Objektno oblikovanje

Seminarski rad

|  |  |
| --- | --- |
| ČLANOVI TIMA | „Lovro i prijatelji“ |
| ČLANOVI GRUPE | KLARA BANIĆ, LOVRO RAČKI, LEON SULJIĆ |
| SAžETAK SEMINARA | APLIKACIJA ZA TRAŽENJE RECEPATA |
| GIT REPOZITORIJ | <HTTPS://GITHUB.COM/RACKILOVRO/SEMINAROBJEKTNO> |

sadržaj

[UVOD 3](#_Toc63019135)

[ZAHTJEVI 4](#_Toc63019136)

[Use cases 7](#_Toc63019137)

[OPIS MODELA 13](#_Toc63019138)

[OPIS IMPLEMENTACIJE PERZISTENCIJE 15](#_Toc63019139)

[Database-first pristup 15](#_Toc63019140)

[Prvi problemi 15](#_Toc63019141)

[Primjer mapiranja 19](#_Toc63019142)

[DESKTOP APLIKACIJA 21](#_Toc63019143)

[Uvod 21](#_Toc63019144)

[Screenshots 22](#_Toc63019145)

[WEB APLIKACIJA 26](#_Toc63019146)

[Uvod 26](#_Toc63019147)

[Screenshots 26](#_Toc63019148)

# UVOD

Odrasli ljudi, ali pogotovo studenti dnevno muče muku što skuhati za doručak, ručak ili večeru, uz određeni budžet i limitirano vrijeme. U današnje vrijeme postoji puno dostupnih aplikacija koje će na temelju upisa naziva recepta vratiti cijeli recept i postupak pripreme određenog jela. Ipak, to nije dovoljno s obzirom da većina nas nije čulo za 90% jela pa nam tražilica po nazivu recepta pretjerano i ne pomaže.

Ideja za aplikaciju je došla upravo od te činjenice. U hladnjaku imamo desetak namirnica, gledamo kako bismo ih mogli povezati i pomoću njih napraviti neko jelo. Nažalost, od prekomjernog učenja i izrade razno raznih projekata, mozak nam je prazan i nije spreman za kreativno povezivanje i izmišljanje recepata. Dovoljni je napor sama priprema namirnica i kuhanje, na koje se moramo odvažiti radi studentskog budžeta.

*RecepTour* je sve što jedan student, ali i zaposleni čovjek treba. *RecepTour* je jednostavna aplikacija, dostupna u web i desktop verziji, u koju korisnik unosi raspoložive namirnice te na temelju toga dobije popis recepata koje može ostvariti koristeći te sastojke. Korisnik zatim može odabrati recept koji mu se najviše sviđa te pogledati detalje recepta i sam postupak pripreme. Svidi mu li se ovaj recept i želi li ga možda ponoviti koji put, korisnik može dodati recept pod omiljene recepte. Kako bi to ostvario, korisnik se najprije treba registrirati kako bi imao svoj račun uz koji bi se svi podaci vezali. Tako prijavljeni korisnik može upravljati svojim omiljenim receptima. Nadalje, prijavljeni korisnik može unijeti svoje recepte u aplikaciju, a nakon toga i upravljati istima. Aplikacija ima dostupne još dvije tražilice – pomoću jedne korisnik može pretraživati recepte po nazivu, a pomoću druge recepte određenog korisnika.

Web aplikacija se može smatrati proširenjem desktop aplikacije koja predstavlja osnovne funkcionalnosti, a to je pretraga recepata. Web aplikacija dodatno pruža mogućnost registracije, prijave, dodavanje vlastitih recepata, omiljenih recepata i upravljanje svojim korisničkim računom.

Aplikacija je oličenje minimalizma i jednostavnosti kako ne bi zbunila korisnike i maksimalno olakšala njenu uporabu. Idealna je za korisnike kojima su prioriteti jednostavnost, organiziranost i pedantnost.

# ZAHTJEVI

Skup mogućnosti koje naš sustav pruža je najlakše prikazati koristeći user storyje.

Kako bi korisnici mogli koristiti sve funkcionalnosti aplikacije najprije će se morati registrirati te potom prijaviti u aplikaciju, iako to nije nužno za osnovno korištenje aplikacije.

Desktop verzija aplikacije predstavlja pojednostavljenu verziju aplikacije bez funkcionalnosti prijavljivanja korisnika u sustav te je namijenjena korisnicima koji imaju računala u blizini kuhinje i žele gledati recept za vrijeme kuhanja. Ako žele dodati svoj recept, to će napraviti kada završe sa kuhanjem i na miru sjesti i prijaviti se u web aplikaciju.

Pomoću jednostavnog izbornika korisnik određuje namirnice koje trenutno ima na raspolaganju, na temelju kojih mu aplikacija nudi prijedloge za recepte koje korisnik može skuhati. Ako se korisnicima nakon kuhanja svidio recept, mogu ga spremiti pod svoje omiljene recepte. Ako korisnici nisu zadovoljni niti jednim pronađenim receptom, mogu objavljivati svoje recepte, ali samo ako su registrirani i prijavljeni u sustav. Svi zahtjevi su dani i opisani u nastavku:

1. Registracija korisnika:

Korisnik se želi registrirati pa odabire opciju za registraciju, unosi potrebne podatke i potvrđuje svoju odluku.

1. Login u sustav:

Korisnik želi pristupiti aplikaciji preko vlastitog korisničkog računa pa unosi potrebne podatke za prijavu i ulazi u aplikaciju preko korisničkog računa.

1. Promjena lozinke:

Korisnik ima potrebu da promijeni svoju postojeću lozinku. Odabire opciju za promjenu lozinke, unosi potrebne podatke i, ako su podatci ispravni, lozinka se mijenja.

1. Promjena podataka o korisniku:

Korisnik želi promijeniti neke svoje podatke. Odabire opciju za promjenu podataka i unosi nove, ažurne podatke. Ukoliko je zadovoljan, potvrđuje svoju odluku i sprema promjene klikom na gumb za potvrdu.

1. Dodavanje recepta pod „Favorites“:

Korisniku se sviđa određeni recept i želi ga spremiti kako bi ga mogao pregledavati i kasnije. To čini lagano, klikom na gumb za spremanje pod „Favorites“ koji se odnosi na pojedini recept.

1. Micanje recepta iz „Favorites“:

Korisniku se više ne sviđa recept pa ga želi maknuti iz „Favorites“ kategorije. To radi klikom na gumb koji to radi, recept se miče iz kategorije „Favorites“, a korisnik može odmah nastaviti s radom.

1. Dodavanje svojih recepata:

Korisnik želi dodati vlastiti recept u aplikaciju. Odabire opciju za dodavanje recepta i unosi sve podatke koji su potrebni za unos recepta. Nakon spremanja, recept je dodan u aplikaciju i ostali korisnici ga mogu vidjeti i pristupiti mu.

1. Brisanje vlastitog recepta:

Korisniku se više ne sviđa recept koji je dodao pa ga želi maknuti iz aplikacije. To čini lagano klikom na gumb za brisanje kod pregleda recepta i recept se odmah briše iz aplikacije.

1. Uređivanje recepta:

Korisnik treba promijeniti neke podatke vezane za svoj recept pa odabire opciju za mijenjanje podataka i unosi nove. Gumbom za spremanje potvrđuje odluku.

1. Pregledavanje recepata:

Korisnik pregledava sve recepte koji se nalaze u aplikaciji.

1. Pretraživanje recepata po sastojcima:

Korisnik označava sastojke koje želi pronaći u receptu što rezultira time da aplikacija prikaže recepte koji sadrže takve sastojke. Nakon toga korisnik može pojedinačno pregledati recept koji mu se sviđa.

1. Pretraživanje recepta po imenu:

Korisnik se sjetio imena recepta kojeg želi naći i unosi njegovo ime u tražilicu te mu aplikacija prikazuje recept koji sadrži to ime, ukoliko takav recept postoji u aplikaciji.

1. Pretraživanje recepta po autoru:

Korisnik se sjetio osobe koja je napravila recept koji on traži ili želi vidjeti recepte svoga prijatelja. Unosi njegovo ime u tražilicu prema autoru i aplikacija prikazuje recepte koje je ta osoba unijela u aplikaciju.

1. Detaljnije prikazivanje recepta:

Korisnik želi više toga saznati o receptu i klikne na njega kako bi vidio detaljnije informacije o njemu, kao npr. potpuni tekst recepta.

1. Prikaz korisničkog profila:

Korisnik želi vidjeti svoj korisnički profil. To ostvari tako da klikne na svoje ime u kutu i pregledava svoj korisnički profil.

Zahtjevi s obzirom na sadržaj zahtjeve bi mogli klasificirati na funkcionalne, nefunkcionalne zahtjeve i zahtjeve domene primjene.

Funkcionalni zahtjevi su izjave o uslugama koje sustav mora pružati, kako će sustav reagirati na određeni ulazni poticaj te kako bi se sustav trebao ponašati u određenim situacijama. Ove zahtjeve možemo lako identificirati tako da si pomognemo sa „*system shall do <requirement>“*.

Često uz ovo vežemo dionike, aktore i njihove funkcionalne zahtjeve. Dionici u ovom projektu su korisnici aplikacije i razvojni tim. Problem kod zahtjeva je u tome što dionici ne znaju što stvarno žele i izražavaju zahtjeve na različite načine. Različiti dionici mogu imati konflikte zahtjeve (moram priznati da ih je bilo 😊), a zahtjevi su se mijenjali za vrijeme procesa analize zahtjeva, ali i kasnije. U nastavku su kratko i jasno, oku ugodno, pobrojani aktori i njihovi funkcionalni zahtjevi:

1. Neregistrirani/neprijavljeni korisnik (inicijator) može:

a. pregledavati recepte

b. pretraživati recepte po sastojcima, imenu ili autoru

c. pristupiti detaljnom prikazu recepta

d. se registrirati u sustav, stvoriti novi korisnički račun za koji su mu potrebni korisničko ime, e-mail i lozinka

2. Registrirani korisnik (inicijator) može:

a. se ulogirati u sustav sa svojim korisničkim podacima

b. pregledavati i mijenjati osobne podatke

d. dodati, izbrisati i promijeniti vlastite recepte

e. pregledavati recepte

f. pretraživati recepte po sastojcima, imenu ili autoru

g. pristupiti detaljnom prikazu recepta

h. dodavati recepte pod „Favorites“ i ukloniti iste

i. pregledavati profile drugih korisnika

3. Administrator (inicijator) može:

a. vidjeti popis svih registriranih korisnika i njihovih osobnih podataka

b. brisati korisnike

c. brisati recepte

4. Baza podataka (sudionik):

a. pohranjuje sve podatke o korisnicima i njihovim ovlastima

b. pohranjuje sve podatke o receptima, sastojcima i drugim detaljima

Nefunkcionalni zahtjevi su ograničenja u uslugama i funkcijama, kao što su vremenska ograničenja, standardi, ograničenja u procesu razvoja i oblikovanja i slično. Naglasak je na karakteristikama sa *„system shall be <requirement>“*.

Naš sustav zadovoljava većinu nefunkcionalnih zahtjeva, a kada bismo htjeli zadovoljiti sve nefunkcionalne zahtjeve, ipak bismo trebali uložiti više novaca i više napora. 😊

Zahtjevi domene primjene su funkcionalni i nefunkcionalni zahtjevi koji proizlaze iz domene primjene sustava, kao i oni koji karakteriziraju tu domenu.

U nastavku je opisano kako se sustav treba ponašati i koja su neka njegova ograničenja – formalni nazivi za ove zahtjeve su nefunkcionalni zahtjevi i zahtjevi domene primjene.

1. Sustav treba omogućiti rad više korisnika u stvarnom vremenu.
2. Jezik korisničkog sučelja i jezik u programskom kodu je engleski
3. Korisničko sučelje i sustav moraju podržavati hrvatsku abecedu (dijakritičke znakove) pri unosu i prikazu tekstualnog sadržaja
4. Vrijeme potrebno da se dohvati neki podatak iz baze podataka treba biti relativno kratko (ne više od nekoliko sekundi)
5. Neispravno korištenje korisničkog sučelja ne smije narušiti funkcionalnost i rad sustava
6. Sustav treba biti jednostavan za korištenje, korisnici se moraju znati koristiti sučeljem bez opširnih uputa
7. Veza s bazom podataka mora biti kvalitetno zaštićena, brza i otporna na vanjske greške
8. Pristup sustavu preko javne mreže je zaštićen pomoću HTTPS protokola
9. Koristi se objektno orijentirana paradigma, a kao posljedica toga je korištenje C# u sklopu *backend*-a

## Use cases

1. Registracija korisnika – samo za web

• Glavni sudionik: Korisnik

• Cilj: Stvoriti korisnički račun za pristup sustavu

• Sudionici: Baza podataka

• Preduvjet: Korisnik ima važeću e-mail adresu

• Opis osnovnog tijeka:

1. Korisnik odabire registraciju

2. Korisnik unosi podatke za svoj korisnički račun

3. Sustav provjerava ispravnost korisničkih podataka

4. Korisnički račun se sprema u bazu podataka

• Opis mogućih odstupanja:

3.a Korisnik odabire zauzet ili neispravan e-mail ili unosi korisničke podatke u krivom formatu

• Sustav obavještava korisnika o pogrešnom formatu e-maila i traži od korisnika da ponovno unese podatke

• Korisnik mijenja krivo unesene podatke te uspješno završava registraciju ili odustaje

2. Prijava korisnika – samo za web

• Glavni sudionik: Korisnik

• Cilj: Prijaviti se u sustav

• Sudionici: Baza podataka

• Preduvjet: Korisnik posjeduje korisnički račun

• Opis osnovnog tijeka:

1. Korisnik unosi svoj e-mail za prijavu te svoju lozinku

2. Sustav provjerava ispravnost korisničkih podataka

3. Korisnik je prijavljen u sustav

• Opis mogućih odstupanja:

2.a Korisnik unosi nevaljane podatke

• Sustav obavještava korisnika kako uneseni podaci nisu valjani te od njega traži da pokuša ponovo

• Korisnik mijenja krivo unesene podatke te se uspješno prijavljuje u sustav ili odustaje

3. Promjena lozinke – samo za web

• Glavni sudionik: Korisnik

• Cilj: Promijeniti lozinku koju korisnik koristi za prijavu u sustav

• Sudionici: Baza podataka

• Preduvjet: Korisnik je prijavljen u sustav

• Opis osnovnog tijeka:

1. Korisnik odabire opciju promjene lozinke

2. Korisnik unosi staru lozinku i novu lozinku dva puta

3. Korisnik potvrđuje promjenu lozinke

• Opis mogućih odstupanja:

3.a Unesena stara lozinka nije ispravna

• Sustav obavještava korisnika o pogrešnoj staroj lozinki i traži od korisnika da ispravno unese lozinku

• Korisnik mijenja krivo unesene podatke te uspješno završava promjenu lozinke ili odustaje

4. Promjena korisničkih podataka – samo za web

• Glavni sudionik: Korisnik

• Cilj: Promijeniti korisničke podatke

• Sudionici: Baza podataka

• Preduvjet: Korisnik je prijavljen u sustav

• Opis osnovnog tijeka:

1. Korisnik odabire opciju promjene korisničkih podataka

2. Korisnik upisuje nove korisničke podatke koje želi promijeniti te sprema promjene

3. Promjene se spremaju u bazu podataka

• Opis mogućih odstupanja:

2.a Korisnik je neispravno upisao nove podatke

• Sustav obavještava korisnika o pogrešno upisanim podacima

• Korisnik mijenja krivo unesene podatke te uspješno završava promjenu korisničkih podataka ili odustaje

5. Dodavanje recepta u favorite – samo za web

• Glavni sudionik: Korisnik

• Cilj: Spremiti željeni recept u kategoriju „Favorites“

• Sudionici: Baza podataka

• Preduvjet: Korisnik je prijavljen u sustav

• Opis osnovnog tijeka:

1. Korisnik pregledava recepte

2. Korisnik odabire recept kojega želi dodati u kategoriju „Favorites“

3. Recept se nalazi u favoritima korisnika

6. Brisanje recepta iz favorita – samo za web

• Glavni sudionik: Korisnik

• Cilj: Ukloniti željeni recept iz kategorije „Favorites“

• Sudionici: Baza podataka

• Preduvjet: Korisnik je prijavljen u sustav

• Opis osnovnog tijeka:

1. Korisnik pregledava recepte koji su u kategoriji “Favorites”

2. Korisnik odabire recept kojeg želi ukloniti iz kategorije „Favorites“

3. Korisnik odabire opciju koja uklanja željeni recept iz favorita

7. Dodavanje vlastitog recepata – samo za web

• Glavni sudionik: Korisnik

• Cilj: Dodati recept u sustav

• Sudionici: Baza podataka

• Preduvjet: Korisnik je prijavljen u sustav

• Opis osnovnog tijeka:

1. Korisnik odabira opciju dodavanja recepta u sustav

2. Korisnik upisuje podatke o receptu – naziv, sastojci, koraci, slike

3. Korisnik sprema novi recept

• Opis mogućih odstupanja:

2.a Podatci o receptu su nepotpuni ili neispravni

• Sustav obavještava korisnika o nepotpunim ili neispravnim podacima

• Korisnik mijenja krivo unesene podatke te uspješno dodaje novi recept ili odustaje

8. Brisanje vlastitog recepta – samo za web

• Glavni sudionik: Korisnik

• Cilj: Obrisati vlastiti recept

• Sudionici: Baza podataka

• Preduvjet: Korisnik je prijavljen u sustav

• Opis osnovnog tijeka:

1. Korisnik pregledava vlastite recepte

2. Korisnik odabire recept koji želi obrisati

3. Korisnik odabire opciju za brisanje

4. Recept se briše iz baze podataka te više ne postoji u sustavu

9. Uređivanje vlastitog recepta – samo za web

• Glavni sudionik: Korisnik

• Cilj: Mijenjanje podataka vezanih za određeni recept (sastojci, ime, ...)

• Sudionici: Baza podataka

• Preduvjet: Korisnik je prijavljen u sustav te je vlasnik recepta kojega želi urediti

• Opis osnovnog tijeka:

1. Korisnik odabire recept kojeg želi uređivati

2. Korisnik uređuje informacije recepta (koristeći forme za unos podataka)

3. Korisnik potvrđuje svoje promjene

10. Pregledavanje recepata

• Glavni sudionik: Korisnik

• Cilj: Pregledati deset recepata rangiranih po odabranom kriteriju

• Sudionici: Baza podataka

• Preduvjet: -

• Opis osnovnog tijeka:

1. Sustav prikazuje recepte iz aplikacije

11. Pretraživanje recepata po sastojcima

• Glavni sudionik: Korisnik

• Cilj: Pronaći sve recepte koji sadržavaju odabrane sastojke

• Sudionici: Baza podataka

• Preduvjet: -

• Opis osnovnog tijeka:

1. Korisnik odabire opciju pretraživanja recepta po sastojcima

2. Korisnik označuje sastojke koje želi naći u receptu

3. Korisnik odabire opciju pretraživanja

4. Sustav prikazuje adekvatne recepte

• Opis mogućih odstupanja:

3.a Ne postoji recept koji zadovoljava filtre pretrage

• Sustav ne prikazuje recepte

12. Pretraživanje recepata po nazivu

• Glavni sudionik: Korisnik

• Cilj: Pronaći recept po nazivu

• Sudionici: Baza podataka

• Preduvjet: -

• Opis osnovnog tijeka:

1. Korisnik odabire opciju pretraživanja recepta po imenu

2. Korisnik unosi naziv recepta

3. Korisnik odabire opciju pretraživanja

4. Sustav prikazuje adekvatne recepte

• Opis mogućih odstupanja:

4.a Ne postoji recept s takvim nazivom

• Sustav ne prikazuje recepte

13. Pretraživanje recepata po autoru

• Glavni sudionik: Korisnik

• Cilj: Pronaći sve recepte određenog autora

• Sudionici: Baza podataka

• Preduvjet: -

• Opis osnovnog tijeka:

1. Korisnik odabire opciju pretraživanja recepta po autoru

2. Korisnik unosi korisničko ime autora čije recepte želi pronaći

3. Korisnik odabire opciju pretraživanja

4. Sustav prikazuje adekvatne recepte

• Opis mogućih odstupanja:

4.a Ne postoji autor s takvim korisničkim imenom

• Sustav ne prikazuje recepte

14. Detaljni prikaz recepta

• Glavni sudionik: Korisnik

• Cilj: Detaljno prikazati odabrani recept

• Sudionici: Baza podataka

• Preduvjet: -

• Opis osnovnog tijeka:

1. Korisnik odabire koji recept želi detaljno prikazati

2. Sustav prikazuje detaljni tekst recepta, popis potrebnih sastojaka,…

15. Prikaz korisničkog profila – samo za web

• Glavni sudionik: Korisnik

• Cilj: Pregledati informacije registriranog korisnika

• Sudionici: Baza podataka

• Preduvjet: Korisnik je prijavljen u sustav

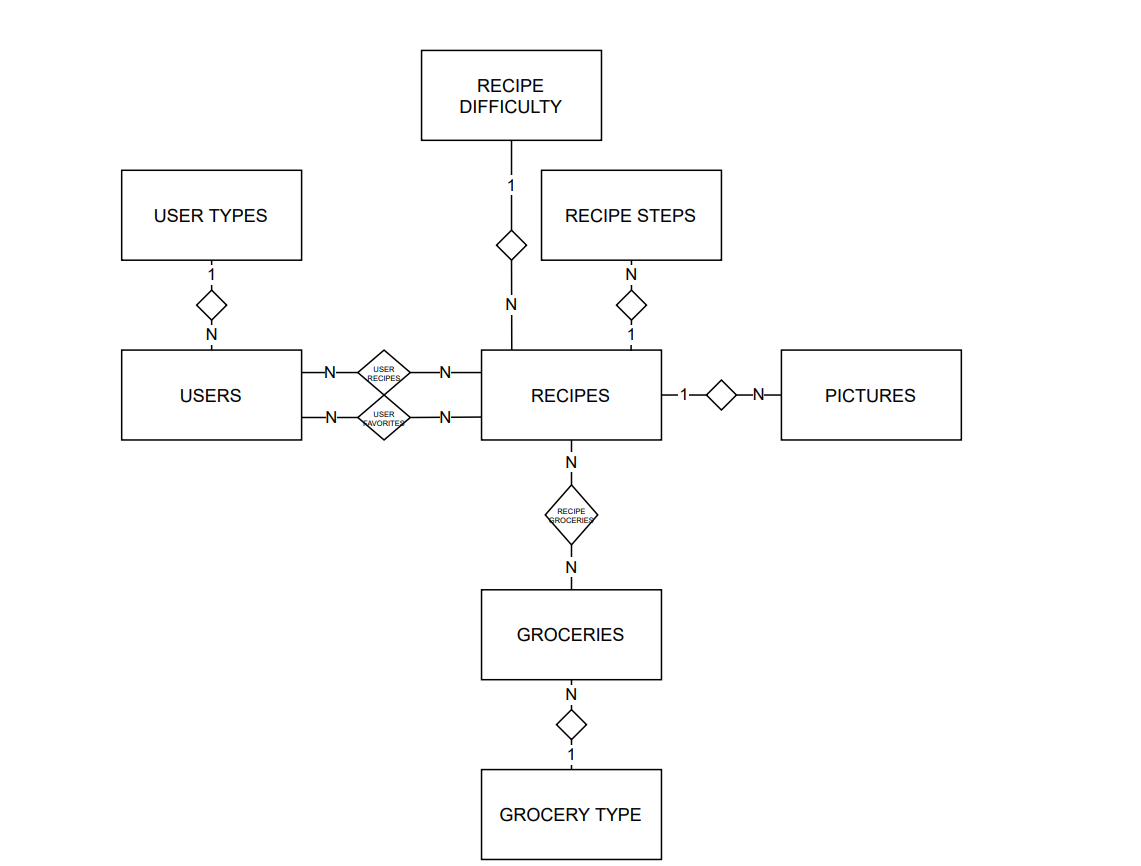
• Opis osnovnog tijeka:

1. Korisnik odabire opciju pregleda vlastitog korisničkog profila

2. Korisnik pregledava vlastite korisničke informacije

# OPIS MODELA

Model domene je sličan rasporedu relacija u bazi podataka za aplikaciju. Ovo je posljedica korištenja *Entity Frameworka* u izradi O/R sloja.



Slika 1 Relacijski dijagram modela domene

Model je prikazan dijagramom klasa i očigledno predstavlja *Model* komponentu u *MVC* i *MVVM* arhitekturnim obrascima. Najvažnije klase su *Recipe* i *User*preko kojih se odrađuju skoro sve funkcionalnosti na aplikacijskom sloju.

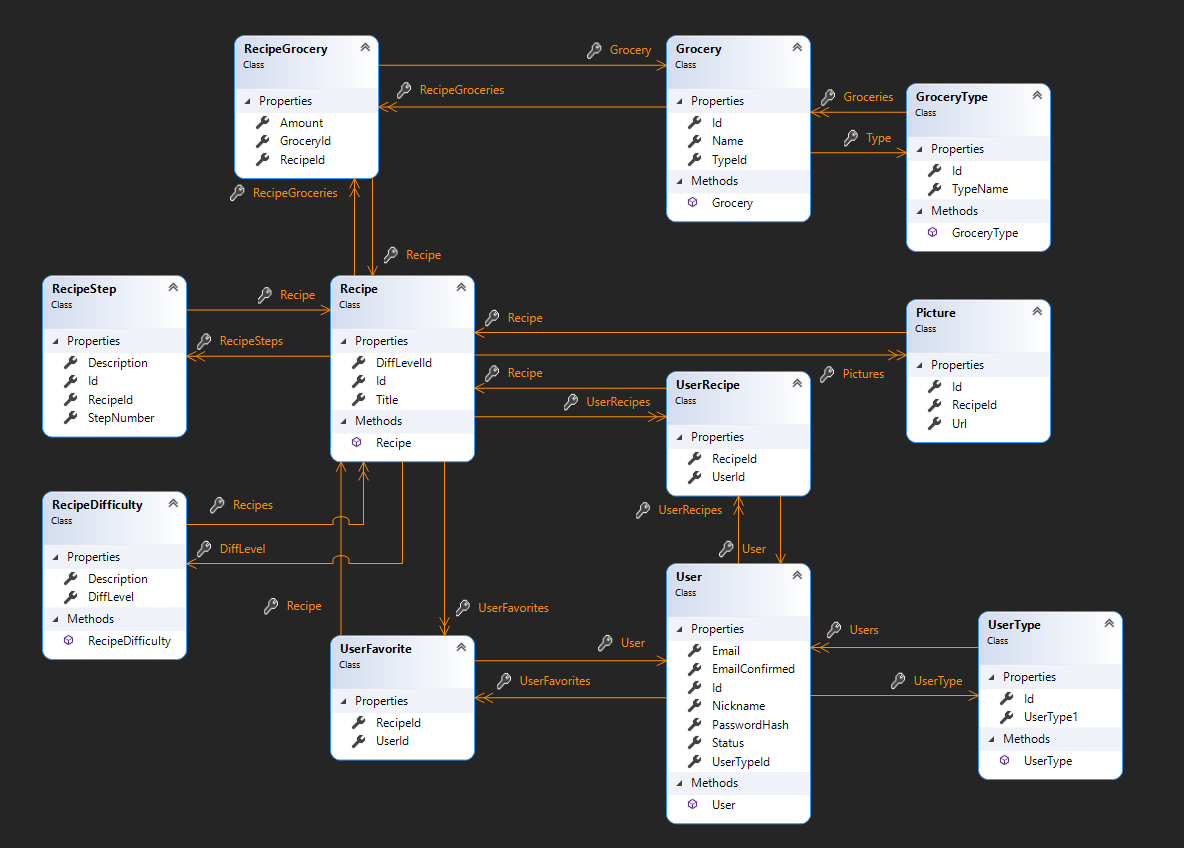


Figure 1 Dijagram klasa

Svaka klasa sadrži potrebna polja koja su mogu lako preslikati u n-torku koja se unosi u prikladnu relaciju u bazi podataka. Uz polja*, Entity Framework* je dodao referencu na klasu koja sadrži strani ključ početne klase. Ovime je omogućeno jednostavno dohvaćanje podataka u složenim operacijama gdje je potrebno spajati instance klasa po određenim ključevima.

Može se primijetiti, iz modela, kako je realizirana jedna od najvažnijih funkcionalnosti aplikacije. Naime, klasa Recept sadrži kolekcije instanci klasa *RecipeStep*, *RecipeGrocery* i reference na instance klasa *Picture*, *RecipeDifficulty* i *UserRecipe*. Povezivanje instanci ovakvih povezanih klasa je uz ovakav model trivijalno i *Entity Framework* nakon spremanja promjena u kontekstu povezuje ove objekte na razini baze podataka.

Svakako je prednost ovakvog modela njegova jednostavnost dodavanja i dohvaćanja podataka. Međutim, poslovna logika (koje u ovoj aplikaciji nema puno) je prebačena u zaseban sloj aplikacije. Konkretno, u slučaju *RecepTour*-a, je ona u *Controller* sloju.

# OPIS IMPLEMENTACIJE PERZISTENCIJE

Glavna odlika perzistencije naše aplikacije je pohrana u bazu podataka. Baza podataka nalazi se na poslužitelju, a objektno relacijsko mapiranje provedeno pomoću *Entity Frameworka*.  
Za implementaciju *Unit of work* oblikovnog obrasca zaslužan je *DbContext* *Entity Frameworka*. Ulogu repozitorija prepustili smo *RecepTourContext* klasi koja nasljeđuje *DbContext*.

Zbog samog izbora tehnologija koje u načelu nisu namijenjene za rad jedno s drugim, najveći problem bio je nesklad *PostgreSQLa i Entity Frameworka*, ne postoji velik izvor informacija na internetu koji bi nas usmjerio u razrješavanju nastalih problema.

## Database-first pristup

Budući da smo kao grupa bolji arhitekti baza podataka, nego inženjeri programskih sustava namijenjenih za *web*, odlučili smo se za *database-first* pristup. Najprije smo definirali model baze podataka te izveli potrebne *SQL* skripte direktno na *Heroku* server. Nakon postavljanja i punjenja baze podataka reprezentativnim podacima, sljedeći korak je povezati *Entity Framework* sa *Heroku* serverom i provesti objektno relacijsko mapiranje.

