Web aplikacija s bazom recepata

Vartušek, Emil

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Preddiplimski studij računarstva

WEB APLIKACIJA S BAZOM RECEPATA

Završni rad

Emil Vartušek

Osijek, 2017.

Sadržaj

1. UVOD

U današnjem vrijeme, Internet ima vrlo veliku ulogu u životu skoro svakoga pojedinca. Pojedinac se može na njega osloniti u svakom trenutku i velika je vjerojatnost da na njemu, ako se dovoljno potrudi, može pronaći odgovor na svako pitanje. Baš zbog toga Internet koristi sve više i više ljudi kako bi došli do recepata za razna jela, a pogotovo za deserte. Prema tome, web aplikacije koje omogućuju svojim korisnicima pronalazak i pisanje određenih recepata za deserte uvijek su dobrodošle i zbog toga se ovaj završni rad i bavi tom tematikom: izradnja web aplikacije s bazom recepata za deserte.

1.1 Cilj

Cilj ovoga završnoga rada je izraditi funkcionalnu web aplikaciju koja će svojim korisnicima omogućiti čitanje i pisanje recepata, te im isto tako omogućiti i provjeru da li posjeduju određene sastojke potrebne za izradu određenog deserta putem dijela aplikacije koji će se nazivati smočnica (engl. *pantry*). Važno je napomenuti kako će svi recepti biti vidljivi svakome tko posjeti ovu web aplikaciju, no recepte će moći pisati samo oni korisnici koji su se prethodno registrirali i prijavili, tj. oni korisnici koji su spremljeni u bazu podataka. Isto tako će biti i sa samom smočnicom koju će također moći koristiti samo prethodno registrirani korisnici. Prethodno registrirani korisnici dijelit će se na obične korisnike koji će imati mogućnost pisanja i brisanja recepata, te mogućnost dodavanja i brisanja sastojaka u svoju smočnicu, te na administratore koji će imati iste mogućnosti kao i obični korisnici, ali će uz njih imati i mogućnost brisanja određenih korisnika.

2. TEHNOLOGIJE KORIŠTENE ZA IZRADU WEB APLIKACIJE

Kako bi se ova web aplikacije realizirala, bilo je potrebno koristiti brojne tehnologije kako bi se postigla njena konačna funkcionalnost i dizajn. Tako je kao mrežni programski okvir (engl. *web application framework*) korišten *Ruby on Rails* koji je zbog svoje pristupačnosti omogućio nesmetan i olakšan rad u HTML-u i CSS-u, te je isto tako omogućio izradu i dobru povezanost baze podataka s ostatkom aplikacije putem *SQLite* baze podataka. Također, kako bi aplikacije ljepše izgledala, korišten je i tzv. *front-end* mrežni okvir *Bootstrap* koji nudi brojne načine uređivanja web aplikacije. Na kraju valja spomenuti kako su kao hosting servis za *Git* repozitorije i server korišteni *GitHub* i *Heroku*.

2.1. Ruby

Ruby je objektno orijentirani višeplatformski i dinamički programski jezik. Otvorenog je koda, a prvi puta se pojavio 1995. godine u Japanu kada ga je razvio Yukihiro Matsumoto. Programski jezici *Pearl, Smalltalk, Eiffel, Ada* i *Lisp* imali su najveći utjecaj na razvoj ovog programskog jezika te su od njih preuzete neke njihove karakteristike i prenesene u novi programski jezik, *Ruby*.

Razvijajući *Ruby*, Matsumotuov glavni cilj je bio razviti jezik koji neće zbunjivati iskusne programere, ali isto tako i jezik koji će biti zabavan i jednostavan za bilo kojega korisnika. Kako bi se to realiziralo, njegov novi jezik maksimalno je smanjio programerski posao i mogućnost pogreške pri pisanju programskog koda.

Nadalje, kako je *Ruby* objektno orijentirani program, svaka njegova vrijednost smatra se objektom, pa čak i *integer* i *boolean* vrijednosti. Varijable se uvijek referenciraju na objekte, a svaka fukncija biva metoda koju poziva neki objekt. Također, *Ruby* podržava i nasljeđivanje klasa, te je opisan kao višepokazni računalni jezik koji dopušta proceduarlno i funkcionalno programiranje. Semantika ovoga jezika veoma je slična semantici programskog jezika *Smalltalk*.

Što se tiče sintakse, ona veoma sliči onoj iz programskih jezika *Pearl* i *Python*. Klase i metode definirane su pomuću ključne riječi, dok se blokovi koda nalaze između ključnih riječi ili između zagrada. Izrazi i naredbe se ne razlikuju, a kraj naredbe ili izraza označava i kraj linije koda. Za razliku od nekih drugih programskih jezika kao što su *Python* i *Pearl*, sve klasne varijable napisane u programskom jeziku *Ruby* bivaju potpuno privatne u klasama i moguće im je

pristupiti preko tzv. pristupnih metoda koje je potrebno samostalno napisati, ali na veoma jednostavan način.

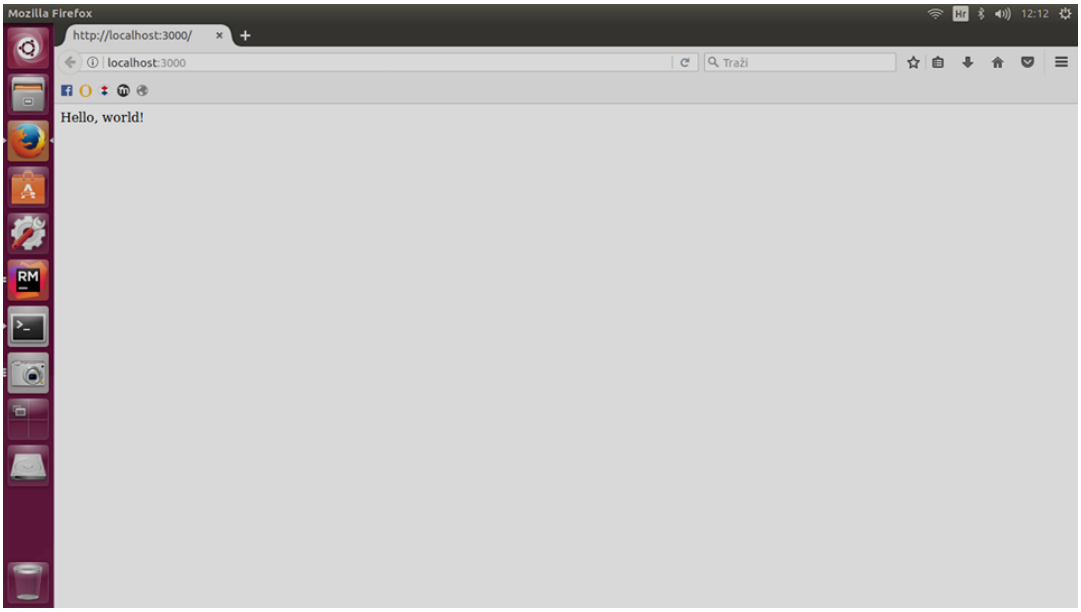
Na kraju, valja spomenuti kako postoje i mnogobrojne alternativne implementacije programskog jezika *Ruby*, a neke od njih su *JRuby* (implementacija Jave), *Rubinius* (implementacija C++), *MacRuby* (implementacija Mac OS X), itd.

```
class ApplicationController < ActionController::Base
  protect_from_forgery with: :exception

  def hello
    render html: "Hello, world!"
  end

end
```

Sl. 2.1. Primjer upravljača i klase zapisane u programskom jeziku *Ruby*

```
Rails.application.routes.draw do
  root 'application#hello'
end
```

Sl. 2.2. Primjer postavljanja tzv. rute za primjer klase s prethodne fotografije



Sl. 2.3. Primjer web-stranice napisane u programskom jeziku *Ruby*

2.2. Ruby on Rails

Ruby on Rails je mrežni programski okvir (engl. *web application framework*) koji uključuje sve što je potrebno kako bi web aplikacija s bazom podataka bila izrađena. Sve aspekte izrade web aplikacije mrežni programski okvir *Ruby on Rails* sjedinjuje po principu model-pogled-upravljač (engl. *Model-View-Controller*), te su isto tako radnja u prezentacijskim i stilskim jezicima HTML i CSS te radnja u bazi podataka i izrada web aplikacije u programskom jeziku *Rubyi* sjedinjeni. Uz sve ovo, *Ruby on Rails* sadrži i tzv. *Action Mailer* koji služi za slanje e-mail poruka u HTML i tekstnom obliku. Zbog svih ovih prednosti rad u *Ruby on Rails* mrežnom programskom okviru je pojednostavljen, a zbog načina na koji su svi aspekti rada na web aplikaciji spojeni, *Ruby on Rails* biva idealan za izradu web aplikacije s bazom recepata za kolače.

```
<header class="navbar navbar-fixed-top navbar-inverse">
  <div class="container">
    <%= link_to "railscipes", root_path, id: "logo" %>
    <nav>
      <ul class="nav navbar-nav navbar-right">
        <li><%= link_to "Home", root_path %></li>
        <% if logged_in? %>
            <li><%= link_to "Users", users_path %></li>
            <li class="dropdown">
              <a href="#" class="dropdown-toggle" data-toggle="dropdown">
                Account <b class="caret"></b>
              </a>
              <ul class="dropdown-menu">
                <li><%= link_to "Profile", current_user %></li>
                <li><%= link_to "Pantry", user_pantries_path(current_user) %></li>
                <li><%= link_to "Edit account", edit_user_path(current_user) %></li>
                <li class="divider"></li>
                <li>
                  <%= link_to "Sign out", logout_path, method: :delete %>
                </li>
              </ul>
            </li>
        <% else %>
            <li><%= link_to "Sign in", login_path %></li>
            <li><%= link_to "Sign up", signup_path %></li>
        <% end %>
      </ul>
    </nav>
  </div>
</header>
```

Sl. 2.4. Dio web-aplikacije koji sadrži *Ruby*, *HTML* i dijelove *CSS*-a, sve u sklopu *Ruby on Rails* okruženja

2.3. HTML

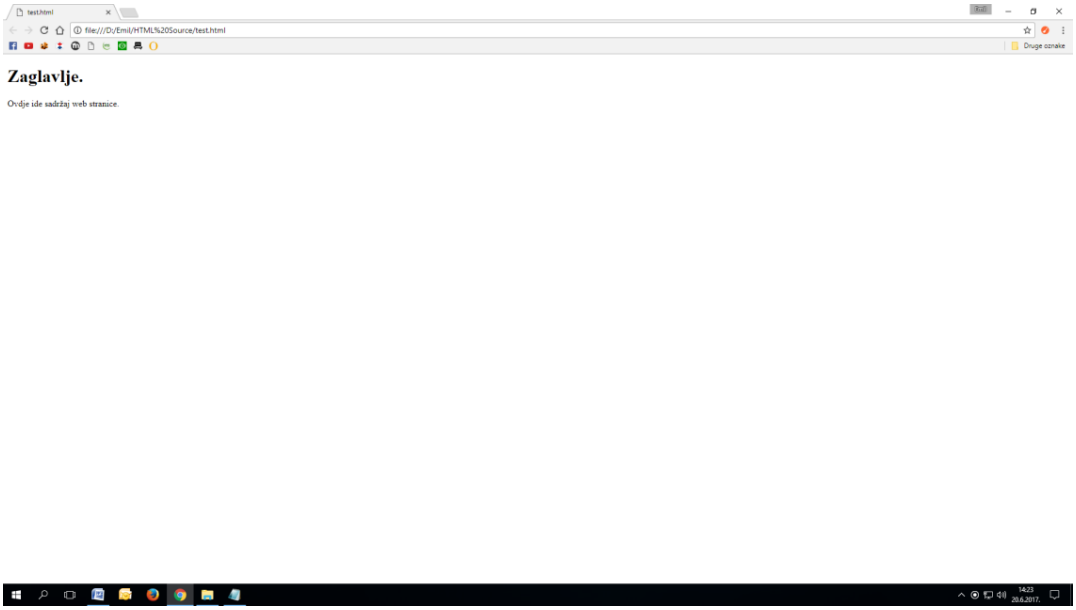
HyperText Markup Language, ili skraćeno HTML, je prezentacijski jezik i njegova glavna svrha je izrada web stranica. HTML ne predstavlja programski jezik, jer u njemu nije moguće izvršiti nikakvu zadaću, nego predstavlja tekstualni dokument koji sadrži neki tekst i hiperveze, koji zajedno tvore tzv. hipertext. Rad u ovom prezentacijskom jeziku veoma je jednostavan i vrlo lako se uči. Ovo, ali i to što je besplatan, su glavni razlozi zbog kojih je HTML veoma popularan i prihvaćen u svijetu.

Prva inačica ovoga prezentacijskoga jezika bila je izdana 1993. godine i ova inačica je bila veoma ograničena. Danas, trenutna inačica HTML-a je HTML5 i ona broji mnogo više pogodnosti nego prijašnje inačice, koje su se također vremenom sve više i više razvijale.

