario:

1. Korisnik putem kompleksne pretrage određuje kriterije po kojima će se uzimati recepti iz baze Spoonacular API-a. Kriteriji pretrage uključuju: vrstu kuhinje, vrstu dijete, intoleranciju, tip jela (predjelo, doručak, ručak i sl.) i sort (sortiranje podataka)
2. Nakon što API pošalje recepte definirane kriterijima, Recipely app putem Browsera prikazuje iste

SK6: Spremanje recepata uz zabilješku

1. Aktori: Korisnik, Spoonacular API
2. Preduslov: Korisnik je prijavljen i sistem je aktivan

Osnovni scenario:

1. Korisnik odabire određeni recepat putem *buttona „Detalji“*
2. Recipely app prikazuje atribute odabranog recepta
3. Korisnik unosi svoju zabilješku i odabirom *buttona “Spremi”*, sprema recept sa zabilješkom u svoj dnevnik
4. UID spremjenog recepta se sprema u bazu aplikacije Recipely
5. Recipely ispisuje notifikaciju o uspjehnosti procesa

Alternativni scenario:

1. Ako se u sistemu desi greška, prikazuje se prikladna Error poruka i nemoguće je spremiti recept

SK7: Pregled spremjenih recepata

1. Akteri: Korisnik, Spoonacular API
2. Preduslov: Korisnik je prijavljen, sistem je aktivan

Osnovni scenario:

1. Korisnik odabire dugme za prikaz Spremljenih recepata
2. Recipely app uzima iz baze podataka sve ID-eve recepata koje je korisnik spremio, te ih traži od Spoonacular API-a
3. API vraća podatke o receptima kao što su naziv, slika, sastojci i koraci pri izradi
4. Recipely app prikazuje specifikacije o receptima

Alternativni scenario:

1. Ako korisnik nije spremio nijedan recept, stranica koja se prikazuje je prazna (bez podataka o receptima)

SK8: Uređivanje zabilješki na spremjenom receptu

1. Akteri: Korisnik, Spoonacular API
2. Preduslov: Recept je spremjen od strane korisnika

Osnovni scenario:

1. Korisnik odabire jedan spremjeni recept
2. Recipely app šalje Spoonacular API-u zahtjev za dobijanje podataka o receptu sa predanim ID-om

3. API vraća podatke o receptu (naziv, slika, sastojci, koraci)
4. Recipely app prikazuje podatke o receptu
5. Korisnik upisuje novu zabilješku
6. Pritisom na tipku Spremi, izmjena se sprema u Recipely bazu podataka
7. Recipely app prikazuje notifikaciju o uspješnom spremanju

Alternativni scenario:

1. Ako se u sistemu desi greška, prikazuje se prikladna Error poruka i nemoguće je urediti recept

SK9: Pregled svih planova ishrane

1. Aktori: Korisnik, Spoonacular API
2. Preduslov: Korisnik je prijavljen, sistem je aktivran

Osnovni scenario:

1. Korisnik odabire dugme za prikaz svih planova ishrane
2. Recipely šalje zahtjev Spoonacular API-u za dobijanje 40 nasumičnih planova ishrane
3. Spoonacular API vraća planove ishrane
4. Recipely app putem Browsera prikazuje prikupljene planove ishrane

Alternativni scenario:

1. Ako se u sistemu desi greška, prikazuje se prikladna Error poruka i nemoguće je prikazati planove ishrane

SK10: Odabir plana ishrane

1. Akteri: Korisnik, Spoonacular API
2. Preduslov: Korisnik je prijavljen, sistem je aktivran

Osnovni scenario:

1. Korisnik odabire dugme za detaljan pregled plana ishrane
2. Recipely app prikazuje detalje o odabranom planu ishrane
3. Korisnik odabire datum od kojeg započinje plan ishrane
4. Pritisom na dugme za spremanje plana, plan započinje od dana koji je odabran

Alternativni scenario:

2. Ako je korisnik već spremio jedan plan ishrane, na taj se dodaju informacije (recepti, nutricione vrijednosti) o novom planu ishrane

SK11: Uređivanje korisničkog računa

1. Akteri: Korisnik,
2. Preduslov: Korisnik je prijavljen, sistem je aktivan

Osnovni scenario:

1. Korisnik odabire dugme za prikaz profila
2. Recipely keni korisničke podatke iz svoje baze te ih prikazuje unutar forme za izmjenu profila
3. Korisnik odabire koje podatke želi izmijeniti, nakon čega odabire dugme Spremi
4. Korisnički podaci u bazi se ažuriraju i Recipely app daje notifikaciju o uspješnoj izmjeni

Alternativni scenario:

1. Ako se novi atribut poklapa sa postojećim atributom u bazi, Recipely obavještava korisnika i zahtjeva novi unos

SK12: Izmjena zaboravljene šifre

3. Akteri: Korisnik, SendGrid
4. Preduslov: Korisnik je prijavljen, sistem je aktivan

Osnovni scenario:

1. Korisnik prilikom prijave odabire ponuđeni link za izmjenu zaboravljene šifre
2. Recipely šalje formu za unos e-mail adrese
3. Na unesenu adresu, nakon što korisnik pritisne dugme Pošalji, ukoliko adresa postoji u bazi, Recipely na istu, putem SendGrid API-a, šalje link za formu za unos nove šifre.
4. Otvaranjem linka, Korisnik unosi i potvrđuje novu šifru, koja se ažurira u bazi podataka nakon što se pritisne dugme Pošalji

Alternativni scenario:

5. Ako korisnik nije u roku od 1min izvršio unos nove šifre, token za promjenu šifre (koji je dio url-a za dobijanje nove forme) ističe, čime taj url postaje neupotrebljiv i korisnik putem njega ne može promijeniti šifru, već mora poslati novi zahtjev.

5. ANALIZA PODSISTEMA

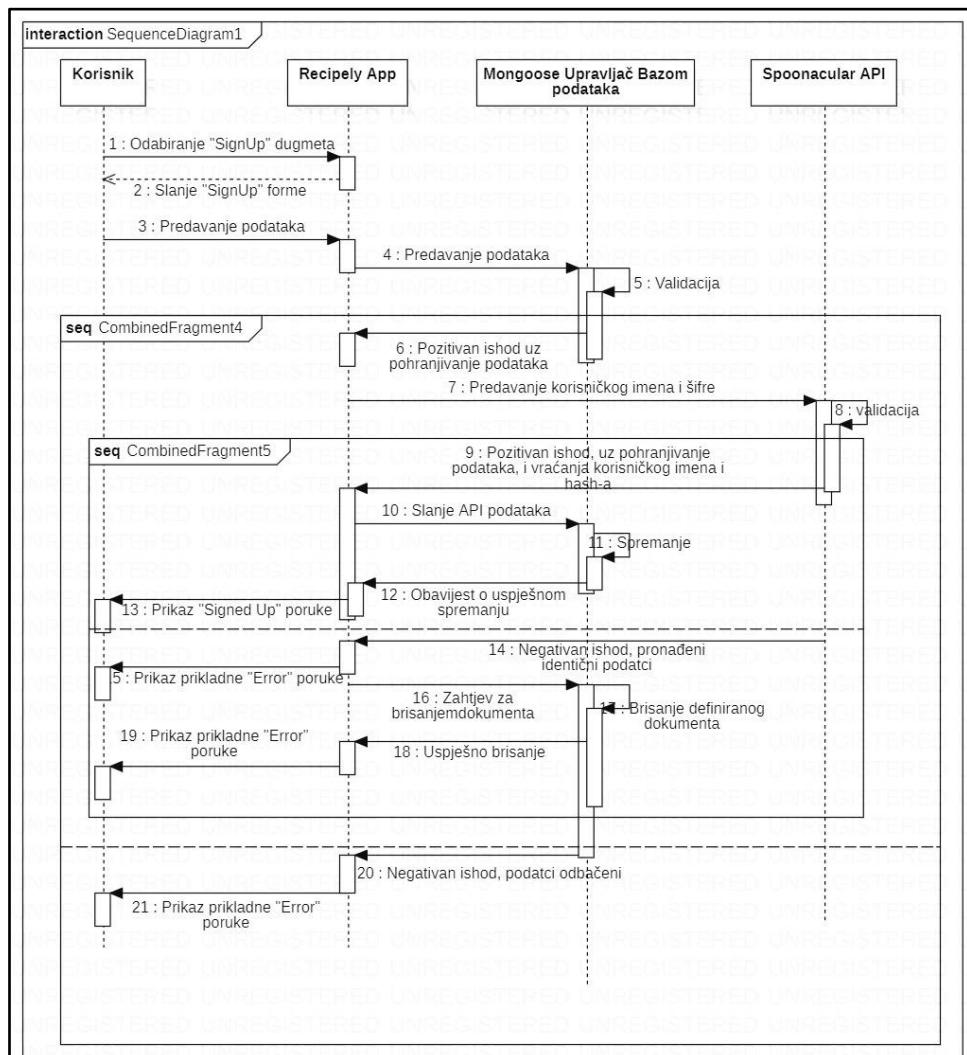
U fazi analize opisuje se logička struktura i ponašanje softverskog sistema (poslovna logika softverskog sistema). Ponašanje softverskog sistema se opisuje pomoću sekvencijalnih dijagrama.

5.1. Sekvencijalni dijagrami

Za razliku od dijagrama korištenja koji prikazuje statičku, dijagram slijeda prikazuje dinamičku saradnju između sudionika u sistemu. Ovaj dijagram prikazuje vremenski slijed poruka koje sudionici razmjenjuju i služi za prikaz redoslijeda njihove interakcije.

Ovaj dijagram ima dvije dimenzije: vertikalnu, koja predstavlja vrijeme i horizontalnu koja prikazuje različite objekte odnosno instance sistema. U ovim dijagramima objekti su predstavljeni uz pomoć pravougaonika, a protok vremena isprekidanim linijama koje idu od objekata na dole.

SD1: Registracija korisnika



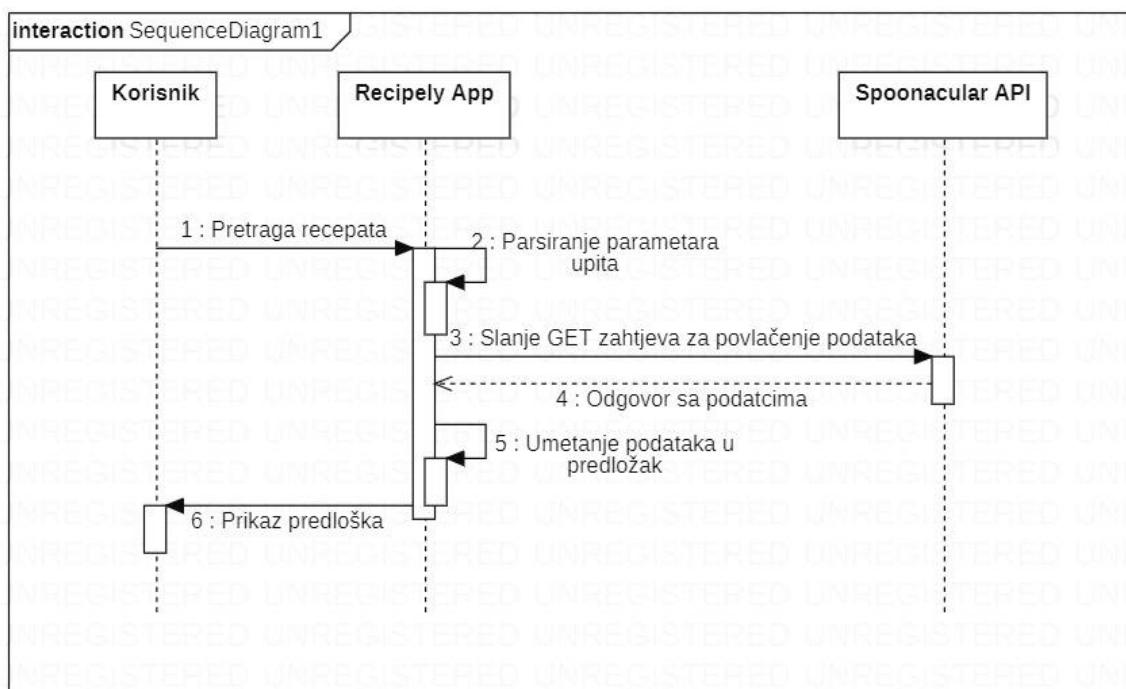
Slika 5.1. Sekvencijalni dijagram podsistema za registraciju korisnika

Iz sekvencijalnog dijagrama se može vidjeti tačno za koju akciju je zadužen koji dio sistema. Na lijevoj strani su predstavljene akcije korisnika koji on može izvršiti. U sredini su predstavljene akcije koje lokalna Recipely aplikacija obavlja u pozadini. Jedan od učesnika sistema je i Mongoose upravljač bazom podataka. Na desnoj strani su predstavljene akcije koje Spoonacular API obavlja kad joj se pošalje zahtjev. Akcije su povezane strelicama koje predstavljaju smjer odnosno tok akcija kojim se izvršavaju.

Analiza dijagrama podsistema za registraciju korisnika

Korisnik zahtjeva formu za registraciju od Recipely aplikacije, koja istu vraća ukoliko korisnik odabere SignUp dugme. Korisnik predaje podatke aplikaciji Recipely, te aplikacija ove podatke prosljeđuje upravljaču internet baze podataka. Validacija se vrši unutar te baze podataka, te kada je upravljač poslao notifikaciju aplikaciji Recipely o pozitivnom ishodu, ista šalje korisničko ime i šifru Spoonacular API-u. API također vrši validaciju dobijenih podataka, te u slučaju pozitivnog ishoda, uz pohranjivanje podataka, vraća korisničko ime i *hash*-iranu šifru aplikaciji. Podaci se spremaju u bazi podataka aplikacije. Tada Recipely aplikacija obavještava korisnika da je uspješno registriran. Ako se desi slučaj negativnog ishoda (pronađeni su identični podaci u bazi API-a), Spoonacular API obavještava aplikaciju Recipely o tome, koja šalje zahtjev za brisanje dokumenta kreiranog u bazi podataka, koji sadrži te korisničke podatke. Upravljač bazom vrši brisanje i daje notifikaciju aplikaciji Recipely o uspješnom brisanju. Tada aplikacija Recipely obavještava korisnika da nije moguća registracija budući da već postoji korisnik s tim podacima.

SD2: Pretraga recepata

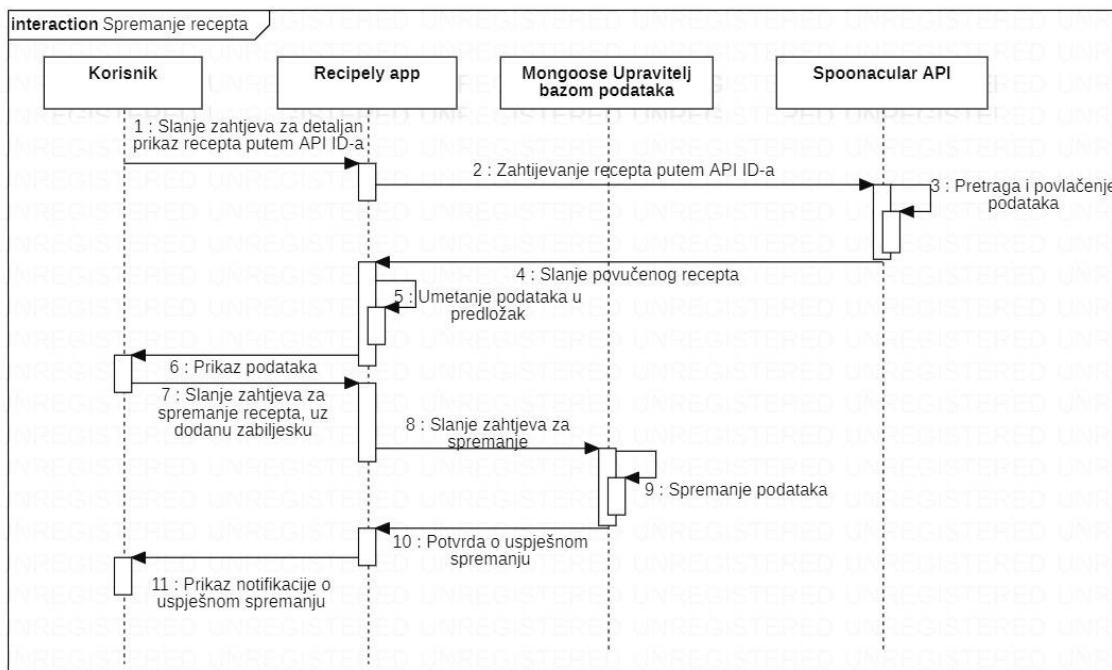


Slika 5.2. Sekvencijalni dijagram podsistema za pretragu recepata

Na lijevoj sekvencijalnog dijagrama strani su predstavljene akcije korisnika koji on može izvršiti. U sredini su predstavljene akcije koje lokalna Recipely aplikacija obavlja u pozadini. Na desnoj strani su predstavljene akcije koje Spoonacular API obavlja kad joj se pošalje zahtjev od strane aplikacije Recipely.

Najprije korisnik odabire opciju za pretragu recepata, a aplikacija Recipely parsira parametre upita, nakon čega šalje Spoonacular API-u GET zahtjev za povlačenje podataka iz baze API-a. Ovaj zahtjev podrazumijeva 12 nasumično odabralih recepata iz baze. Spoonacular API vraća Recipely aplikaciji odgovor sa podacima, koja ove podatke pohranjuje u predložak, koji se ispisuje Korisniku.

SD3: Spremanje recepta



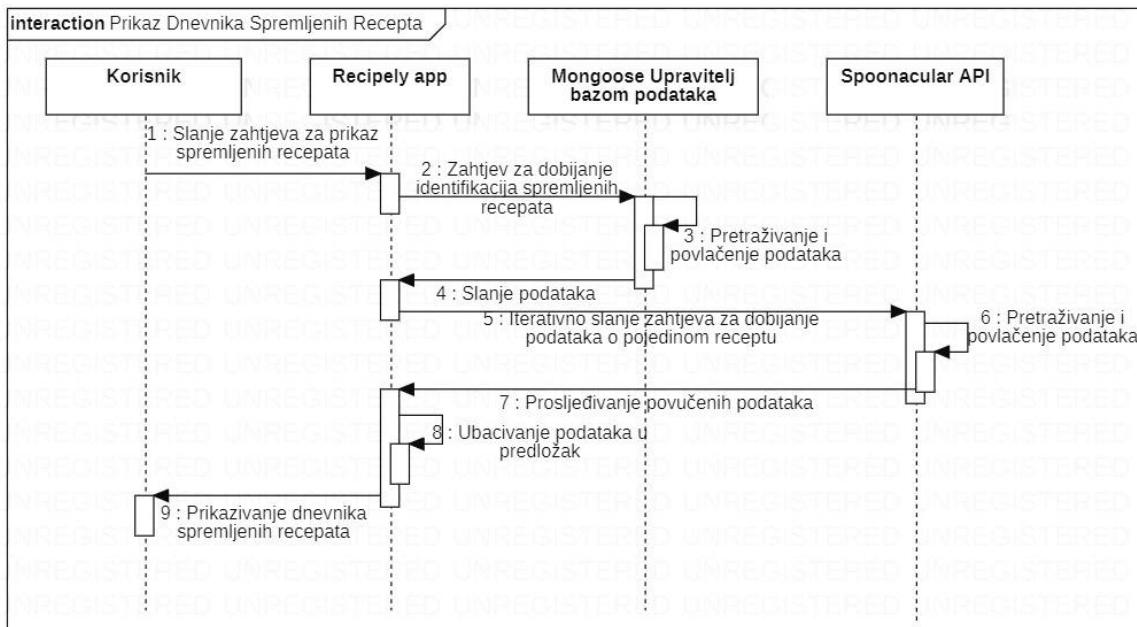
Slika 5.3. Sekvencijalni dijagram podsistema za spremanje recepta

Na lijevoj sekvencijalnog dijagrama strani su predstavljene akcije korisnika koji on može izvršiti. U sredini su predstavljene akcije koje lokalna Recipely aplikacija obavlja u pozadini i Mongoose upravitelj bazom podataka. Na desnoj strani su predstavljene akcije koje Spoonacular API obavlja kad joj se pošalje zahtjev od strane aplikacije Recipely.