## Prvi problemi

*Entity Framework* jednostavan je za koristiti jer nudi dobru i razvijenu podršku kroz dobro osmišljeno korisničko sučelje…ako koristite neku od službeno podržanih podržanih *SQL* standarda. Nažalost, odlučili smo se za *PostgreSQL* i time si otežali posao. Morali smo se odreći korisničkog sučelja i daljnji razvoj odvijao se kroz terminal.  
Prvi problem na koji smo naišli je *connection string* potreban za povezivanje *Entity Frameworka* i baze podataka na serveru. *Heroku connection stringove* zapisuje u jednom formatu, a *Entity Framework*, odnosno *Npgsql*, očekuje ih u drugom formatu.

Figure 2 Heroku connection string

Figure 3 Heroku connection string

postgres://mcdgxeedppzugi:56fd5710c4601eaa2c022a7a28332a5983e48c97af7cc3cc39b6b8a69da9b9b1@ec2-54-220-229-215.eu-west-1.compute.amazonaws.com:5432/d3jgof5caojkns

Kako bismo riješili taj problem pronašli smo parser koji pretvara *connection string* iz jednog formata u drugi.

Figure 4 Entity Framework connection string

postgres://mcdgxeedppzugi:56fd5710c4601eaa2c022a7a28332a5983e48c97af7cc3cc39b6b8a69da9b9b1@ec2-54-220-229-215.eu-west-1.compute.amazonaws.com:5432/d3jgof5caojkns

Kada smo napokon dobili pravi *connection string* mogli smo dalje nastaviti sa procesom objektno relacijskog mapiranja.



Figure 5 Pokretanje objektno relacijskog mapiranja

Nakon otvaranja *developer command prompta* potrebno je pozicionirati se u *root* direktorij *solutiona* i pokrenuti potrebnu *dbcontext scaffold* naredbu sa navedenim *connection strignom*.  
Ovdje dolazimo do drugog problema na koji smo naišli. Naime, za potrebe *scaffolda* sa baza podataka koje su *hostane* na *Heroku* potrebno je u *connection string* nadodati *SSL Mode* i *TrustServerCertificate* polja. Iako se čini jednostavno na ovaj korak smo izgubili određeno vrijeme, jer ovako specifičan izbor tehnologija nije previše dokumentiran niti popularan.  
Nakon izvršavanja u *Models* direktoriju *solutiona* generirani su svi potrebni razredi modela domene. Prilikom generiranja vidljive su neke negativne posljedice izbora navedenih tehnologija. *Many-to-many* veze nisu predstavljene neko složenom kolekcijom kao što bi bile u *NHibernateu*, nego kao *one-to-many* veze između polazne tablice, spojne tablice i naposljetku završne tablice. Osim toga automatski generirani *dbcontext* razred dobio je nerazumljivi naziv preuzet iz *connection stringa* (d3jgof5caojkns) što dodatno ukazuje na činjenicu da ove tehnologije nisu usklađene, no ovaj problem smo razriješili jednostavnom promjenom naziva razreda.  
Postupak provođenja objektno relacijskog mapiranja proveden je dvaput. Jednom za *desktop* aplikaciju, a jednom za *web* aplikaciju.   
Nakon generiranja modela domene, automatski smo generirali i sve potrebne bazne kontrolere koje smo kasnije nadograđivali po potrebi. U automatski generiranim baznim kontrolerima nalaze *GET*, *POST* i *DELETE* rute za sve potrebne modele.

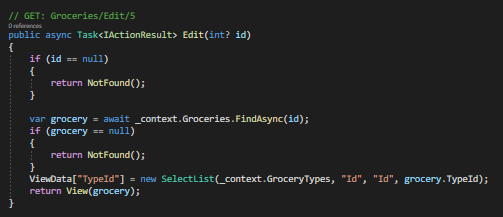


Figure 6 GET u kontroleru za namirnice

Kontroleri funkcioniraju na principu komuniciranja sa objektom *\_context* koji je tipa *db context*. *Db context* razred sadrži reference na sve tablice modela domene te se npr. svi zapisi tablice *Groceries* dobivaju sa *\_context.Groceries.*

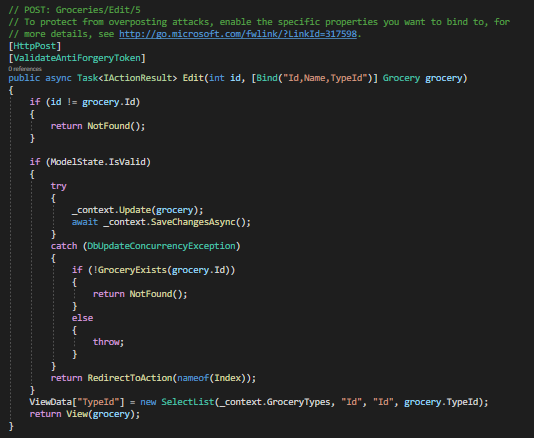


Figure 7 Update za namirnicu

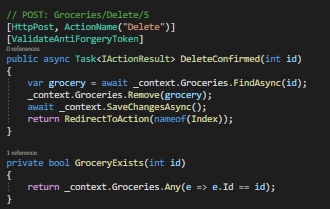


Figure 8 Delete za namirnicu

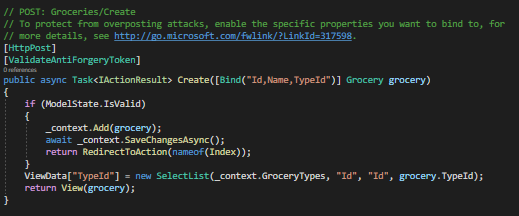


Figure 9 Create metoda za namirnice

Zadnji ozbiljniji problem s kojim smo se suočili vezan je uz generiranje primarnih ključeva.  
Naime, prilikom postavljanja baze na server, u bazu smo direktno ubacili nekoliko podataka za testiranje i razvoj. Kada smo napokon uspjeli pokrenuti *web* aplikaciju krenuli smo testirati automatski generirani *viewove* za pregled, dodavanje, ažuriranje i brisanje podataka. Zbog nekog čudnog razloga nismo mogli smo samo pregledavati podatke. Ispostavilo se da je ključna pogreška bilo dodavanje podataka izravno u bazu podataka, zbog kojeg je došlo do nesklada u komunikaciji prilikom generiranja vrijednosti primarnih ključeva. Brojač primarnih ključeva u bazi bio je na razini većoj od 1, dok je brojač primarnog ključa *Entity Frameworka* ostao na jedan te je uporno pokušavao u bazu podatka dodati zapis sa primarnim ključem 1, što je izazivalo pogreške. Problem je razriješen brisanjem svih zapisa iz baze podataka te smo se u budućnosti ograničili na dodavanje zapisa isključivo kroz *web* aplikaciju. Ovo je samo još jedan primjer nesklada između korištenih tehnologija.

## Primjer mapiranja

Tablica *Groceries* mapirana je u razred *Grocery.*

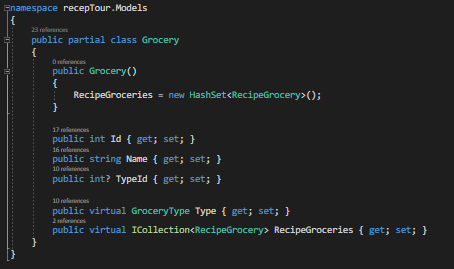
**

Figure 10 Primjer mapiranja tabice

Tablica *Groceries* *many-to-many* vezom povezana je sa tablicom *Recipes*.

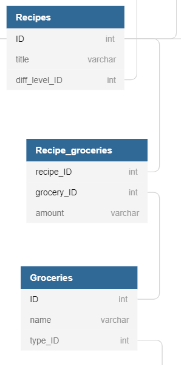


Figure 11 Tablice veze

Navedene tablice su mapirane u tri istoimena razreda gdje je veza između *Groceries* i *Recipe\_groceries* ostvarena pomoću kolekcije *RecipeGroceries*.

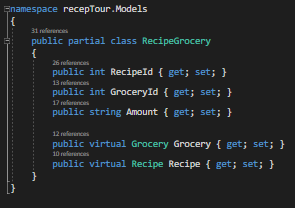


Figure 12 Mapiranje spojne tablice

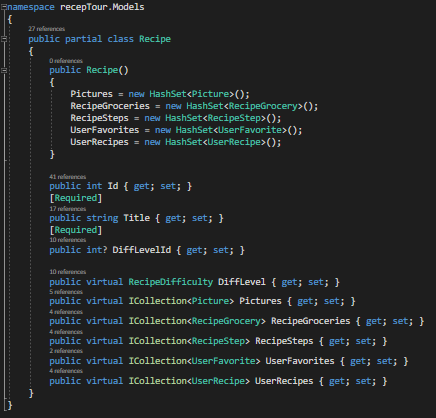


Figure 13 Mapiranje tablice Recipes

# DESKTOP APLIKACIJA

## Uvod

Desktop aplikacija implementirana je pomoću *windows presentation frameworka (WPF)*. WPF koristi *model-view-viewmodel (MVVM)* oblikovni obrazac. Ključan koncept *MVVM* oblikovnog obrasca je razdvajanje rukovanja prezentacijskim sadržajem i podacima te njihove prezentacije.  
Model domene definira se u *model* dijelu obrasca, koji je u *RecepTour desktop* aplikaciji *(DesktopTour)* ostvaren i *O/R* mapiranjem preslikan pomoću *Entity Frameworka* iz baze podataka u projekt.   
*Viewmodel* dio *MVVM* obrasca zadužen je za dohvat i pripremu podataka koji su potrebni *View* dijelu obrasca. U *DesktopTouru*, *Viewmodel* je implementiran kroz nekoliko jednostavnih razreda, koji nemaju previše odgovornosti jer je *DesktopTour* namijenjen samo za pregledavanje sadržaja.  
*View* dio *MVVM* oblikovnog obrasca namijenjen je za prezentaciju sadržaja. Prilikom inicijalizacije *View* dijela oblikovnog obrasca postavlja se *context* razreda u kojemu se nalaze svi potrebni podaci za prikaz. Ključno je istaknuti da *code-behind* dio *WPF view* razreda ne dodaje niti mijenja podatke koji su mu u *context* postavljeni prilikom inicijalizacije na bilo koji način, u skladu sa razdvajanjem odgovornosti *MVVM* oblikovnog obrasca. U razredu *view* definiran je izgled i prikaz podataka sintaksom sličnom *HTMLU*.  
Kako bi se *View* razred povezao sa konkretnim podacima iz modela, *View* u svojim konstruktorima ili metodama stvara novi primjerak razreda *Viewmodel*, koji u svom konstruktoru iz repozitorija povlači potrebne podatke, zatim se kao *context* *View* razreda postavlja novostvoreni *Viewmodel*.

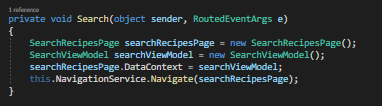


Figure 2 Primjer povezivanja VIewmodela i viewa preko DataContext svojstva

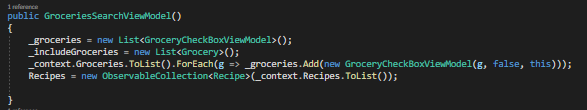


Figure 3 Dohvaćanje svih podataka modela u konstruktoru razreda Viewmodel

## Screenshots

Nakon pokretanja aplikacije korisnici dolaze na početnu stranicu gdje imaju mogućnost odabira između pretraživanja recepata po jednom od tri kriterija, ili pregleda svih recepata koji se nalaze u aplikaciji.

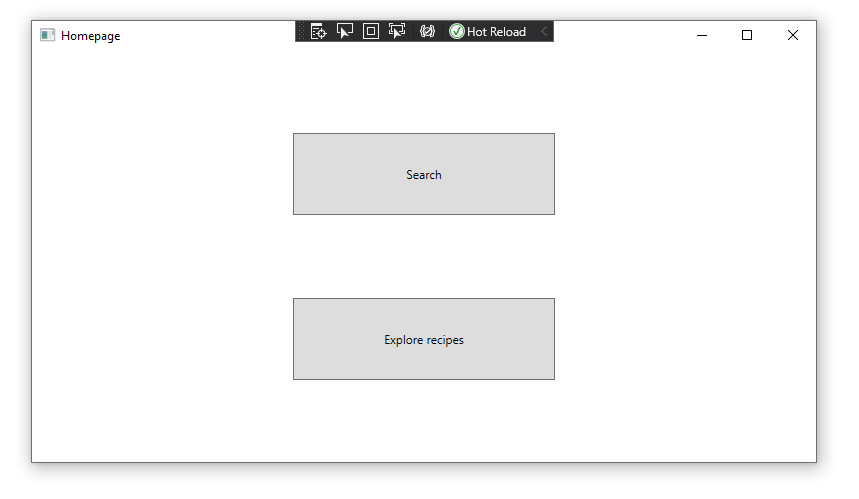


Figure 4 Početna stranica

Odabirom mogućnosti pretraživanja recepata po kriterijima korisnici se preusmjeravaju na stranicu pretraživanja recepata po sastojcima. U početku nijedna namirnica nije označena te se prikazuju svi recepti u bazi podataka. Kako bi korisnik pretražio bazu podataka po receptima koji sadrže neke sastojke mora označiti koje sastojke za koje želi doznati recepte, lista recepata se osvježava pri svakom označavanju ili odznačavanju sastojka.

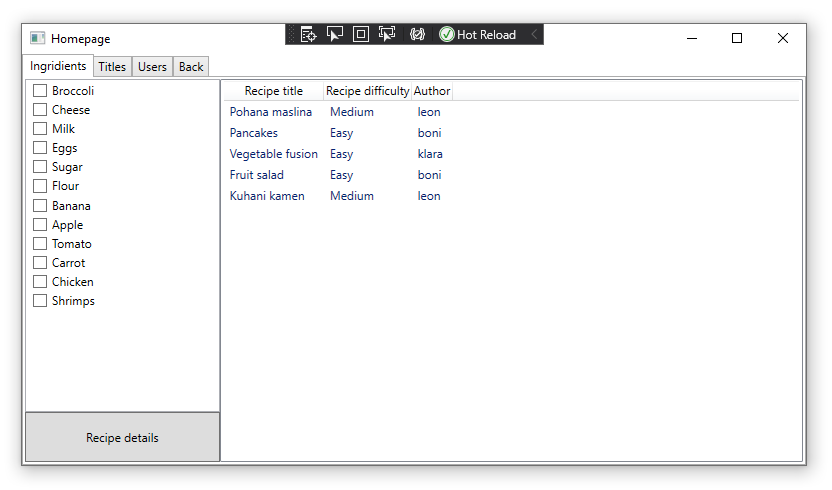


Figure 5 Pretraživanje recepata po sastojcima

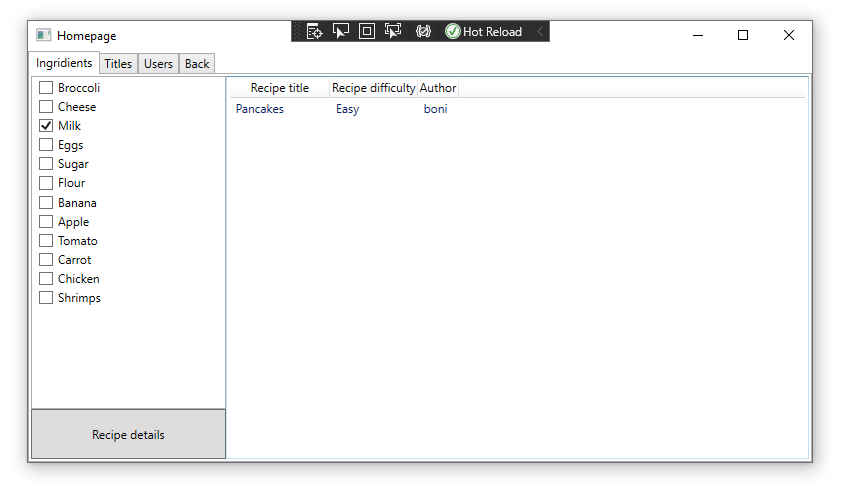


Figure 6 Recepti koji pri izradi zahtijevaju jaja

Ako korisnik želi doznati više informacija o nekom receptu najprije mora odabrati recept koji ga zanima iz liste recepata koja se nalazi na desnoj strani prozora te potom pritisnuti gumb *Recipe details*. Nakon pritiska na gumb korisnici su preusmjereni na prozor gdje se nalaze detalji o receptu poput težine izvedbe, slika, popisa sastojka i njihove količine.