Struktura svakoga HTML dokumenta sastoji se od građevnih blokova, tzv. HTML elemenata od kojih se svaki dalje sastoji od para HTML oznaka (engl. *tags*) između kojih se zapisuju vrijednosti, tj. tekst koji će biti prikazan, a također mogu imati i atribute koji mogu odrediti neka svojstva tog HTML elementa. Preporučljivo je da svaki HTML dokument počinje s <!DOCTYPE> elementom koji označava inačicu standarda korištenog za izradu HTML dokumenta, zatim se stavlja <html> element koji označava početak HTML dokumenta. Svaki element se sastiji od tzv. početnog elementa (engl. *opening tag*) i završnog elementa (engl. *closing tag*), ali postoje i elementi koje nije potrebno zatvarati završnim elementom. Kako je <html> element potrebno zatvoriti završnim elementom </html>, sve što će sadržavati web stranica treba biti napisano unutar ta 2 elementa. Tako unutar njih možemo imati <head> element koji označava početak zaglavnja dokumenta, a završava </head> elementom, <body> element u koji upisujemo sadržaj HTML dokumenta i koji se zatvara </body> elementom, te brojne druge elemente, ovisno o potrebama korisnika. Na slici 2.5. moguće je vidjeti jedan primjer HTML koda, dok je na slici 2.6. prikazan sadržaj web stranice nastale kodom sa prijašnje slike.

```
<!doctype html>
<html>
<h1>
Zaglavlje.
</h1>
<body>
Ovdje ide sadržaj web stranice.
</body>
</html>
```

Sl. 2.5. Primjer HTML koda



Sl. 2.6. Web stranica nastala kodom s prijašnje slike

2.4. CSS

Cascading Style Sheets, ili skraćeno CSS, je stilski jezik koji se koristi kako bi se neki dokument napisan u nekom prezentacijskom jeziku opisao, tj. uredio. Neko vrijeme nakon što su se pojavili prezentacijski jezici, samo uređivanje i opis dokumenta bili su odrađivani u tom istom dokumentu pomoću elemenata za definiranje prezentacije dokumenta, ali je ubrzo uočena potreba za posebnim stilskim jezikom koji će osloboditi prezentacijski jezik od njegovog oblikovanja kako bi dokumenti napisani u prezentacijskim jezicima radili ono što i trebaju raditi, a to je prikazivanje sadržaja. Tako dolazi do razvoja stilskog jezika CSS čija je glavna uloga definiranje prikaza prezentacijskih jezika (HTML) na način da se uređuje njihov izgled i raspored stranica.

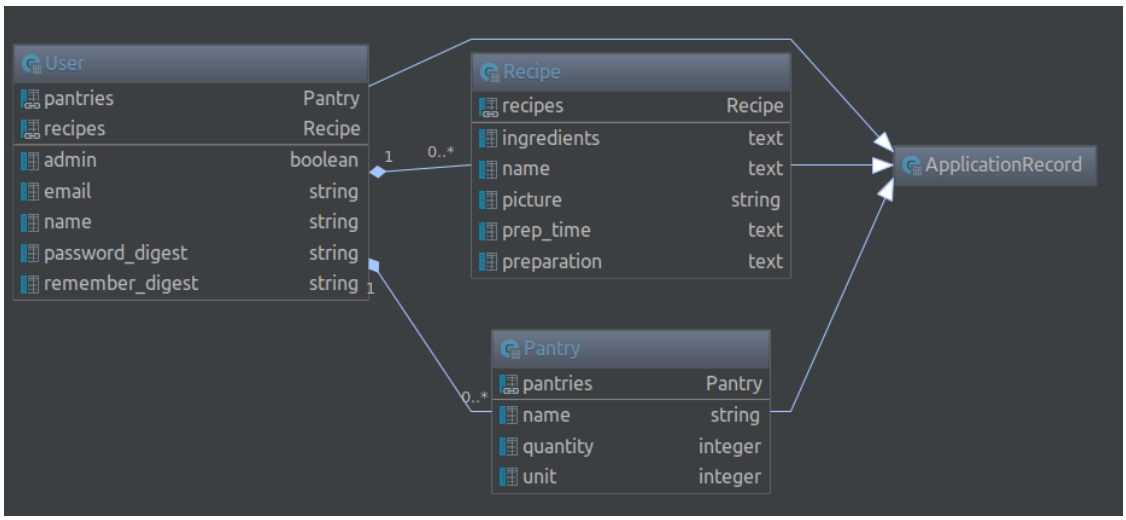
Što se tiče sintakse u stilskom jeziku CSS, postoji nekoliko pravila, od kojih se svako pravilo sastoji od tzv. *selectora*, te od deklaracijskog bloka. *Selector* predstavlja dio prezentacijskog jezika na koji primjenjujemo određeni stil, dok se deklaracijski blok nalazi unutar vitičastih zagrada i sastoji se od svojstva i vrijednosti koji su odvojeni dvotočkom. Sve deklaracije napisane u jednom deklaracijskom bloku bivaju razdvojene pomoću ; (točka zarez).

```
body {
  padding-top: 60px;
}
```

Sl. 2.7. Primjer sintakse u stilskom jeziku CSS

2.5. SQLite

SQLite je relacijska baza podataka koja je utemeljena na maloj C programskoj biblioteci. Neke od karakteristika *SQLite* baze podataka su: smještena je u jednoj datoteci na disku, podijela podataka između više računala, brže izvodi standardne operacije, ima jednostavno korisničko sučelje itd. Glavna primjena *SQLite* baze podataka je u izradi web stranica, za izradu raznih sprava i aplikacija, te kao zamjena za *ad hoc disk* datoteke.



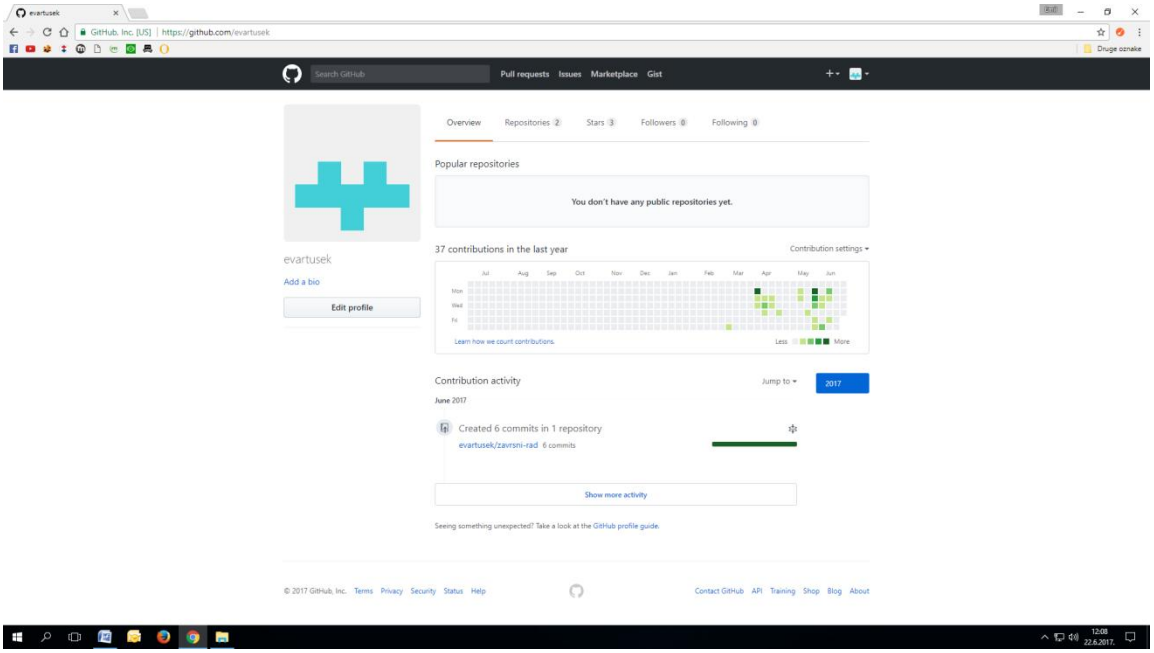
Sl. 2.8. Primjer baze napisane u *SQLite* relacijskoj bazi podataka

2.6. Bootstrap

Bootstrap je mrežni okvir koji pomaže pri izradi *front-end*-a web stranica i aplikacija. Besplatan je i otvorenoga je koda, a sadrži dizajne bazirane na HTML i CSS elementima. *Bootstrap* se sastoji od niza tzv. *stylesheets* koje su organizirane u skupinu i koje su uključene u izgled web aplikacije, ali također je moguće ubaciti ili izbaciti neke individualne komponente. Svaki *Bootstrap* element se satoji od strukture napisane u prezentacijskom jeziku HTML i deklaracija napisanih u stilskom jeziku CSS.

2.7. GitHub

GitHub je platforma koja omogućava razvoj, vođenje i izgradnju web aplikacija, te isto tako i pregledavanje koda web aplikacija, a sve to uz mogućnost da na vašem projektu rade i ostali developeri web aplikacija. Baš zbog toga, *GitHub* omogućava rad na privatnim, ali i na javnim repozitorijima i to preko istog korisničkog računa, zbog čega je često i korišten za projekte otvorenoga koda. *GitHub* trenutno ima 20 milijuna korisnika i 57 milijuna repozitorija, što ga čini najvećim spremištem koda na svijetu.



Sl. 2.9. Primjer profila na *GitHub*-u

2.8. Heroku

Heroku predstavlja kategoriju računarstva u oblaku koja se naziva platforma kao usluga (*Platform as a Service*), što znači da pruža svojim korisnicima platformu na kojoj oni tada mogu razvijati, pokretati i upravljati aplikacije, bez da se moraju baviti gradnjom i održavanjem okoline pogodne za rad njihove aplikacije. *Heroku* podržava nekoliko programskih jezika i koristi se za razvoj web aplikacija. Isto tako, *Heroku* je jedna od prvih platformi za računarstvo u oblaku, a prvi programski jezik koji je podržavaja je *Ruby*. Danas *Heroku* podržava i programske jezike *Java*, *PHP*, *Python*, *Node.js*, *Scala*, itd.

3. IZRADA WEB APLIKACIJE

Izrada web aplikacije veoma je složen problem, no uz pomoć mrežnog programskog okvira *Ruby on Rails* to postaje jednostavnije. *Ruby on Rails* spaja sve elemente potrebne za izradu ove web aplikacije. Neki od elementa koji su potrebni za izradu ovakve web aplikacije u programskom jeziku *Ruby* su modeli, upravljači, pogledi, baza podataka itd., a kako bi to bilo moguće ostvariti, potreban je i dokument pod nazivom *Gemfile*. Sve ove stvari neophodne su za rad web aplikacije na način na koji je to zamišljeno. Web aplikacija s bazom recepata za kolače baš to i treba raditi: biti baza recepata. Prema tome, cijela baza treba biti izrađena na način da to dogovara problemu s kojim smo se suočili, a svi upravljači i modeli trebaju utjecati na ispravan rad te aplikacije. Pogledi, zajedno s prezentacijskim jezikom HTML i stilskim jezikom CSS, trebaju uvelike pridonijeti izgledu, dok tzv. rute (engl. *routes*) trebaju osigurati da svaki zahtjev dođe do određenog upravljača ili akcije.
Kao što je već rečeno, svaki od ovih elemenata, tj. dijelova web aplikacije je neizostavan i zbog toga je važno objasniti na koji način i što radi svaki od njih.

3.1. Gemfile

Gemfile je dokument koji koristimo kako bi opisali o kojim to tzv. *gem*-ovima svaki *Ruby* program ovisi. Svaki gem predstavlja kolekciju koda napisanog u programskom jeziku *Ruby* koji se kasnije, za vrijeme razvoja aplikacije, mogu pozvati. Različiti *gem*-ovi čine *Gemfile* koji bi se uvijek trebao nalaziti u korijenskom direktoriju kako bi ga tzv. *bundler* (služi za instalaciju *gem*-ova) mogao lako pronaći i instalirati potrebne *gem*-ove iz tog *Gemfile*-a. Na početku svakog *Gemfile*-a potrebno je staviti izvor (engl. *source*) *gem*-ova, tj. mjesto na kojem će *Gemfile* moći pronaći svoje *gem*-ove i pozvati ih. Za to koristimo *source* metodu, te nakon nje navodimo *gem*-ove. Ovo se može vidjeti i na slici 3.1., koja prikazuje *Gemfile* web aplikacije s bazom recepata.