Korisnik šalje zahtjev za detaljan prikaz recepta putem API ID-a. Zahtjev se šalje odabirom *buttona* Detalji. Recipely app tada šalje novi zahtjev Spoonacular API-u za povlačenje podataka o receptu s datim ID-em. Spoonacular API pretražuje svoju bazu i povlači recept koji je pretražen. Nadalje se povučeni recept šalje Recipely aplikaciji, koja umeće podatke u odgovarajući predložak, koji se prikazuje korisniku. Korisnik upisuje zabilješku ako želi, te odabirom na dugme Spremi, Korisnik šalje zahtjev za spremanje recepta u svoj dnevnik. Recipely aplikacija šalje zahtjev Mongoose upravitelju baze podataka za spremanje objekta koji se sastoji od ID-a recepta i zabilješke, u obliku umetnutog dokumenta u dokument tog

korisnika. Upravitelj baze podataka, kao povratnu informaciju, šalje Recipely aplikaciji potvrdu o spremanju recepta u bazu, te aplikacija obavještava Korisnika o uspješnom spremanju recepta u svoj dnevnik.

SD4: Prikazivanje dnevnika spremjenih recepata

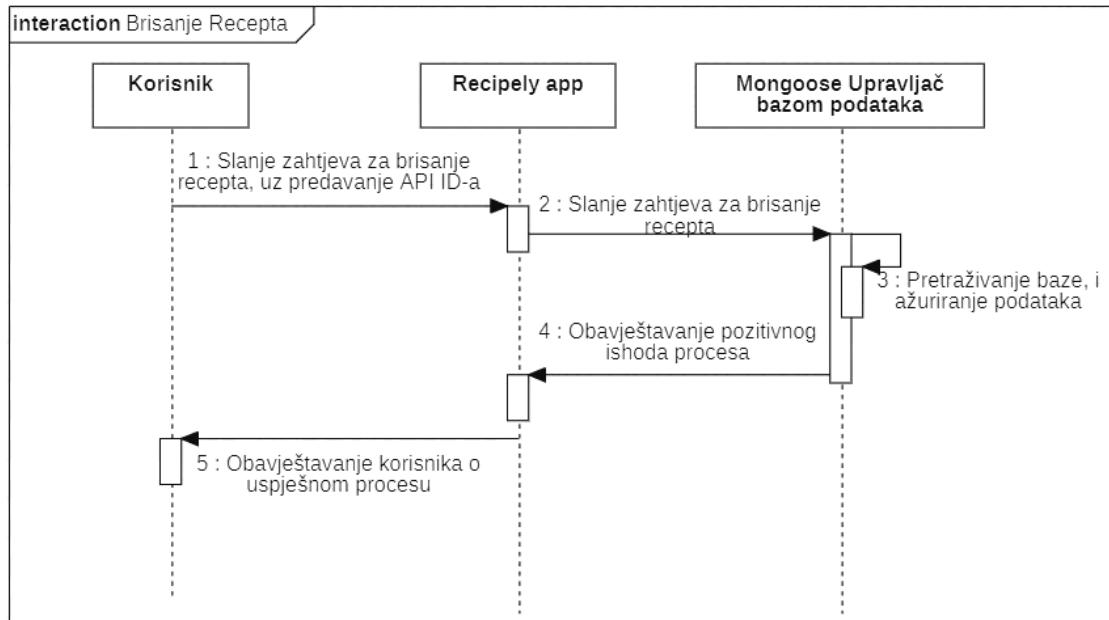


Slika 5.4. Sekvencijalni dijagram podsistema za prikazivanje dnevnika recepta

Na lijevoj sekvencijalnog dijagrama strani su predstavljene akcije korisnika koji on može izvršiti. U sredini su predstavljene akcije koje lokalna Recipely aplikacija obavlja u pozadini i Mongoose upravitelj bazom podataka. Na desnoj strani su predstavljene akcije koje Spoonacular API obavlja kad joj se pošalje zahtjev od strane aplikacije Recipely.

Korisnik šalje aplikaciji Recipely zahtjev za prikaz spremjenih recepata. Recipely app šalje Upravitelju baze podataka zahtjev za povlačenje identifikacija spremjenih recepata tog korisnika. Upravitelj pretražuje i povlači podatke iz baze, te ih šalje aplikaciji Recipely. Recipely app iterativno šalje zahtjev Spoonacular API-u za dobijanje podataka o pojedinom receptu. Spoonacular API pretražuje i povlači podatke iz svoje baze, te ih proslijeđuje aplikaciji Recipely putem povratne poruke. Nadalje, Recipely app ubacuje dobijene podatke u predložak, koji se prikazuje Korisniku putem Browzera.

SD5: Brisanje recepta



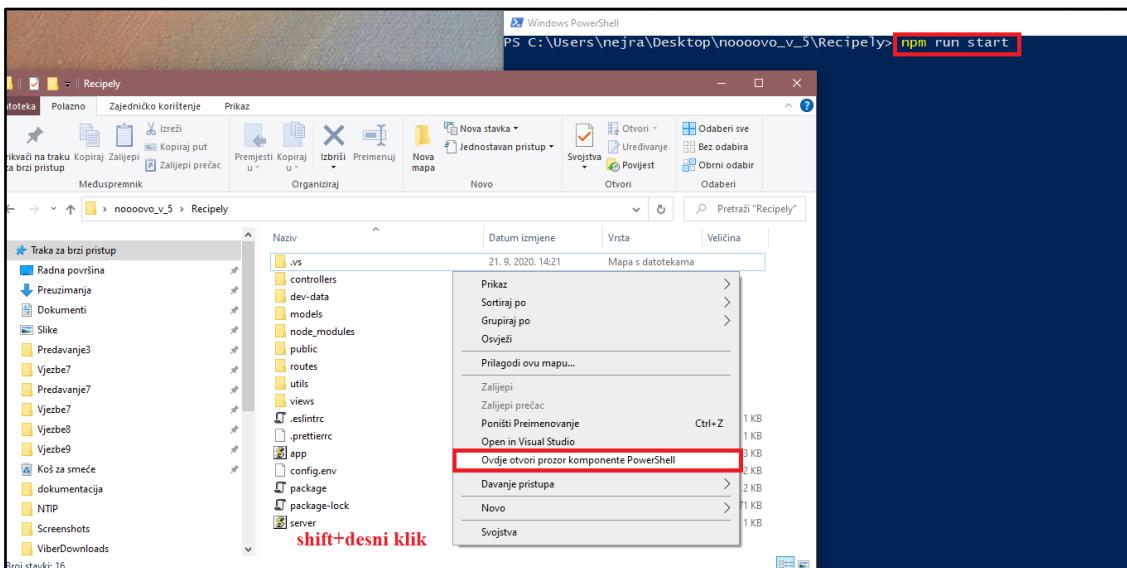
Slika 5.5. Sekvencijalni dijagram podsistema za spremanje recepta

Na lijevoj sekvencijalnog dijagrama strani su predstavljene akcije korisnika koji on može izvršiti. U sredini su predstavljene akcije koje lokalna Recipely aplikacija obavlja u pozadini i Mongoose upravitelj bazom podataka.

Unutar svog dnevnika, korisnik odabire koji recept će obrisati, pritiskom na dugme za Brisanje. Time korisnik šalje zahtjev za brisanje recepta uz predavanje API ID-a tog odabranog recepta. Nadalje, Recipely app šalje Upravitelju baze podataka slanje zahtjeva za brisanje tog recepta. Upravitelj vrši pretraživanje i ažuriranje podataka u bazi. Upravitelj obavještava aplikaciju Recipely o pozitivnom ishodu procesa, nakon čega Recipely obavještava Korisnika da je brisanje izvršeno.

6. IMPLEMENTACIJA PODSISTEMA

Implementacija sistema podrazumijeva treću komponentu metodologije korištene u sklopu dokumentacije priloženog projekta. Unutar ovog dijela biće prikazani dijelovi aplikacije Recipely koji se prikazuju u Browseru Chrome. Najprije se prikazuje izgled generalnih stranica (kao što je početna, stranica About, Kontakt i Galerija recepata) neregistriranim korisnicima. U nastavku će biti opisan kod samo onih dijelova sistema koji su vezani uz komunikaciju sa Spoonacular API-em.



Slika 6.1. Pokretanje aplikacije Recipely

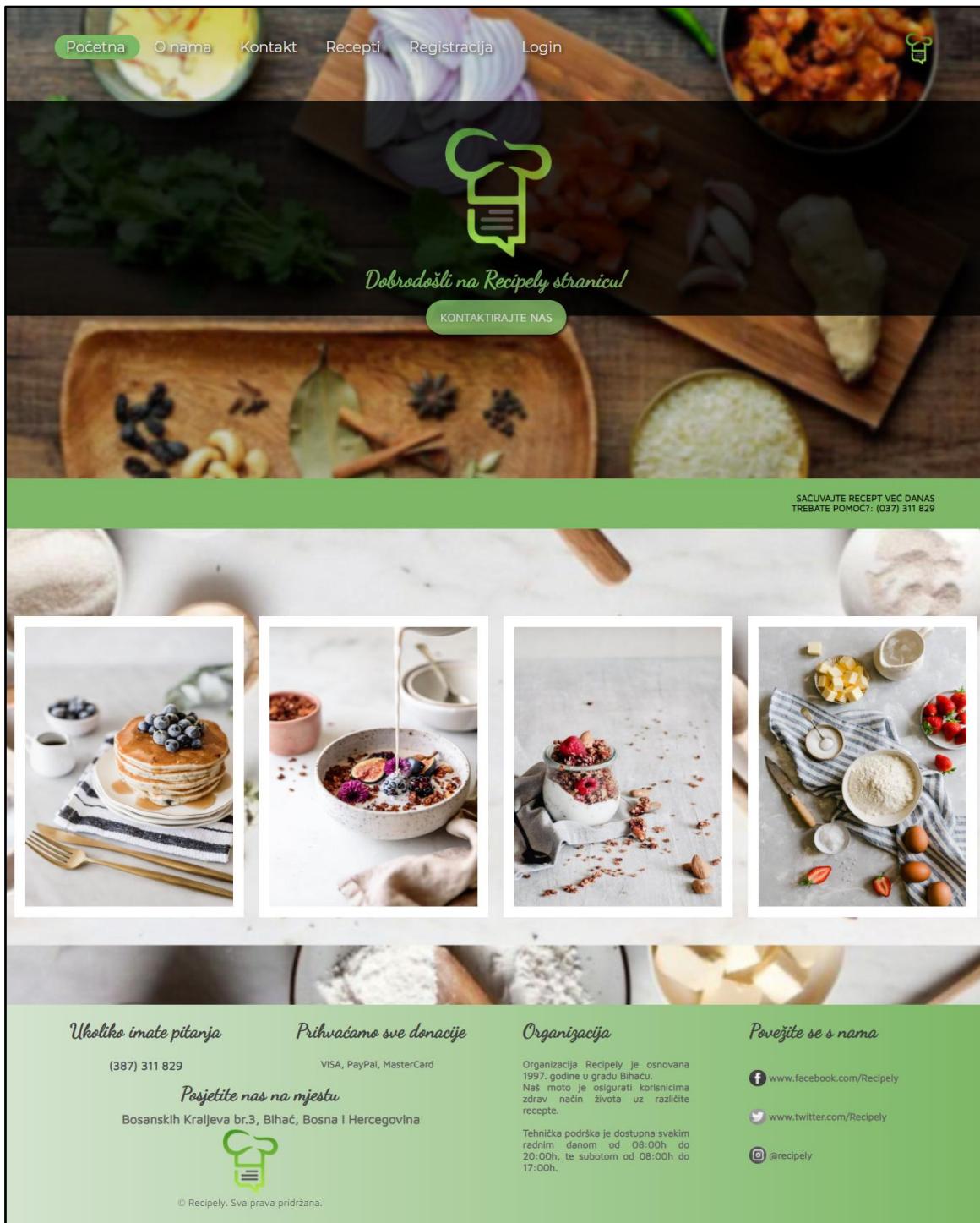
Uslov pokretanja aplikacije je da korisnik ima pristup internetu, budući da se putem interneta šalju podaci iz API-a. Postupak pokretanja aplikacije Recipely je prikazan na slici 6.1. Najprije se otvara projektni folder, u kojem se pomoću "shift+desni klik" otvara lista opcija. Odabire se pokretanje u PowerShell-u. Kada se otvori Powershell, potrebno je unijeti naredbu "npm run start", nakon čega se pritisne enter. Ukoliko je povezivanje s bazom uspješno, u terminalu Powershell se ispisuje poruka koja govori da je konekcija s bazom uspješna i da se projekat pokreće na portu 3000. Nakon toga se otvara Browser unutar kojeg se unosi ruta localhost:3000. Ova ruta korisnika odvodi na početnu stranicu. Također, ukoliko korisnik posjeduje Visual Studio Code, na isti način je moguće ovaj projekat otvoriti unutar terminala te aplikacije, nakon dodavanja foldera u nju.

6.1. Neregistrirani korisnici

Neregistrirani korisnici su oni korisnici koji nemaju registriran korisnički račun. Neregistrirani korisnik može pregledati početnu stranicu, stranicu "O nama", stranicu "Kontakt", na kojoj može poslati e-mail organizaciji kroz priloženu formu, može pregledati Galeriju nasumičnih 12 recepata, koji se mijenjaju svakim osvježavanjem stranice, te može pretražiti recepte iz Spoonacular API-a prema određenim kriterijima.

6.1.1. Početna stranica

Izgled početne stranice je prikazan na slici 6.2. Budući da u ovom dijelu ne postoji komunikacija sa Spoonacular API-em, kod za implementaciju ovog dijela nije potrebno objašnjavati.



Slika 6.2. Početna stranica aplikacije Recipely

Početna stranica je dostupna i registriranim i neregistriranim korisnicima. Ukoliko korisnik nije trenutno prijavljen/registriran, u sklopu navigacijskog menija se nude opcije za prijavu ili

registraciju. Ukoliko je korisnik prijavljen, u sklopu navigacijskog menija se nude opcije za korisnički profil i za logout.

6.1.2. Stranica "O nama"

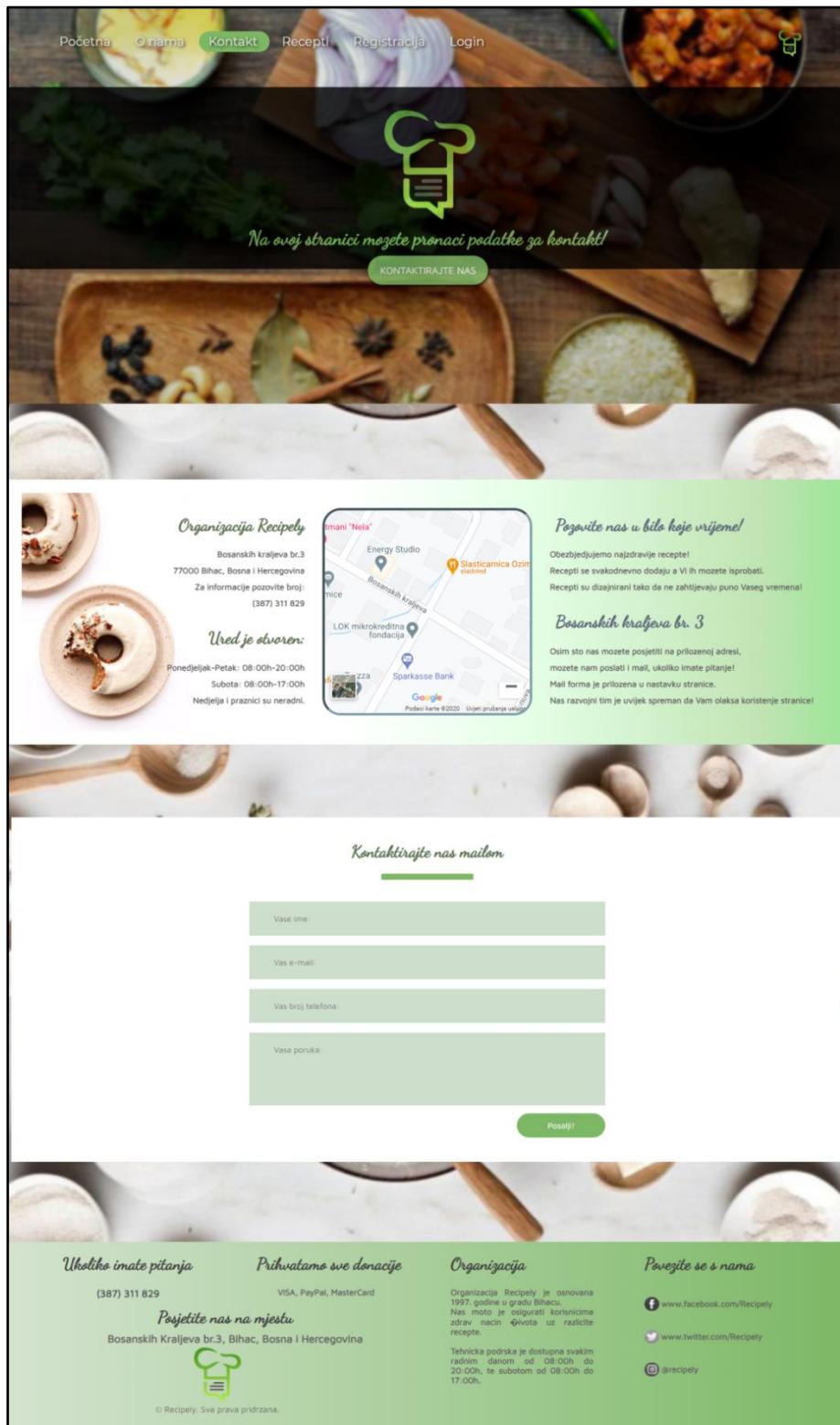
Izgled stranice "O nama" je prikazan na slici 6.3. Budući da u ovom dijelu ne postoji komunikacija sa Spoonacular API-em, kod za implementaciju ovog dijela nije potrebno objašnjavati.