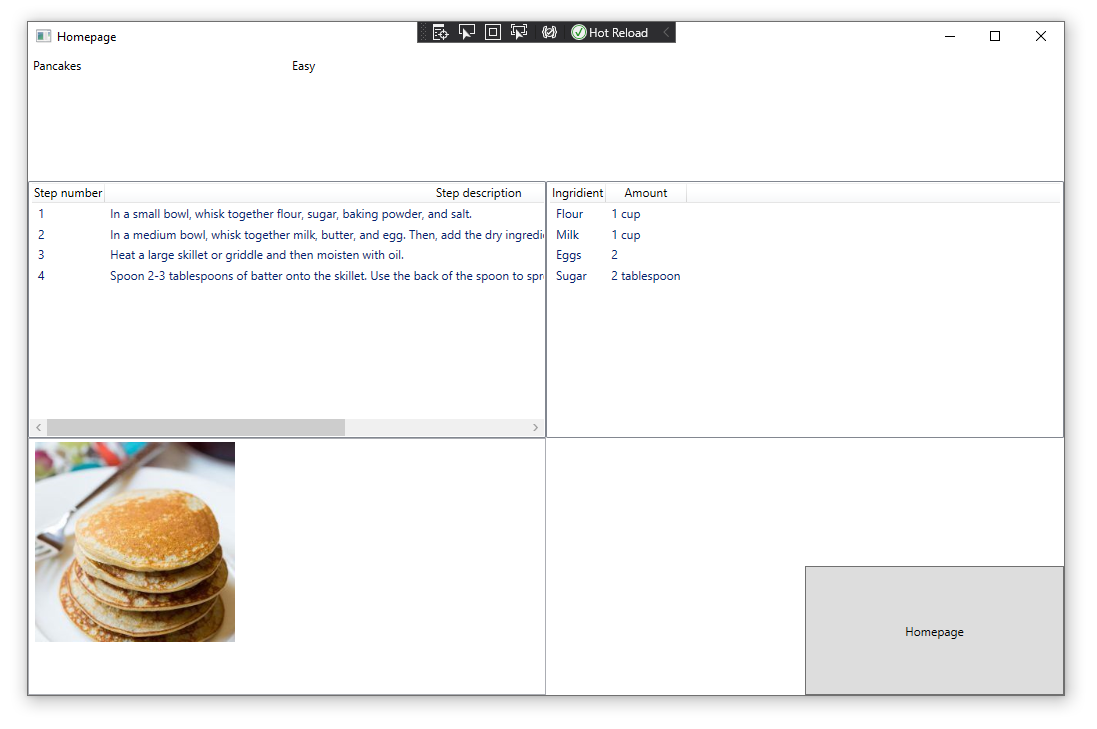


Figure 7 Detalji recepta za palačinke

Pritiskom na gumb *Homepage* korisnici se vraćaju na početnu stranicu.

Ako korisnici žele pretraživati recepte po naslovu najprije moraju kliknuti na karticu *Titles*, potom upisati naslov recepta koji ih zanima te naposljetku pritisnuti gumb *Search by title*.  
Sličan je pristup i pretraživanju recepta po autoru, samo što korisnici umjesto naslova recepta upisuju ime autora.

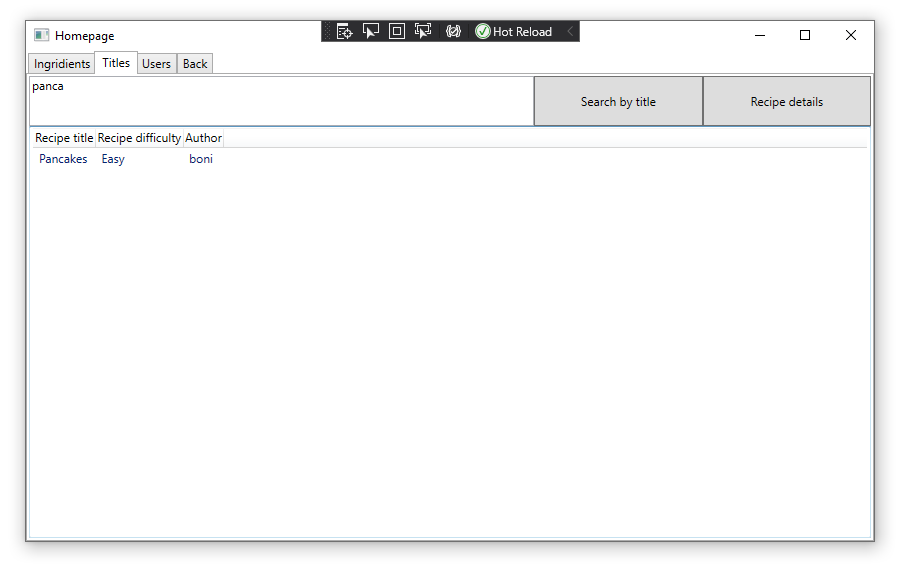


Figure 8 Pretraživanje recepta koji u naslovi sadrže „panca“

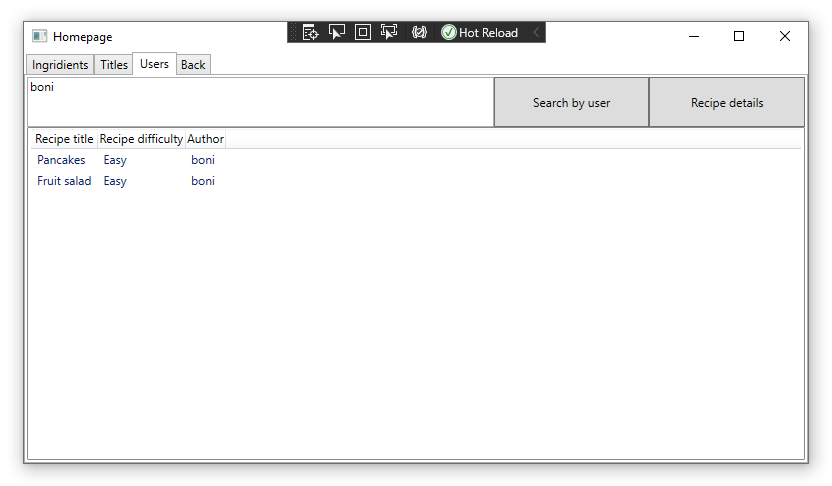


Figure 9 Pretraživanje recepta korisnika "boni"

Kako bi se vratio na početnu stranicu korisnik odabire karticu *Back* i pritiskom na gumb *Back to homepage* vraća se na početnu stranicu.

Ako korisnici žele pregledavati sve recepte u bazi podataka odabiru gumb *Explore recipes*, nakon čega dolaze na stranicu na kojoj se nalaze svi recepti u bazi podataka.

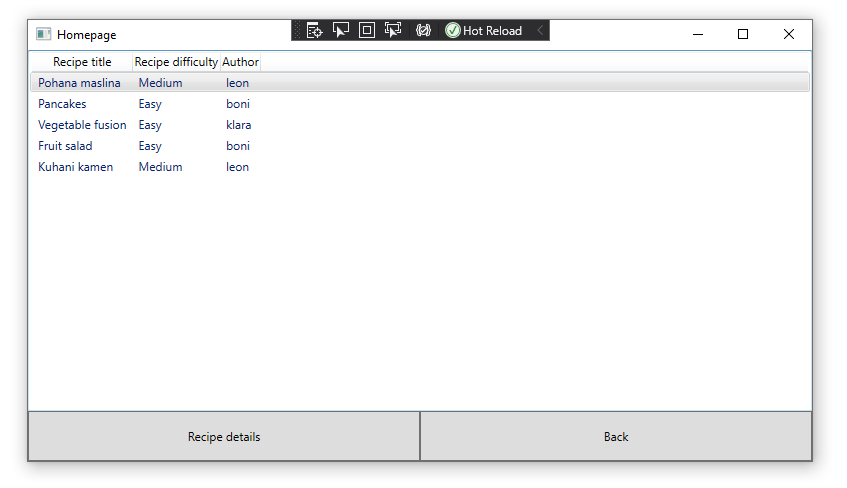


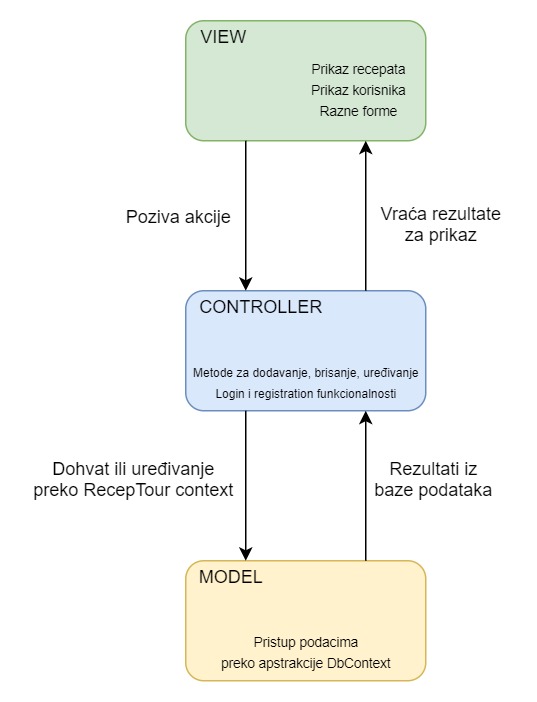
Figure 10 Pregledavanje svih recepata u bazi podataka

# WEB APLIKACIJA

## Uvod

Web aplikacija je implementirana pomoću ASP.NET MVC (*Model-View-Controller*) oblikovnog obrasca koji je korišten za razdvajanje korisničkog sučelja (*View*), podataka (*Model*) i aplikacijske logike (*Controller*). Model domene je ovdje također definiran u *model* dijelu obrasca, a preslikan je *O/R* mapiranjem preslikan pomoću *Entity Frameworka* iz baze podataka u projekt.

Konkretno, korisnik pristupa aplikaciji preko korisničkog sučelja. Korisničko sučelje poziva metode sloja kontrolera i dobiva rezultate od njega koje prikaže korisniku nakon izvršenog upita. Budući da se koriste tzv. *fat controlleri*, logika dohvata podataka se nalazi na sloju kontrolera te se najviše toga dešava upravo ovdje. Kontroleri preko *RecepTourContext*-a pristupa podacima iz baze podataka i koristi rezultate koje filtrira i šalje na korisničko sučelje. Visoki pregled MVC arhitekture u aplikaciji je prikazan na sljedećoj slici:



Dijelimo korisničko sučelje na *front-end* dio, a kontroleri i model su dio *back-enda*. *RecepTourContext* je ubačen u kontrolerske klase preko *Depenency Injection*-a i vraća podatke iz baze podataka koji se deserijaliziraju u klase modela (*Recipe*, *User*, *Grocery*, …). Također, uz *RecepTourContext*, koriste se i klase *UserManager* i *SignInManager* koje služe za login i registraciju. One također se mogu koristiti tj. Injektirati u sloj korisničkog sučelja kako bi se određeni elementi sakrili ako korisnik nije logiran u sustav.

Korištenje aplikacije kroz korisničko sučelje je prikazano u nastavku.

## Screenshots

Nakon pokretanja aplikacije korisnik je pozicioniran na početnoj stranici gdje može odabrati registraciju, login ili da mu se prikažu svi recepti koji se nalaze u aplikaciji.

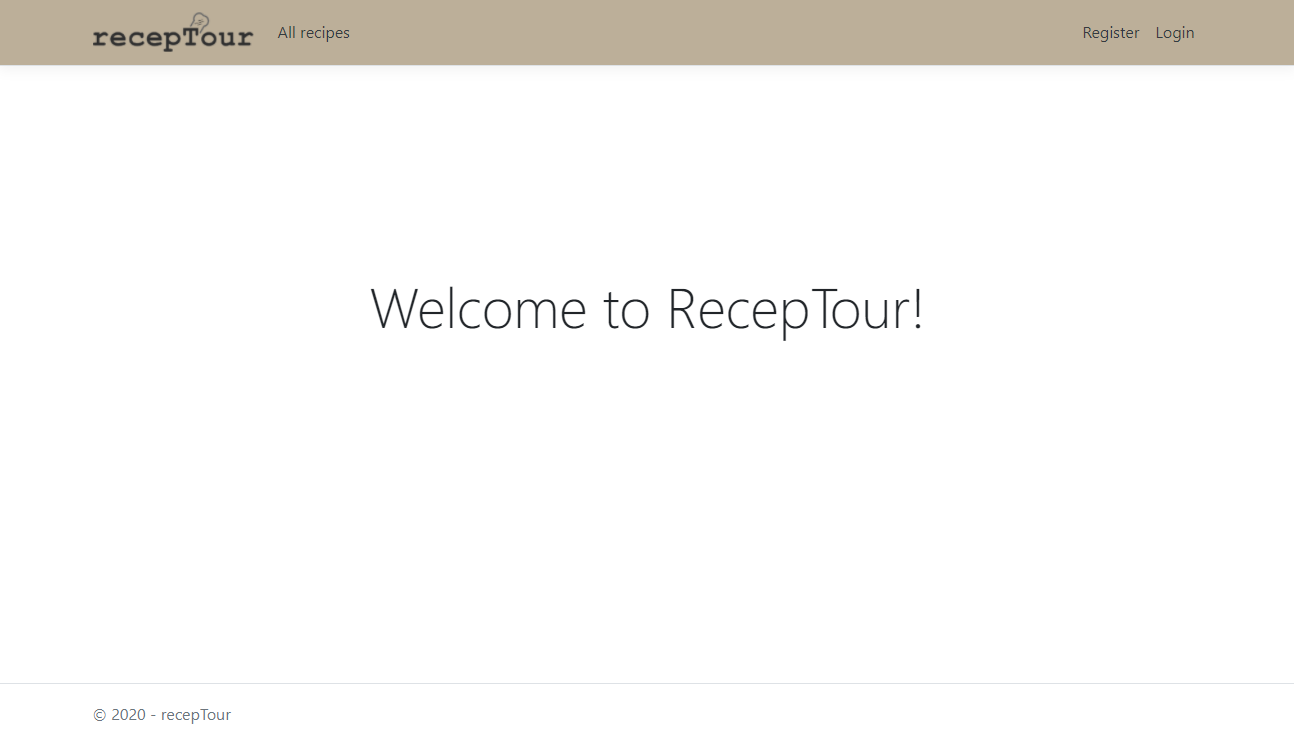


Figure 11 Početna stranica

Klikom na *All recipes* korisniku se prikazuje popis svih recepata, kao i tražilice za pretragu prema nazivu recepta, korisniku ili sastojcima.