Sl. 3.1. Gemfile web aplikacije s bazom recepata

Na slici 3.1.1. vidimo *Gemfile* za web aplikaciju s bazom recepata koji započinje s metodom *source*, te koji svoje *gemove* poziva s https://rubygems.org web stranice. Također se može vidjeti kako je potrebno navesti inačicu programskog jezika *Ruby*, uz sve ostale *gem*-ove. Neki od *gem*-ova korišteni za izradu web aplikacije s bazom recepata su: '*faker*' (korišten za izradu lažnih korisnika kako bi se provjerio i testirao rad web aplikacije), '*will-paginate*' (korišten za raspoređivanje korisnika i recepata po stranicama), '*bootstrap-sass*' (omogućava korištenje mrežnog okvira *Bootstrap*), '*puma*' (služi kao server za razvoj web aplikacije), '*turbolinks*' (čini kretanje web aplikacijom bržim) i mnogi drugi. Moguće je vidjeti i kako su neki od *gem*-ova grupirani, a to su *gem*-ovi koji služe za razvoj baze podataka i testiranje, te za produkciju, tj. za slanje web aplikacije u pogon. Vidimo kako je bio potreban velik broj *gem*-ova kako bi web aplikacija radila ispravno. Zbog toga je važno napomenuti kako ovi *gem*-ovi nisu stavljeni u *Gemfile* na početku rada web aplikacije, već su nadodavani kako je koji bio potreban i kako je razvoj web aplikacije sve dalje napredovao. Naravno, neki od *gem*-ova bili su potrebni od samoga početka, te su tada oni i bili nadodani odmah na početku rada. Neovisno o tome, možemo zaključiti da je *Gemfile* i njegovi gem-ovi veoma bitan aspekt ove, ali i svake druge, web aplikacije napisane u programskom jeziku *Ruby*.

3.2. Modeli

Modeli su klase napisane u programskom jeziku *Ruby*, a služe kao posrednici između web aplikacije i baze podataka, pohranjuju i potvrđuju podatke, te izvode svu programsku logiku koja je potrebna za rad web aplikacije. Model predstavlja M u MVC (*Model-View-Controller*) obrascu softverske arhitekture po kojoj je programski jezik *Ruby* poznat. Model je središnji dio tog obrasca jer prikazuje ponašanje same web aplikacije i direktno upravlja svim podacima

unesenim u bazu podatka. Modeli su neovisni o korisničkom sučelju o kojem se brinu preostala dva slova iz MVC: V (*Views*, tj. pogledi) i C (*Controllers*, tj. upravljači).

Za izradu web aplikacije s bazom recepata koristili smo 3 modela. Prvo je bio potreban model korisnika (model *user.rb*) koji će koristiti web aplikaciju i preko kojega će se odrađivati glavnina rada ove web aplikacije, a isto tako i preko kojega ćemo povezati ostala dva modela. Drugi model koji je bio potreban je model recepta (model *recipe.rb*) koji je povezan sa svakim korisnikom i predstavlja svaki recept koji je korisnik napisao i objavio. Posljednji model potreban za izradu ove web aplikacije je model smočnice (model *pantry.rb*), tj. model svakog sastojka koji će se spremati u smočnicu svakog korisnika i koji će isto tako biti povezani sa svakim korisnikom. Na slikama 3.2., 3.3., i 3.4. se može vidjeti kako izgledaju ti modeli.

```ruby
class User < ApplicationRecord
  has_many :recipes, dependent: :destroy
  has_many :pantries, dependent: :destroy

  validates :name, presence: true, length: { maximum: 50 }
  VALID_EMAIL_REGEX = /\A[\w+\-.]+@[a-z\d\-.]+\.[a-z]+\z/i
  validates :email, presence: true, length: { maximum: 100 },
            format: { with: VALID_EMAIL_REGEX },
            uniqueness: { case_sensitive: false }
  has_secure_password
  validates :password, presence: true, length: { minimum: 6 }, allow_nil: true

  class << self
    # Returns the hash digest of the given string.
    def digest(string)
      cost = ActiveModel::SecurePassword.min_cost ? BCrypt::Engine::MIN_COST :
          BCrypt::Engine.cost
      BCrypt::Password.create(string, cost: cost)
    end

  # Returns a random token.
    def new_token
      SecureRandom.urlsafe_base64
    end
  end
```

```ruby
  # Remembers a user in the database for use in persistent sessions.
  def remember
    self.remember_token = User.new_token
    update_attribute(:remember_digest, User.digest(remember_token))
  end

  # Forgets a user.
  def forget
    update_attribute(:remember_digest, nil)
  end

  # Defines a proto-feed.
  def feed
    Recipe.where('user_id = ?', id)
  end

end
```

Sl. 3.2. Model *user.rb*

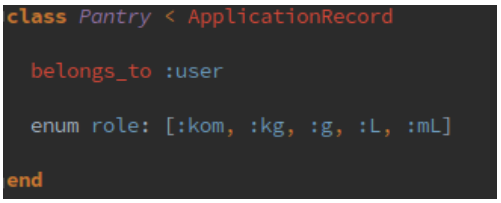
Model *user.rb* predstavlja model korisnika sa svim funkcijama koje su bitne da bi taj model radio na način na koji je to potrebno. Ostala dva modela su povezana s ovim modelom i to na način na koji svaki korisnik može imati jedan ili više recepata i jedan ili više sastojaka u svojoj smočnici. Model *user.rb* također i potvrđuje postojanje imena, lozinke i *e-maila* svakog od

korisnika, te isto tako i propisuje očekivani izgled svake *e-mail* adrese i duljinu lozinke koja mora biti dulja od 6 znakova. Unutar tzv. *self* klase, model *user.rb* ima brojne metode, od kojih su neke metoda *digest* (metoda koja uzima lozinku korisnika i pretvara ju u skup slova i brojki i služi za prikrivanje lozinke u bazi podataka), metoda *remember* (metoda koja pamti korisnika), metoda *forget* (metoda koja zaboravlja korisnika), te metoda *feed* (metoda koja omogućuje prikaz svih recepata korisnika).

```ruby
class Recipe < ApplicationRecord

  belongs_to :user
  default_scope -> { order(created_at: :desc) }
  mount_uploader :picture, PictureUploader
  validates :user_id, presence: true
  validates :name, presence: true, length: { maximum: 50 }
  validates :ingredients, presence: true, length: { maximum: 1000 }
  validates :prep_time, presence: true, length: { maximum: 5 }
  validates :preparation, presence: true, length: { maximum: 10000 }
  validate  :picture_size

  private

  # Validates the size of an uploaded picture.
  def picture_size
    if picture.size > 5.megabytes
      errors.add(:picture, "should be less than 5MB")
    end
  end

  def self.search(search)
    where("name LIKE ? OR ingredients LIKE ? OR preparation LIKE ?",
          "%#{search}%",
          "%#{search}%",
          "%#{search}%")
  end

end
```

Sl. 3.3. Model *recipe.rb*

Model *recipe.rb* predstavlja model recepta sa svim svojim funkcijama i metodama. Ovaj model pripada modelu *user.rb* koji ga poziva, a svi recepti su poredani od najnovijeg prema najstarijem. Također se u ovom modelu nalazi *PictureUploader*, koji omogućuje objavljivanje slike uz napisani recept. Model *recipe.rb* potvrđuje postojanje ID-a korisnika koji je napisao taj recept, te postojanje imena, liste sastojaka, vremena pripreme i načina pripreme pojedinoga recepta. Svim ovim elementima određena je i maskimalna veličina znakova koju je moguće zapisati. Također, imamo i dvije metode, od kojih za prvu metodu, metodu *picture_size* (metoda koja određuje maksimalnu veličinu slike, 5 MB), imamo i potvrdu. Druga metoda je metoda *self.search(search)*, a ona predstavlja metodu pomoću koje recepti pretražuju sami sebe nakon što korisnik u tražilicu recepata upiše riječ koja ga zanima. Nakon upisivanja željene riječi, ova metoda prolazi kroz sve nazive, sastojke i načine pripreme svakog recepta i prikazuje sve one recepte koji sadrže tu željenu riječ.



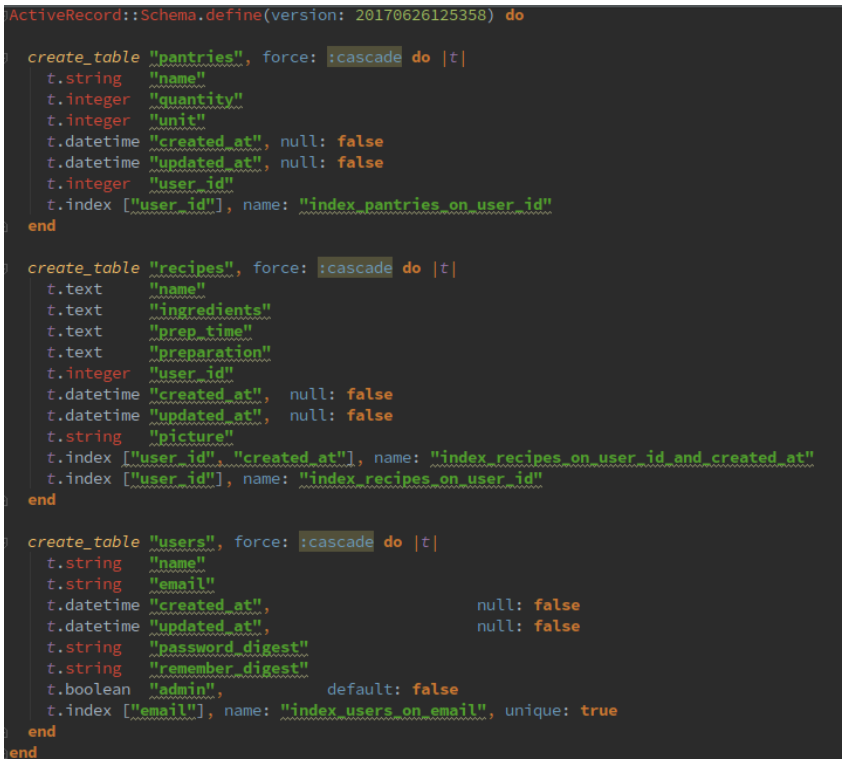
Sl. 3.4. Model *pantry.rb*

Model *pantry.rb* predstavlja model smočnice, tj. model pojedinog sastojka koji će u tu smočnicu biti spremljen. Ovaj model također pripada modelu *user.rb*, što znači da će svaki korisnik upisivati svoje sastojke, na isti način kao što će to raditi i za recepte. U ovom modelu također su deklarirane i pojedine tzv. uloge (engl. *roles*) koje će se moći izabrati pri upisivanju svakog sastojka, a radi se o ulogama koje će označavati količinu određenog sastojka (komad, kilogram, gram, litra i mililitra).

3.3. Baza podataka

Baza podataka predstavlja organiziranu skupinu podataka zapisanih u tablice kojima računalni program ili web aplikacija mogu pristupiti kada je to potrebno. Svaka tablica sastoji se od stupaca koji imaju nekakav naziv i u čijim su retcima zapisane vrijednosti koje se odnose na taj stupac. Uz tablice, baza podataka se sastoji i od shema, upita, izvješća, pogleda i ostalih objekata.

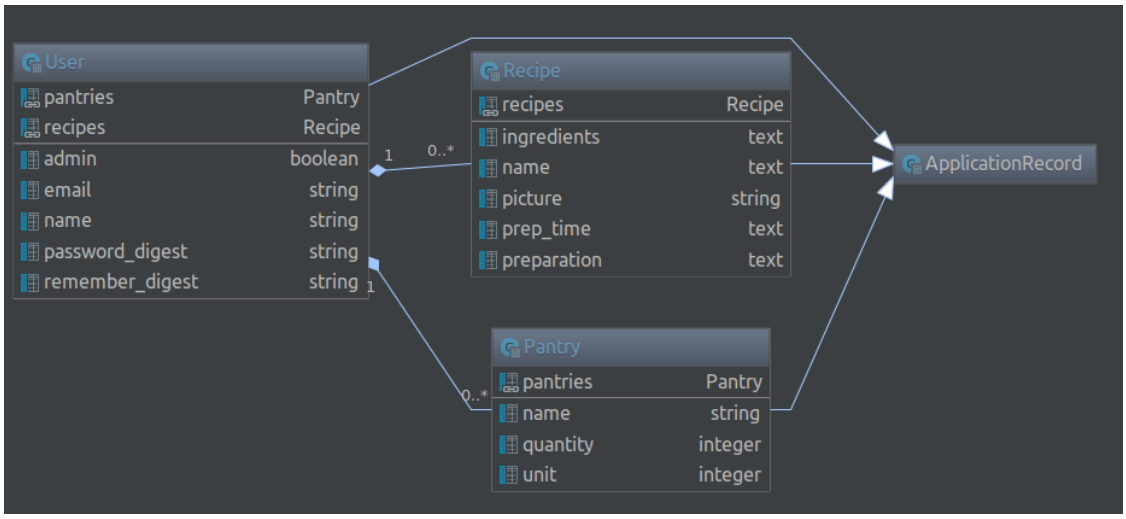
Za izradu baze podataka za web aplikaciju s bazom recepata bilo je potrebno onoliko tablica koliko imamo i modela, što znači da imamo 3 tablice. Prva tablica je tablica s korisnicima, druga tablica je tablica s receptima, a treća tablica sadrži sastojke koji se upisuju u smočnicu svakoga korisnika. Na slici 3.5. se može vidjeti što svaka od tih tablica sadrži.



```
ActiveRecord::Schema.define(version: 20170626125358) do

  create_table "pantries", force: :cascade do |t|
    t.string   "name"
    t.integer  "quantity"
    t.integer  "unit"
    t.datetime "created_at", null: false
    t.datetime "updated_at", null: false
    t.integer  "user_id"
    t.index ["user_id"], name: "index_pantries_on_user_id"
  end

  create_table "recipes", force: :cascade do |t|
    t.text     "name"
    t.text     "ingredients"
    t.text     "prep_time"
    t.text     "preparation"
    t.integer  "user_id"
    t.datetime "created_at",  null: false
    t.datetime "updated_at",  null: false
    t.string   "picture"
    t.index ["user_id", "created_at"], name: "index_recipes_on_user_id_and_created_at"
    t.index ["user_id"], name: "index_recipes_on_user_id"
  end

  create_table "users", force: :cascade do |t|
    t.string   "name"
    t.string   "email"
    t.datetime "created_at",                    null: false
    t.datetime "updated_at",                    null: false
    t.string   "password_digest"
    t.string   "remember_digest"
    t.boolean  "admin",           default: false
    t.index ["email"], name: "index_users_on_email", unique: true
  end
end
```