Slika 6.3. Informacijska stranica aplikacije Recipely

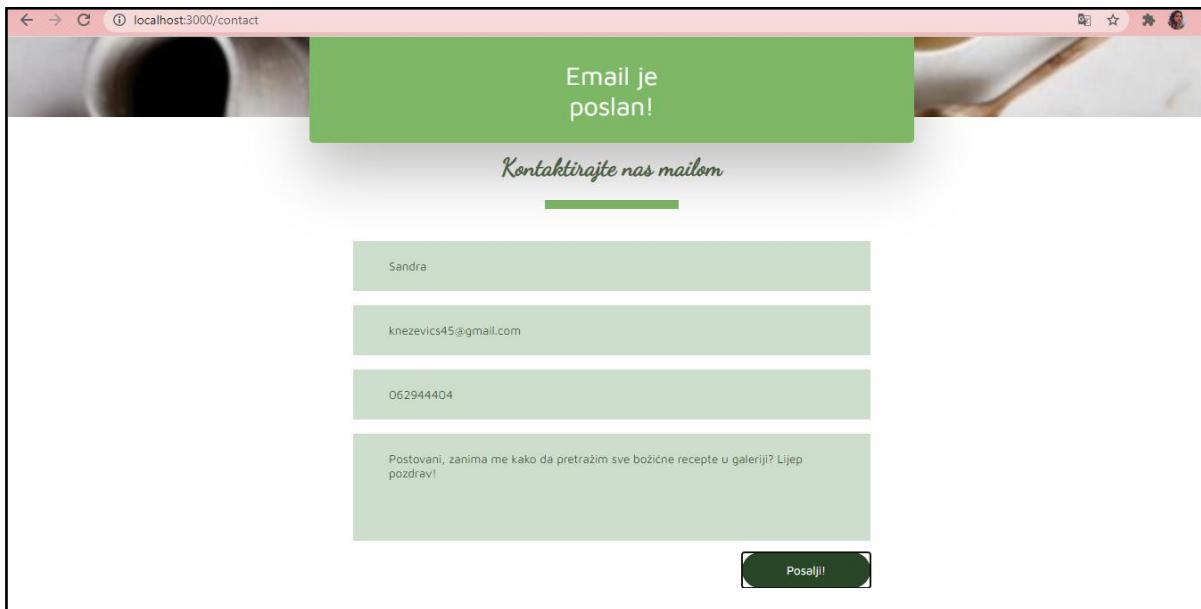
6.1.3. Stranica “Kontakt”

Izgled stranice “Kontakt” je prikazan na slici 6.4. Na prvom dijelu stranice se nalazi umetnuta mapa sa lokacijom sjedišta organizacije koja se može detaljnije pogledati na Google kartama. Osim toga, u ovom dijelu se nalazi forma za slanje e-maila organizaciji Recipely.



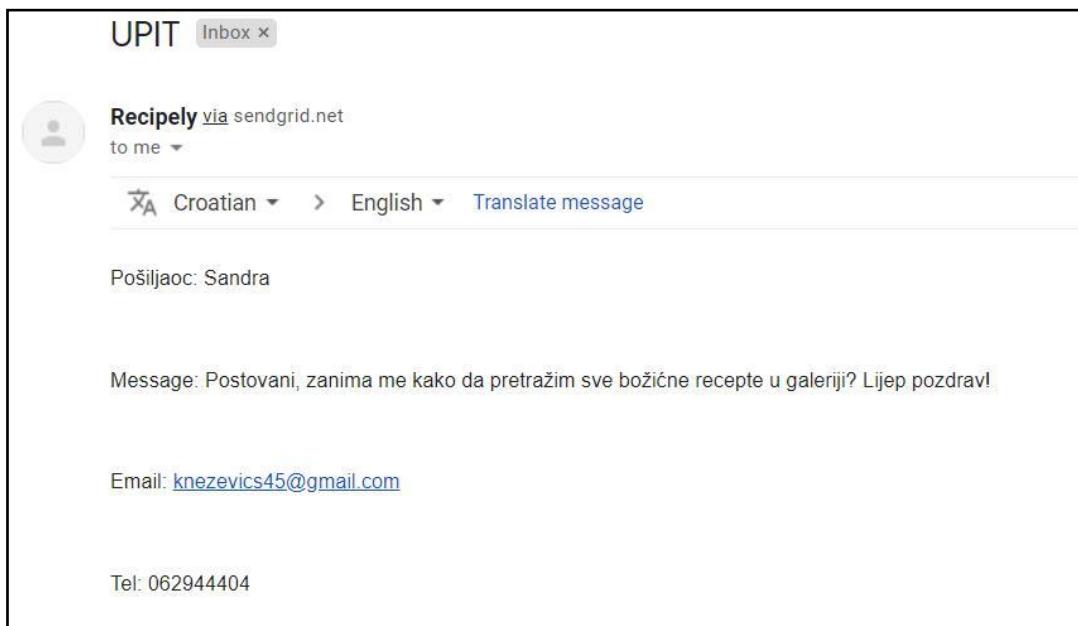
Slika 6.4. Kontaktna stranica aplikacije Recipely

Na slici 6.5 prikazana je notifikacija sistema koja se prikazuje korisniku nakon što on popuni formu za slanje maila organizaciji i pritisne dugme *Posalji*.



Slika 6.5. E-mail forma na aplikaciji Recipely

Na slici 6.6 prikazan je e-mail koji je organizacija dobila od strane jednog korisnika. Komunikacija Recipely aplikacije i korisnika putem maila je ostvarena servisa SendGrid.

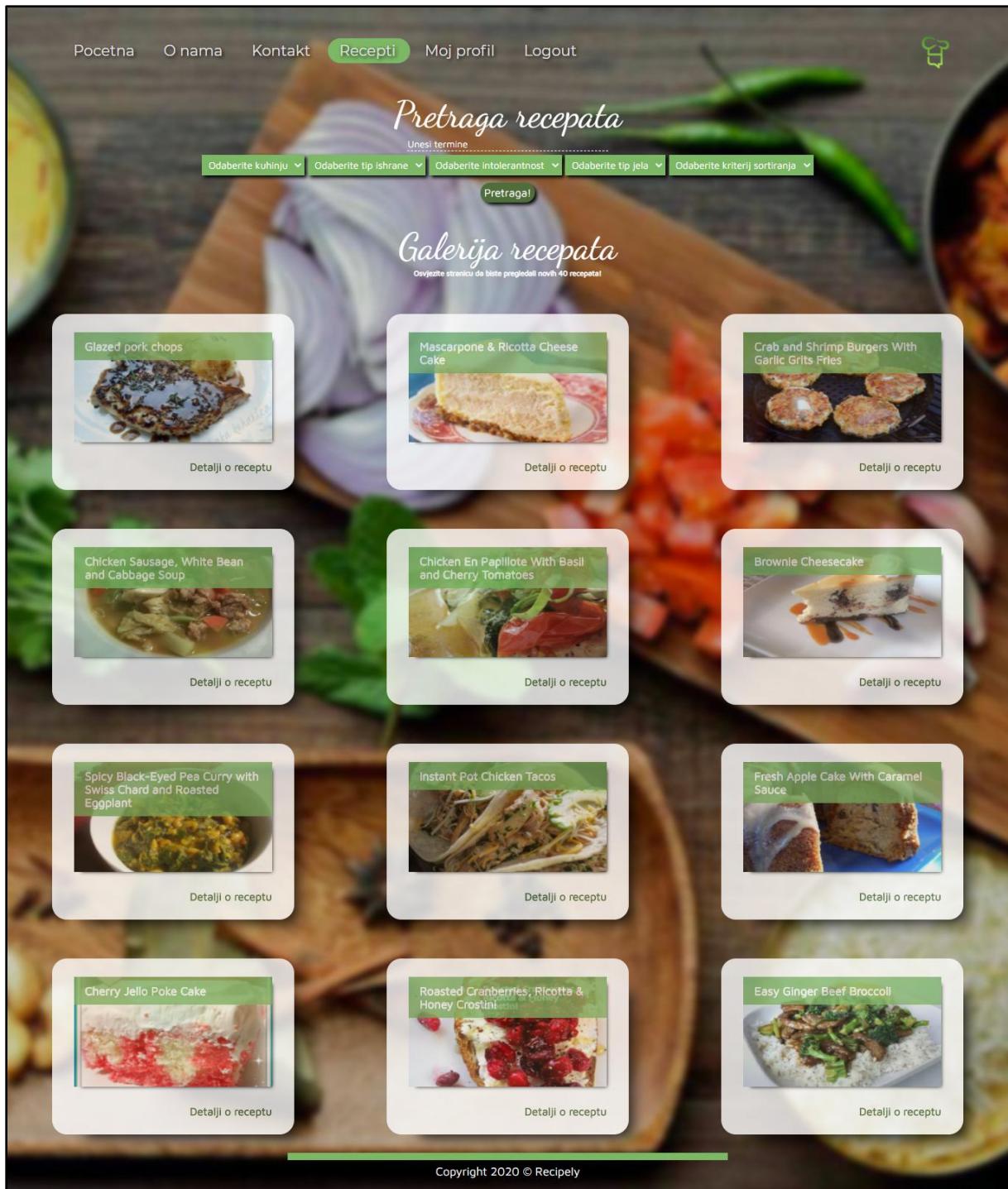


Slika 6.6. E-mail koji je organizacija dobila od strane korisnika

SendGrid je je servis koji se ponaša kao menadžer koji rješava sve tehničke detalje vezane uz slanje mailova. Također, u sklopu sistema se ovaj servis koristi za slanje mailova dobrodošlice svakom novom prijavljenom korisniku, kao što će biti prikazano u poglavljju o prijavi.

6.1.4. Galerija recepata

Galerija recepata podrazumijeva prikaz 12 nasumično odabralih recepata iz baze Spoonacular API-a. Ova stranica je prikazana na Slici 6.7.



Slika 6.7. Galerija 12 nasumičnih recepata aplikacije Recipely

Budući da se recepti pozivaju iz baze Spoonacular API-a, u nastavku će biti objašnjen način komunikacije Recipely aplikacije s API-em prilikom dobijanja 12 nasumičnih recepata.

Korišteni modul za ostvarivanje komunikacije:

receiptController

Prikaz koda:

```
1. exports.getReceipts = catchAsync(async (req, res, next) => {
2.   let odg;
3.   try {
4.     odg = await axios.get(
5.       `${process.env.URL}/recipes/random?apiKey=${
6.         process.env.API_KEY
7.       }&number=12`
8.     );
9.   } catch (err) {
10.    return next(new AppError('Upps. Try again later', 400));
11.  }
12.  res.status(200).render('receipts', {
13.    receipts: odg.data.recipes,
14.    type: req.type
15.  });
16.});
```

URL koji se koristi za dobijanje recepata iz baze API-a je sljedeći [6]:

GET | <https://api.spoonacular.com/recipes/random>

Modul receiptController nudi metodu *getReceipts*, koja kao rezultat vraća nasumično odabrane recepte. Za komunikaciju s API-em se koristi modul *axios*.

Ovaj modul je javascript biblioteka, korištena za generiranje HTTP zahtjeva iz *node.js* okruženja ili XMLHTTP zahtjeva iz Browsera.

Odgovor se dobija u obliku *promise-a*. Kako bi se dobili recepti, kreira se GET zahtjev (*request*) na prethodno prikazani URL. Kako bi se ograničio broj povratnih recepata na 12, upitu se predaje parametar *number* = 12. Ako se desi pogreška, korisnik ne može dobiti povratne informacije od API-a. U suprotnom, *response* se sastoji od predloška *receipts*, u kojeg su ubaćeni dobijeni podaci. Osim toga, pohranjuje se i tip korisnika. Ovim se provjerava da li je korisnik prijavljen ili ne. Ako je korisnik prijavljen *type* će imati vrijednost *korisnik*, a u suprotnom će imati vrijednost *none*.

6.1.5. Pretraga recepata

Pretraga recepata je omogućena i za neregistriranog i za registriranog korisnika i ona se nalazi u sklopu galerije recepata. Postoji pretraga prema terminu, koja funkcioniše tako da korisnik može unijeti jedan ili više termina koje želi, odvojene razmakom. Osim toga, postoji pretraga prema tipu kuhinje, tipu ishrane, tipu intolerantnosti, tipu jela, kriteriju sortiranja. U nastavku će biti prikazan primjer korištenja pretrage recepata.



Slika 6.8. Primjer korištenja opcija za pretragu

Na slici 6.9. prikazani su rezultati za unešene upite (termin chicken, kuhinja američka, intolerantnost na mlijeko i sortiranje prema broju kalorija max->min).

Pocetna O nama Kontakt Recepti Moj profil Logout

Pretraga recepata

Unesite termine

Odaberite kuhinju Odaberite tip ishrane Odaberite intolerantnost Odaberite tip jela Odaberite kriterij sortiranja

Pretraga!

Galerija recepata

Osvježite stranicu da biste pregledali novih 40 recepta

Chicken and Chickpea Chili
Detalji o receptu

Chicken Avocado Burger (Whole 30, PALEO, & Simple Fit Forty)
Detalji o receptu

Chicken Brats & Root Beer BBQ Sauce
Detalji o receptu

Spicy Chicken Corn Dogs with Homemad... Chilli
Detalji o receptu

Chili Lime Chicken Burgers
Detalji o receptu

Chili and Garlic Spiced Beef and Broccoli Stir Fry
Detalji o receptu

Chili Chicken Salad
Detalji o receptu

17 Bean White Chicken Chili
Detalji o receptu

Chicken and Green Pepper Chili
Detalji o receptu

Chicken Ranch Burgers
Detalji o receptu

Copyright 2020 © Recipely

Slika 6.9. Rezultati korištenja opcija za pretragu

Budući da se povratne informacije koje su rezultat pretrage, dobijaju iz baze Spoonacular API-a, u nastavku će biti objašnjen kod kojim se implementira komunikacija između aplikacije Recipely i Spoonacular API-a.

Korišteni modul za ostvarivanje komunikacije:

receiptController

Prikaz koda:

```
1. exports.searchSpecificReceipts = catchAsync(async (req, res, next) => {
2.   const {
3.     query,
4.     cuisine,
5.     diet,
6.     intolerances,
7.     type,
8.     sort
9.   } = req.body;
10.
11.  const url = `${process.env.URL}/recipes/complexSearch?apiKey=${process.env.API_KEY}
12.    &query=${query}&cuisine=${cuisine}&diet=${diet}&intolerances=${intolerances}&type=${type}&sort=${sort}`;
13.  let odg = await axios.get(url);
14.
15.
16.  res.status(200).render('receipts', {
17.    receipts: odg.data.results,
18.    type: req.type
19.  });
20. });


```

URL koji se koristi za dobijanje specifičnih recepata iz baze API-a je sljedeći [6]:

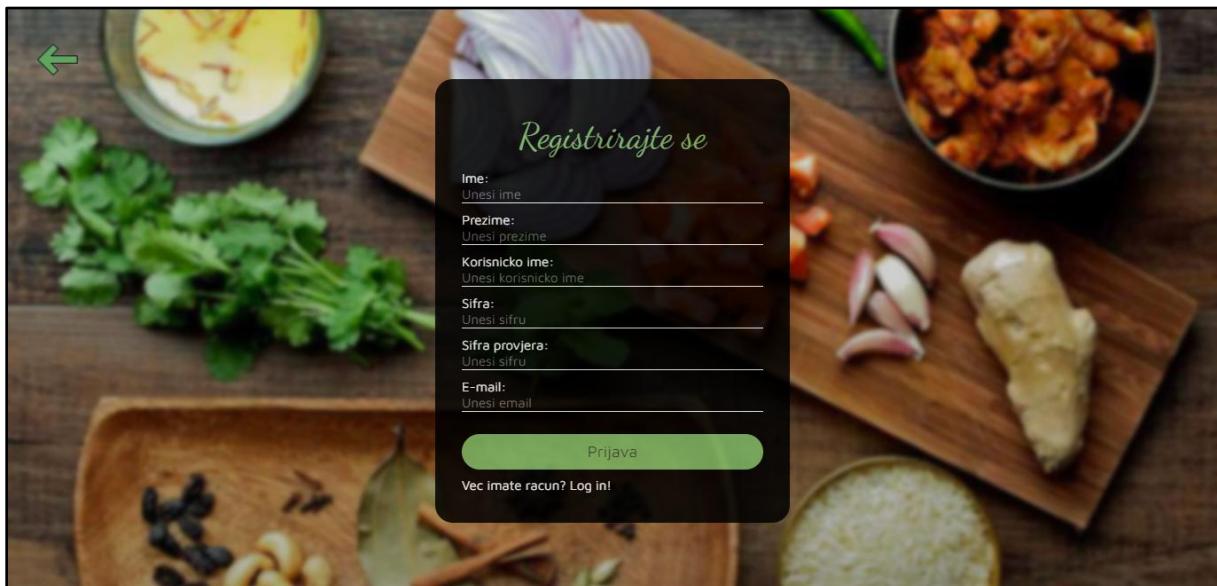
GET | <https://api.spoonacular.com/recipes/complexSearch>

Priložena metoda se nalazi unutar modula receiptController. Prvo se vrši destrukturalizacija objekta *req.body*. Dobivene vrijednosti se predaju parametrima upita koji je specifiran varijablom *url*. Varijabla *url* sadrži link za dobijanje željenih recepata, sa upitima koje je korisnik unio. Modulom *axios* se šalje HTTP zahtjev na taj url, te API vraća željene podatke.

Ako se desi pogreška, korisnik ne može dobiti povratne informacije od API-a. U suprotnom, *response* se sastoji od predloška *receipts*, u kojeg su ubačeni dobijeni podaci. Osim toga, pohranjuje se i tip korisnika. Ovim se provjerava da li je korisnik prijavljen ili ne. Ako je korisnik prijavljen *type* će imati vrijednost *korisnik*, a u suprotnom će imati vrijednost *none*.

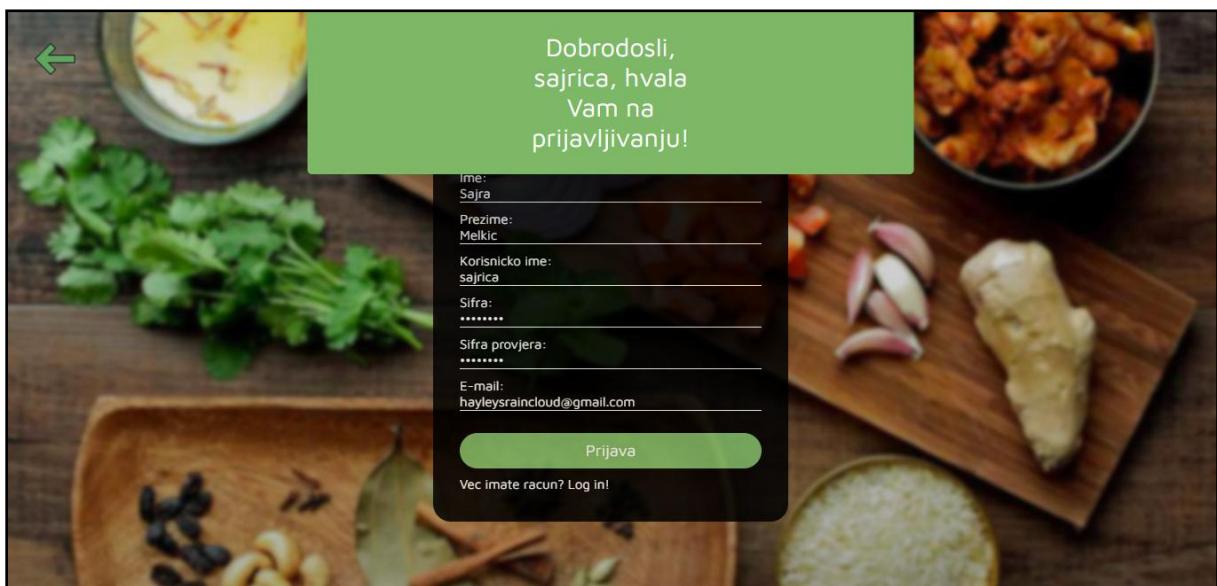
6.1.6. Registracija

Da bi neregistrirani korisnik dobio mogućnost kreiranja dnevnika spremlijenih recepata i planova ishrane, obavezno mora obaviti registraciju, koja se sastoji od popunjavanja forme korisničkih podataka. Sa lijeve strane nalazi se strelica koja korisnika vraća na početnu stranu.