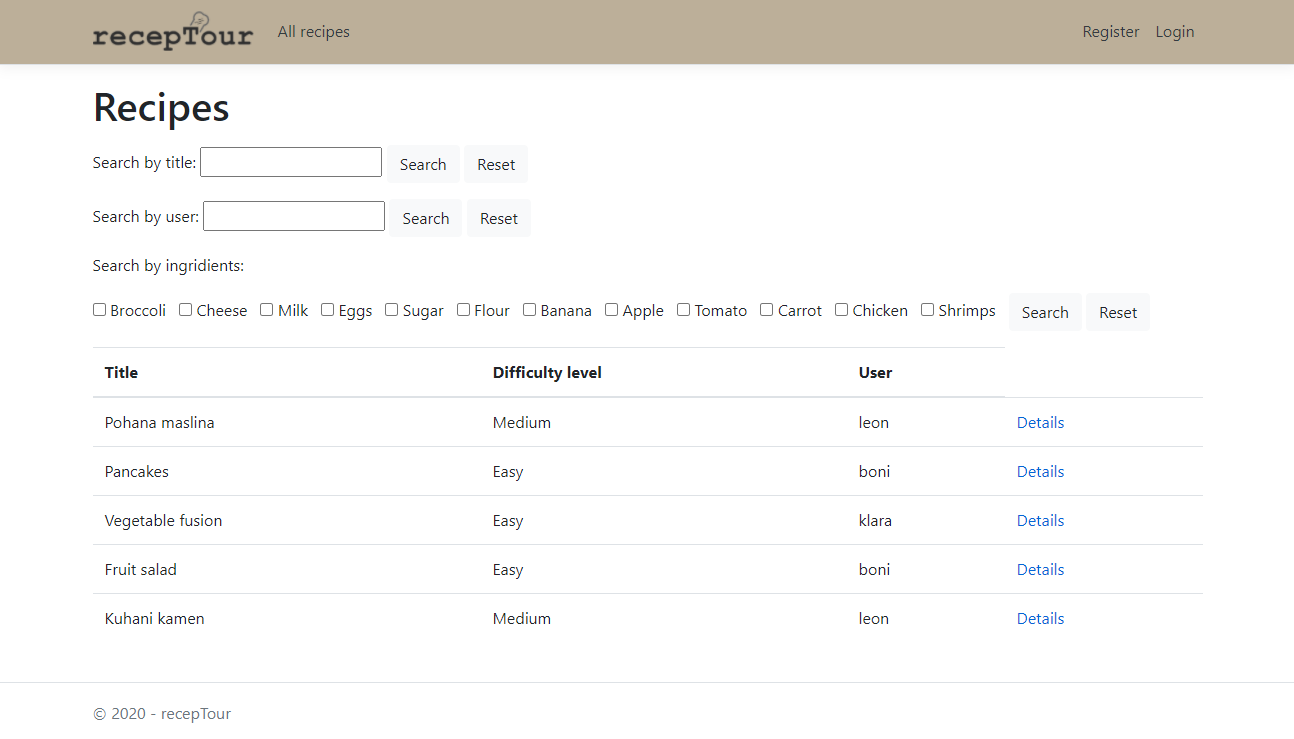


Figure 12 Prikaz svih recepata

Ukoliko korisnik traži recepte po naslovu, dovoljno je da u *textbox* upiše naziv recepta kojeg želi pronaći i klikne na gumb *Search* koji će korisniku prikazati sve recepte tog naziva.

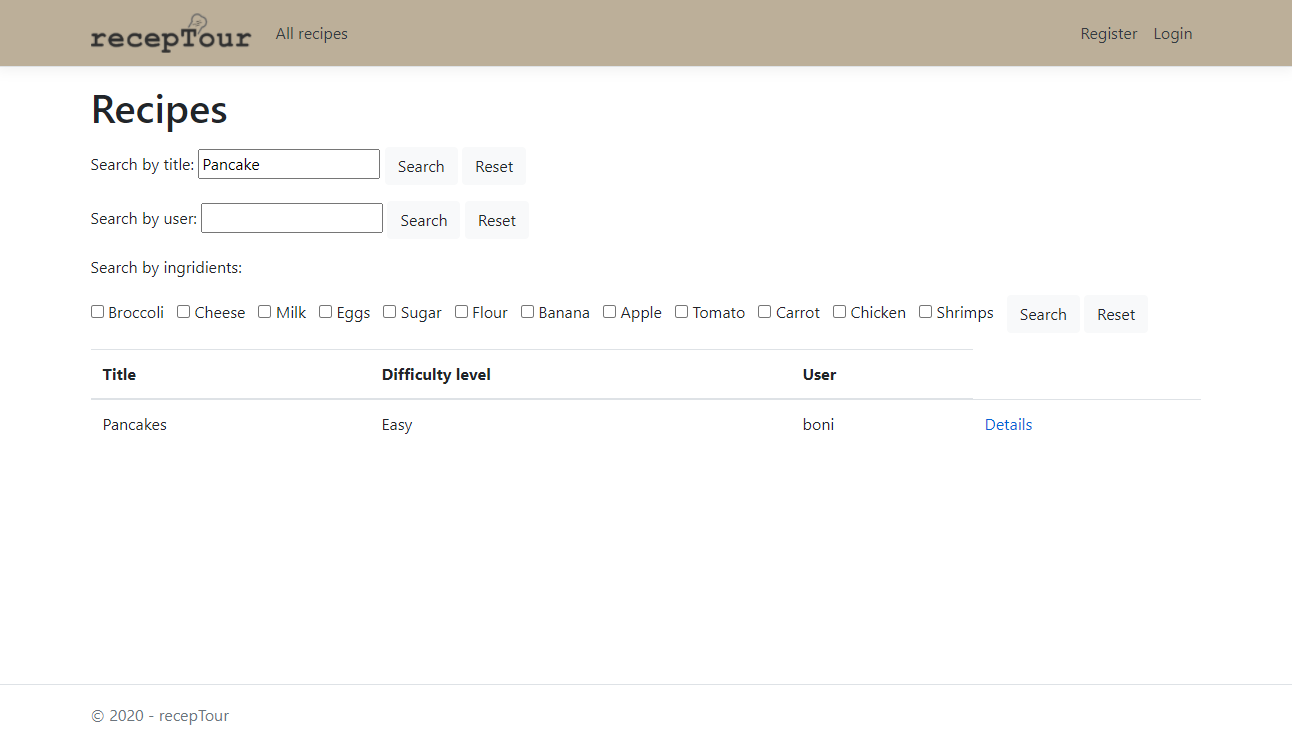


Figure 13 Pretraga recepta po naslovu

Ako korisnika zanima potpuni recept, može kliknuti na *Details* što će mu prikazati detaljan opis recepta, kao i potrebne namirnice te sliku jela, ukoliko ona postoji.

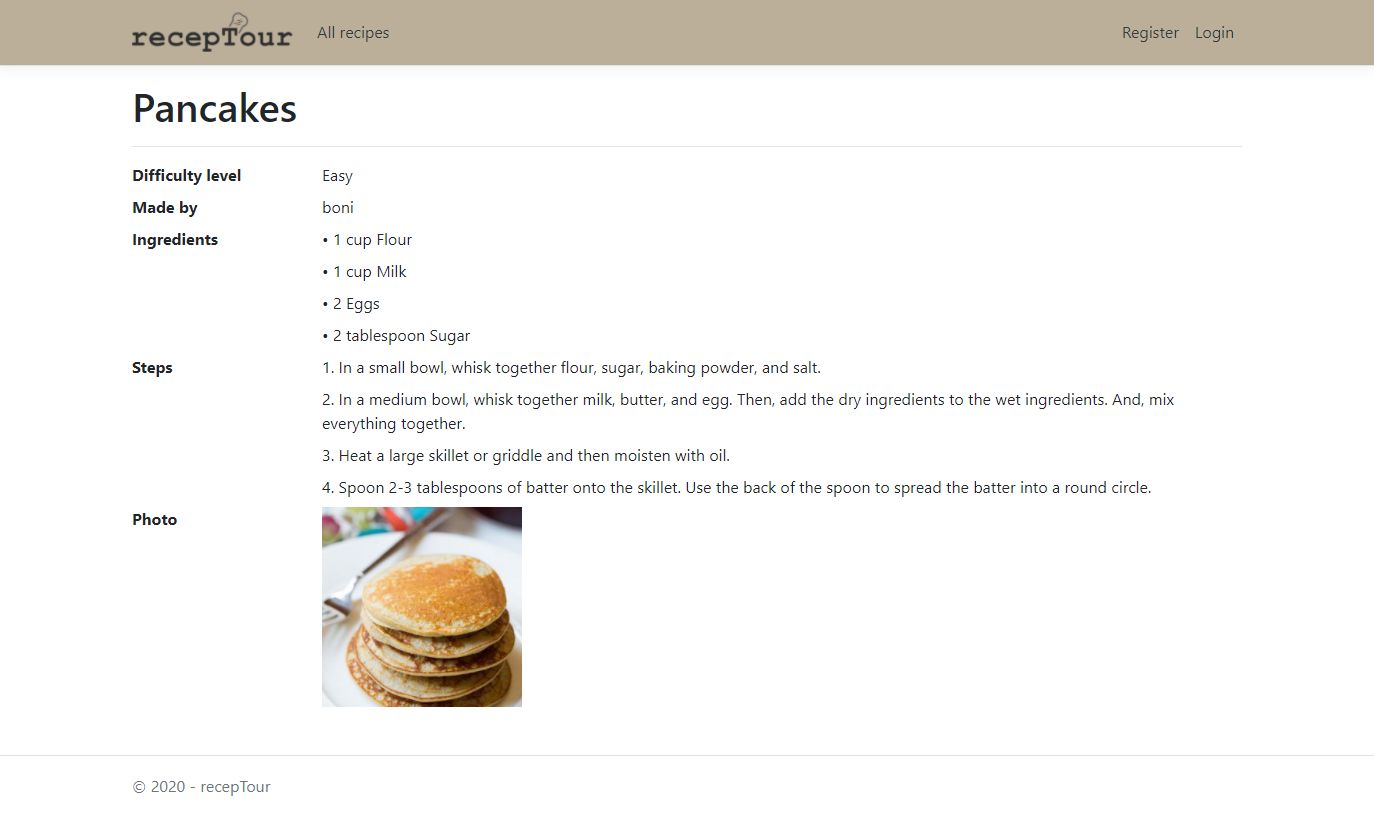


Figure 14 Prikaz detalja recepta

Korisnik ima opciju tražiti recepte po *username*-u korisnika. U tražilicu upiše *username* osobe čije recepte želi pregledati, te mu se recepti te osobe izlistaju.

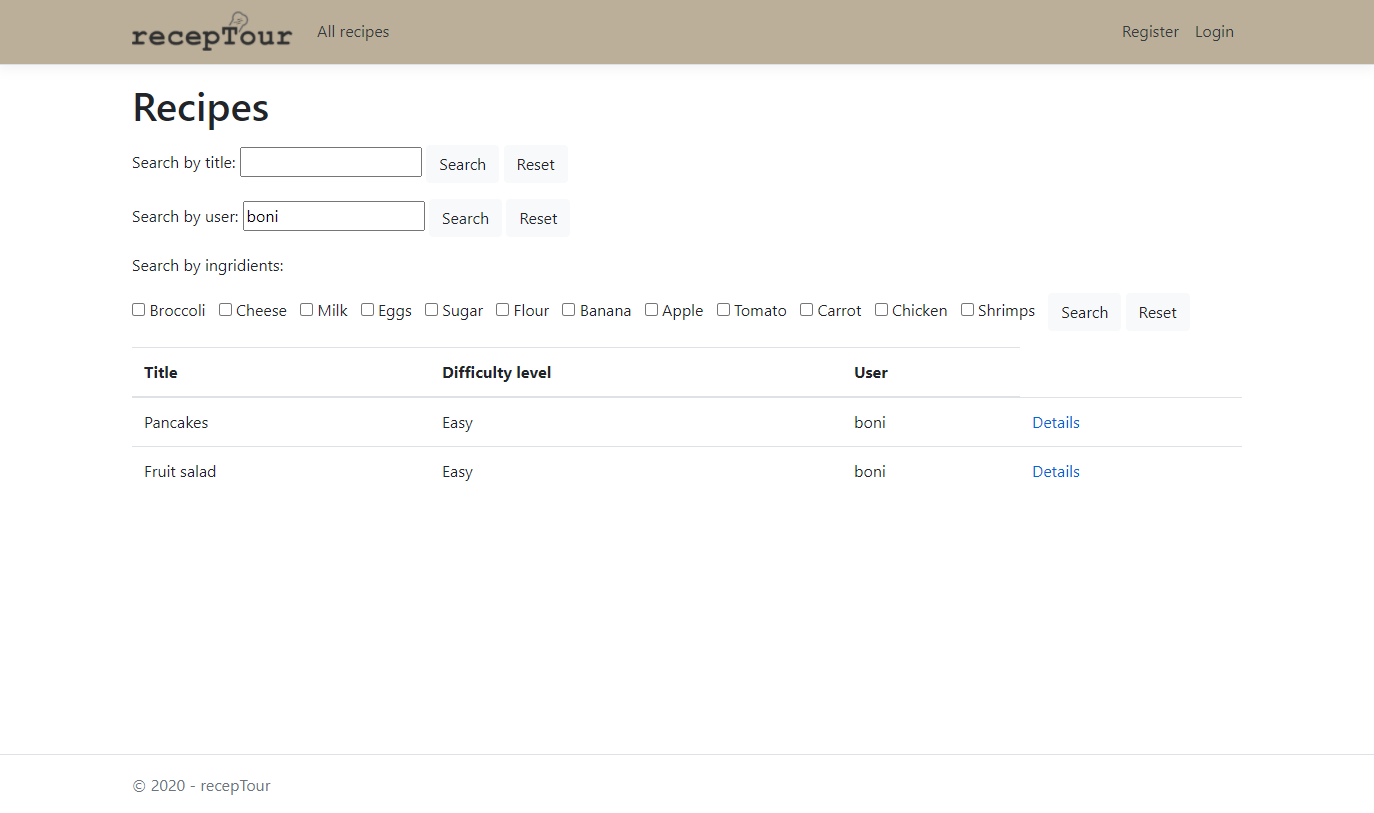


Figure 15 Traženje po username-u korisnika

Glavna funkcionalnost aplikacije leži u pretrazi po namirnicama. Korisnik može odabrati više namirnica od kojih želi napraviti jelo te to čini klikom na *checkbox* uz pojedinu namirnicu. Nakon što je označio sve namirnice, treba kliknuti na *Search* što će mu prikazati sve recepte koje može pripremiti od odabranih namirnica.

