Mrežna stranica za recepte izrađena u PHP-u pomoću Laravel programskog okvira

Sedlarević, Tomislav

MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU

STRUČNI PRIJEDIPLOMSKI STUDIJ RAČUNARSTVO

Tomislav Sedlarević, 03160017876

Mrežna stranica za recepte izrađena u PHP-u pomoću Laravel programskog okvira

Recipe website built in PHP using the Laravel framework

Završni rad

Mentor: dr. sc.
Sanja Brekalo,
prof. struč. stud.

Čakovec, srpanj 2024.

MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU

PRIJAVA TEME I OBRANE ZAVRŠNOG/DIPLOMSKOG RADA

Stručni prijediplomski studij:

☑ Računarstvo ☐ Održivi razvoj ☐ Menadžment turizma i sporta

Stručni diplomski studij Menadžment turizma i sporta: ☐

Pristupnik: Tomislav Sedlarević (ime i prezime), JMBAG: 03160017876

Kolegij: PHP programiranje (na kojem se piše rad)

Mentor: dr.sc. Sanja Brekalo, prof. struč. stud. (ime i prezime, zvanje)

Naslov rada: Mrežna stranica za recepte izrađena u PHP-u pomoću Laravel programskog okvira

Naslov rada na engleskom jeziku: Recipe website built in PHP using the Laravel framework

Članovi povjerenstva:
1. Bruno Trstenjak (ime i prezime, zvanje), predsjednik
2. Jurica Trstenjak (ime i prezime, zvanje), član
3. Sanja Brekalo (ime i prezime, zvanje), mentor
4. Marija Mlšančuk (ime i prezime, zvanje), zamjenski član

Broj zadatka: 2023-RAČ-1

Kratki opis zadatka: Mrežna stranica se izrađuje pomoću Laravel programskog okvira. Izrađena aplikacija bi služila za dijeljenje i istraživanje kulinarskih recepata. Ideja je omogućiti korisnicima registraciju, dodavanje vlastitih recepata, pretraživanje po različitim kriterijima, ocjenjivanje i komentiranje recepata te prilagođavanje profila prema preferencijama. Za spremanje recepata koristila bi se baza podataka MYSQL.

Datum: 25.06.2024.

Potpis mentora: Sanja Brekalo

Predgovor

Tema ovog završnog rada je razvoj web aplikacije za recepte koristeći Laravel okvir zbog njegove fleksibilnosti i snage u razvoju modernih web aplikacija. *Laravel* pruža set alata i funkcionalnosti koje omogućuju razvijanje aplikacija sa MVC strukturom. Također, *Laravel* ima detaljnu dokumentaciju i stalna ažuriranja što ga čini idealnim izborom za ovaj projekt. Zahvaljujem se svim profesorima s Međimurskog Veleučilišta koji su mi pružili znanje i podršku tijekom studija, posebno mentorici koja me vodila kroz izradu ovog rada. Veliku zahvalnost dugujem i mojim kolegama i prijateljima koji su mi pomagali savjetima i idejama. Također, zahvaljujem se i obitelji na neizmjernoj podršci i razumijevanju tijekom cijelog procesa izrade ovog rada. Ovaj rad posvećujem svojoj obitelji, prijateljima i mentorici koji su uvijek vjerovali u mene i poticali me na uspjeh.

Sažetak

U ovom radu detaljno je opisana izrada web aplikacije korištenjem PHP-a, *Laravel*-a, MySQL-a, te *front-end* tehnologija kao što su HTML5, CSS3, JavaScript i *jQuery*. U radu je objašnjen proces izrade aplikacije korištenjem MVC arhitekture, s posebnim naglaskom na upotrebu *Laravel*-ovih alata poput *Artisan*-a, *Blade* sistema i *Eloquent ORM*-a. Također, pojašnjena je integracija *Livewire*-a za dinamičke komponente, kao i korištenje *Bootstrap*-a za razvoj korisničkog sučelja. Rad detaljno prikazuje korake kreacije aplikacije, počevši od instalacije projekta, kreiranja baze podataka uz ER dijagrame i migracije, pa sve do izrade stranica kroz modele, poglede, *Livewire* komponente i kontrolere. Posebna pažnja posvećena je rutama i *middleware*-u za upravljanje zahtjevima te tvornici u *Laravel*-u za generiranje testnih podataka. Dodatno, opisani su slučajevi upotrebe aplikacije te proces spajanja na udaljeni server i prikaz konačnog izgleda stranice.