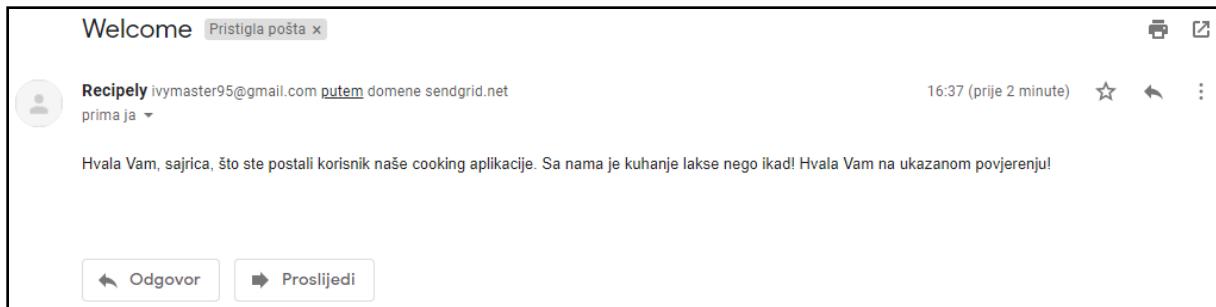
Slika 6.10. Stranica za registraciju korisnika

Na slici 6.11 je prikazana notifikacija o uspješnosti registracije korisnika koji sada ima pristup zaštićenim rutama. Nakon obavijesti, korisnik se preusmjerava na početnu stranicu.



Slika 6.11. Stranica za registraciju korisnika

Na slici 6.12 prikazan je izgled mail-a koji se šalje korisniku kao poruka dobrodošlice za prijavljivanje na Recipely aplikaciju. Slanje mail-a je implementirano kroz servis SendGrid koji je ranije spomenut.



Slika 6.12. Stranica za registraciju korisnika

Budući da je registracija povezana sa bazom Spoonacular API-a, u nastavku će se objasniti implementacija ove komunikacije.

Korišteni modul za ostvarivanje komunikacije:

authController

Prikaz koda:

```
1. exports.signup = catchAsync(async (req, res, next) => {
2.   const newUser = await User.create({
3.     username: req.body.username,
4.     firstname: req.body.firstname,
5.     lastname: req.body.lastname,
6.     email: req.body.email,
7.     password: req.body.password,
8.     passwordConfirm: req.body.passwordConfirm,
9.     photo: 'default.jpg'
10.   });
11.
12.   await new Email(newUser, '').sendWelcome();
```

Unutar metode *signup* se kreira objekat *newUser*, čije atributi čine podaci koji su pokupljeni iz forme za registraciju. Korisniku se dodjeljuje *defaultna* profilna slika, koja se kasnije može promijeniti. Potom se na predanu e-mail adresu šalje poruka dobrodošlice. Slanje poruke je obavljeno putem servisa SendGrid.

```
1.   params = {
2.     username: newUser.username,
3.     firstname: newUser.firstname,
4.     lastname: newUser.lastname,
5.     email: newUser.email
6.   };
7.
```

U nastavku se u objekat *params* spremaju korisnički podaci, koji će biti predani Spoonacular API-u na registraciju. Podaci se šalju u obliku *body-a* na link za generiranje korisnika. Link za generiranje korisnika je:

POST | <https://api.spoonacular.com/users/connect>

```

1. let odg = await axios.post(
2.   `${process.env.URL}/users/connect?apiKey=${process.env.API_KEY}`,
3.   params
4. );
5. if (!odg.data.username) {
6.   return next(new AppError('Korisnicko ime je zauzeto! Unesite drugo.', 4
00));
7. }
8. newUser.APIusername = odg.data.username;
9. newUser.APIpassword = odg.data.hash;
10.
11. await User.findByIdAndUpdate(newUser.id, newUser, {
12.   new: true,
13.   runValidators: true
14. });
15. createSendToken(newUser, 201, res);
16. );

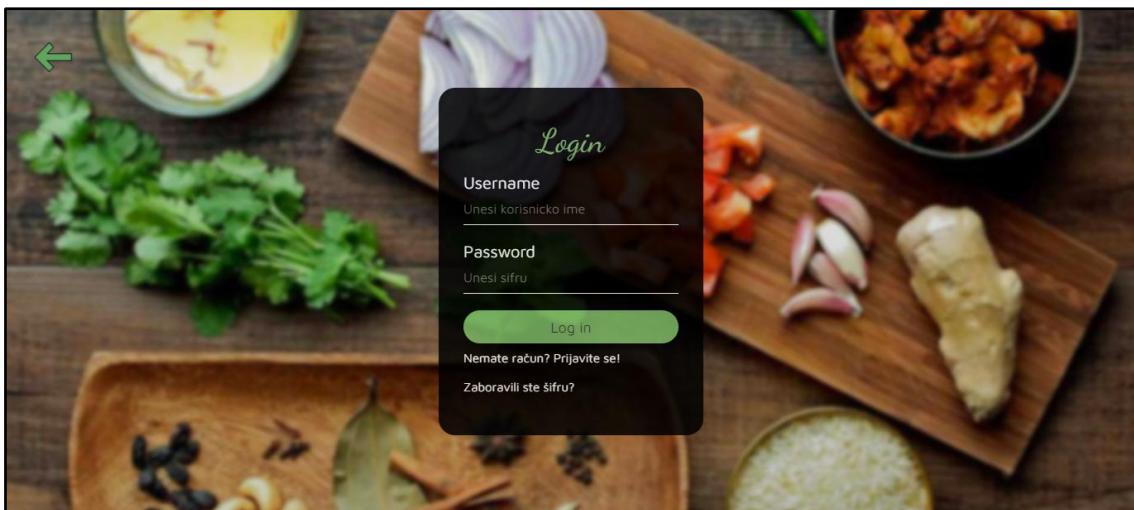
```

U varijabli *odg* su pohranjeni povratni podaci koje tvore API kreirano korisničko ime (*username*) i *hash*-ovana šifra (*hash*). Oba podatka su spremljena u Spoonacular API bazu podataka, i pomoću njih se uspostavljaju funkcije korisnički definisanih linkova. Korisnički definisani linkovi su oni linkovi Spoonacular API-a koji zahtijevaju predavanje korisničkih informacija za njihov rad [6].

Dobiveni API podaci se spremaju u Recipely app MongoDB bazu podataka. Kao *response* se Browseru šalje *token* koji omogućuje pristup zaštićenim rutama. Zaštićene rute podrazumijevaju rute koje su dostupne samo prijavljenim korisnicima (npr. planovi ishrane, spremljeni recepti i sl.) [6].

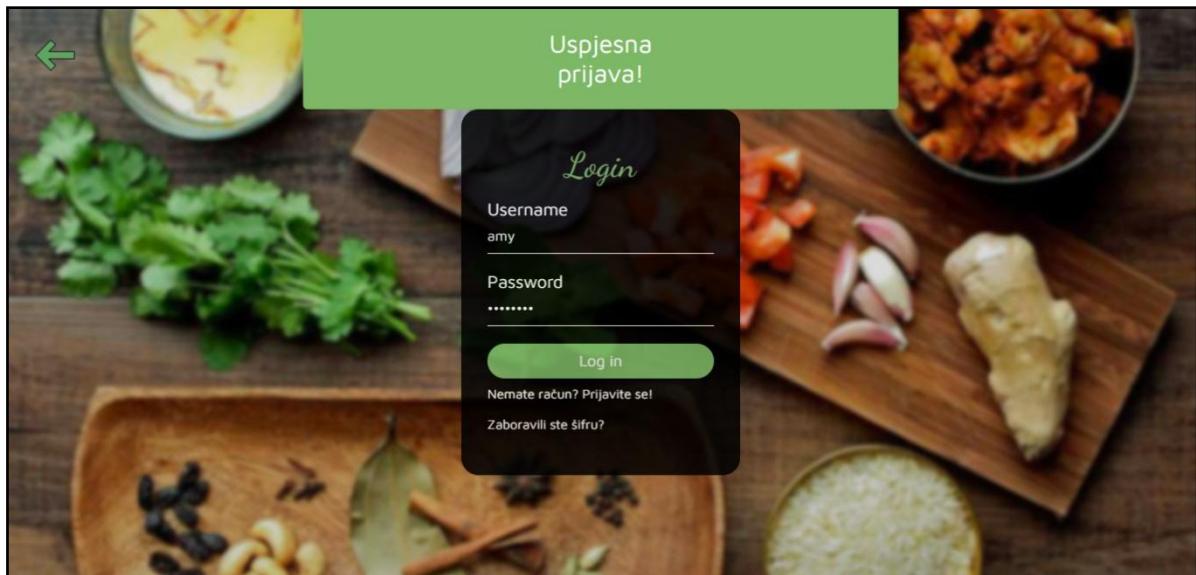
6.1.7. Prijava

Prijaviti se mogu samo registrirani korisnici pritiskom na dugme *Prijava*. Tokom rada sistema, metodom *isLoggedIn()* iz *authController-a* se provjerava da li je neki korisnik trenutno prijavljen. Ukoliko jeste, njegov *cookie token* se proslijeđuje rutama zaštićenim za registrirane korisnike. U ovisnosti o tome da li je prijavljen, korisniku će u navigacijskom meniju biti priloženi linkovi za registraciju/prijavu ili odjavu.



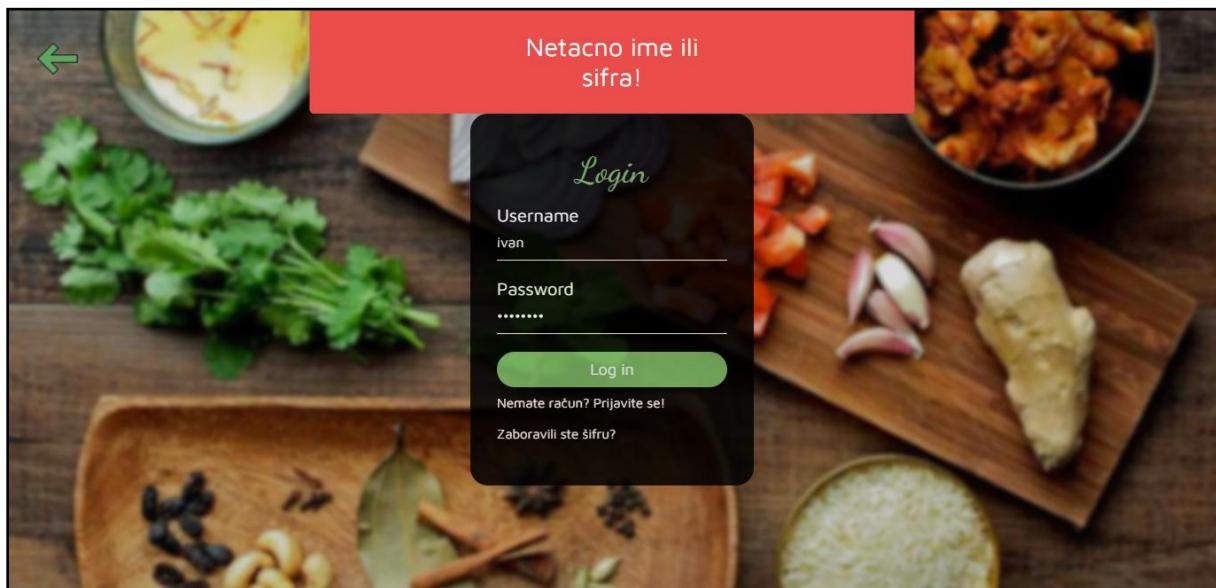
Slika 6.13. Stranica za prijavu korisnika

Na slici 6.13 prikazana je stranica za prijavu korisnika. Proces prijave je u potpunosti realiziran pomoću Recipely aplikacijom i nema doticaja sa Spoonacular API-em, stoga nije potrebno objašnjavati kod vezan uz isti.



Slika 6.14. Uspješna prijava korisnika

Na slici 6.14 prikazana je notifikacija o uspješnoj prijavi korisnika, nakon koje se korisnik preusmjerava na početnu stranicu, ovaj put sa mogućnostima pristupa zaštićenim rutama. Naravno, ukoliko korisnik unese korisničko ime koje ne postoji u bazi ili podatke koji nisu tačni, ispisuje mu se za to prikladna notifikacija, te mora ponovo izvršiti unos podataka.



Slika 6.15. Nespješna prijava korisnika

Na slici 6.15 prikazan je slučaj notifikacije ukoliko je korisnik unio netačne ili nepostojeće podatke.

6.2. Registrirani korisnici

Registrirani korisnici imaju mogućnost izmjene korisničkih informacija, koje uključuju ime, prezime, korisničku sliku, šifru i slično. Osim toga, ako se korisnik ne može prijaviti u sistem zbog toga što je zaboravio šifru, moguće je poslati zahtjev za promjenu iste. Registrirani korisnik može spremati recepte u svoj dnevnik. Svaki recept se može pregledati detaljno i urediti zabilješku recepta. Omogućeno je da korisnici mogu i brisati recepte iz dnevnika spremljenih recepata. Pored recepata, korisnici mogu spremiti i planove ishrane, unutar kojih dobijaju dnevna uputstva o načinu ishrane, te o receptima koji će se taj dan praviti.

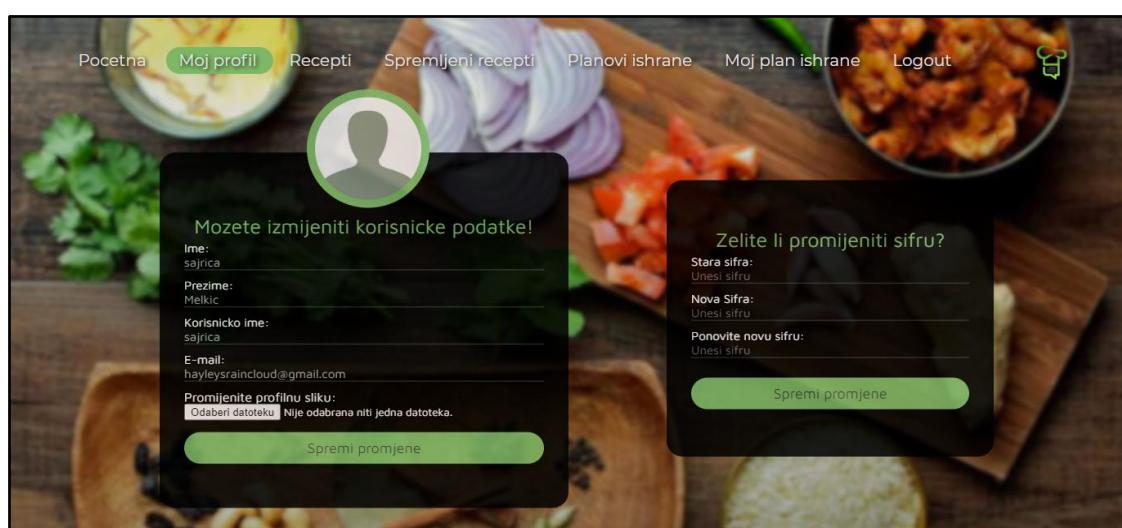
6.2.1. Izmjena korisničkih informacija

Korisničke informacije su dostupne u sklopu korisničkog profila. Kada je korisnik prijavljen, unutar navigacijskog menija se, na svakoj ruti, prikazuje link za korisnički profil. Navigacijski meni na generalnim stranicama (početna, informacijska, kontaktna, galerija recepata) kada je korisnik prijavljen izgleda kao na slici 6.16.



Slika 6.16. Navigacijski meni u slučaju prijavljenog korisnika

Kada korisnik odabere link *Moj profil*, prikazuje mu se profil sa podacima, kao na slici 6.17.

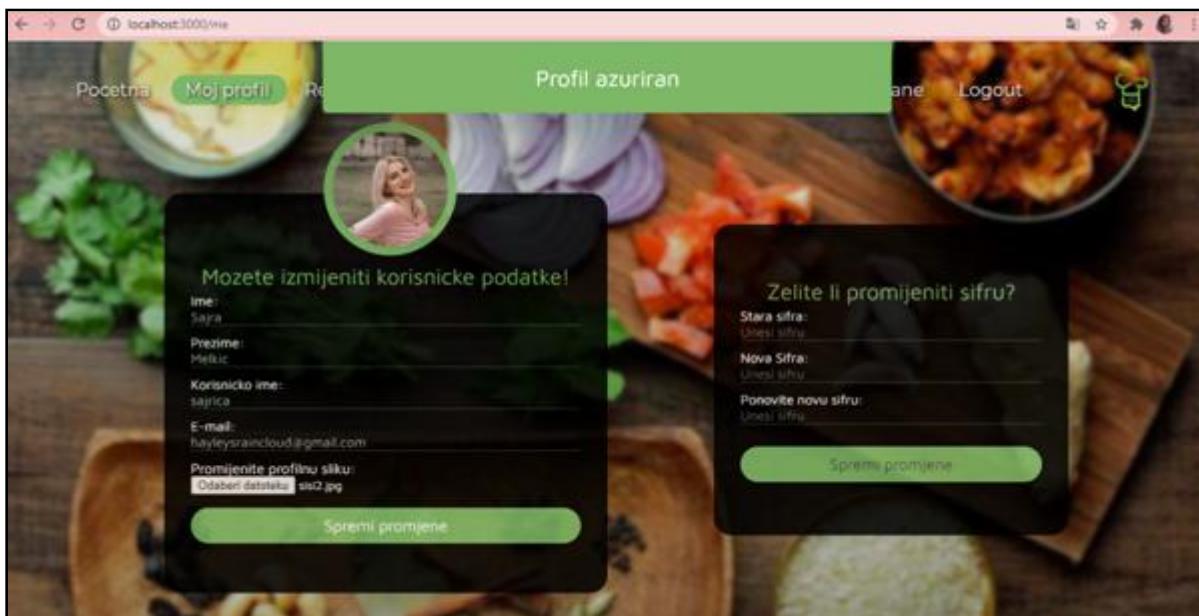


Slika 6.17. Prikaz korisničkog profila

Kada se otvori korisnički profil, unutar navigacijskog menija se pojavljuju linkovi za pristup spremjenim receptima, planovima ishrane i vlastitom planu ishrane. Ove opcije nisu dodane na navigacijske menije generalnih stranica zbog toga što bi tada svaki navigacijski meni izgledao neuredno. Na ovaj način se svakoj zaštićenoj akciji može pristupiti preko korisničkog profila.

Korisnički profil ima dva dijela – na lijevom dijelu se prikazuje *default-na* korisnička slika koja se dodaje prilikom registracije. Ova slika se može izmijeniti tako da korisnik odabere novu datoteku i pritisne dugme *Spremi promjene*. Osim slike, moguće je mijenjati i sve druge korisničke podatke, kao što su ime, prezime, korisničko ime, e-mail i šifra. Šifra se sprema u posebnoj formi, tako što se unosi stara šifra i nova šifra (dvaput).

Na slici 6.18. prikazana je notifikacija koja se daje korisniku nakon što odabere *Spremi promjene*. Notifikacija se na stranici zadržava nekoliko sekundi. Na profilu je promijenjeno korisničko ime i profilna slika.