Sl. 3.5. Dokument schema.rb sa sve 3 tablice i njihovim elementima

Na slici se mogu vidjeti tablice koje zapravo predstaljvalju modele koje smo ranije objasnili, te tako imamo tablicu *pantries* (model *pantry.rb*), tablicu *recipes* (model *recipe.rb*) i tablicu *users* (model *user.rb*). Svaka od ovih tablica ima elemente koji ju čine, što se također može vidjeti sa slike. Tako tablica *pantries* ima polja: *name* tipa *string* (naziv sastojka koji ulazi u smočnicu), *quantity* tipa *integer* (količina nekog sastojka), *unit* tipa *integer* (odabir uloge za količinu određenog sastojka), te *user_id* tipa *integer* (označava kojem korisniku sastojak pripada). Tablica *recipes* ima polja: *name*, *ingredients*, *prep_time*, i *preparation* tipa *text* (ime, sastojci, vrijeme pripreme i način pripreme), *user_id* tipa *integer* (označava kojem korisniku recept pripada), te *picture* tipa *string* (slika). Tablica *user* sadrži polja: *name*, *email*, *password_digest* i *remember_digest* tipa *string* (ime korisnika, email korisnika, *digest* lozinke i *digest* za pamćenje korisnika), te *admin* tipa *boolean* (označava da li je korisnik admin). Sve 3 tablice sadrže i polja *created_at* i *updated_at* tipa *datetime* koji govore kada su korisnik, recept ili sastojak umetniti, te kada su posljednji puta mijenjani u bazi podataka. Ova dva polja tablice *recipes* i *pantries* nasljeđuju od tablice *users*, što se može vidjeti i na slici 3.6., koja prikazuje odnos između ove 3 tablice, te izgled baze podatak web aplikacije s bazom recepata.



Sl. 3.6. Prikaz odnosa unutar tablica baze podataka web aplikacije s bazom recepata

3.4. Upravljači

Upravljači predstavljaju C u MVC (*Model-View-Controller*) obrascu softverske arhitekrute i koriste se za upravljanje korisničkim zahtjevima. Nakon što se odredi koji upravljač treba pozvati, pozvani upravitelj obavlja ono što je od njega traženo i za što je zadužen. Upravljači šalju naloge modelima pomoću kojih ažuriraju njihovo stanje, a također služe i kako bi se poslale naredbe pogledima (*Views*) kako bi se promijenio prikaz modela.

Određeni korisnički zahtjevi obrađeni su uz pomoć nekoliko upravljača koji su potrebni za izradu web aplikacije. Ti upravljači su: *application_controller.rb*, *pantries_controller.rb*, *recipes_controller.rb*, *sessions_controller.rb*, *static_pages_controller.rb* i *users_controller.rb*.

Upravljač *application_controller.rb* uz *protect_from_forgery* metodu, u svom privatnom dijelu definira i metodu *logged_in_user* koja provjerava da li je korisnik prijavljen, te ako nije, preusmjerava korisnika na dio web aplikacije za prijavu.

```
class ApplicationController < ActionController::Base
  protect_from_forgery with: :exception
  include SessionsHelper

  private

  # Confirms a logged-in user.
  def logged_in_user
    unless logged_in?
      store_location
      flash[:danger] = "Please log in."
      redirect_to login_url
    end
  end
end
```

Sl. 3.7. Prikaz upravljača *application_controller.rb*

Model *pantry.rb*, tj. sastojci koji se upisuju u smočnicu korisnika, upravljani su *pantries_controller.rb upravljačem.* Ovaj upravljač sadrži nekoliko metoda. Metoda *index* ispisuje sve sastojke određenog korisnika, metoda *update* služi kako bi se promijenila količina određenog sastojka, metoda *create* stvara novi sastojak prilikom njegovog upisivanja u smočnicu tako da svakom određenom sastojku pridruži određenu mjernu jedinicu, metoda *destroy* briše određeni sastojak iz smočnice, a metoda *update_params* daje dopuštenje metodi *update* da koristi parametar količina (engl. *quantity*) za njene akcije.

```ruby
class PantriesController < ApplicationController
  def index
    if current_user == User.find(params[:user_id])
      @pantries = User.find(params[:user_id]).pantries.all
    else
      redirect_to user_pantries_path(current_user), :notice => "You can't view other users pantries."
    end
    @pantry = Pantry.new
  end

  def update
    @pantry = Pantry.find(params[:id])
    if params[:quantity]
      if @pantry.update_attributes(update_params)
        redirect_to user_pantries_path(current_user), :notice => "Quantity updated."
      else
        redirect_to user_pantries_path(current_user), :alert => "Unable to update quantity."
      end
    end
  end
```

```ruby
  def create
    @unit = params[:pantry][:unit]
    @unit_number = 0

    case @unit
      when 'kom'
        @unit_number = 0
      when 'kg'
        @unit_number = 1
      when 'g'
        @unit_number = 2
      when 'L'
        @unit_number = 3
      when 'mL'
        @unit_number = 4
    end

    @pantry = current_user.pantries.build(:name => params[:name], :quantity => params[:quantity],
                                          :unit => @unit_number)
    if @pantry.save
      redirect_to user_pantries_path(current_user), :notice => "Ingredient added."
    else
      redirect_to user_pantries_path(current_user), :alert => "Unable to add ingredient."
    end
  end
```

```ruby
  def destroy
    Pantry.find(params[:id]).destroy
    redirect_to user_pantries_path(current_user), :notice => "Ingredient deleted"
  end

  private

  def update_params
    params.permit(:quantity)
  end

end
```

Sl. 3.8. Prikaz upravljača *pantries_controller.rb*

Model *recipe.rb* je upravljan *recipes_controller.rb* upravljačem. Sve metode ovog upravljača izvršavaju se tek nakon što se provjeri da li je korisnik prijavljen i da li se radi o ispravnom korisniku. Metoda *create* služi kako bi se stvorio novi recept, metoda *index* ispisuje sve recepte

od najnovijeg prema najstarijem, metoda *destroy* briše napisani recept, a u privatnom dijelu ovog upravljača imamo metode *recipe_params* i *correct_user* koje označavaju parametre recepta i ispravnog korisnika.

Upravljač *sessions_controller.rb* upravlja tzv. sesijama, tj. poziva se metodom *create* koja stvara novu sesiju koja započinje prijavom koirnsika na web aplikaciju, a završava odjavom korisnika, tj. pozivanjem metode *destroy*.

```
class SessionsController < ApplicationController

  def new
  end

  def create
    user = User.find_by(email: params[:session][:email].downcase)
    if user && user.authenticate(params[:session][:password])
        log_in user
        params[:session][:remember_me] == '1' ? remember(user) : forget(user)
        redirect_back_or user
    else
      flash.now[:danger] = 'Invalid email/password combination'
      render 'new'
    end
  end

  def destroy
    log_out if logged_in?
    redirect_to root_url
  end

end
```

Sl. 3.9. Prikaz upravljača *sessions_controller.rb*

Radom statičkih stranica upravlja *static_pages_controller.rb* upravljač. Web aplikacija s bazom recepata ima 4 statičke stranice: *home*, *help*, *about* i *contact*. *Home* statička stranica predstavlja početnu stranicu web aplikacije na kojoj se prikazuju svi recepti, dok ostale 3 statičke stranice prikazuju samo sadržaj napisan u prezentacijskom jeziku HTML.

Na kraju imamo upravljač *users_controller.rb* koji upravlja svim korisnicima koji se nalaze u bazi podataka web aplikacije. Kao i kod *recipes_controller.rb* upravljača, metode ovog upravljača obavljaju se tek nakon što se provjeri da li je korisnik logiran, ispravan i da li je admin. Upravljač *user_controller.rb* sadrži brojne metode. Metoda *show* pokazuje sve recepte svakog korisnika, metode *new* i *create* stvaraju novog korisnika prilikom registracije, metode *edit* i *update* omogućuje promjenu atributa korisnika (ime, e-mail i lozinka), metoda *index* prikazuje sve korisnike, a metoda *destroy* briše određenog korisnika (obrisati drugog korisnika može samo admin). U privatnom dijelu ovog upravljača imamo metodu koja određuje parametre korisika, metodu koja potvrđuje ispravnog korisnika i metodu koja potvrđuje admina.

```ruby
class UsersController < ApplicationController
 before_action :logged_in_user, only: [:index, :edit, :update, :destroy]
 before_action :correct_user,   only: [:edit, :update]
 before_action :admin_user,     only: :destroy


 def show
   @user = User.find(params[:id])
   @recipes = @user.recipes.paginate(page: params[:page])
 end

 def new
   @user = User.new
 end

 def create
   @user = User.new(user_params)
   if @user.save
     log_in @user
     flash[:success] = "Welcome to the Railscipes!"
     redirect_to @user
   else
     render 'new'
   end
 end

 def edit
   @user = User.find(params[:id])
 end
```

```ruby
 def update
   @user = User.find(params[:id])
   if @user.update_attributes(user_params)
     flash[:success] = "Account updated"
     redirect_to @user
   else
     render 'edit'
   end
 end

 def index
   @users = User.paginate(page: params[:page])
 end

 def destroy
   User.find(params[:id]).destroy
   flash[:success] = "User deleted"
   redirect_to users_url
 end
```

```ruby
 private

 def user_params
   params.require(:user).permit(:name, :email, :password,
                                :password_confirmation)
 end

 # Confirms the correct user.
 def correct_user
   @user = User.find(params[:id])
   redirect_to(root_url) unless current_user?(@user)
 end

 # Confirms an admin user.
 def admin_user
   redirect_to(root_url) unless current_user.admin?
 end

end
```

Sl. 3.10. Prikaz upravljača *users_controller.rb*

3.5. Pogledi

Pogled predstavlja V u MVC (*Model-View-Controller*) obrascu softverske arhitekture. Svaki podatak koji smo prethodno modelirali prikazujemo pomoću pogleda koji, kako bi stvorio prikaz modela korisniku, od modela zahtjeva informacije koje mu on tada prosljeđuje. Također, svaki pogled je povezan i s određenim upravljačem za kojega ima predložak kako bi on trebao izgledati. Ti predlošci su napisani pomoću ugrađenog programskog jezika *Ruby* i dijelova koji su napisani u prezentacijskom jeziku HTML.

U web aplikaciji s bazom recepata imamo 5 pogleda za 5 upravljača i uz njih imamo i poglede koji se odnose na opći izgled web aplikacije i na izgled nekih njenih svojstava.

Prvo imamo poglede kojima određujemo opći izgled, tj. raspored (*layout*) web aplikacije. Ovaj izgled sastoji se od dijela koji određuje izgled zaglavlja *(_header.html.erb*) i podnožja (*_footer.html.erb*) web aplikacije. Također, imamo i dio pogleda *application.html.erb* koji određuje nekakav opći oblik svake stranice web aplikacije.

```
<header class="navbar navbar-fixed-top navbar-inverse">
  <div class="container">
    <%= link_to "railscipes", root_path, id: "logo" %>
    <nav>
      <ul class="nav navbar-nav navbar-right">
        <li><%= link_to "Home", root_path %></li>
        <% if logged_in? %>
            <li><%= link_to "Users", users_path %></li>
            <li class="dropdown">
              <a href="#" class="dropdown-toggle" data-toggle="dropdown">
                Account <b class="caret"></b>
              </a>
              <ul class="dropdown-menu">
                <li><%= link_to "Profile", current_user %></li>
                <li><%= link_to "Pantry", pantries_path(current_user) %></li>
                <li><%= link_to "Edit account", edit_user_path(current_user) %></li>
                <li class="divider"></li>
                <li>
                  <%= link_to "Sign out", logout_path, method: :delete %>
                </li>
              </ul>
            </li>
        <% else %>
            <li><%= link_to "Sign in", login_path %></li>
            <li><%= link_to "Sign up", signup_path %></li>
        <% end %>
      </ul>
    </nav>
  </div>
</header>
```

Sl. 3.11. Prikaz pogleda zaglavlja *(_header.html.erb*) web aplikacije s bazom recepata

Pogledi *pantries* označava poglede vezane uz *pantries* upravljač, a određuju izgled smočnice. Ovaj pogled sastoji se od nekoliko dijelova. Dio *_pantry.html.erb* određuje izgled sastojka upisanog u smočnicu, te pomoću forme omogućuje da se količina nekog sastojka promijeni. Dio *destroy.html.erb* omogućuje brisanje određenog sastojka pritiskom na dugme, dok dio

update.html.erb potvrđuje promjenu količine određenog sastojka. Na kraju imamo i dio *index.html.erb* koji ispisuje sve sastojke koji se nalaze u smočnici, i to jedan ispod drugoga te isto tako sadrži i formu za upisivanje novoga sastojka.