Ključne riječi: *PHP, Laravel, MVC arhitektura, Artisan, Blade, Eloquent ORM, Livewire*

Abstract

The final project provides a detailed description of the development of a web application using PHP, Laravel, MySQL, and front-end technologies such as HTML5, CSS3, JavaScript, and jQuery. The project describes the process of building the application using the MVC architecture, with a particular focus on the use of Laravel tools such as Artisan, Blade system, and Eloquent ORM. It also details the integration of Livewire for dynamic components, as well as the use of Bootstrap for developing the user interface. The project meticulously outlines the steps of creating the application, starting from project installation, database creation with ER diagrams and migrations, to the development of pages through models, views, Livewire components, and controllers. Special attention is given to routes and middleware for request management, and Laravel's factory for generating test data. Additionally, the use cases of the application are described, along with the process of connecting to a remote server and showcasing the final appearance of the website.

Keywords: *PHP, Laravel, MVC architecture, Artisan, Blade, Eloquent ORM, Livewire*

Popis korištenih kratica

PHP Hypertext Preprocessor

HTML HyperText Markup Language

CSS Cascading Style Sheets

DOM Document Object Model

CLI Command Line Interface

AJAX Asynchronous JavaScript and XML

CMS Content Management System

Sadržaj

1. UVOD

Digitalno doba zahtjeva izgradnju web aplikacija kao vitalan dio informatičke industrije. Korištenjem suvremenih tehnologija i alata, razvoj aplikacija postaje sve pristupačniji, te olakšava kreiranje stvaranje web stranica i aplikacija. Rad se fokusira na analizu i kreaciju web aplikacija korištenjem PHP programskog jezika i *Laravel* okvira. PHP ostaje jedan od najčešće korištenih programskih jezika za razvoj stranica na strani poslužitelja uz okvir kao što je *Laravel*. Prema istraživanju, PHP je značajno prisutniji u odnosu na .NET okruženje za izradu web sadržaja. [1] Danas se za izradu web stranica koriste razne tehnologije koje omogućuju različite pristupe u razvoju. JavaScript je ključna tehnologija za *front-end* razvoj, pri stvaranju dinamičkih i interaktivnih korisničkih sučelja. Okviri kao što su *React*, *Angular* i *Vue* postali su industrijski standardi zbog svoje fleksibilnosti i performansi. Pored PHP-a za logiku na strani poslužitelja i JavaScript-a za *front-end* interakcije, moderni web razvoj uključuje upotrebu raznih alata i okvira, kako bi se pojednostavio razvojni proces. To može uključivati biblioteke kao što su *jQuery* za lakšu manipulaciju DOM-om i rad s AJAX-om. Istodobno, pomaže i kod CMS platforme, kao što su WordPress ili Joomla za kreiranje i upravljanje sadržajem na web stranicama. Uz tehnologije specifične za razvoj, web stranice su ključni alat u digitalnom marketingu. Pružaju online prisutnost i informacije o proizvodima i uslugama. S druge strane, analitički alati kao što su *Google Analytics* pomažu u praćenju performansi web stranica, analiziranju ponašanja korisnika i optimizaciji marketinških strategija. [2] Fokus rada biti na kreaciji web aplikacija korištenjem PHP programskog jezika i *Laravel* okvira.

2. Korištene Tehnologije

1.1 PHP

PHP je dinamički programski jezik koji je osmišljen za kreiranje interaktivnih web stranica na serveru. PHP kod se obrađuje na web serveru, a ne na klijentskom uređaju. Kada se posjeti stranica koja sadrži PHP kod, web preglednik na uređaju korisnika šalje zahtjev serveru za prikazivanje stranice. Na serveru se obrađuje PHP kod i generira HTML koji se zatim šalje nazad do korisničkog uređaja. Dinamičko generiranje HTML sadržaja omogućava se ovisno o različitim faktorima kao što su korisnički unos ili podaci iz baze podataka. PHP stranice mogu se izrađivati bez programskih okvira, ali to zahtjeva više vremena. Okviri poput *Laravel*-a, *Symfony*, *CakePHP* i *FuelPHP* razvijeni su kako bi se omogućila brža izrada mrežnih stranica. [3]

1.2 Laravel

Laravel, okvir za PHP, izradio je Taylor Otwell 2011. godine za izradu web aplikacija programerima. Neke od značajki uključuju potporu za autentifikaciju i autorizaciju korisnika, *Eloquent ORM* za bazu podataka, rute i *Blade*. Za sigurnost se koriste CSRF tokeni koji se automatski kreiraju prilikom podnošenja forme. *Laravel* ima dokumentaciju koja omogućava učenje i rješavanje grešaka. Baziran je na MVC arhitekturi. [4]

1.2.1 MVC Arhitektura

MVC je skraćenica od *Model*, *View*, *Controller*, gdje svaka od komponenti ima svoju specifičnu ulogu u organizaciji i funkcioniranju aplikacije.