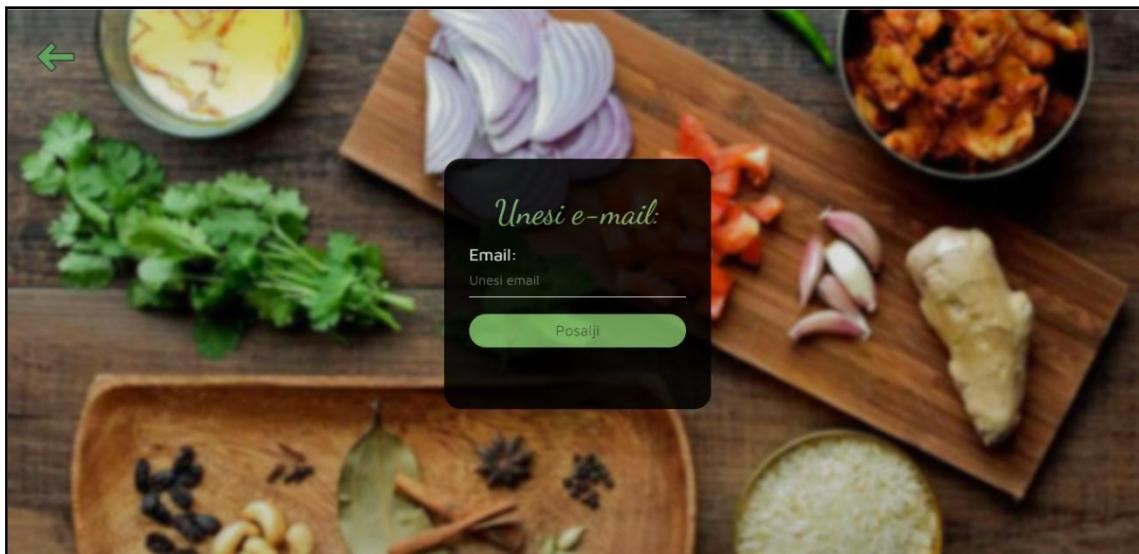


Slika 6.18. Prikaz izmjene korisničkog profila

Izmjena korisničkog profila se vrši u Recipely bazi podataka, odnosno bez komunikacije sa Spoonacular API-em, stoga taj dio koda nije potrebno objašnjavati.

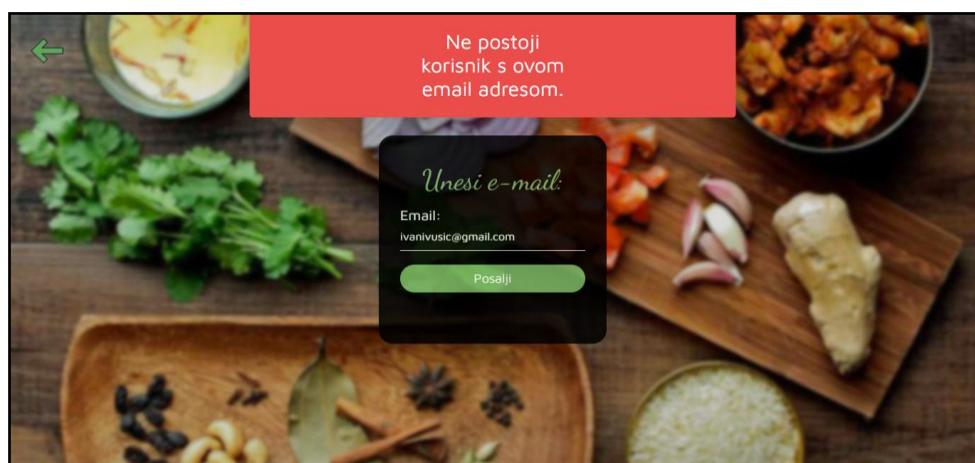
6.2.2. Zaboravljeni šifra

Link za promjenu zaboravljenih šifri je dostupan na stranici za prijavu. Pritisom na link otvara se stranica čiji je izgled prikazan na slici 6.19. Na toj stranici korisnik unosi svoj e-mail, na koji će se poslati URL za unos nove šifre. Ovaj proces je ostvaren pomoću servisa SendGrid.



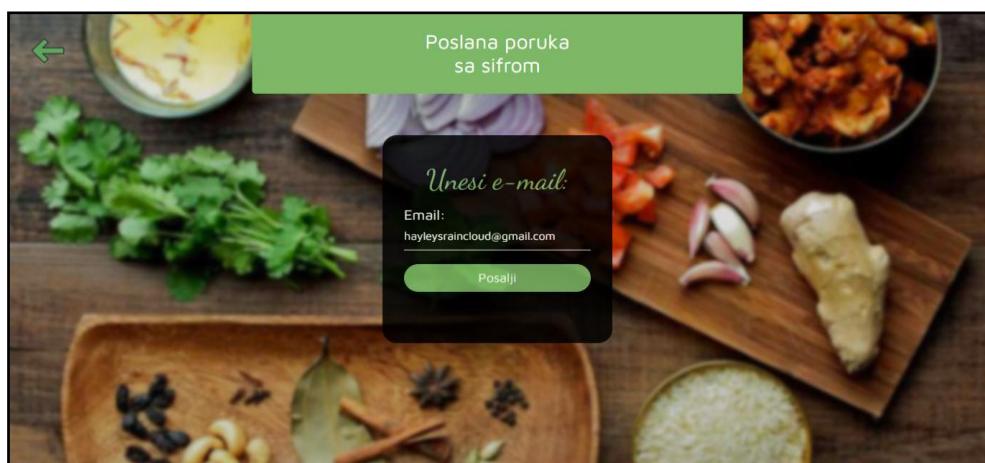
Slika 6.19. Stranica za unos e-maila na koji će se poslati zahtjev

Ako korisnik unese e-mail koji nije registriran u bazi, pojavljuje mu se notifikacija kao na slici 6.20, te se čeka da korisnik unese novi e-mail, koji je važeći.



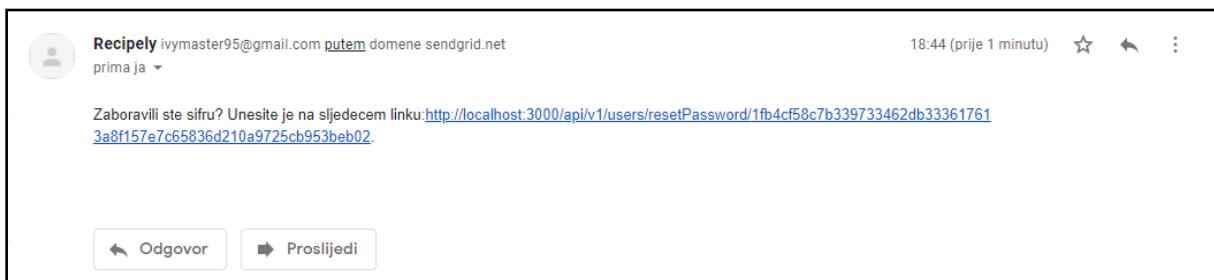
Slika 6.20. Nepostojeci korisnik

Ako je korisnik unio e-mail koji je registriran, pojavljuje se sljedeća notifikacija:



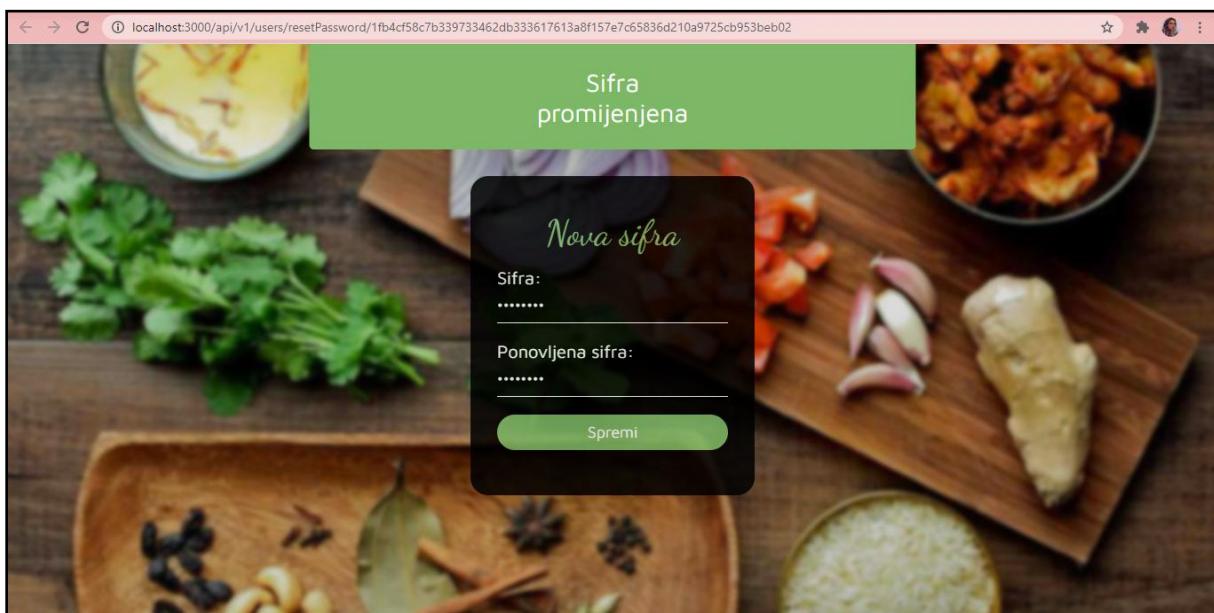
Slika 6.21. Uspješno poslan e-mail

Kada se otvorí e-mail, dobije se poruka koja je prikazana na slici 6.22.



Slika 6.22. Uspješno primljen e-mail

Ovim linkom se prosljeđuje *hash*-ovani token, čija se originalna verzija spremi u bazi podataka. Upotrebljivost ovog tokena traje određeni vremenski interval, nakon čega postaje nevažeći. Ako korisnik ne promijeni šifru u tom vremenskom intervalu, ponovo će morati slati zahtjev za promjenu zaboravljenih šifri. Na slici 6.23 prikazana je obavijest koja se ispisuje korisniku ukoliko je sifra uspješno promijenjena.



Slika 6.23. Uspješno promijenjena korisnička šifra

6.2.3. Prikaz jednog recepta

Detaljan prikaz informacija o receptu korisnik dobija pritiskom na dugme *Detalji* u sklopu galerije svih recepata. Time Recipely app šalje zahtjev Spoonacular API-u za povlačenje podataka o receptu sa ID-om koji se nalazi u linku.

Na slici 6.24 prikazan je jedan specifični nasumično odabran recept iz galerije recepata.

Pocetna Moj profil Recepti Spremljeni recepti Planovi ishrane Moj plan ishrane Logout

Vegan Taco bowls with Cilantro Lime Cauliflower Rice

SACUVAJTE RECEPT VEC DANAS
TREBATE POMOC?: (037) 311 829

Sacuvajte recept uz komentar:

Vas komentar:

Sacuvaj

Koraci recepta:

1.

Sastojci:
cauliflower - spread - water - meat - nuts - rice - cooking oil -
Posudje:
food processor aluminum foil baking sheet bowl oven
Postupak:
Set the nuts to soak in a bowl of water 2-8 hours before preparing the meat. Preheat the oven to 375 F and line a baking sheet with aluminum foil. Finely chop the cauliflower into rice-sized pieces, or pulse in the food processor until the desired consistency. Toss with the oil and spread onto the baking sheet in an even layer.

2.

Sastojci:
cauliflower rice - guacamole - meat -
Posudje:
oven
Postupak:
Bake for 18 minutes, stirring halfway through. While the cauliflower rice is cooking, prepare the guacamole and taco meat.

3.

Sastojci:
avocado -
Posudje:
bowl
Postupak:
Cut the avocado into large chunks and place into a small bowl.

4.

Sastojci:
lime juice - cilantro - jalapeno pepper - avocado - spices - onion -
Posudje:

Postupak:
Add the lime juice and mash the avocado to a chunky puree or your preferred consistency. Stir in the cilantro, jalapeno, onion and spices. Set aside.

5.

Sastojci:
ground beef - water - nuts -
Posudje:
food processor
Postupak:
Remove the nuts from the water and place into a small food processor. Discard the water. Pulse the nuts until they've reached a ground meat consistency. I find about 5-7 pulses works.

6.

Sastojci:
cauliflower rice - chili powder - fresh cilantro - garlic powder - cauliflower - guacamole - cilantro - tomato - cumin - lime - meat - salt -
Posudje:
mixing bowl
Postupak:
Transfer to a bowl and stir in the salt, cumin, garlic powder, and red chili powder. Set aside. When the cauliflower is done, transfer to a mixing bowl while still warm and stir in the lime, cilantro, and salt. To assemble the bowls, layer 1 cup of the cauliflower rice in a bowl with taco nut meat, a dollop of guacamole, fresh tomatoes, and a sprinkling of fresh cilantro.

Slika 6.24. Prikaz jednog recepta

Prikaz karakteristika recepta zavisi od podataka iz baze Spoonacular API-a, stoga će se konekcija objasniti u nastavku.

Korišteni modul za prikaz recepta:

receiptController

Prikaz koda:

```
1. exports.getReceptInfo = catchAsync(async (req, res, next) => {
2.   //Dobijanje recepta
3.   const url = `${process.env.URL}/recipes/${req.params.id}/information?apiKey=${process.env.API_KEY}`;
4.   let odg = await axios.get(url);
```

URL koji se koristi za dobijanje specifičnog recepta iz baze API-a je sljedeći [6]:

GET | <https://api.spoonacular.com/recipes/{id}/information>

Modul receiptController nudi metodu *getReceptInfo*, koja kao rezultat vraća jednu instancu modela Recept. Za komunikaciju s API-em se, kao i u prethodnim slučajevima, koristi modul *axios*.

U varijabli *url* se spremaju navedeni URL, kojem se predaje API ID traženog recepta. Odgovor, koji sadrži podatke o receptu, se spremaju u varijablu *odg*.

U nastavku je objašnjeno dobijanje tzv. *widgeta*. Widgeti predstavljaju kolekciju informacija u obliku HTML podataka. Widget koji se koristi u aplikaciji služi za prikaz sastojaka određenog recepta [6].

```
1. //Widgeti
2. const url2 = `${process.env.URL}/recipes/${req.params.id}/ingredientWidget?apiKey=${process.env.API_KEY}`;
3. let odg2 = await axios.get(url2);
4. odg2 = odg2.data.split('<div id="spoonacular-ingredient-vis-list">')[0];
5. odg2 = odg2.split('<div style="clear:both"></div>')[1];
```

URL koji se koristi za dobijanje widgeta iz baze API-a je sljedeći [6]:

GET | <https://api.spoonacular.com/recipes/{id}/ingredientWidget>

U varijabli *url2* se spremaju navedeni URL, kojem se predaje API ID traženog recepta. Odgovor onosno *widget* se spremaju u varijablu *odg*. Svaki widget sadrži ime sastojka, količinu i sliku. Pošto HTML odgovor sadrži dosta nepotrebnih elemenata, njih je potrebno izbaciti. To se radi funkcijom *split()*, te pažljivim odabirom stringa prema kojem se vrši dijeljenje odgovora. Taj string predstavlja *div* element u HTML-u, kao što je vidljivo u priloženom kodu. Parsirani HTML odgovor se prosljeđuje predlošku.

U nastavku se opisuje način na koji se dobijaju koraci recepta iz baze Spoonacular API.

```
1. //Koraci
2. const url4 = `${process.env.URL}/recipes/${
3.   req.params.id
4. }/analyzedInstructions?apiKey=${process.env.API_KEY}`;
5. let odg4 = await axios.get(url4);
6. res.status(200).render('SingleReceipt', {
7.   receipt: odg4.data,
8.   widget: odg2,
9.   koraci: odg4.data[0].steps,
10.  type: req.type
11. });
12. );
```

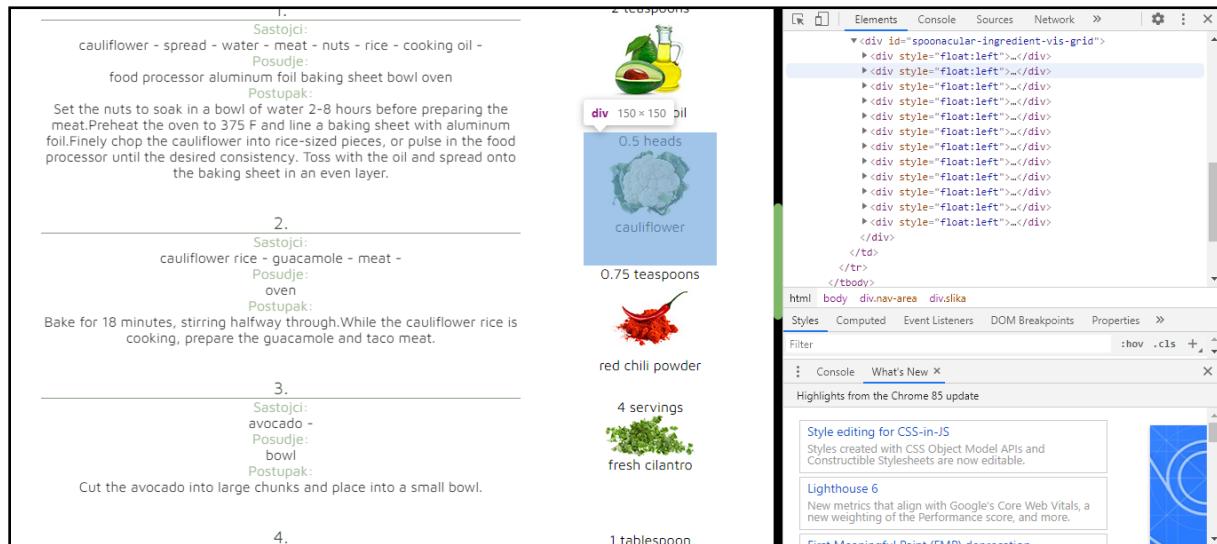
URL koji se koristi za dobijanje widgeta iz baze API-a je sljedeći [6]:

GET | <https://api.spoonacular.com/recipes/{id}/analyzedInstructions>

U varijabli *url4* se spremaju navedeni URL, kojem se predaje API ID traženog recepta. Odgovor od Spoonacular API-a sadrži detaljno objašnjene korake izrade jela. Pošto u odgovoru postoje drugi nebitni podaci, kao koraci se prosljeđuju samo prvi članovi niza steps.

Ovim se završava prosljeđivanje detalja o receptu od API-a do aplikacije Recipely.

U nastavku se opisuje na koji način se uređuju CSS atributi dobijenog widgeta, koji se sastoji od imena sastojka, količine i slike.



Slika 6.25. Provjera (*inspect*) stranice

Da bi se izmijenile CSS klase kojima pripadaju widgeti, potrebno je dobiti informaciju o njihovom nazivu. To se obavlja provjerom stranice, te prikazom imena klase kojoj pripada widget. ID kojim se uređuje način prikaza widgeta je *id = spoonacular-ingredient-vis-grid* a klasa kojom se upravlja karakteristikama jednog widgeta je *class = spoonacular-ingredient*.

Korišteni modul za uređivanje klasa API-a:

savedChange.css

Prikaz koda:

```
1. #spoonacular-ingredient-vis-grid {  
2.     margin-left: 40px;  
3. }  
4.  
5. /*kolicina doze, slika, sastojak*/  
6. .spoonacular-ingredient {  
7.     font-size: 16px;  
8.     text-decoration: solid;  
9.     text-align: center;  
10.    height: 150px;  
11.    width: 150px;  
12. }
```

Cjelokupni niz widgeta se pomijera ulijevo za 40px. Svaki sastojak ima veličinu fonta 16px, centriira se i rezolucije je 150x150px. Na taj način su widgeti raspodijeljeni na stranici za prikaz recepta.