Pogled *recipes* označava poglede vezane uz *recipes* upravljač, a određuju izgled svakog recepta. Sastoje se od dva dijela: *_recipe.html.erb* i *index.html.erb*. Prvi dio određuje kako će izgledati recept kojega je neki korisnik objavio, te isto tako omogućuje i brisanje toga recepta. Drugi dio ovog pogleda oblikuje izgled i ispisuje recepte koji sadrže određenu riječ koju je korisnik upisao, ako recepti koji sadrže tu riječ uopće postoje.

Pogledi *sessions* označavaju poglede vezane uz *sessions* upravljač, a sastoje se samo od jednog dijela, *new.html.erb*. Ovaj dio predstavlja prikaz dijela web aplikacije na kojem korisnik vrši prijavu na web aplikaciju.

```
<% provide(:title, "Sign in") %>
<h1>Sign in</h1>

<div class="row">
  <div class="col-md-6 col-md-offset-3">
    <%= form_for(:session, url: login_path) do |f| %>

      <%= f.label :email %>
      <%= f.email_field :email, class: 'form-control' %>

      <%= f.label :password %>
      <%= f.password_field :password, class: 'form-control' %>

      <%= f.label :remember_me, class: "checkbox inline" do %>
        <%= f.check_box :remember_me %>
        <span>Remember me on this computer</span>
      <% end %>

      <%= f.submit "Sign in", class: "btn btn-primary" %>
    <% end %>

    <p>New user? <%= link_to "Sign up now!", signup_path %></p>
  </div>
</div>
```

Sl. 3.12. Prikaz pogleda *new.html.erb* za prijavu novog korisnika

Pogledi koji utječu na izgled nekih svojstava web aplikacije predstavljeni su pogledom *shared*. Sastoje se od nekoliko dijelova. Dio *_error_messages.html.erb* služi kako bi se korisniku prikazala poruka u slučaju neke pogreške, *_feed.html.erb* služi za prikaz recepata svih korisnika, *_recipe_form.html.erb* prikazuje sučelje za upisivanje novoga recepta u bazu podataka, a *_user_info.html.erb* omogućuje pregledavanje profila prijavljenoga korisnika, te koliko taj korisnik ima napisanih recepata.

Pogledi *static_pages* predstavljaju poglede kojima uređujemo statičke stranice web aplikacije. Ovi pogledi određuju što će se na svakoj od ovih stranica prikazati. Tako pogled statičke stranice

about.html.erb govori nešto o samoj web aplikaciji, pogled *contact.html.erb* prikazuje kako kontaktirati tvorca ove web aplikacije, pogled *help.html.erb* pruža pomoć pri navigiranju web aplikacijom, a pogled *home.html.erb* govori što se sve prikazuje na početnoj stranici web aplikacije.

Pogledi *users* predstavljaju poglede vezane uz *users* upravljač, a oni utječu na izgled svojstava koja se odnose na korisnike i rad nad njihovim podacima. Tako imamo pogled *_user.html.erb* koji omogućuje brisanje drugih korisnika, ali samo ako je korisnik admin. Pogled *edit.html.erb* prikazuje izgled dijela web stranice u kojem svaki korisnik može promijeniti svoje podatke (ime, e-mail, lozinka) i spremiti ih. Pogled *index.html.erb* prikazuje sve korisnike i svrstava ih po stranicama, a pogled *show.html.erb* prikazuje korisnikov profil i sve njegove recepte. Na kraju imamo pogled *new.html.erb* koji određuje prikaz dijela web aplikacije na kojem se novi korisnici mogu registrirati.

```
<% provide(:title, 'Sign up') %>
<h1>Sign up</h1>

<div class="row">
  <div class="col-md-6 col-md-offset-3">
    <%= form_for(@user) do |f| %>
        <%= render 'shared/error_messages', object: f.object %>

        <%= f.label :name %>
        <%= f.text_field :name, class: 'form-control' %>

        <%= f.label :email %>
        <%= f.email_field :email, class: 'form-control' %>

        <%= f.label :password %>
        <%= f.password_field :password, class: 'form-control' %>

        <%= f.label :password_confirmation, "Confirmation" %>
        <%= f.password_field :password_confirmation, class: 'form-control' %>

        <%= f.submit "Create my account", class: "btn btn-primary" %>
    <% end %>
  </div>
</div>
```

Sl. 3.13. Prikaz pogleda *new.html.erb* za registraciju novog korisnika

3.6. Routes, helpers, stylesheets

Routes predstavlja način na koji web aplikacija prepoznaje internet adrese i povezuje ih s određenim upravljačem, a također i olakšavaju rad u pogledima generirajući puteve i internetske adrese.

U ovoj web aplikaciji imamo mnogo puteva, tj. ruta koje se nalaze u njenom *routes.rb* dijelu. Neke od njih služe kako bi se došlo do određene statičke stranice, neke služe kako bi se došlo do

određenih svojstava iz smočnice korisnika, neke kako bi se došlo do recepata, a neke kako bi se stvorila nova sesija prijave korisnika za rad u web aplikaciji.

```
Rails.application.routes.draw do
  get 'pantries/index'
  get 'pantries/update'
  get 'pantries/destroy'
  get 'password_resets/new'
  get 'password_resets/edit'
  get 'sessions/new'
  get 'users/new'

  root 'static_pages#home'
  get  '/help',    to: 'static_pages#help'
  get  '/about',   to: 'static_pages#about'
  get  '/contact', to: 'static_pages#contact'
  get  '/pantries',  to: 'pantries#index'
  post '/pantry', to: 'pantries#update'
  get '/recipes', to: 'recipes#index'
  get  '/signup',  to: 'users#new'
  get   '/login',   to: 'sessions#new'
  post  '/login',   to: 'sessions#create'
  delete '/logout',  to: 'sessions#destroy'

  resources :users do
    resources :pantries
  end

  resources :recipes,          only: [:create, :destroy]


end
```

Sl. 3.14. Prikaz *routes.rb* dijela web aplikacije

Helpers, tj. pomoćnici ili pomagači, najčešće se koriste kada se određeni dio koda treba prikazati samo u pogledima i kada je u njima potrebno napraviti neki složeni kod napisan u prezentacijskom jeziku HTML, ili kada se radi s podacima koji nisu povezani s bazom podataka. Koliko imamo pogleda, toliko ćemo imati i pomagača, ali nije ih potrebno sve koristiti, već samo one za koje mislimo da su potrebni.

U web aplikaciji s bazom recepata imamo 2 pomagača: *sessions_helper.rb* i *users_helper.rb*. Pomagač *sessions_helper.rb* sadrži sve što je potrebno kako bi se korisnik mogao uspješno prijaviti i odjaviti na web aplikaciju. Neke od metoda su *log_in* koja prijavljuje određenog korisnika, *remember* koja pamti korisnika za vrijeme njegovog rada u web aplikaciji, *logged_in?* koja provjerava da li je korisnik prijavljen, *log_out* koja odjavljuje korisnika i prekida trenutnu sesiju itd. Pomagač *users_helper.rb* služi samo kako bi korisnik mogao postaviti sliku svoga profila preko servisa za postavljanje *avatara* koji se zove *Gravatar*.

```ruby
# Returns true if the given user is the current user.
def current_user?(user)
  user == current_user
end

# Returns the current logged-in user (if any).
def current_user
  if (user_id = session[:user_id])
    @current_user ||= User.find_by(id: user_id)
  elsif (user_id = cookies.signed[:user_id])
    user = User.find_by(id: user_id)
      log_in user
      @current_user = user
  end
end
```

Sl. 3.15. Primjer nekih metoda iz *sessions_helper.rb* pomagača

Stylesheets predstavlja dio web aplikacije koji uz pomoć stilskog jezika CSS uređuje sam izgled te web aplikacije. Ovaj dio predstavlja kod koji je u potpunosti napisan u stilskom jeziku CSS. Kako bi uredili izgled ove web aplikacije, koristili smo dokumet *custom.scss* u koji je zapisan kod uz kojega web aplikacije izgleda onako kako smo to i zamislili.

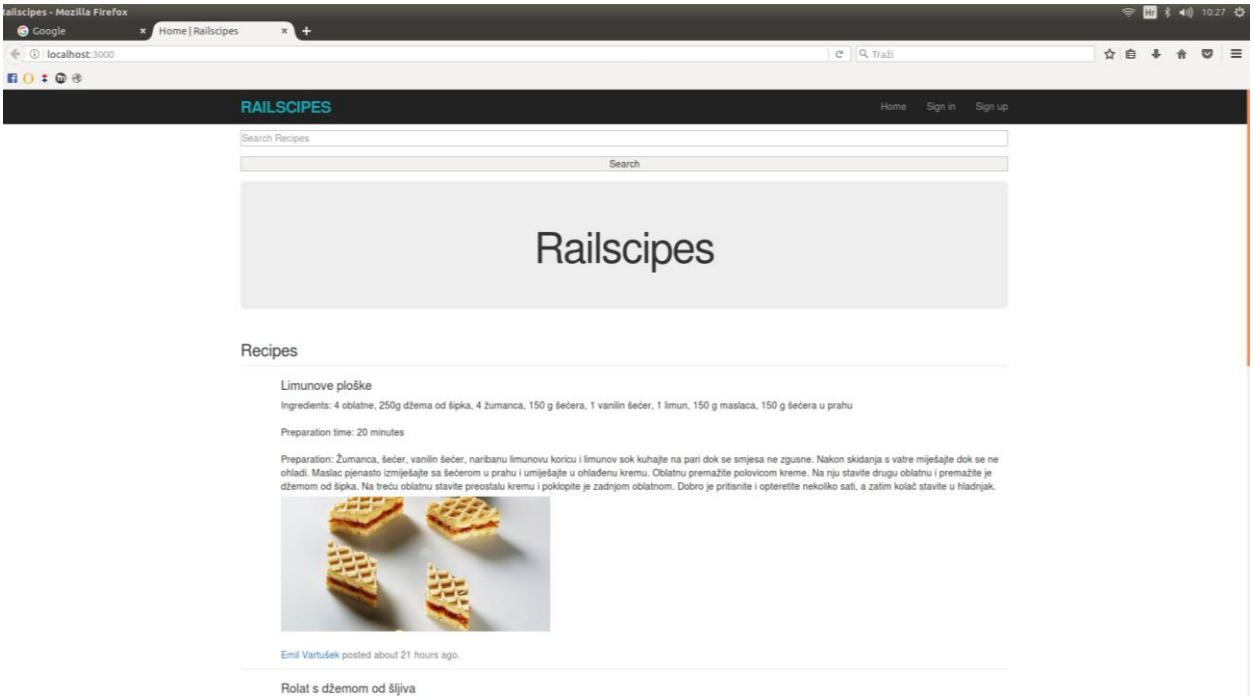
```css
#logo {
  float: left;
  margin-right: 10px;
  font-size: 1.7em;
  color: #19a8b4;
  text-transform: uppercase;
  letter-spacing: -1px;
  padding-top: 9px;
  font-weight: bold;
}

#logo:hover {
  color: #fff;
  text-decoration: none;
}
```

Sl. 3.16. Dio *custom.scss* dokumenta koji određuje izgled logotipa web aplikacije

4. TESTIRANJE WEB APLIKACIJE

Kako bi rad ove web aplikacije bio testiran, biti će prikazan njen izgled, registracija i prijava korisnika, prikaz profila korisnika prije i nakon pisanja recepata, pretraživanje recepata, prikaz svih ostalih korisnika te prikaz smočnice korisnika, prikaz dijela web aplikacije na kojemu korisnik može promijeniti svoje podatke i na kraju odjava korisnika.

Prvo će biti prikazan izgled početne stranice web aplikacije na kojoj se prikazuju svi recepti koji se nalaze zapisani u bazi podataka.