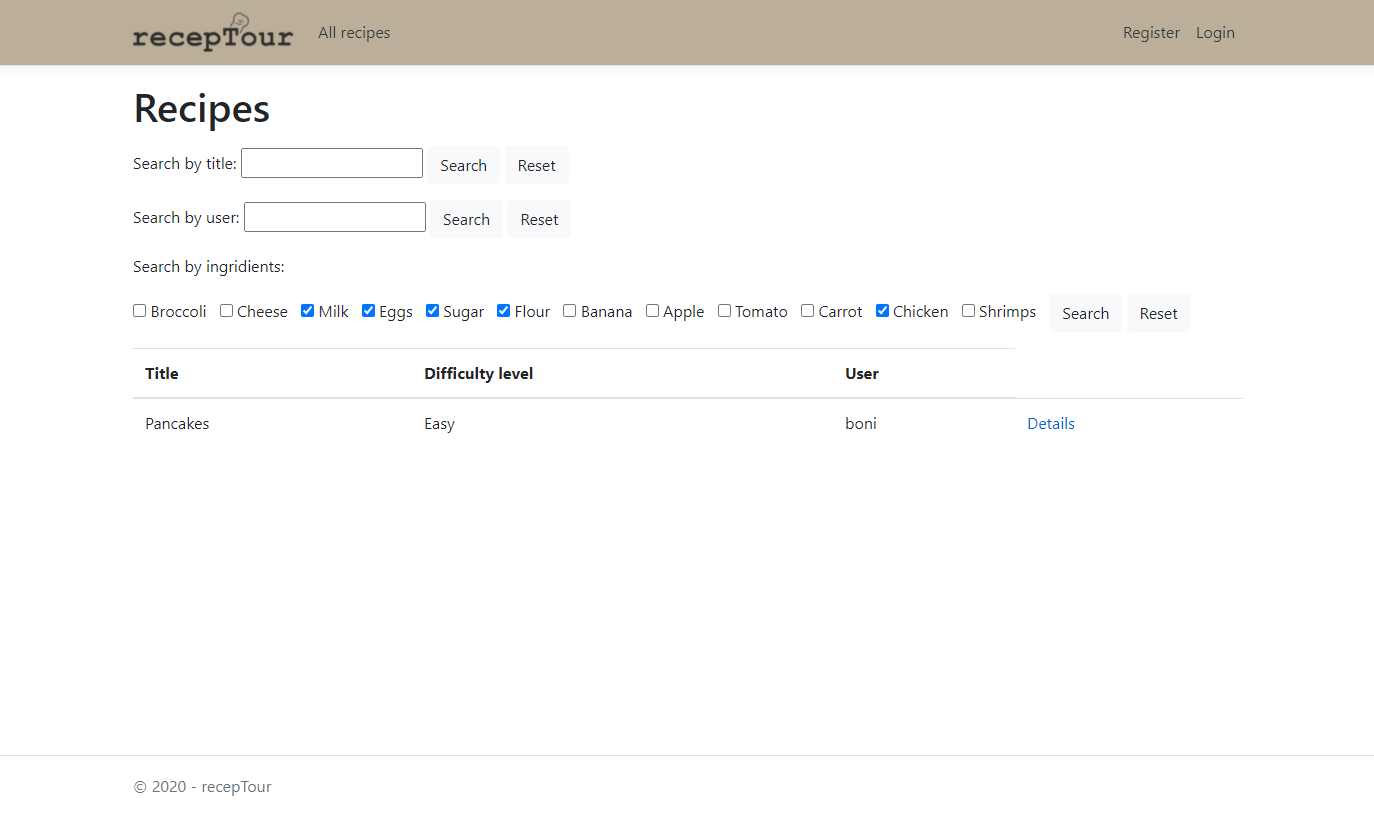


Figure 16 Pretraga recepata po namirnicama

Ukoliko korisnik želi koristiti dodatne funkcionalnosti poput dodavanja vlastitih recepata, u tom slučaju se prvotno treba registrirati. To čini klikom na *Register* u navigacijskoj traci što ga preusmjerava na stranicu za registraciju. Bitno je da korisnik upiše dobar format e-maila, da lozinka i potvrda lozinke budu jednaki unosi te da *username* već ne postoji u aplikaciji.

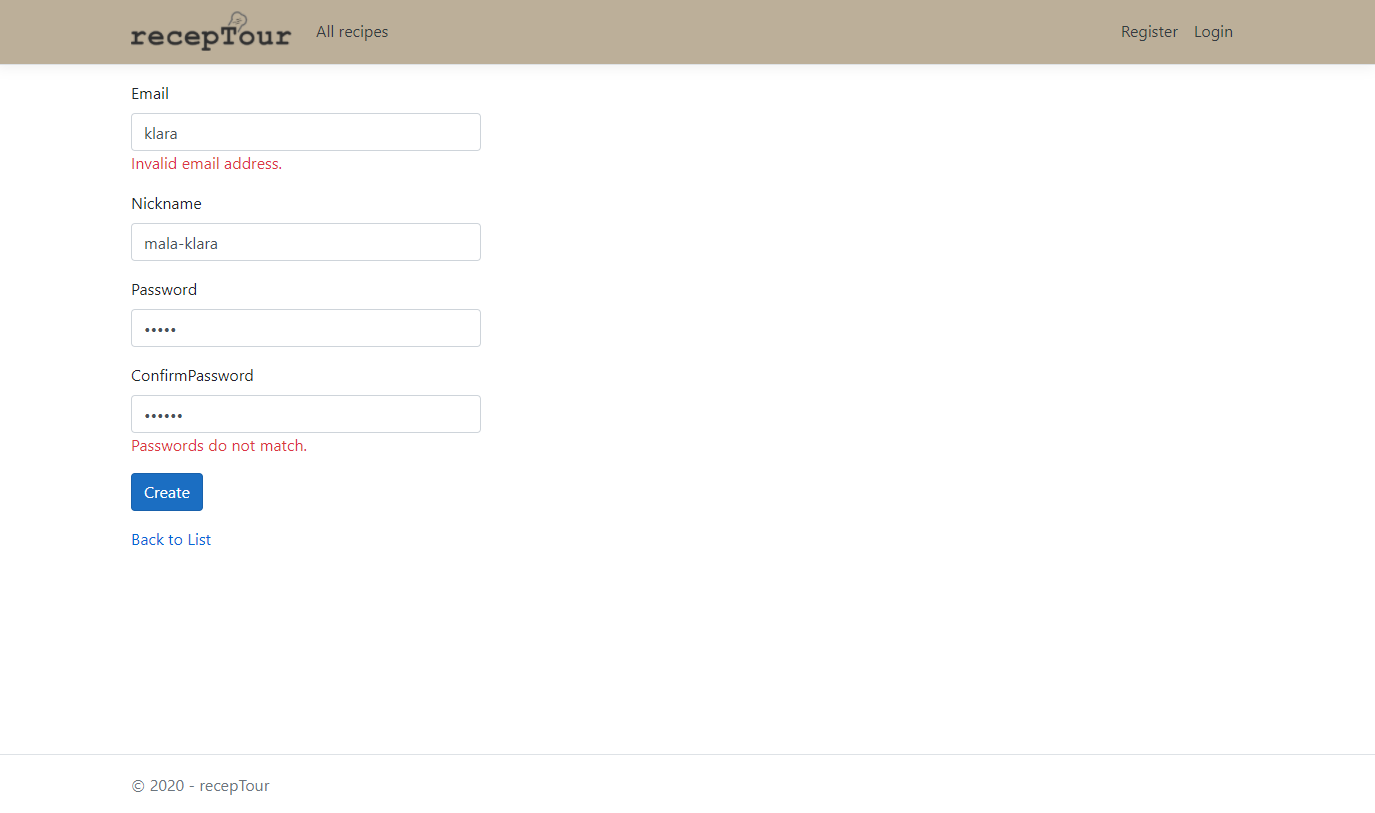


Figure 17 Registracija korisnika

Nakon što se uspješno registrirao, korisnik može pristupiti stranici za prijavu klikom na Login u navigacijskoj traci. Treba unijeti svoju e-mail adresu i lozinku čime se uspješno prijavljuje u sustav.

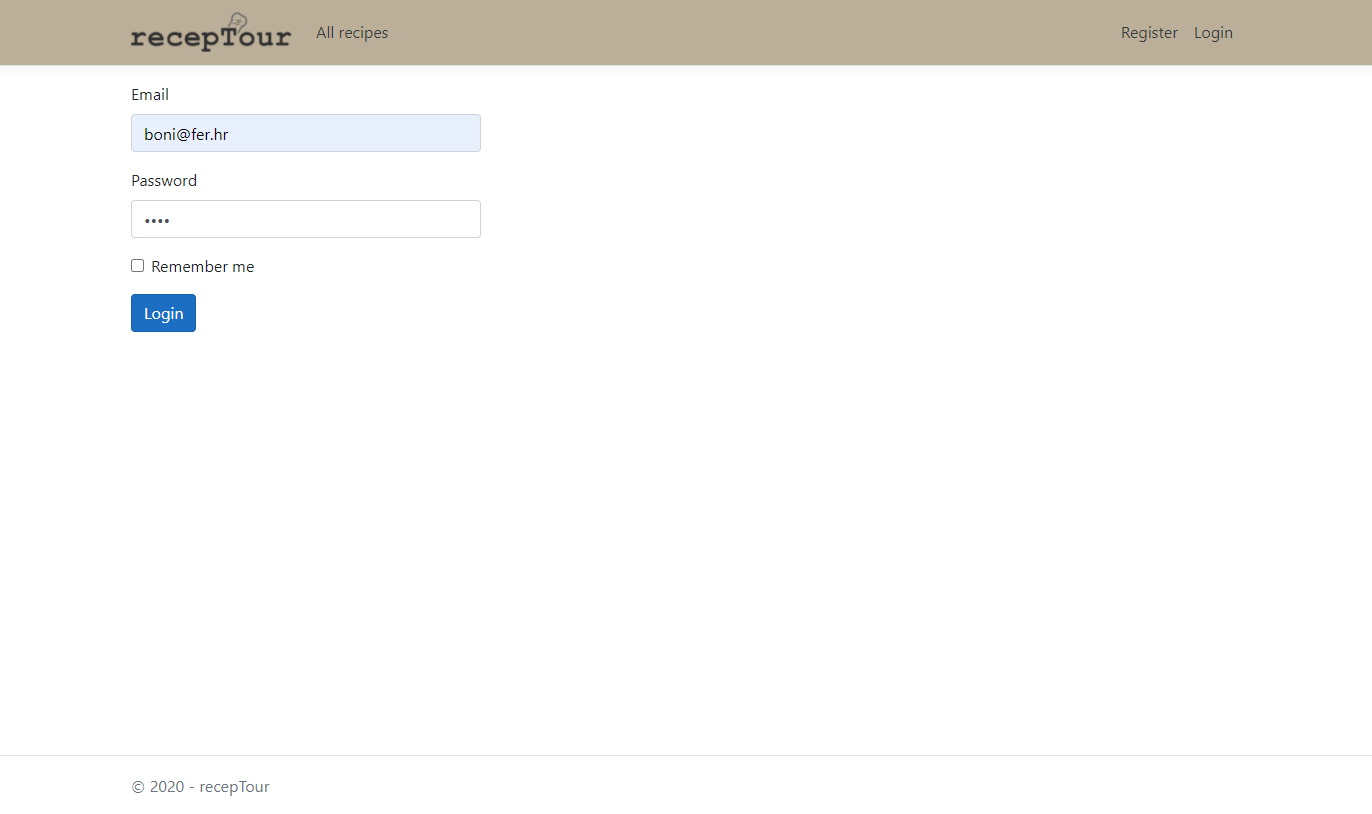


Figure 18 Prijava korisnika u sustav

Nakon što se korisnik prijavio u sustav, može vidjeti da su mu se u navigacijskoj traci dodale nove opcije. Korisnik može kreirati vlastiti recept klikom na *Create a recipe*, pregledati vlastite recepte na *My recipes*, pregledati svoje favorite na *Favorites*, pristupiti svom korisničkom računu klikom na e-mail adresu te odjaviti se iz sustava klikom na *Logout*.



Figure 19 Opcije prijavljenog korisnika u navigacijskoj traci

Klikom na Create a recipe korisnika se preusmjeri na stranicu gdje treba prvo unijeti naziv recepta i pripadnu težinu pripreme.

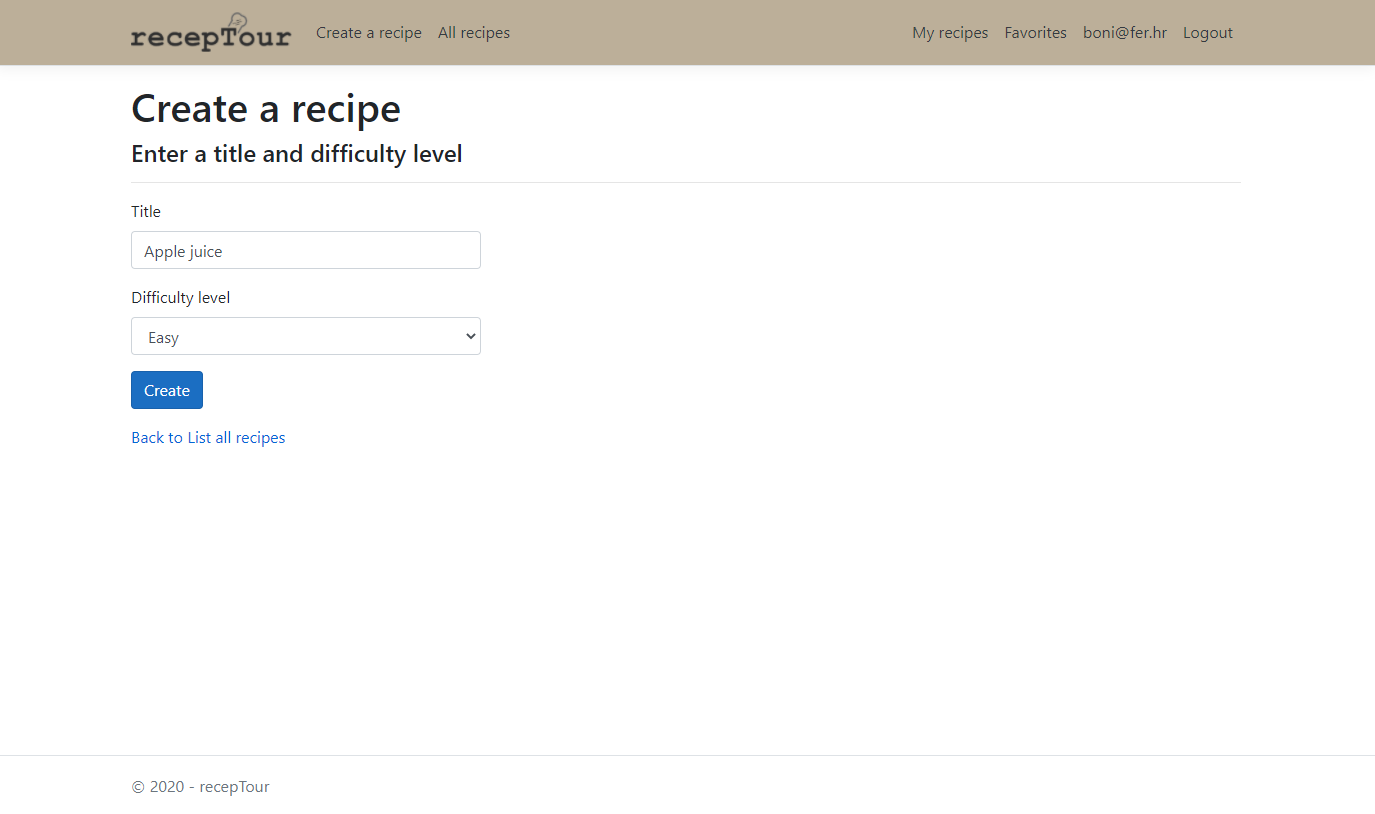


Figure 20 Unos novog recepta

Klikom na Create se kreira recept kojemu korisnik treba dodati detalje. Prvo treba unijeti postupak pripreme jela od više koraka što čini na sljedećoj stranici:

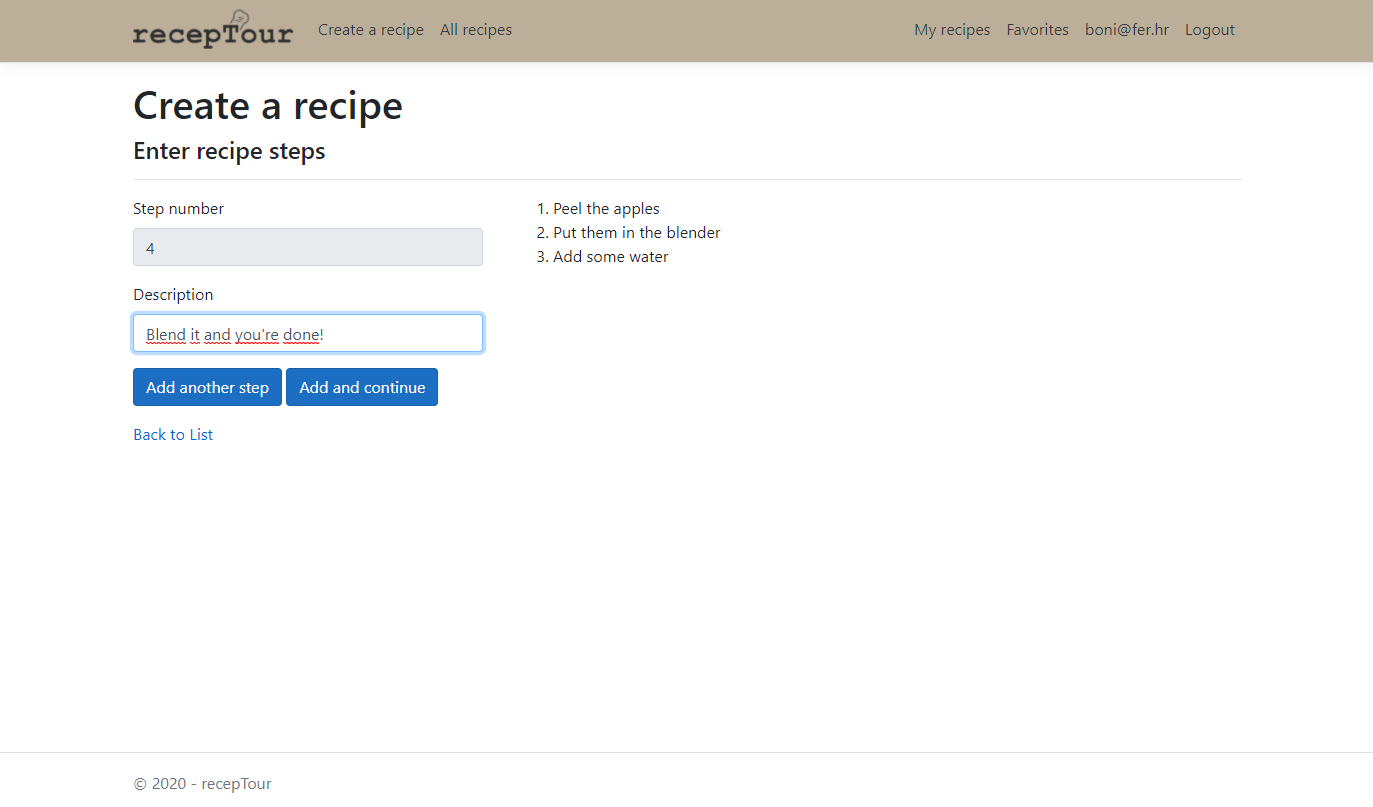


Figure 21 Unos koraka recepta

Zatim korisnik u aplikaciju treba unijeti sve sastojke, odnosno namirnice i količinu pojedine namirnice:

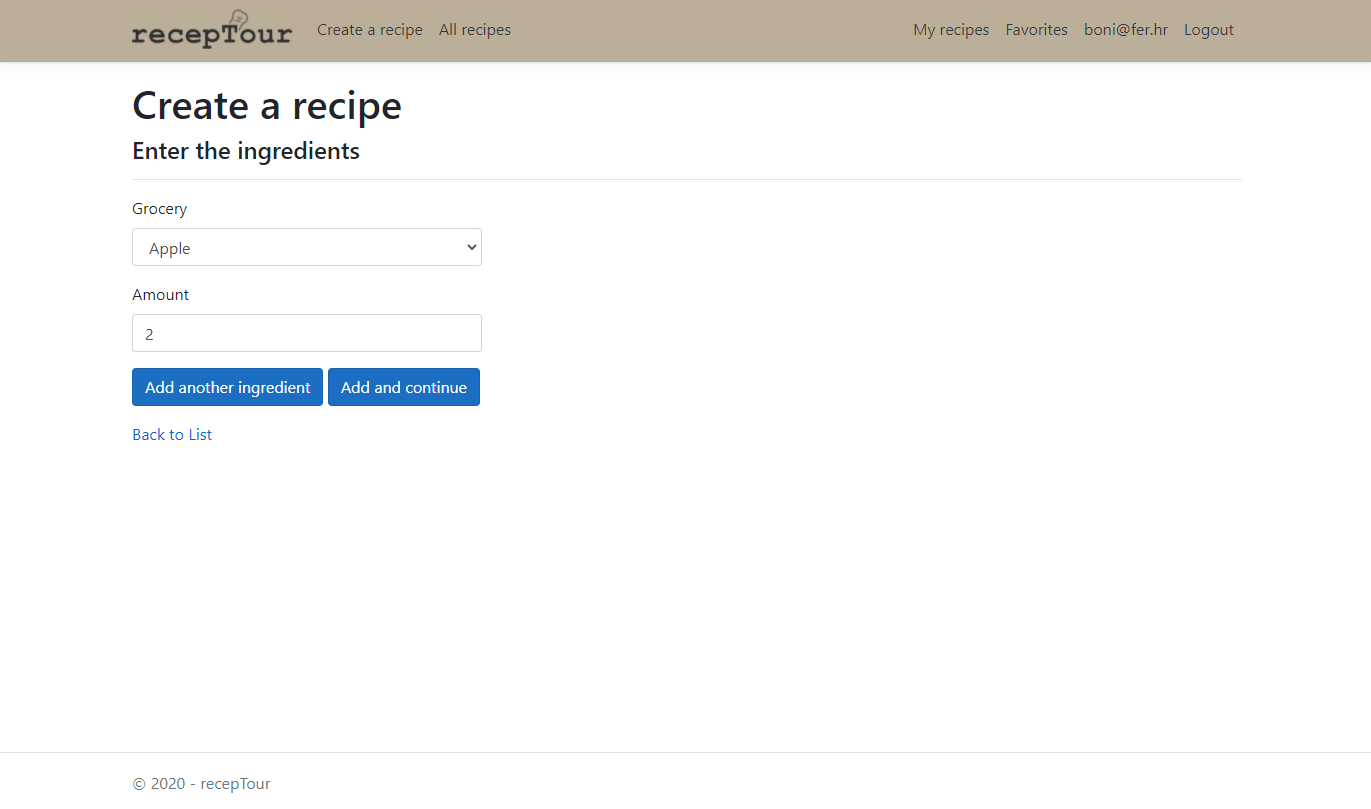


Figure 22 Unos namirnica i pripadne količine

Na samom kraju korisnik opcionalno može dodati sliku pripadnog recepta:

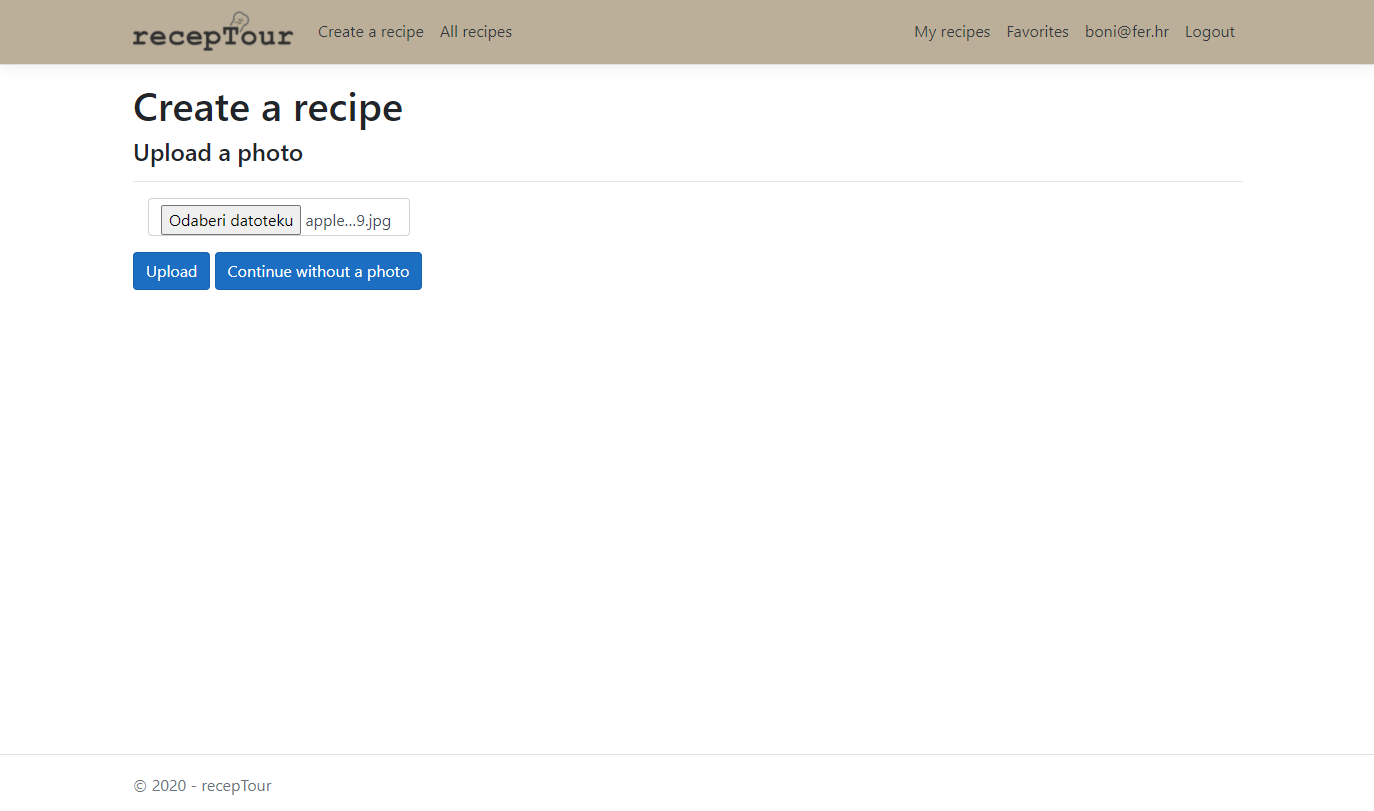


Figure 23 Unos slike za recept

Klikom na *Upload* ili *Continue without a photo*, korisnika se preusmjerava na gotov recept koji se sastoji od svih unesenih detalja:

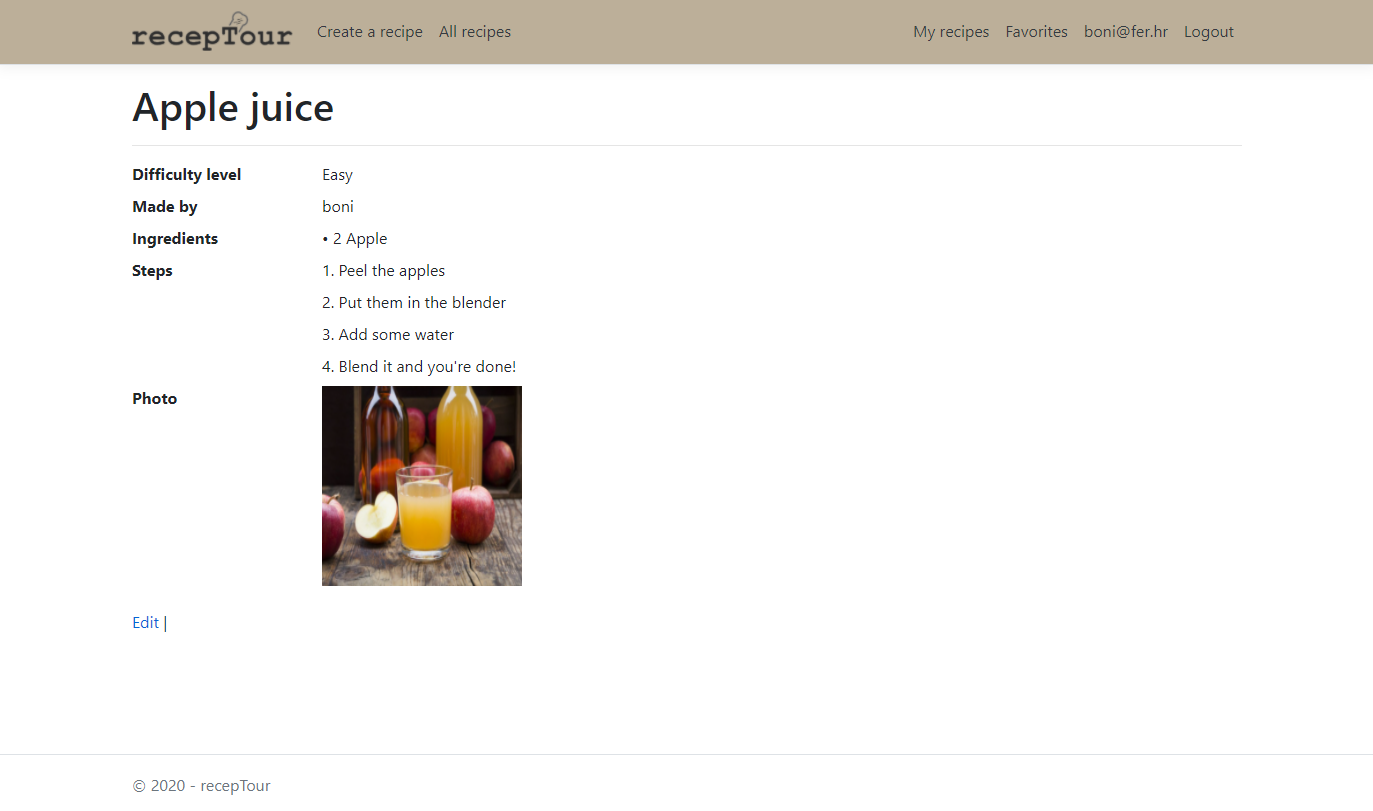


Figure 24 Detalji novounesenog recepta

U navigacijskoj traci korisnik može pristupiti receptima koje je on sam unio u aplikaciju te ih može mijenjati ili brisati:

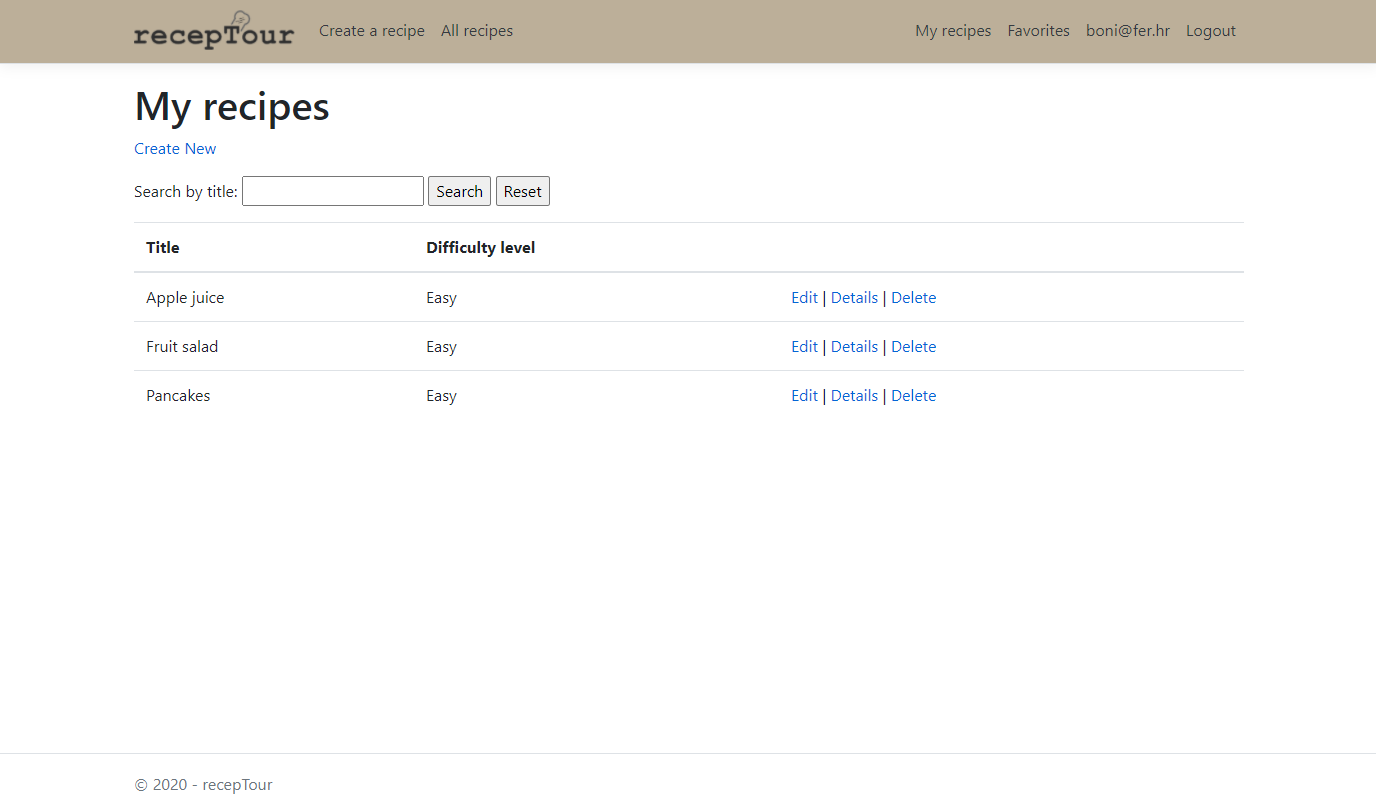


Figure 25 Prikaz recepata prijavljenog korisnika

Klikom na Favorites, korisnika se preusmjerava na stranicu gdje su popisani svi recepti koje je on spremio u svoje favorite. Tamo može pogledati detalje tih recepata ili ih maknuti iz favorita.

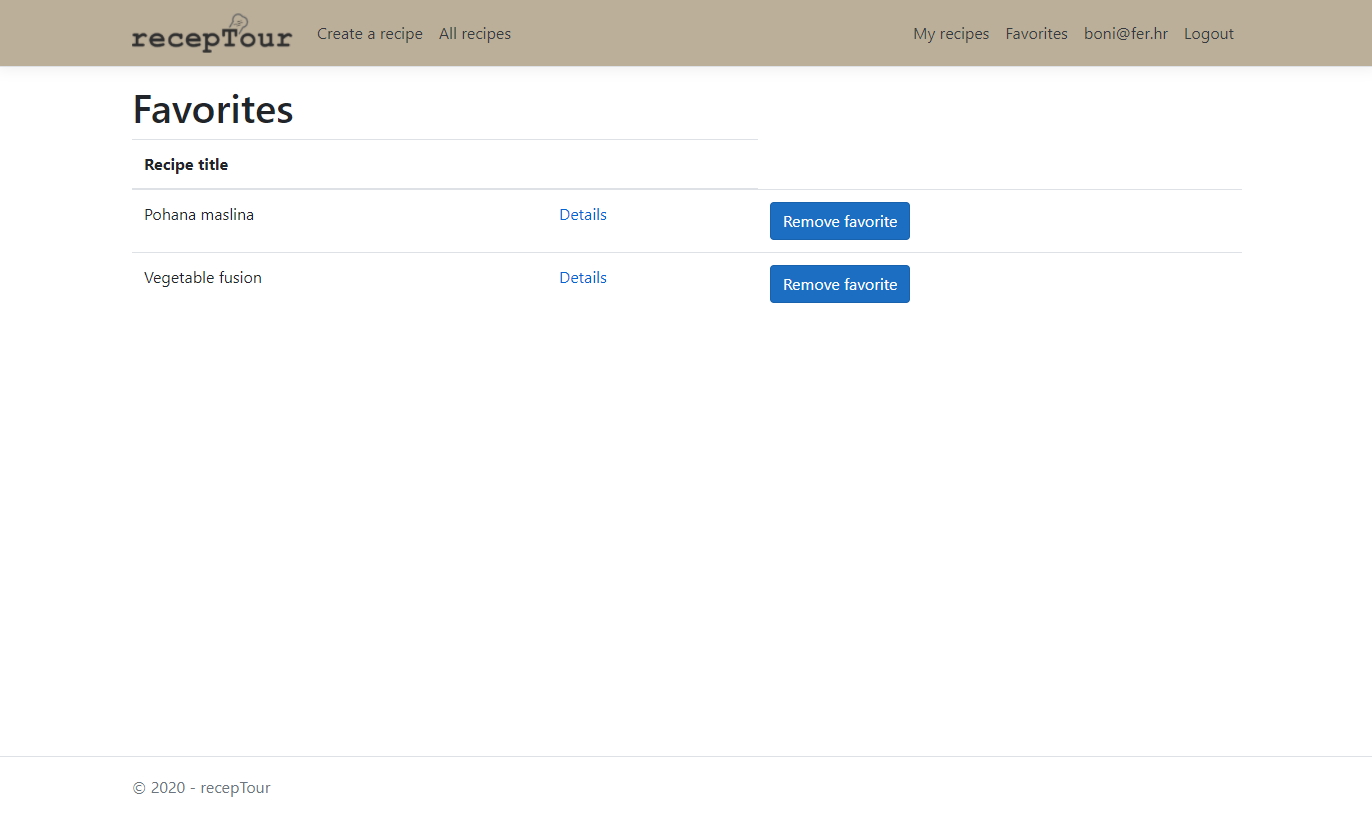


Figure 26 Prikaz omiljenih recepata prijavljenog korisnika

Kada prijavljeni korisnik pregledava sve recepte, vidi više opcija nego neprijavljeni korisnik. Pojedini recept automatski može dodati u omiljene recepte klikom na Favorite uz taj recept, a svoje recepte može vidjeti i automatski ih tamo mijenjati ili obrisati.

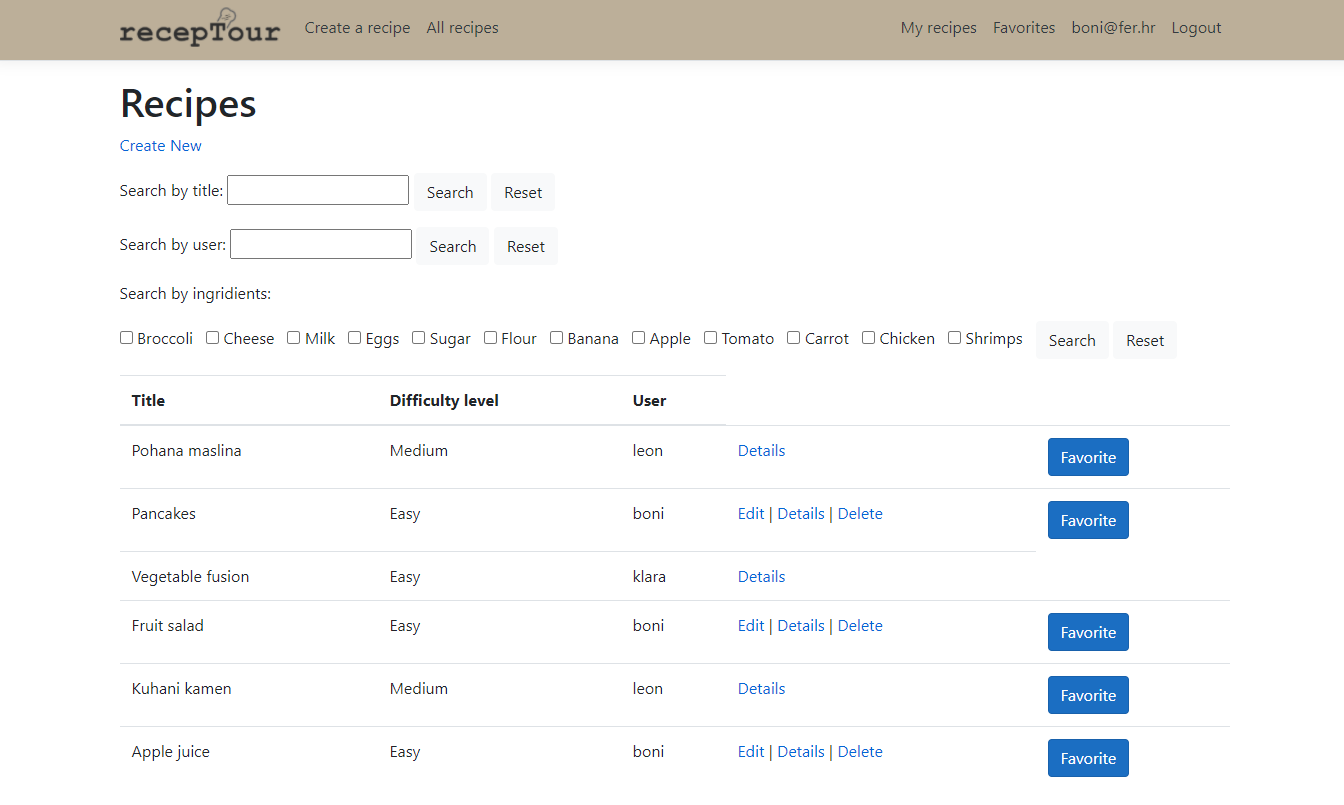


Figure 27 Prikaz svih recepata za prijavljenog korisnika

Korisnik može pristupiti svom profilu klikom na vlastiti e-mail u navigacijskoj traci. To ga preusmjerava na stranicu na kojoj može promijeniti svoje informacije – *nickname* i status, ali i lozinku.

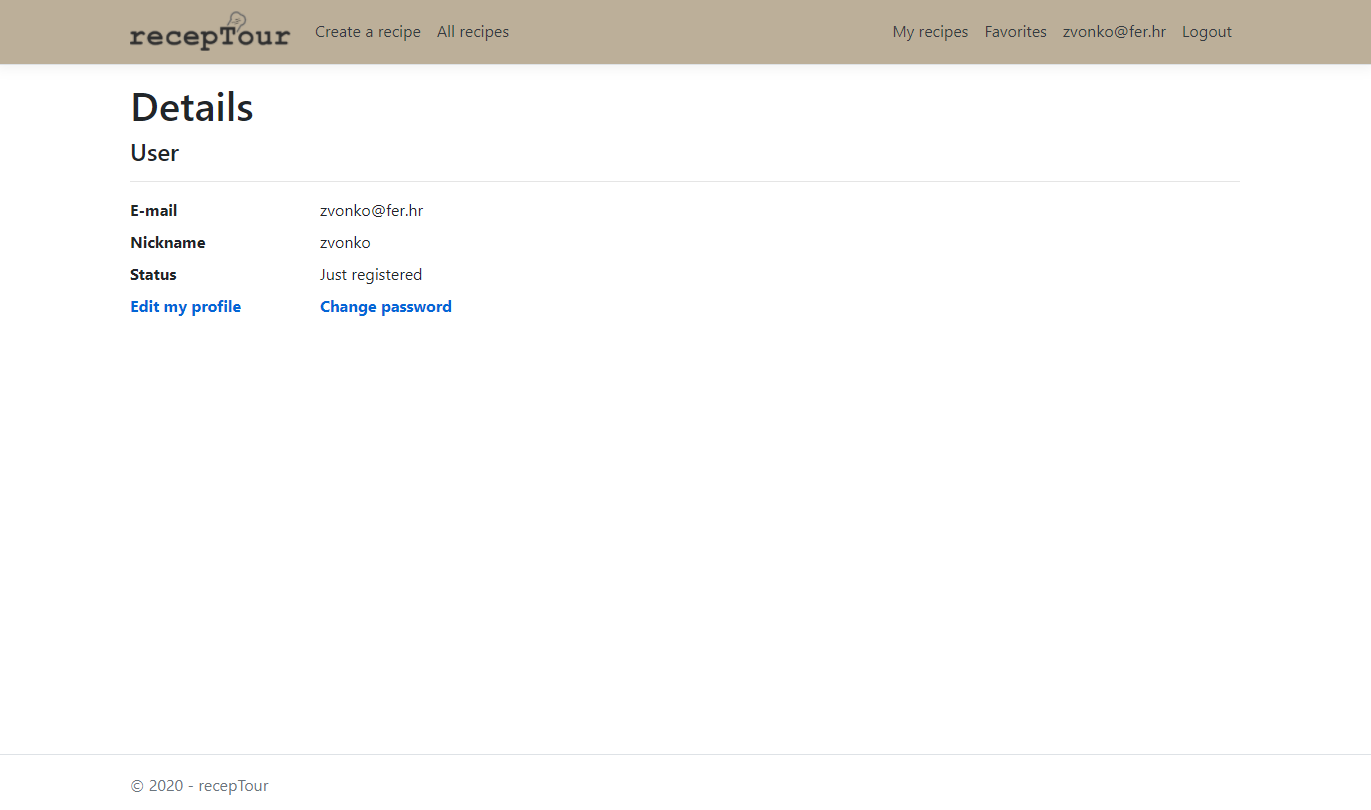


Figure 28 Prikaz korisničkog profila

Korisnik se može odjaviti klikom na Logout u navigacijskog traci čime ga se preusmjerava na početnu stranicu aplikacije.