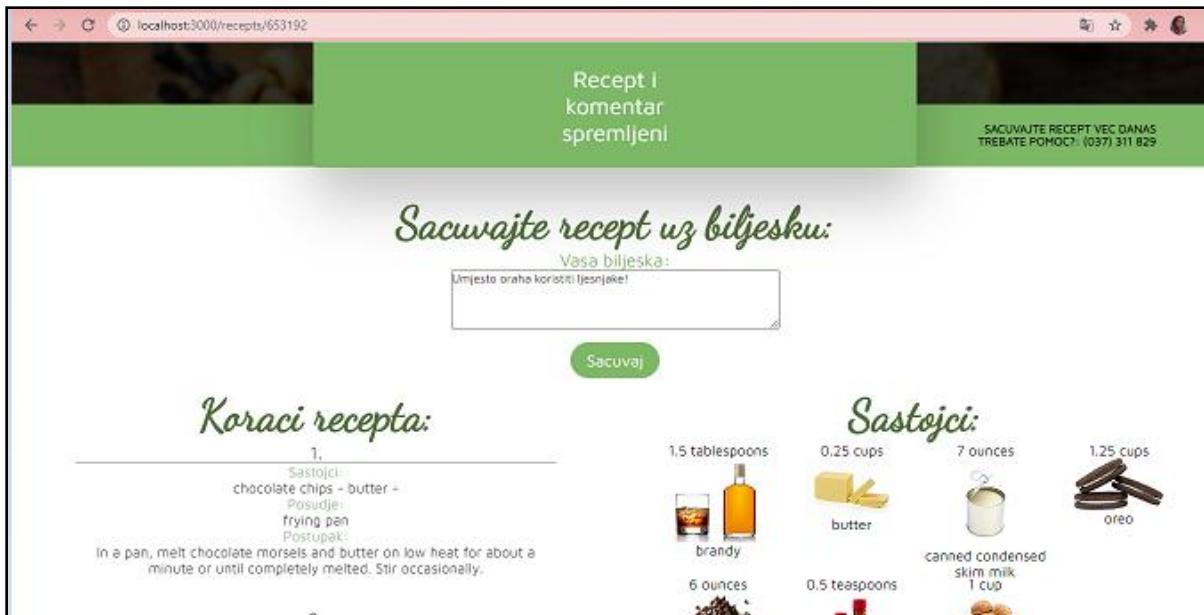
6.2.4. Spremanje recepta

Spremanje recepta je moguće obaviti pritiskom na dugme *Spremi*. Korisnik, naravno, može dodati zabilješku uz taj recept, koju kasnije može urediti. Opcija dodavanja zabilješke je opcionalna. Zabilješka, sa ID-em recepta se spremava u bazu Recipely aplikacije. Na slici 6.26 prikazan je primjer dodavanja zabilješke na jedan recept „No-Bake Fudge Brandy Brownies“



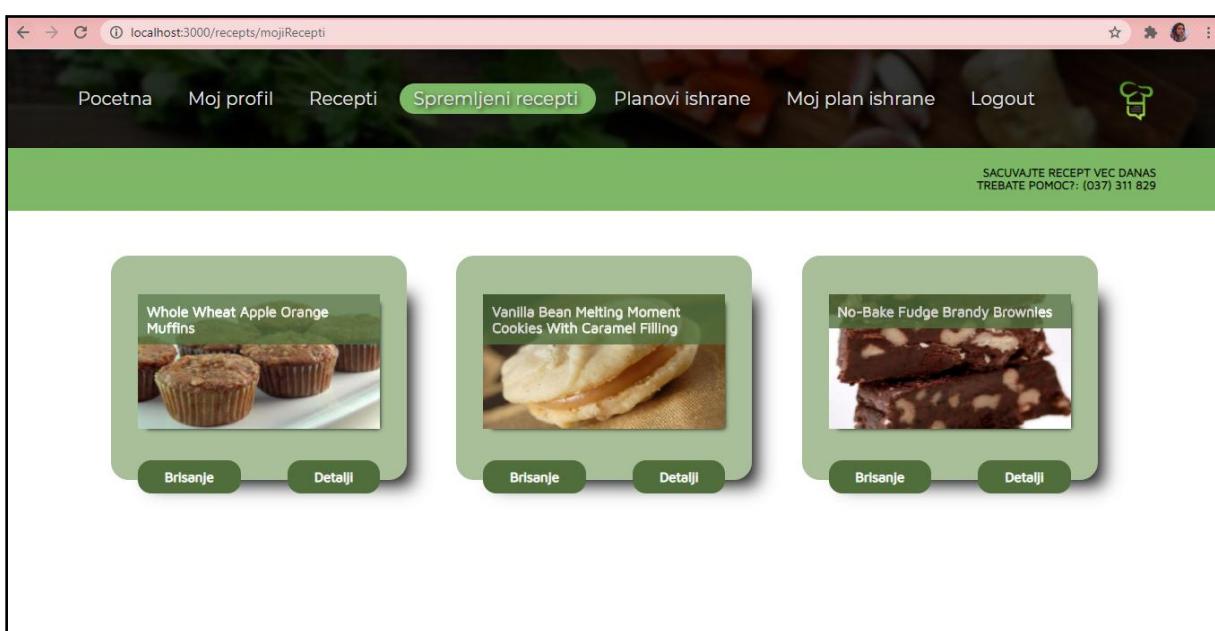
Slika 6.26. Primjer zabilješke

Na slici 6.27 prikazana je notifikacija koja se pojavljuje korisniku na nekoliko sekundi, nakon što spremi recept. Nakon toga, osvježava se stranica i korisnik se preusmjerava na dnevnik receptata.



Slika 6.27. Spremanje recepta uz zabilješku

Nakon što se recept spremi, korisnik se preusmjerava na stranicu spremljenih recepata odnosno dnevnik, kao što je prikazano na slici 6.28.



Slika 6.28. Dnevnik recepata tekućeg korisnika

Za prikaz dnevnika, Recipely aplikacija uzima podatke iz Spoonacular API-a kao što su naziv recepta i slika recepta. Ukoliko se recept iz dnevnika želi izbrisati, potrebno je odabratи dugme Brisanje, a ukoliko se želi detaljnije prikazati spremljeni recept, potrebno je odabratи dugme Detalji. O pregledu i izmjeni spremljenih recepata će biti detaljnije opisano u idućem poglavljju.

U nastavku će se opisati kod koji poziva naziv i sliku recepta iz baze podataka Spoonacular API-a.

Korišteni modul za ostvarivanje komunikacije:

receiptController

Prikaz koda:

```
1. exports.showUsersReceipts = catchAsync(async (req, res, next) => {
2.   const user = await User.findById(req.user._id);
3.
4.   let recepti = [];
5.   recepti = user.recepti.map(async el => {
6.     const url = `${process.env.URL}/recipes/${el.id}/information?apiKey=${
7.       process.env.API_KEY
8.     }`;
9.     let odg = axios.get(url);
10.    return odg;
11.  });
12.
13.  recepti = await Promise.all(recepti);
14.  recepti = recepti.map(el => {
15.   return el.data;
16. });
17.
18.  res.status(200).render('savedReceipts', {
19.   status: user.status,
20.   receipts: recepti,
21.   type: req.type
22. });
23.});
```

URL koji se koristi za dobijanje recepata iz baze API-a je sljedeći [6]:

GET | <https://api.spoonacular.com/recipes/{id}/information>

Modul receiptController nudi metodu *showUsersReceipts*, koja kao rezultat vraća podatke o receptu sa odgovarajućim ID-em. Za komunikaciju s API-em se koristi modul *axios*.

U niz *recepti* se pohranjuje atribut *recepti* instance trenutnog *user-a*. Za svaki element tog niza, koji sadrži ID jednog po jednog recepta iz baze Recipely app, kreira se HTTP zahtjev, koji je spremlijen u obliku *promise-a*. Svi *promises* se u isto vrijeme šalju API-u, nakon čega API vraća podatke o željenim receptima. Podaci se stavljuju u predložak i prikazuju korisniku.

6.2.5. Pregled i izmjena spremljenih recepata

Pregled spremljenog recepta je sličan kao i detaljni pregled bilo kojeg recepta iz galerije, uz dodatak prikaza dodane zabilješke, što znači da se komunikacija sa Spoonacular API-em implementira na isti način. Osim toga, iz Recipely baze se povlači zabilješka koja je spremljena uz taj recept.

Na slici 6.29 prikazan je izgled stranice za prikaz jednog određenog spremljenog recepta.



SACUVAJTE RECEPT VEC DANAS
TREBATE POMOC?: (037) 311 829

Promijenite biljesku:

Vasa biljeska:

Umjesto oraha koristiti lješnjake

Spremi

Koraci recepta:

1.
Sastojci:
chocolate chips - butter -
Posudje:
frying pan
Postupak:
In a pan, melt chocolate morsels and butter on low heat for about a minute or until completely melted. Stir occasionally.



2.
Sastojci:
walnuts - oreo cookies -
Posudje:
bowl
Postupak:
Remove from heat. In a large bowl, mix Oreo crumbs (set aside a tablespoon or two) and walnuts.

3.
Sastojci:
vanilla extract - sweetened condensed milk - chocolate -
Posudje:
Postupak:
Add condensed milk, vanilla extract, and chocolate mixture.

4.
Sastojci:
brandy - butter - sugar -
Posudje:
aluminum foil
Postupak:
Add brandy and mix well. If you're feeling a little gutsy, add another half a tablespoon of brandy for a solid kick! Line whatever container you want to put it in with foil or grease it with butter and sugar. Press the mixture firmly onto bottom of container.

5.
Sastojci:
oreo cookies -
Posudje:
Postupak:
Garnish with Oreo crumbs on top. Refrigerate for about two hours and enjoy!

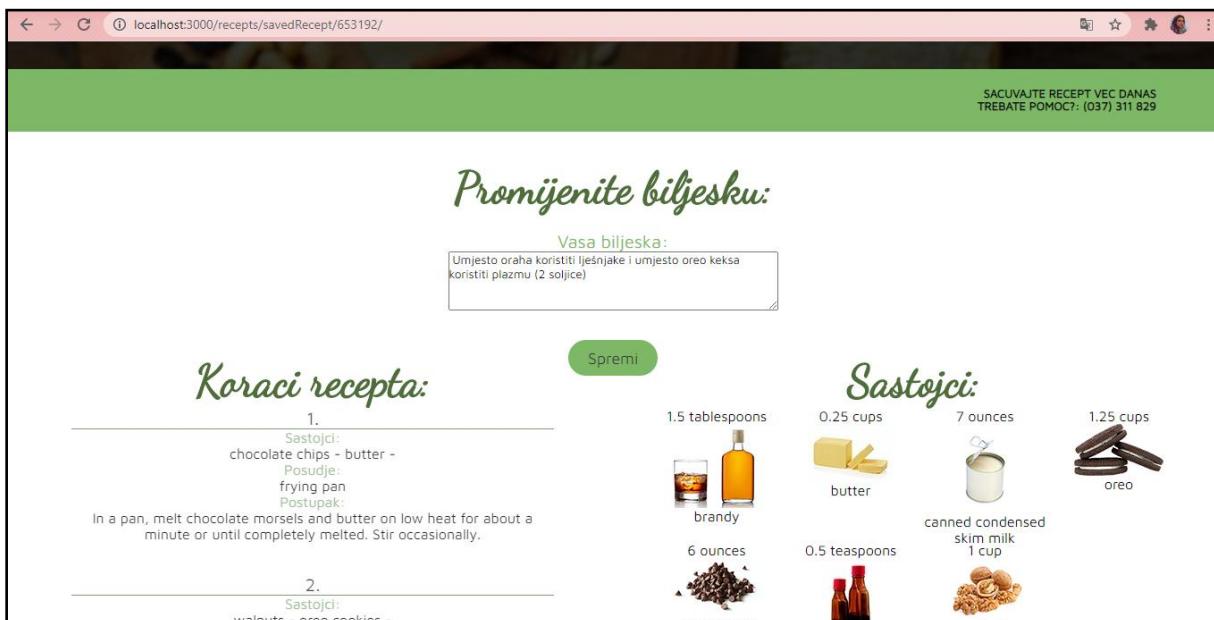
Slika 6.30. Prikaz spremlijenog recepta

Na slici 6.31 prikazana je notifikacija koja se nakratko pojavljuje korisniku nakon što promijeni zabilješku na receptu.



Slika 6.31. Mijenjanje bilješke na receptu

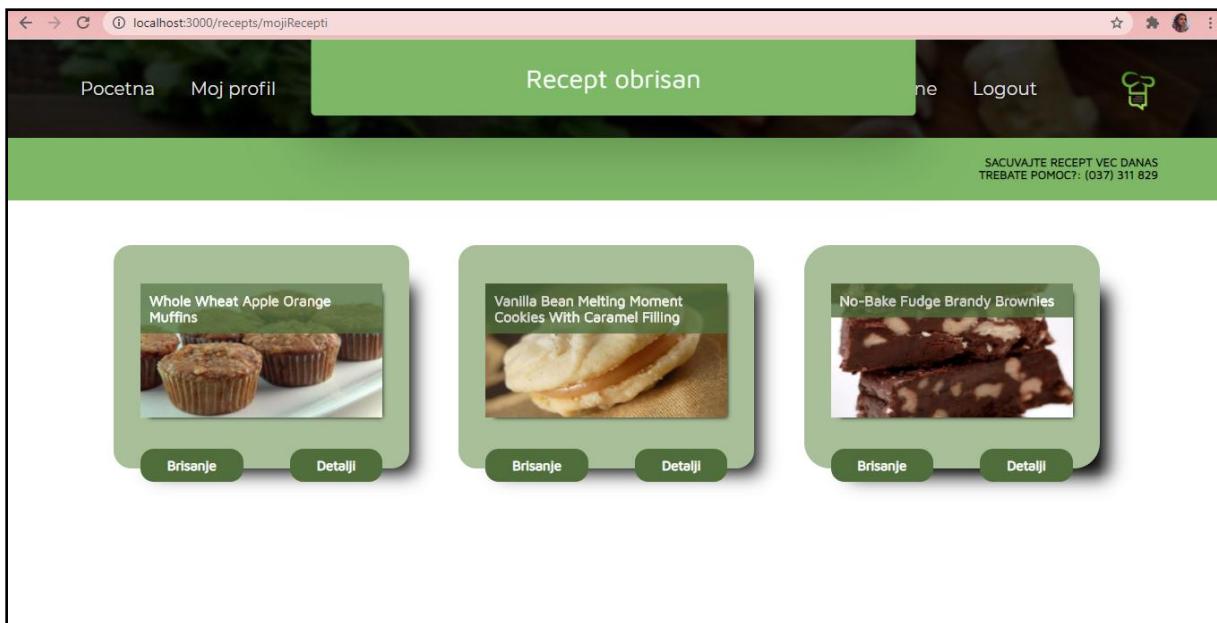
Na slici 6.32 prikazana je nova bilješka koja se opet kasnije, ukoliko korisnik želi, može izmijeniti!



Slika 6.32. Mijenjanje bilješke na receptu

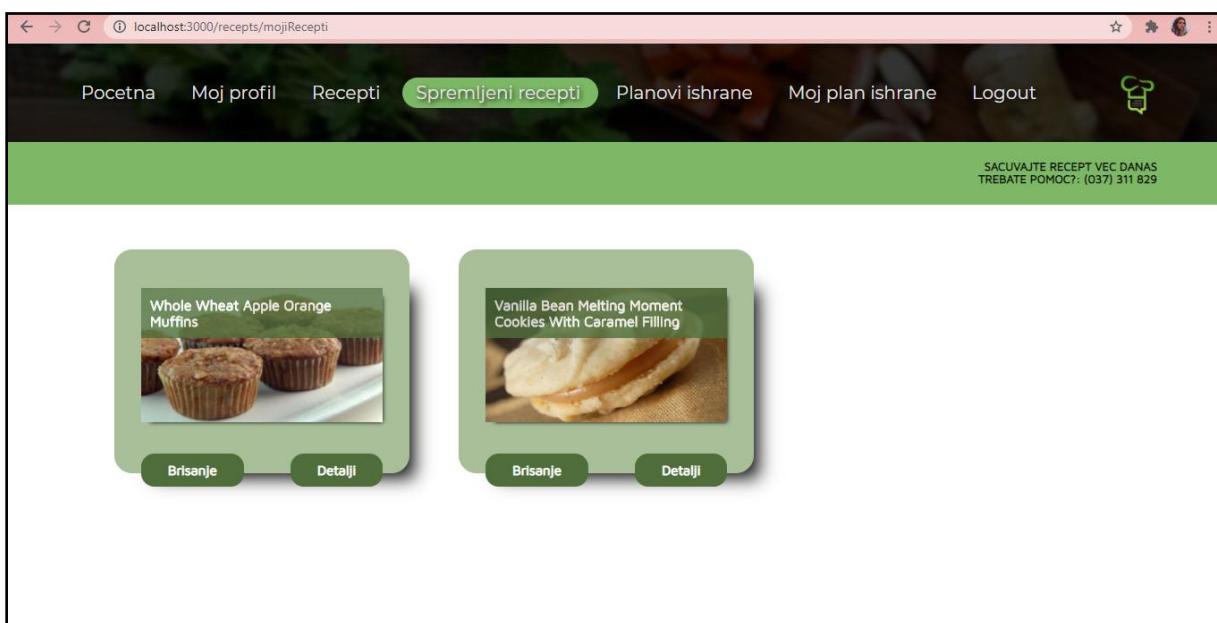
6.2.6. Brisanje recepata

Svaki registrirani korisnik može obrisati recept iz svog dnevnika spremljenih recepata. Opcija brisanja je dostupna unutar dnevnika putem *buttona Brisanje*. Ako korisnik pritisne ovo dugme, pojavljuje mu se notifikacija o uspješnom brisanju recepta, stranica se osvježava i prikazuje se dnevnik bez obrisanog recepta. Notifikacija koja se prikazuje korisniku prikazan je na slici 6.33. Izvršeno je brisanje trećeg recepta „No-Bake Fudge Brandy Brownies“.



Slika 6.33. Brisanje zadnje dodanog recepta

Na slici 6.34 prikazan je dnevnik spremljenih recepata nakon što se stranica automatski osvježi poslije brisanja. Kao što se može vidjeti, ne postoji obrisani recept.