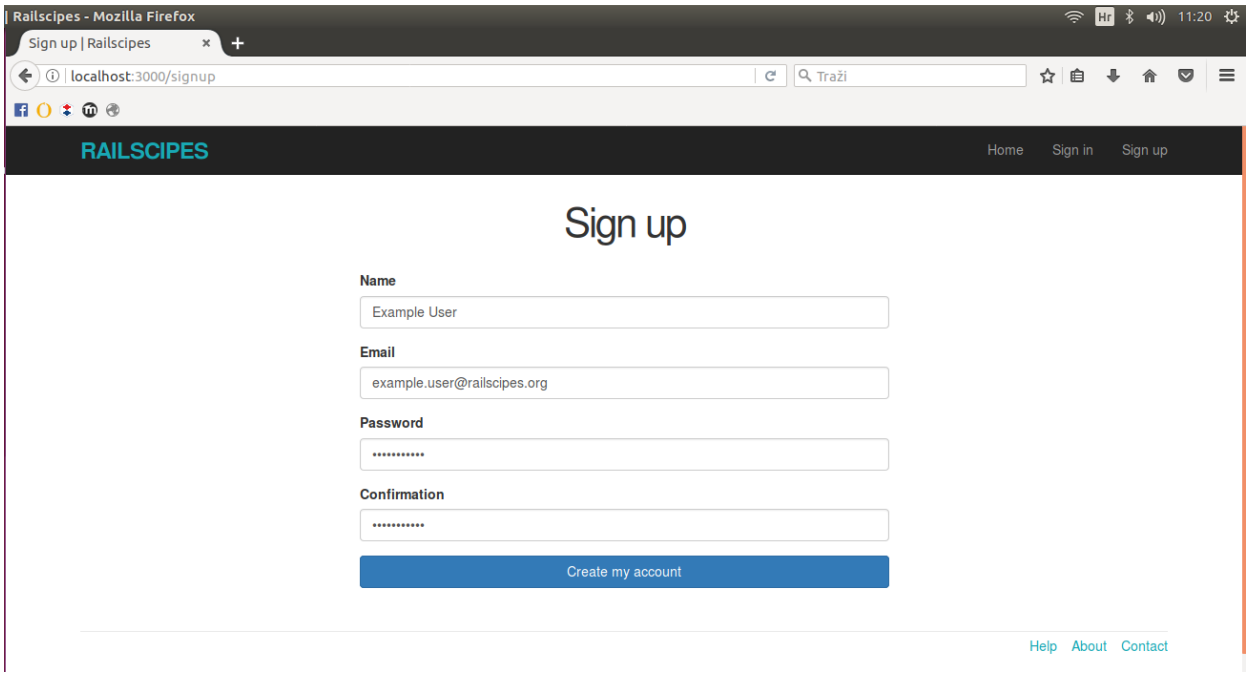


Sl. 4.1. Prikaz početne stranice web aplikacije

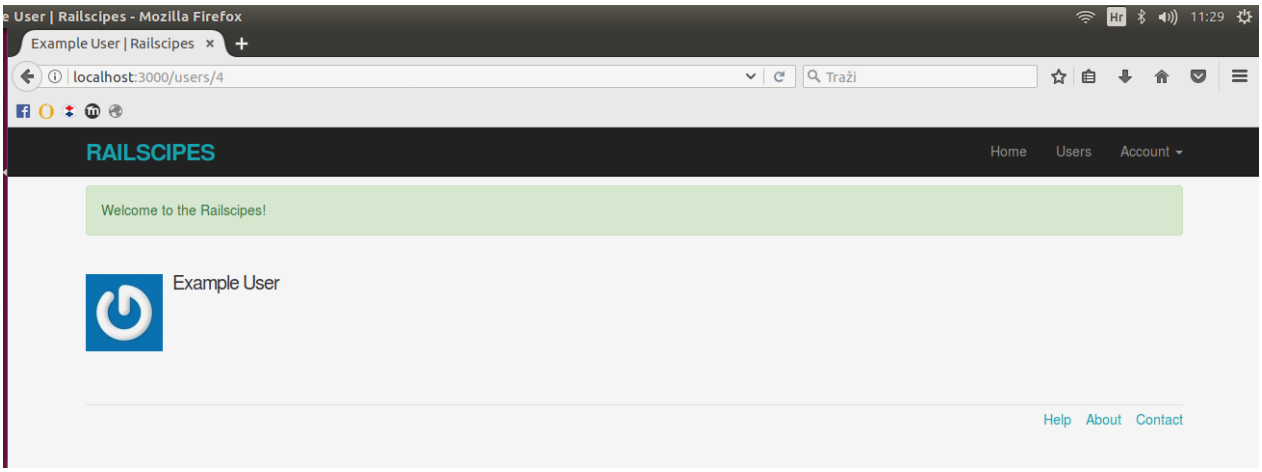
Na slici 4.1. može se vidjeti prikaz početne stranice web aplikacije. Početna stranica sadržava logotip, te tipke *Home* (vraćanje na početnu stranicu u slučaju odlaska na neki drugi dio web aplikacije), *Sign in* (prijava korisnika) i *Sign up* (registracija korisnika). Na početnoj stranici se još može vidjeti i *Search* pomoću kojega se pretražuju recepte upisivanjem pojma u tražilicu i pritiskom na *Search*. Ako rezultata s traženim pojmom ima, oni će se ispisati, a ako nema, ispisati će se prigodna poruka.

Kako bi web aplikaciju bila dalje testirana, jedan korisnik bit će registriran i uz pomoć njega bit će pokazane sve mogućnosti ove web aplikacije.

Dio web aplikacije za registraciju novih korisnika popunjenu s podacima novoga korisnika koji će biti registriran prikazan je na sljedećoj slici.

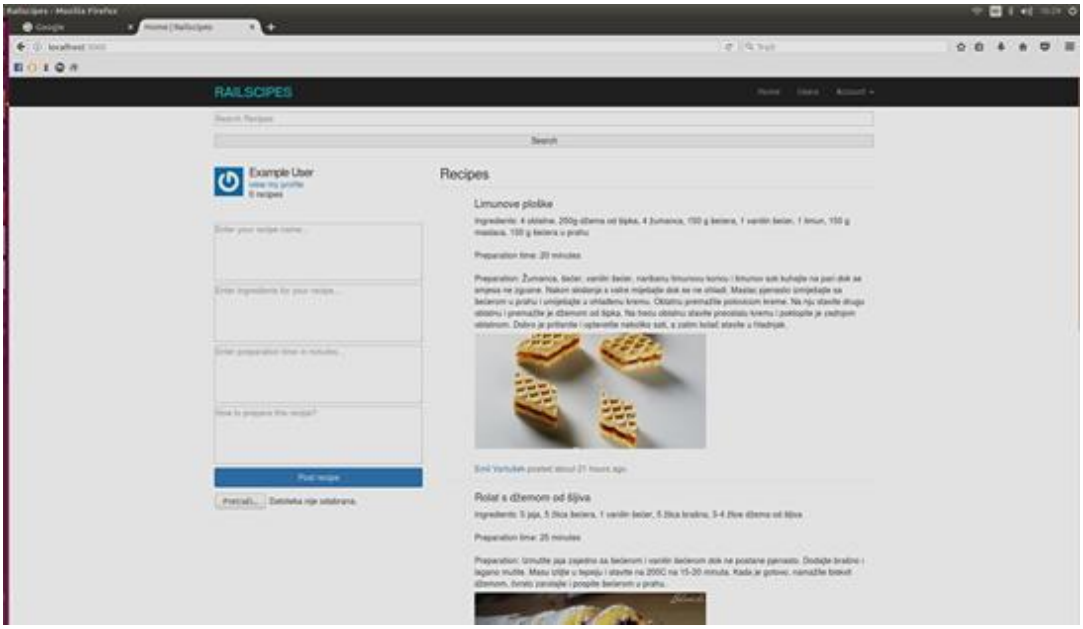


Sl. 4.2. Registracija novoga korisnika



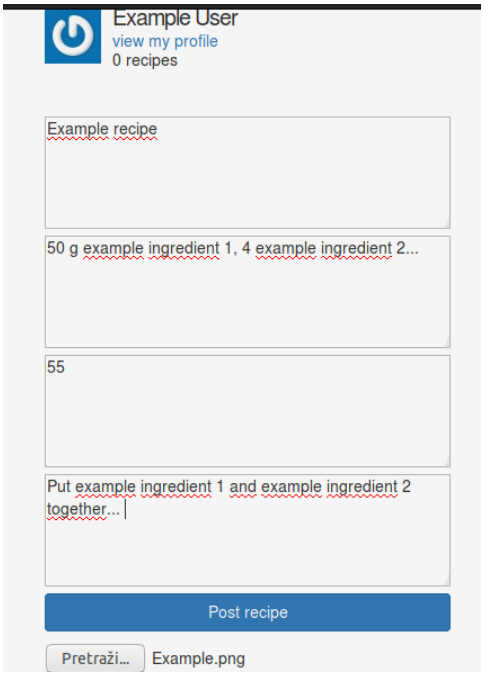
Sl. 4.3. Uspješna registracija korisnika

Nakon što je korisnik registriran, on je odmah prijavljen za rad na web aplikaciji. Izgled početne stranice prijavljenoga korisnika može se vidjeti na slici 4.4. Jedina razlika između početne stranice prijavljenoga i neprijavljenoga korisnika je u tome što početna stranica prijavljenoga korisnika, uz prikaz svih recepata, sadži i formu za popunjavanje, tj. zapisivanje novoga recepta te njegovo objavljivanje.



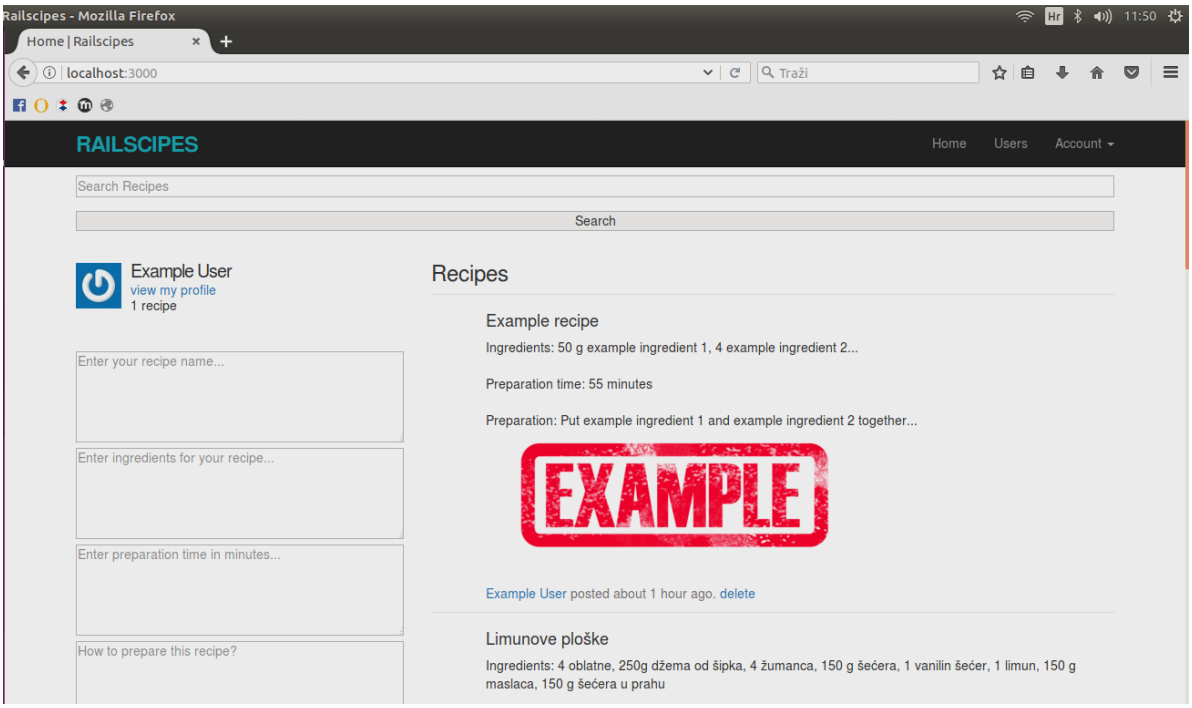
Sl. 4.4. Izgled početne stranice nakon prijave korisnika

Kako je glavna prednost prijavljenog korisnika objavljivanje recepata, jedan novi recept u ime novoga korisnika bit će objavljen. Popunjena forma za objavljivanje recepta prikazana je na slici 4.5.



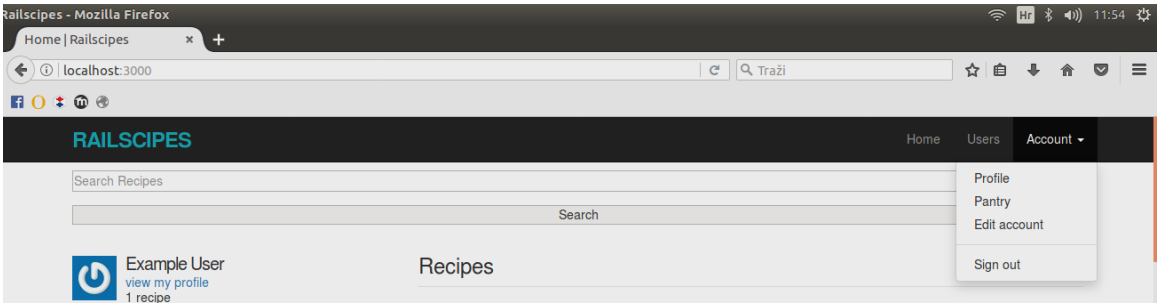
Sl. 4.5. Ispunjena forma za objavu novoga recepta

Nakon što je novi recept objavljen, on se pojavljuje na prvom mjestu na početnoj stranici. Na sljedećoj slici može se vidjeti početna stranica s novonapisanoim receptom.



Sl. 4.6. Novonapisani recept novoga korisnika na početnoj stranici

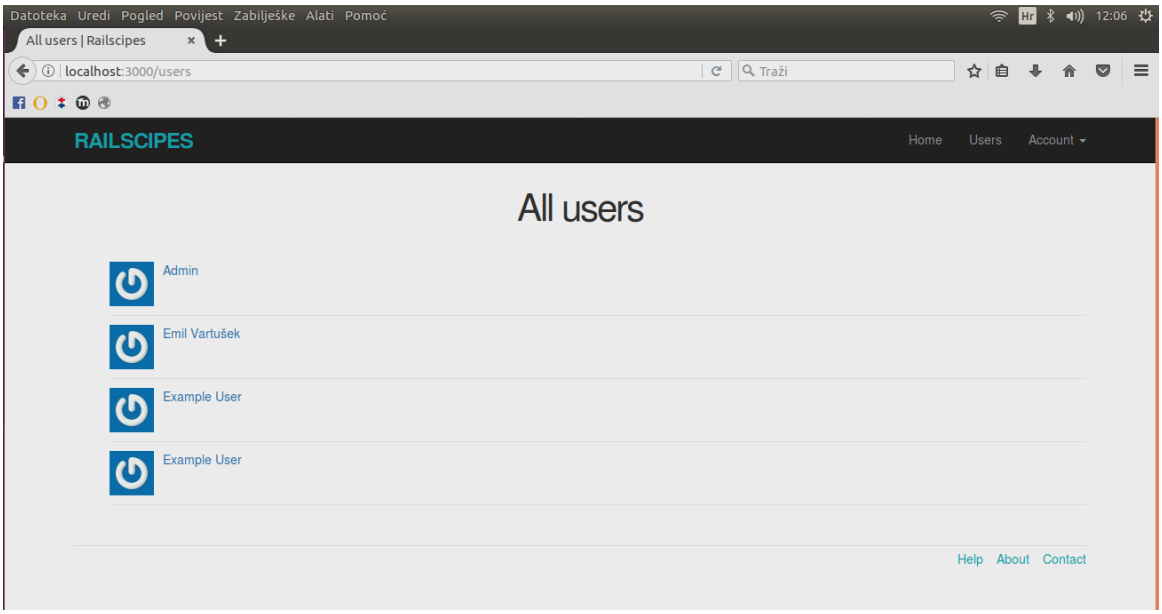
Kako se prijavom korisnika promijenila početna stranica, tako su se promijenile i tipke koje se nalaze desno od logotipa web aplikacije. Sada je moguće vidjeti *Home* tipku, *Users* tipku koja prikazuje sve korisnike web aplikacije, te dropdown izbornik *Account* u kojemu se nalaze tipke *Profile* (prikazuje profil prijavljenoga korisnika), *Pantry* (prikaz smočnice), *Edit account* (promjena podataka korisnika) i *Sign out* (odjava korisnika).



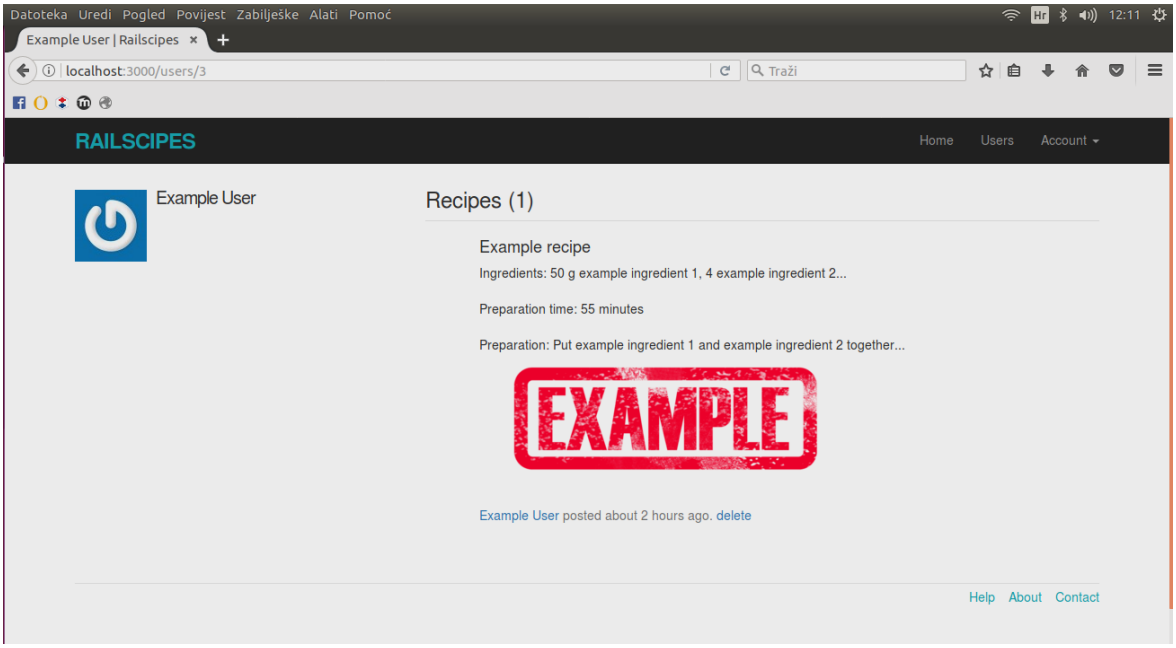
Sl. 4.7. Izgled tipki početne stranice prijavljenoga korisnika

Već je viđeno kako izgleda početna stranica prijavljenoga korisnika, sada preostaje vidjeti što se dobije pritiskom na ostale tipke. Slika 4.8. prikazuje sve korisnike koji su prikazani pritiskom na tipku *Users* (važno je napomenuti kako admin ima opciju brisanja korisnika), slika 4.9. prikazuje profil prijavljenoga korisnika pritiskom na tipku *Profile*, slika 4.10. prikazuje dio web aplikacije za promjenu podataka pritiskom na *Edit account* tipku, a tipka *Sign out* odjavljuje trenutnog

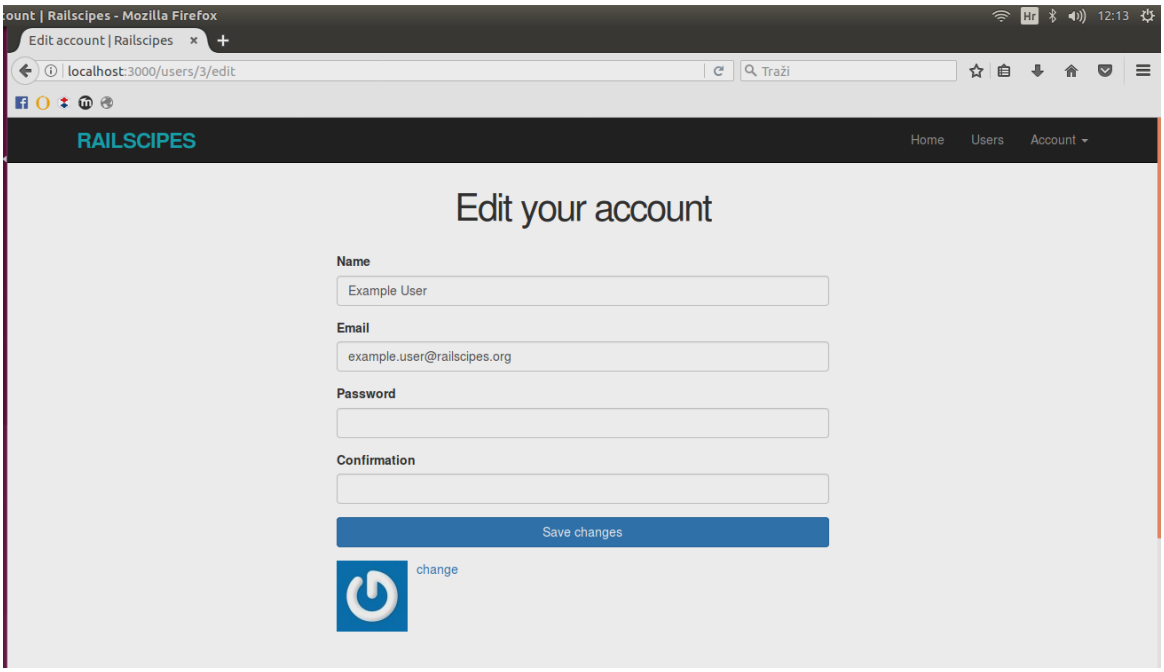
korisnika. Nešto više o prikazu smočnice pritiskom na tipku *Pantry* biti će rečeno kasnije, jer taj dio web aplikacije sadrži neke složenije radnje koje je također potrebno objasniti.



Sl. 4.8. Svi korisnici web aplikacije prikazani pritiskom na tipku *Users*

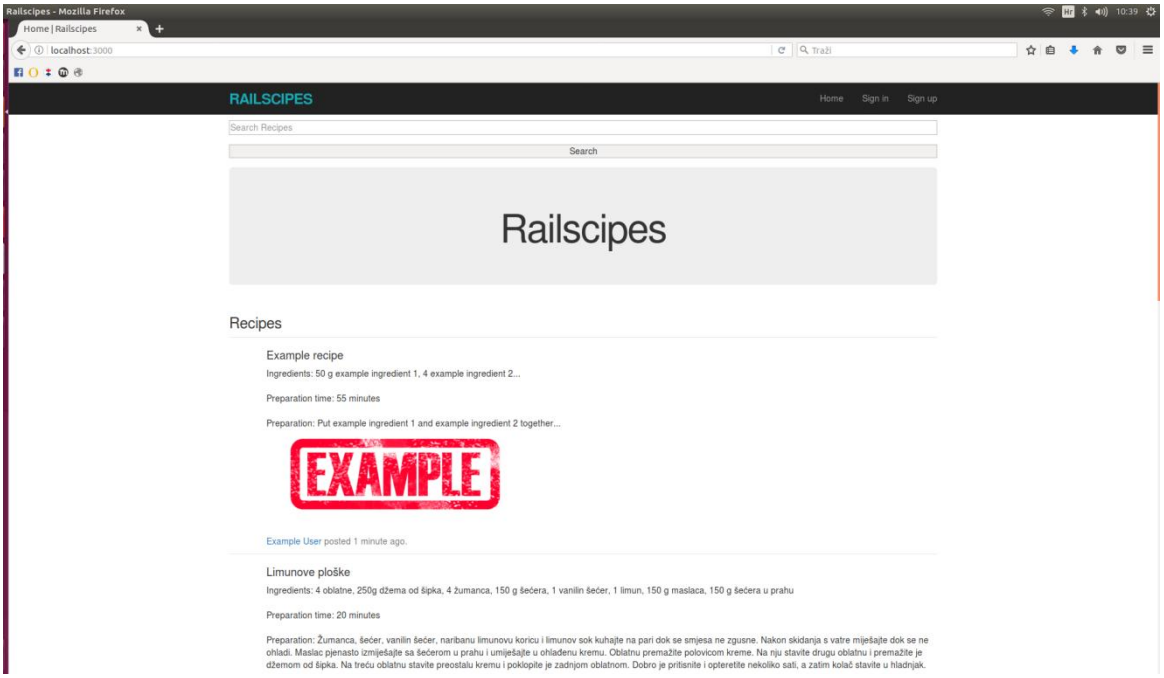


Sl. 4.9. Prikaz profila prijavljenoga korisnika



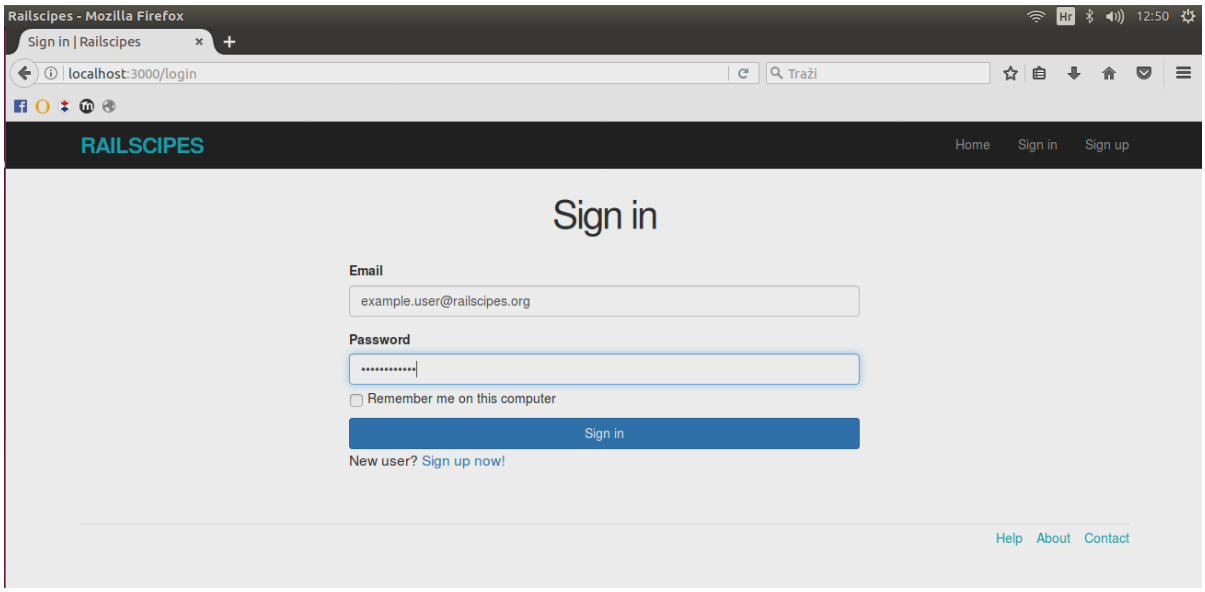
Sl. 4.10. Prikaz dijela web aplikacije za promjenu podataka korisnika

Nakon što se korisnik odjavi, napisani recept prikazan je na početnoj stranici koju mogu vidjeti i neprijavljeni korisnici. To se može vidjeti na slici 4.11.