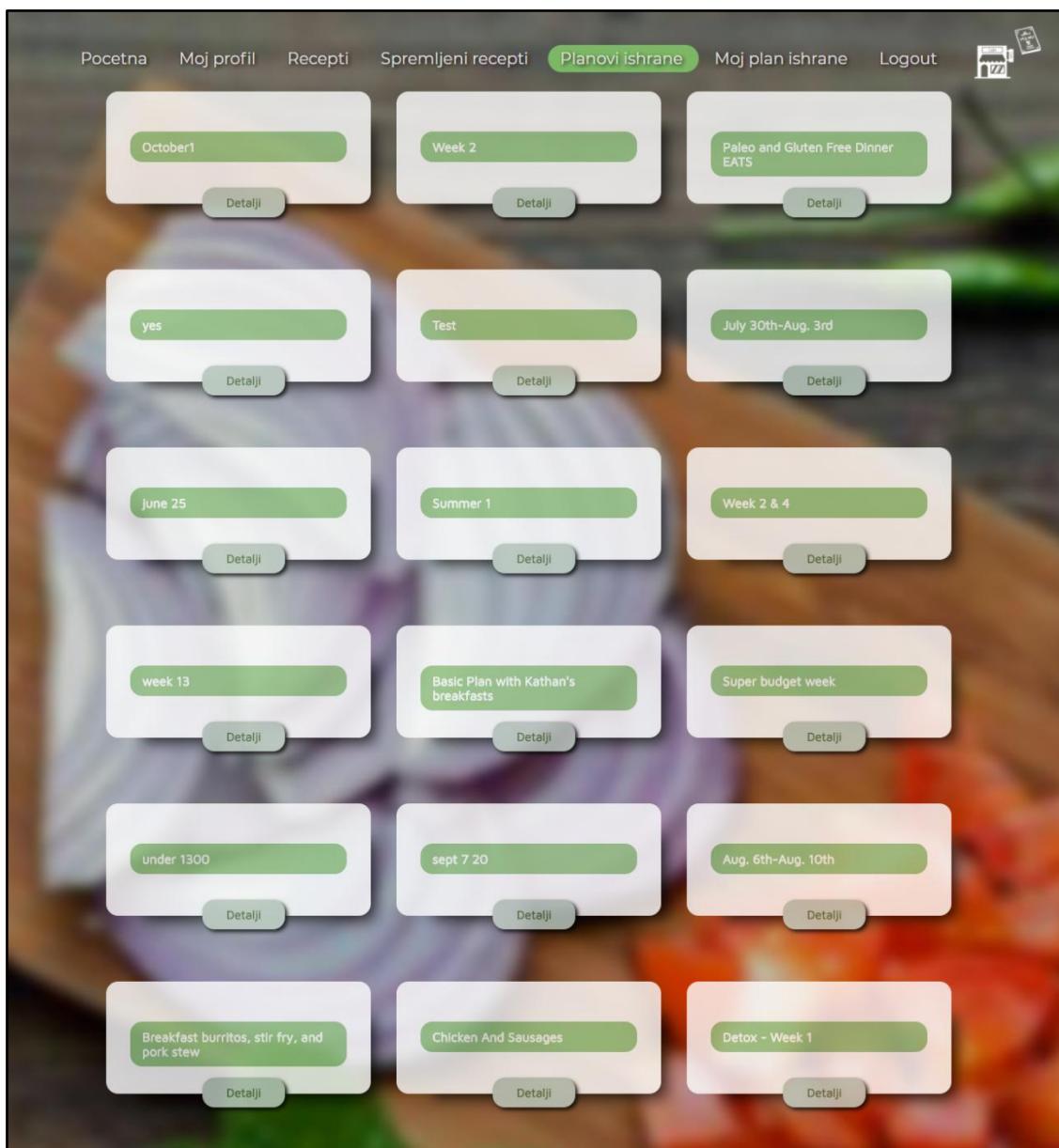
Slika 6.34. Dnevnik spremljenih recepata nakon brisanja jednog recepta

Naravno, brisanje se ne odvija na Spoonacular API-u već u Recipely bazi podataka. Pri zahtjevu za brisanje, predaje se ID recepta, koji se pronađe u bazi i izbacuje se iz niza spremljenih recepata tekućeg korisnika [6].

6.2.7. Pregled planova ishrane

Plan ishrane je višednevno uputstvo korisniku za pravljenje recepata. Svaki plan ishrane traje određeni broj dana i prikazuje korisniku koje tačno recepte da pravi. U slučaju pridržavanja konzumiranja preporučenih recepata, plan ishrane prikazuje i koliki je dnevni unos nutricionih vrijednosti kao što su kalorije, masnoće, karbohidrati i proteini. Pregled svih planova ishrane je moguć samo registriranim korisnicima.

Planovi ishrane su vidljivi na linku „*Planovi ishrane*“, gdje se odabire 40 nasumičnih planova iz baze Spoonacular API-a. Na slici 6.35 prikazan je jedan dio planova ishrane na stranici.



Slika 6.35. Planovi ishrane

Pošto se planovi ishrane dobijaju iz baze Spoonacular API-a, u nastavku će biti objašnen princip povezivanja Recipely aplikacije i Spoonacular baze podataka.

Korišteni modul za ostvarivanje komunikacije:

mealController

Prikaz koda:

```
1. exports.getAllMeals = catchAsync(async (req, res, next) => {
2.   let odg;
3.   try {
4.     odg = await axios.get(
5.       `${process.env.URL}/mealplanner/public-templates?apiKey=${
6.         process.env.API_KEY
7.       }`
8.     );
9.   } catch (err) {
10.    return next(new AppError('Uppps. Try again later', 400));
11.  }
12.  res.status(200).render('templates', {
13.    templates: odg.data.templates.sort(() => Math.random() -
14.      0.5).slice(1, 40),
15.    type: req.type
16.  });
17.});
```

URL koji se koristi za dobijanje planova ishrane iz baze API-a je sljedeći [6]:

GET | <https://api.spoonacular.com/mealplanner/public-templates>

Modul mealController nudi metodu *getAllMeals*, koja kao rezultat vraća sve planove ishrane koji postoje u bazi Spoonacular API-a. Za komunikaciju s API-em se, kao i u prethodnim slučajevima, koristi modul *axios*. Budući da API sadrži ogroman broj planova ishrane, što nije poželjno prikazivati na samo jednoj stranici, implementiran je nasumični odabir i prikaz samo prvih 40 planova. Modifikacija je obavljena na nizu svih dobijenih planova.

Ako se desi pogreška, korisnik ne može dobiti povratne informacije od API-a. U suprotnom, *response* se sastoji od predloška *templates* (to su planovi ishrane), u kojem su ubaćeni dobijeni podaci. Osim toga, pohranjuje se i tip korisnika. Ovim se provjerava da li je korisnik prijavljen ili ne. Ako je korisnik prijavljen *type* će imati vrijednost *korisnik*, a u suprotnom će imati vrijednost *none*.

6.2.8. Spremanje planova ishrane

Da bi korisnik spremio plan ishrane, mora otvoriti određeni plan, pritiskom na dugme *Detalji*. Tada se korisniku prikazuje stranica koja je prikazana na slici 6.36. Najprije se na stranici nalazi opcija za spremanje tog plana. Spremanje plana se implementira kroz odbir datuma od kojeg će započeti prvi dan plana ishrane. Ako se odabere današnji datum (što je preporučljivo), danas će biti prvi dan plana. Ako se odabere neki datum u budućnosti, na primjer 25.09.2020, plan ishrane će započeti tek od tog datuma, što znači da dotad korisnik

neće moći vidjeti sadržaj prvog dana ishrane, sve do 25.09. Ukoliko se odabere datum iz prošlosti, taj datum postaje prvi dan plana ishrane. Ako je odabran datum prije pet dana, znači da će danas biti peti dan plana ishrane i korisniku će se ispisati recepti koji se prave petog dana. Na slici 6.36 prikazan je izgled jednog plana ishrane.

Korisnici mogu brisati svoj plan ishrane ili jednostavno dodati novi – čime se zbrajaju nutricione vrijednosti i recepti odnosno obroci koji se koriste.

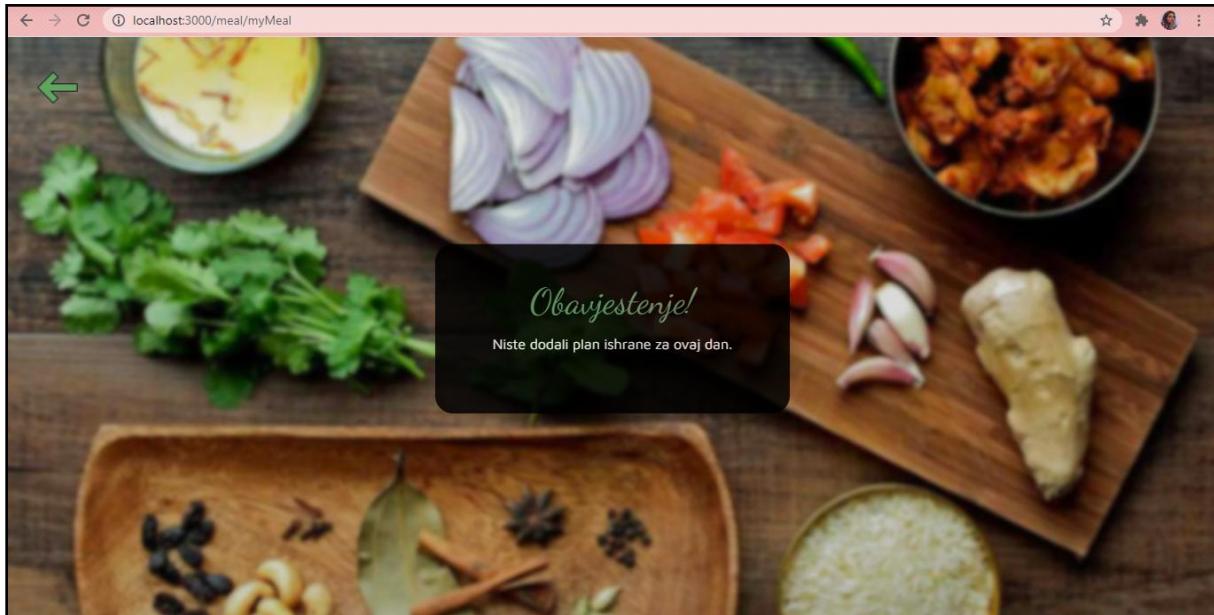
The screenshot shows a meal planning application interface. At the top, there is a navigation bar with links: Pocetna, Moj profil, Recepti, Spremljeni recepti, Planovi ishrane, Moj plan ishrane, Logout, and a user icon. Below the navigation bar, a green header bar displays the text "SACUVAJTE RECEPT VEC DANAS TREBATE POMOC? (037) 311 829". The main content area is titled "Spremite plan ishrane*" and includes a date input field "dd. mm. 9999." with a calendar icon. A note below the date input states: "Plan spremate tako da odaberete datum od kojeg će započeti. Svakim dodavanjem se podaci zbrajuju na prethodno dodane. Ukoliko se želi novi početak, potrebno je obrisati stare." A green "Spremi" button is located below the date input. The page then displays seven days of the meal plan:

- 1. dan**
 - Nutriciona vrijednost:**
 - Calories - 990cal
 - Fat - 69g
 - Carbohydrates - 53g
 - Protein - 47g
 - Obroci za taj dan:**
 - Egg & Cheese Sandwich On Whole Wheat
 - Peanut Butter Sandwich and Carrots
 - Veggie Quesadillas with Cilantro Yogurt Dip
- 2. dan**
 - Nutriciona vrijednost:**
 - Calories - 658cal
 - Fat - 28g
 - Carbohydrates - 90g
 - Protein - 19g
 - Obroci za taj dan:**
 - Nature Valley Crunchy Granola Bars Peanut Butter - 6 CT
 - Tuna Sandwich With Apple
 - Turkey Salad
- 3. dan**
 - Nutriciona vrijednost:**
 - Calories - 348cal
 - Fat - 12g
 - Carbohydrates - 52g
 - Protein - 12g
 - Obroci za taj dan:**
 - Quaker Oatmeal Instant Oatmeal - Flavor Multi-Pack 3 Flavors
 - Turkey Salad
 - Zucchini Tomato Bake
- 4. dan**
 - Nutriciona vrijednost:**
 - Calories - 826cal
 - Fat - 64g
 - Carbohydrates - 32g
 - Protein - 38g
 - Obroci za taj dan:**
 - Egg & Cheese Sandwich On Whole Wheat
 - Peanut Butter Sandwich and Carrots
- 5. dan**
 - Nutriciona vrijednost:**
 - Calories - 415cal
 - Fat - 14g
 - Carbohydrates - 60g
 - Protein - 15g
 - Obroci za taj dan:**
 - Nature Valley Crunchy Granola Bars Peanut Butter - 6 CT
 - Zucchini Tomato Bake
 - My Personal Pizza
- 6. dan**
 - Nutriciona vrijednost:**
 - Calories - 444cal
 - Fat - 10g
 - Carbohydrates - 69g
 - Protein - 20g
 - Obroci za taj dan:**
 - Quaker Oatmeal Instant Oatmeal - Flavor Multi-Pack 3 Flavors
 - My Personal Pizza
 - Pumpkin Ricotta Stuffed Shells
- 7. dan**
 - Nutriciona vrijednost:**
 - Calories - 679cal
 - Fat - 23g
 - Carbohydrates - 99g
 - Protein - 24g
 - Obroci za taj dan:**
 - Quaker Oatmeal Instant Oatmeal - Flavor Multi-Pack 3 Flavors
 - Tuna Sandwich With Apple
 - Quick Chili

Slika 6.36. Detalji o određenom planu ishrane

Osim opcije za spremanje tog plana ishrane, na drugom dijelu stranice se nalaze detalji o svakom danu u tom planu. Prikazani plan ishrane ima 7 dana. Za svaki dan se ispisuje nutriciona vrijednost (kalorije, masnoće, karbohidrati i proteini) te obroci u tom danu. U obroke mogu spadati recepti i obični sastojci. Korisnici mogu na svaki recept kliknuti, što ih

preusmjerava na detalje o tom receptu (gdje mogu spremiti recept). Na slici 6.37 prikazan je izgled linka „Moj plan ishrane“ ukoliko korisnik nije dodao nijedan plan.



Slika 6.37. Prikaz stranice za plan ishrane ukoliko nije spremljen plan

U nastavku će se objasniti na koji način se izvršava dobijanje podataka iz Spoonacular API-a vezanih za planove ishrane.

Korišteni modul za ostvarivanje komunikacije:

mealController

Prikaz koda:

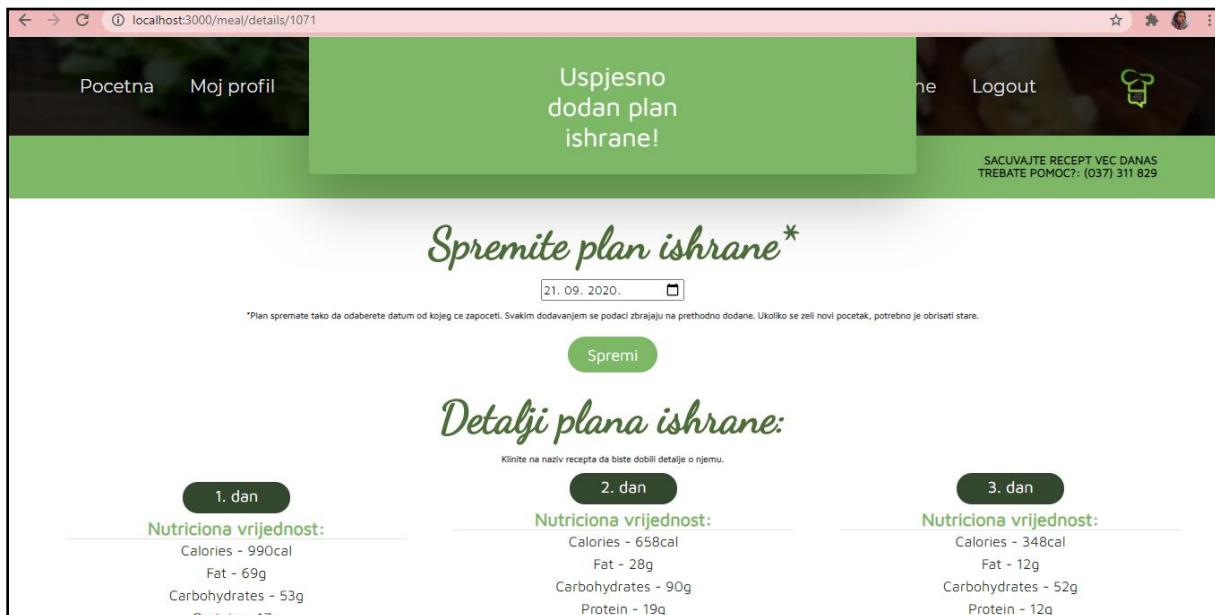
```
1. exports.getMealInfo = catchAsync(async (req, res, next) => {
2.   let odg = await axios.get(
3.     `${process.env.URL}/mealplanner/${req.user.APIusername}/templates/${
4.       req.params.id
5.     }?hash=${req.user.APIpassword}&apiKey=${process.env.API_KEY}`
6.   );
7.
8.   res.status(200).render('templateInfo', {
9.     template: odg.data,
10.    type: req.type
11.  });
12.});
```

URL koji se koristi za dobijanje informacija o određenom planu ishrane iz baze API-a je sljedeći [6]:

GET | <https://api.spoonacular.com/mealplanner/{username}/templates/{id}>

Modul mealController nudi metodu `getMealInfo`, koja kao rezultat vraća sve detalje plana ishrane koji postoje u bazi Spoonacular API-a. Za komunikaciju s API-em se, kao i u prethodnim slučajevima, koristi modul `axios`. Priloženom URL-u se dodaje API `username` korisnika i `id` plana ishrane koji će se prikazati. Osim toga, dodaje se i `hash`-ovana API šifra, kao parametar upita. API vraća podatke iz baze, koji se spremaju u `odg`.

Kada korisnik odabere dan za početak plana ishrane, pokazuje mu se notifikacija kao na slici 6.38, te se korisnik automatski preusmjerava na stranicu za prikaz vlastitog plana ishrane.



Slika 6.38. Spremanje plana ishrane

U nastavku će se objasniti na koji način se dodaje plan ishrane

Korišteni modul za ostvarivanje komunikacije:

mealController

Prikaz koda:

```

1. exports.addMealTemplate = catchAsync(async (req, res, next) => {
2.   let odg = await axios.post(
3.     `${process.env.URL}/mealplanner/${req.user.APIusername}/items/?hash=${
4.       req.user.APIpassword
5.     }&apiKey=${process.env.API_KEY}`,
6.     {
7.       mealPlanTemplateId: req.params.id,
8.       startDate: Math.round(new Date(req.body.date).getTime() / 1000)
9.     }
10.   );
11.
12.   if (odg.data.status != 'success') {
13.     return next(new AppError('Upssss. Try again later', 400));
14.   }
15.   req.user.datumiObroka.push(req.body.date);
16.   await User.findByIdAndUpdate(req.user.id, req.user, {

```

```

17.    new: true,
18.    runValidators: true
19.  });
20.  res.status(222).json({
21.    status: 'success',
22.    type: req.type,
23.    data: {
24.      data: null
25.    }
26.  });
27. });

```

URL koji se koristi za dobijanje informacija o određenom planu ishrane iz baze API-a je sljedeći [6]:

POST | <https://api.spoonacular.com/mealplanner/{username}/items>

Spoonacular API nudi mogućnost dodavanja svih elemenata određenog plana ishrane registriranom korisniku u svoju bazu podataka. Dakle, planovi ishrane za određenog korisnika se spremaju **isključivo** u bazu podataka Spoonacular API-a. Istovremeno, u Recipely bazu se sprema datum početka spremljenog plana ishrane.

U URL-u POST zahtjeva se prosljeđuje API username (kao i hash-ovana API šifra), dok tijelo zahtjeva nosi ID želenog plana ishrane, kao i vrijednost početnog datuma. API Recipely aplikaciji vraća odgovor o uspješnosti proteklog procesa.



Slika 6.39. Prvi plana ishrane

Na slici 6.39 prikazan je izgled današnjeg plana ishrane. Prikazan je dan, nutricionalna vrijednost koja se unosi receptima koji se konzumiraju. Svaki recept je prikazan u obliku linka, što znači da se može detaljno pregledati ukoliko se klikne na njegov naziv – to je pregled recepta koji je opisan u jednom od prethodnih poglavlja. Svakim danom se, naravno, mijenja sadržaj plana ishrane.

U nastavku će biti opisan način na koji su preuzeti podaci iz Spoonacular API-a da bi se prikazali u ovoj formi.

Korišteni modul za ostvarivanje komunikacije:

mealController

Prikaz koda:

```
1. exports.getMyMeal = catchAsync(async (req, res, next) => {
2.   let date = new Date().toJSON().split('T')[0];
3.   let odg;
4.   try {
5.     odg = await axios.get(
6.       `${process.env.URL}/mealplanner/${{
7.         req.user.APIusername
8.       }}/day/${date}?hash=${req.user.APIpassword}&apiKey=${process.env.API_K
EY}`;
9.     );
10.   } catch (err) {
11.     return next(new AppError('Niste dodali plan ishrane za ovaj dan.', 401)
12.   }
13.   res.status(200).render('myMeal', {
14.     template: odg.data,
15.     type: req.type
16.   });
17. });


```

URL koji se koristi za dobijanje informacija o određenom planu ishrane iz baze API-a je sljedeći [6]:

GET | <https://api.spoonacular.com/mealplanner/{username}/day/{date}>

Kako bi se dobio plan ishrane za današnji dan, Recipely app šalje HTTP zahtjev na priloženi URL, kojem se predaje API username, kao i datum u formatu „YYYY/MM/DD“. Spoonacular API kao odgovor na taj zahtjev šalje sve podatke vezane za taj dan plana ishrane (nutricia vrednost, recepti i sl), za tog korisnika [6].

Dodavanje novog plana ishrane

Korisnik može, na već postojeći plan ishrane, dodati i nove planove. Tada se nutricione vrijednosti svakog dana sabiraju a niz recepata koji se primjenjuju u tom danu se dodaje na niz recepata koji su sadržani u danu proteklog plana ishrane. Ukratko – dodavanjem novog plana ishrane na već postojeći plan, podaci se zbrajaju. Naravno, moguće je dodati novi plan ishrane na bilo koji datum, te će novi plan započeti od tog datuma.

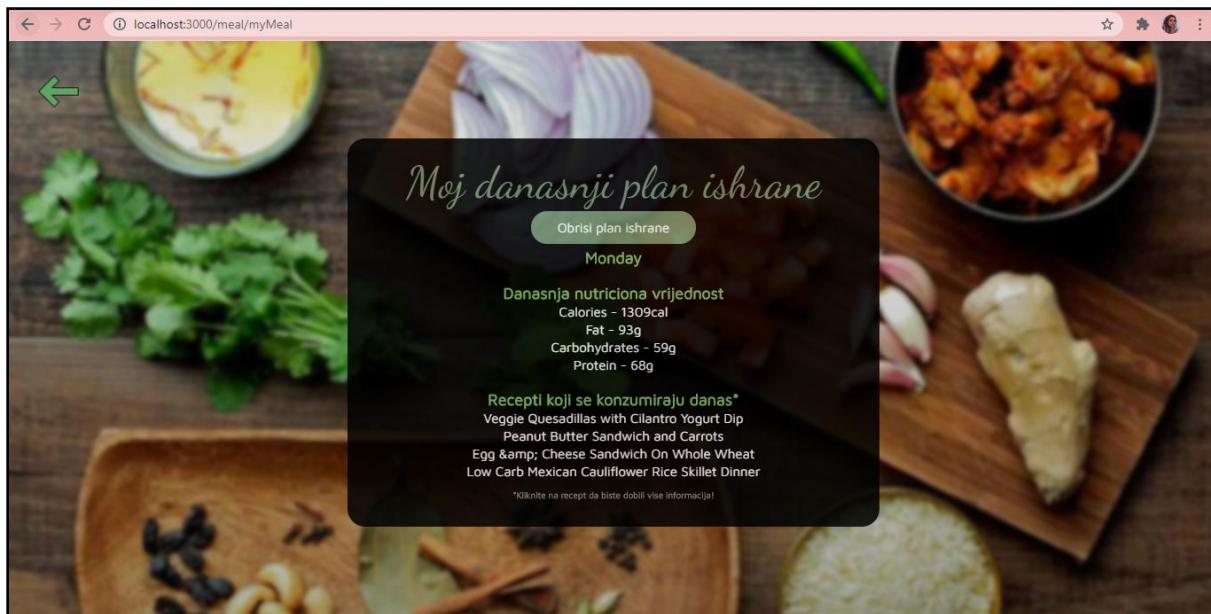
Kao primjer, na postojeći plan ishrane će se dodati plan koji je prikazan na slici 6.40.

The screenshot shows a meal planning interface. At the top, there are navigation links: Pocetna, Moj profil, Recepti, Spremljeni recepti, Planovi ishrane, Moj plan ishrane, Logout, and a user icon. A green header bar contains the text "SACUVAJTE RECEPT VEC DANAS" and "TREBATE POMOC? (037) 311 829". Below the header, the title "Spremite plan ishrane*" is displayed, followed by a date input field "dd. mm. 9999." and a "Spremi" button. A note below the date input states: "Plan spremate tako da odaberete datum od kojeg će započeti. Svakim dodavanjem se podaci zbrajaju na prethodno dodane. Ukoliko se zeli novi početak, potrebno je obrisati stare." The main content area is titled "Detalji plana ishrane:" and includes a note: "Kliknite na naziv recepta da biste dobili detalje o njemu." The plan is divided into five days:

- 1. dan**: Nutricionalna vrijednost: Calories - 486cal, Fat - 12g, Carbohydrates - 78g, Protein - 25g. Obroci za taj dan: 10-Spice Vegetable Soup (Freezer Friendly, Vegan, Gluten-Free).
- 2. dan**: Nutricionalna vrijednost: Calories - 319cal, Fat - 23g, Carbohydrates - 6g, Protein - 21g. Obroci za taj dan: Low Carb Mexican Cauliflower Rice Skillet Dinner.
- 3. dan**: Nutricionalna vrijednost: Calories - 241cal, Fat - 10g, Carbohydrates - 19g, Protein - 21g. Obroci za taj dan: Joseph's Flax Oat Bran & Whole Wheat Pita Bread - 8 CT, Lemon Garlic Grilled Shrimp, Greek Salad - 2 Points.
- 4. dan**: Nutricionalna vrijednost: Calories - 458cal, Fat - 27g, Carbohydrates - 33g, Protein - 24g. Obroci za taj dan: Healthy Salmon Quinoa Burgers, Roasted Acorn Squash Salad.
- 5. dan**: Nutricionalna vrijednost: Calories - 368cal, Fat - 22g, Carbohydrates - 14g, Protein - 27g. Obroci za taj dan: Balsamic Roasted Vegetables, Baked Lemon Chicken.

Slika 6.40. Primjer plana ishrane koji će se dodavati na postojeći

Da bi se testiralo dodavanje plana ishrane na dan koji se desio u prošlosti, dodat' će se plan tako da započinje od 20.09.2020. (datum od jučer). Novi plan ishrane prikazan je na slici 6.41.



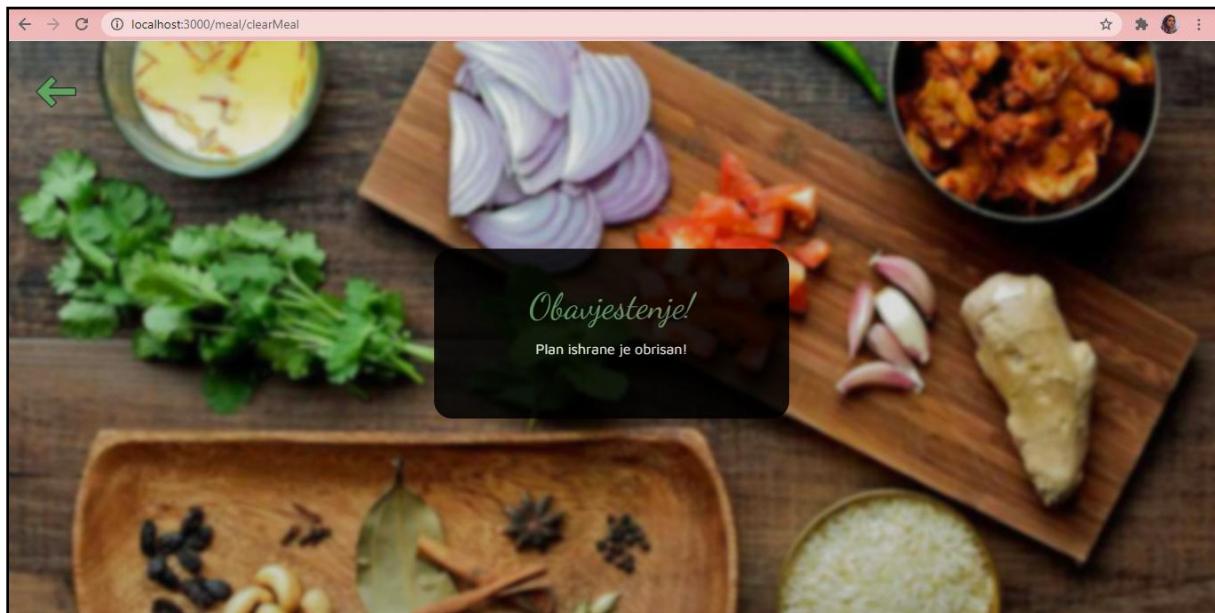
Slika 6.41. Dodavanje plana na prethodni

Kao primjer, ovaj plan će se dodati tako da počinje od 20.09.2020. (datum od jučer), što znači da bi se danas sadržaj *dana 2* trebao dodati na postojeći plan ishrane. Postojeći plan ishrane korisnika obavještava da ima 990 kalorija, 69g masnoće, 53g karbohidrata, 47g proteina i tri recepta, kao što se može vidjeti na slici 6.39. Ako se na taj sadržaj doda sadržaj iz dana 2, korisnik će imati 1309cal, 92g masnoće, 59g karbohidrata, 68g proteina i četiri recepta, kao što je prikazano na slici 6.41.

6.2.9. Brisanje plana ishrane

Preporučuje se da korisnik ne dodaje nove planove ishrane sve dok ne obriše stari, zbog toga što se, prilikom odabira *buttona* za brisanje, brišu svi planovi koje korisnik trenutno prati. Ovaj *button* je dostupan na stranici za prikaz korisničkog plana ishrane, kao što se može vidjeti na slici 6.41.

Kada se pritisne dugme za brisanje, korisnik se obavještava da je njegov trenutni plan ishrane obrisan, kao što je prikazano na slici 6.42.



Slika 6.42. Obavještenje o obrisanom planu ishrane

U nastavku će biti objašnjen princip brisanja plana ishrane, kao vrsta komunikacije sa Spoonacular API-em.

Korišteni modul za ostvarivanje komunikacije:

mealController

Prikaz koda:

```
1. exports.deleteMeal = catchAsync(async (req, res, next) => {
2.   let weekTIme = 1000 * 60 * 60 * 24;
3.   let d = req.user.datumiObroka;
4.   req.user.datumiObroka = [];
5.   const user = await User.findByIdAndUpdate(req.user.id, req.user, {
6.     new: true,
7.     runValidators: true
8. });
9.   try {
10.     for (let i = 0; i < d.length; i++) {
11.       let t = new Date(d[i]).getTime();
12.       for (let j = 0; j < 600; j++) {
13.         let date = new Date(t).toJSON().split('T')[0];
14.         //poziv na api sa date
15.         let odg = await axios.get(
16.           `${process.env.URL}/mealplanner/${{
17.             req.user.APIusername
18.           }/day/${date}?hash=${req.user.APIpassword}&apiKey=${{
19.             process.env.API_KEY
20.           }}`
21.         );
22.         console.log(odg.data);
23.         for (let k = 0; k < odg.data.items.length; k++) {
24.           await axios.delete(
25.             `${process.env.URL}/mealplanner/${req.user.APIusername}/items/${
26.               odg.data.items[k].id
27.             }?hash=${req.user.APIpassword}&apiKey=${process.env.API_KEY}`
28.           );
29.         }
30.         t = t + weekTIme;
31.         if (!odg.data.items) {
32.           break;
33.         }
34.       }
35.     }
36.   } catch (err) {
37.     return next(new AppError('Plan ishrane je obrisan!', 401));
38.   }
39.   res.status(200).redirect('/meal/myMeal');
40. });


```

Postoje dva korištena URL-a: jedan je za dobijanje plana ishrane za određeni datum [6]:

GET | <https://api.spoonacular.com/mealplanner/{username}/day/{date}>

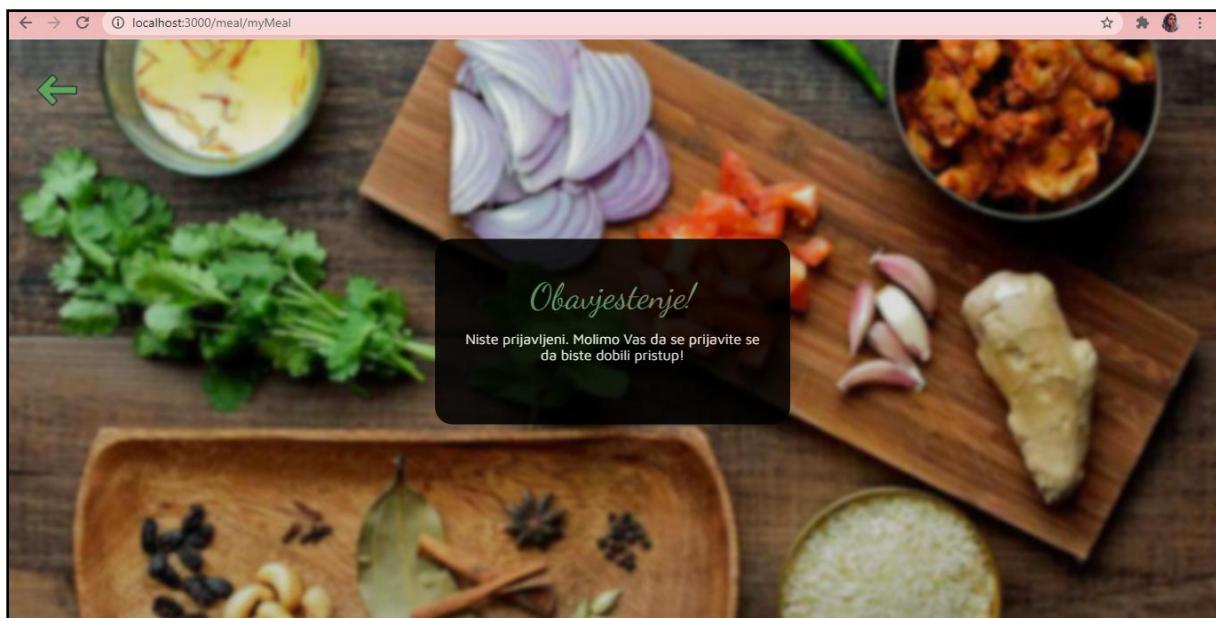
Drugi URL je za brisanje određenog elementa iz plana ishrane (recepti ili sastojci) [6]:

`DELETE | https://api.spoonacular.com/mealplanner/{username}/items/{id}`

Početni datumi svakog spremlijenog plana ishrane određenog korisnika se spremljaju u varijablu d . Kako bi se obrasio plan ishrane, potrebno je za svaki dan dobiti API ID svakog elementa (recepti). Nadalje se šalje DELETE zahtjev ka Spoonacular API-u, čime se briše element koji je označen tim ID-em [6]. Taj postupak se ponavlja za svaki dan trenutnog plana ishrane. Ako je korisnik spremio više od jednog plana ishrane, postupak brisanja koji je prethodno opisan se ponavlja za svaki, odnosno za svaki početni dan plana.

6.2.10. Odjava

Proces odjave se pokreće ukoliko prijavljeni korisnik odabere dugme *Logout* u sklopu navigacijskog menija. Recipely app u svom *response* (odgovoru) šalje *logout cookie*, čije je vrijeme važenja veoma kratak i traje nekoliko sekundi. Istekom tog intervala *cookie* postaje neupotrebljiv i dalji pristup zaštićenim rutama postaje onemogućen, sve dok se korisnik ponovo ne prijavi ili ne registruje na novi račun.



Slika 6.43. Obavještenje o nemogućnosti pristupa zaštićenoj ruti

Ukoliko se pristupi zaštićenim rutama korisniku se prikazuje prikladna poruka kojom se korisnik obavještava da se za tu radnju mora prijaviti (ili registrirati). Izgled ove poruke je prikazan na slici 6.43.

ZAKLJUČAK

U sklopu projektne dokumentacije opisana je metodologija implementacije web servisa koji omogućuje korisnicima kreiranje vlastitog dnevna recepata i planova ishrane. Web servis Recipely podatke prikuplja iz Spoonacular API baze podataka i sprema u online MongoDB bazu podataka. Svrha Recipely web servisa je omogućiti korisniku da na jednom mjestu objedini sve recepte koji mu se sviđaju, uz dodavanje zabilješke na svaki. Naravno, korisnik mora biti registriran. Svaki put, prilikom registracije novog korisnika, na njegov mail se šalje poruka dobrodošlice, što je realizirano putem servisa SendGrid.. Ukoliko je korisnik neregistriran, može pregledati generalne stranice aplikacije kao što su Početna, Informacijska, Kontakt, te galerija recepata koji se uzimaju iz Spoonacular API-a. Osim toga, neregistrirani korisnik može obavljati pretragu tih recepata, te slati e-mail organizaciji putem definisane forme na stranici Kontakt, što je omogućeno pomoću servisa SendGrid. Registrirani korisnici mogu spremati recepte uz zabilješku, koju kasnije mogu i promijeniti. Osim toga, korisnici mogu mijenjati sve svoje podatke, ali i dodati profilnu sliku. Ukoliko je korisnik zaboravio šifru, pomoću servisa SendGrid, na njegov mail se unosi link na kojem može promijeniti svoju šifru. Pored toga, korisnik može spremati planove ishrane, te iste i brisati. Ovi planovi se spremaju na Spoonacular bazi podataka. Korištena programska okruženja u toku razvijanja projekta su *Sublime Text*, kao pomoćno okruženje, *Visual Studio Code*, kao okruženje za implementaciju kôda, *Node JS*, kao okruženje za kreiranje kôda, te *MongoDB*, kao okruženje za bazu podataka.

LITERATURA

- [1] Link: <https://devdocs.io/node/>, Posjećeno: Septembar. 2020. (podaci pronađeni na Internetu).
- [2] D. Čubranić, M. Kaluža, J. Novak: Standardne metode u funkciji razvoja softvera u Republici Hrvatskoj Zbornik Veleučilišta u Rijeci, Vol. 1 (2013), No. 1, pp. 239-256
- [3] Jović A, Horvat M, Grudenić I, UML – dijagrami: Zbirka primjera i riješenih zadataka, Manualia Universitatis studiorum Zagrabiensis, Zagreb 2013.
- [4] Link: <https://www.taniarascia.com/es6-syntax-and-feature-overview/>, Posjećeno: Septembar 2020. (podaci pronađeni na Internetu)
- [5] Link: https://mongoosejs.com/docs/api.html#mongoose_Mongoose, Posjećeno: Septembar, 2020. (podaci pronađeni na Internetu)
- [6] Link: <https://spoonacular.com/food-api/docs> Posjećeno: Septembar, 2020. (podaci pronađeni na Internetu)