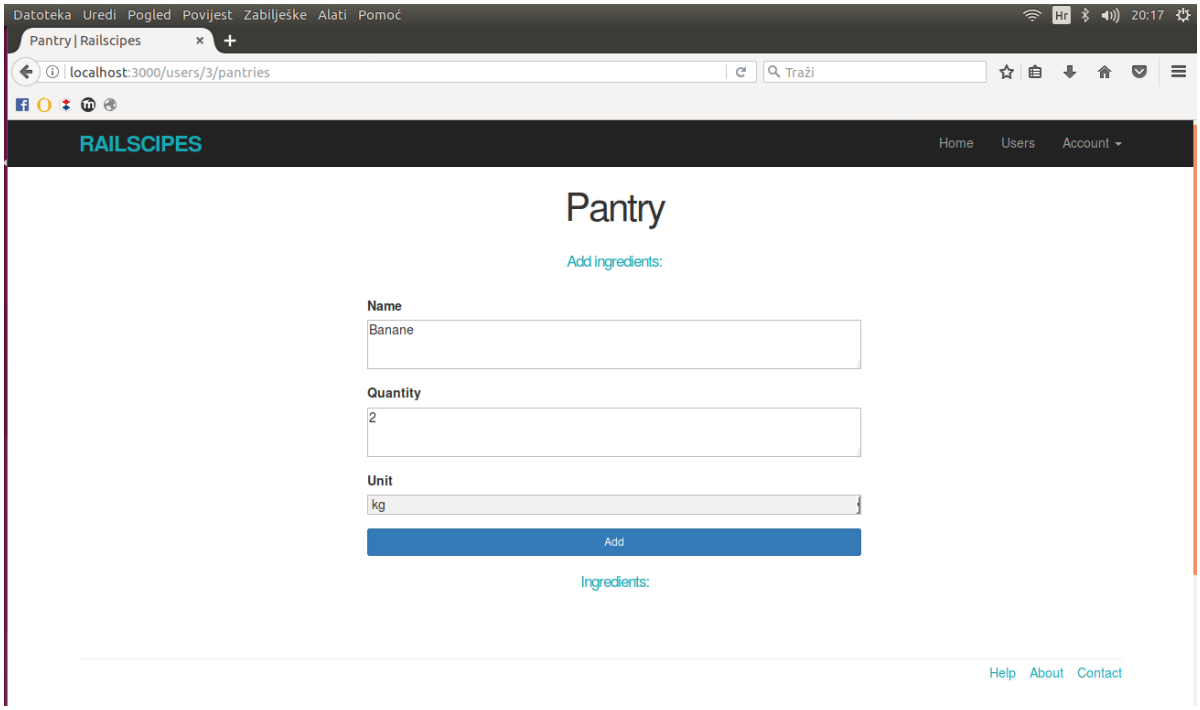
Sl. 4.11. Prikaz početne strane neprijavljenoga korisnika s novonapisanim receptom

Na kraju preostaje objasniti princip rada smočnice. Kako je korisnik odjavljen s web aplikacije, prvo je potrebno prijaviti ga. Forma za prijavu korisnika popunjena podacima korisnika koji se prijavljuje prikazana je na slici 4.12.

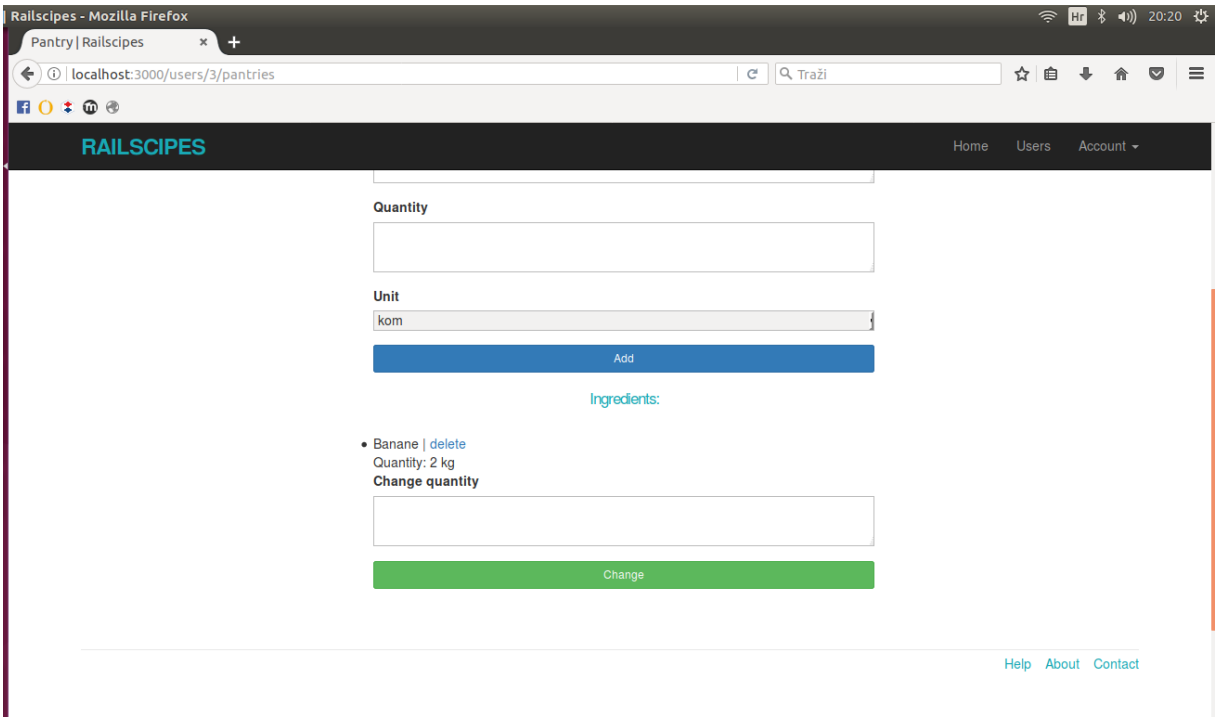


Sl. 4.12. Forma za prijavu korisnika

Nakon što je korisnik prijavljen, može unijeti sastojke pritiskom na tipku *Pantry* u *Account dropdown* izborniku. Potrebno je samo upisati naziv sastojka, njegovu količinu, te iz padajućeg izbornika izabrati mjernu jedinicu koja odgovara upisanom sastojku.



Sl. 4.13. Ispunjena forma za upis novoga sastojka



Sl. 4.14. Novi sastojak upisan u smočnicu korisnika

Popunjena forma za upisivanje novoga sastojka te novi sastojak u smočnici korisnika nakon što je dodan tamo može se vidjeti na slikama 4.13. i 4.14. Također se može vidjeti kako se upisani sastojak može i obrisati ili mu se može promijeniti količina nakon što se sastojak potroši ili nabavi.

5. ZAKLJUČAK

Izrada web aplikacije s bazom recepata pokazla se kao složen poduhvat. Bilo je potrebno primjeniti mnoga znanja i vještine, od rada u programskom jeziku *Ruby* do rada u prezentacijskim i stilskim jezicima (HTML i CSS). No, kombinacija ovih tehnologija pokazala se kao odlično okruženje za rješavanje problema izrade web aplikacije. *Ruby*, zajedno s mrežnim programskim okvirom *Ruby on Rails*, uvelike je pridonjeo i olakšao posao izrade web aplikacije jer je od samoga početka povezao sve glavne aspekte izrade jedne web aplikacije. Neometano smo koristili programski jezik *Ruby* zajedno s prezentacijskim jezikom HTML i stilskim jezikom CSS, a sve to odmah smo bili u mogućnosti i povezati s bazom podataka koja predstavlja temelj web aplikacije s bazom recepata. Baze potrebne za izradu ove web aplikacije izrađene su u *SQLite* relacijskoj bazi podataka koja se također pokazala kao dobar odabir zbog odlične povezanosti s *Ruby on Rails* mrežnim programskim okvirom. Važno je još napomenuti kako su sve ove tehnologije otvorenoga koda, što znači da je rad u njima bio besplatan i neometan.

Što se tiče same web aplikacije s bazom recepata, ona je funkcionalna i odrađuje posao koji joj je zadan, ali ima još mnogo prostora za napredak. Ovaj napredak mogao bi se izkazati na poljima logike (dorađivanje smočnice) i uređivanja dizajna web aplikacije, no on trenutno nije potreban.

Na kraju, nakon što je web aplikacija s bazom recepata izrađena, možemo reći kako smo uspješno ispunili zadatak izrade web aplikacije koja će korisniku omogućiti čitanje, ali i objavljivanje recepata, te mu pružiti uvid u stanje njegove smočnice.

LITERATURA

[1] https://www.railstutorial.org/book/_single-page, svibanj 2017.

[2] http://rubyonrails.org/, lipanj 2017.

[3] https://www.ruby-lang.org/en/about/, lipanj 2017.

[4] https://en.wikipedia.org/wiki/Ruby_(programming_language), lipanj 2017.

[5] https://www.plus.hr/blog/2011/06/08/ruby-on-rails/, lipanj 2017.

[6] https://www.w3schools.com/html/, lipanj 2017.

[7] https://en.wikipedia.org/wiki/HTML, lipanj 2017.

[8] https://www.w3schools.com/css/, lipanj 2017.

[9] https://en.wikipedia.org/wiki/Cascading_Style_Sheets, lipanj 2017.

[10] http://tosbourn.com/what-is-the-gemfile/, lipanj 2017.

[11] https://betterexplained.com/articles/intermediate-rails-understanding-models-views-and-controllers/, lipanj 2017.

[12] https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller, lipanj 2017.

[13] http://guides.rubyonrails.org/active_model_basics.html, lipanj 2017.

[14] http://guides.rubyonrails.org/action_controller_overview.html, lipanj 2017.

[15] http://guides.rubyonrails.org/action_view_overview.html, srpanj 2017.

[16] http://api.rubyonrails.org/classes/ActionDispatch/Routing.html, srpanj 2017.

[17] http://www.linuxzasve.com/uvod-u-github, srpanj 2017.

SAŽETAK

Cilj ovoga završnoga rada bio je izraditi funkcionalnu web aplikaciju s bazom recepata za kolače iz koje će korisnici moći čitati recepte, ali ih i upisivati u bazu podataka, te isto tako i provjeriti stanje njihove smočnice, tj. provjeriti koje sastojke za izradu kolača posjeduju i u kojim količinama. Tehnologije korištene za izradu ove web aplikacije su programski jezik *Ruby*, za izradu baze podataka korištena je *SQLite* relacijska baza podataka, a dizajn web aplikacije rađen je pomoću prezentacijskog jezika HTML i stilskog jezika CSS. Sve ove tehnologije dostupne su u okruženju mrežnog programskog okvira za izradnju web aplikacija *Ruby on Rails* koji sjedinjuje sve ove tehnologije i pospješuje rad na web aplikaciji.

Ključne riječi: web aplikacija, *Ruby*, *Ruby on Rails*, HTML, CSS, *SQLite*

ABSTRACT

Web application with recipes database

The main task of this final paper was to create a functional web application with cookies recipes database which will allow users not only to read recipes, but to also write them into the database and to check their pantry, ie. to check which ingredients for making a cookie they have and how much. Technologies used to develop this web application were Ruby programming language, SQLite to create a database and for web application's design presentation language HTML and CSS language. All these technologies are available within the Ruby on Rails web application framework which unites all these technologies and helps in web application development.

Key words: web application, Ruby, Ruby on Rails, HTML, CSS, SQLite

ŽIVOTOPIS

Emil Vartušek rođen je 1.8.1995. godine u Osijeku. Osnovnu školu završio je 2010. godine u Osnovnoj školi Ante Starčevića u Viljevu. Nakon toga upisuje opću gimnaziju u Srednjoj školi Donji Miholjac, te ju 2014. godine i završava. Iste godine upisuje i Elektrotehnički fakultet u Osijeku (kasnije preimenovan u Fakultet elektrotehnike, računarstva i informacijskih tehnologija), smjer računarstvo, koji i danas pohađa.

POTPIS:

_____