Model upravlja podacima i komunikacijom s bazom podataka. Podaci se čuvaju, obrađuju i manipuliraju kako bi aplikacija mogla djelovati na njima. Logika koja upravlja podacima i osigurava njihovu konzistentnost nalazi se u Modelu.

View, ili pogled, prezentira podatke korisnicima. Informacije se prikazuju na korisničkom sučelju i pruža se mogućnost interaktivne komunikacije korisnicima s aplikacijom. Pogledom se upravlja dizajnom i korisničkim iskustvom, te se preko njega korisnicima prikazuju i omogućavaju interakcije s aplikacijom.

Kontroler je posrednik između Modela i Pogleda. Korisnički zahtjevi se obrađuju i određuje se prikaz podataka iz Modela korisnicima putem Pogleda. Logika aplikacije se upravlja putem Kontrolera, koji također osigurava pravilnu obradu i prikaz podataka korisnicima. [5]

1.2.2 Artisan

Artisan je CLI (*Command Line Interface*) koji je uključen uz *Laravel*. Skripta *Artisan* nalazi se u korijenu aplikacije i nudi mogućnost korištenja naredbi za izgradnju aplikacije. Naredbe omogućuju generiranje modela, kreiranje migracija za bazu podataka, kreiranje kontrolera i dr. Naredbom *php artisan serve* pokreće se ugrađeni razvojni server na lokalnom računalu, uobičajeno na adresi *http://localhost:8000*, što dozvoljava pregled izgleda stranice prije stavljanja na stvarni server. [6]

1.2.3 Blade

Blade je alat koji dolazi s osnovnim *Laravel* okvirom i ima mogućnost kombiniranja PHP koda s HTML-om na način koji je jednostavan za razumijevanje i upotrebu, što čini razvoj korisničkih sučelja pristupačnijim. Svaki *blade* predložak sastavljen je u običan PHP kod i stavljen je u *Cache*(pred memoriju) dok se ne izmijeni. *Blade* datoteke obično koriste ekstenziju *.blade.php* i locirane su u *resources/view* datoteki. [7]

1.2.4 Eloquent ORM

Kroz *Eloquent*, omogućeno je efikasno upravljanje bazom podataka u aplikaciji. Objekti i metode se koriste za komunikaciju s bazom podataka umjesto tradicionalnih SQL upita. Modeli se definiraju u *Eloquent*-u kako bi se strukturirale tablice baze i njihovi međuodnosi. Nadalje, nude se napredne mogućnosti upita, uključujući upite s više uvjeta, grupiranje i sortiranje podataka. [8]

1.2.5 Middleware

Middleware u *Laravel*-u se koristi kao mehanizam koji omogućuje postavljanje filtera između dolaznih HTTP zahtjeva i odgovora koje aplikacija prima i šalje. Koristan je za obavljanje raznih zadataka poput provjere autentifikacije, autorizacije korisnika. Kroz *middleware*, provjere autentifikacije mogu se upravljati prije nego što korisniku bude omogućen pristup određenim dijelovima aplikacije, ili za izvođenje bilo kakvih dodatnih zadataka poput zapisivanja zahtjeva. [9]

1.3 Livewire

Laravel je napisan u PHP-u i izvršava se na strani servera, što može rezultirati osvježavanjem stranice za svaki zahtjev. Kako bi se to izbjeglo, potrebno je koristiti *Javascript* okvire koji se izvršavaju na strani klijenta poput *React*-a, *Vue* i *Svelte*. Za uspješnu integraciju s *Laravel*-om, potrebno je znanje *Javascript*-a i tih okvira. *Laravel* ima paket koji rješava taj problem, nazvan *Livewire*. *Livewire* je paket u *Laravel*-u koji omogućuje izradu reaktivnih stranica bez pisanja *Javascript*-a. Ključna mogućnost *Livewire*-a je kreiranje vlastitih komponenti. Komponente dozvoljavaju ponovnu upotrebu koda, olakšavajući organizaciju i održavanje projekta. Svaka komponenta može sadržavati svoju vlastitu logiku, prikaz i stilove. Korištenjem *Livewire* komponenti, moguće je izgraditi dinamička sučelja s interaktivnim elementima poput formi, tablica, kartica i još mnogo toga, bez potrebe za ručnim upravljanjem stanja sučelja ili korištenjem JavaScript biblioteka. [10]

1.4 XAMPP

XAMPP je besplatan softverski paket koji omogućuje kreiranje lokalnog web servera za razvoj i testiranje web aplikacija. Instalacija XAMPP-a uključuje preuzimanje softverskog paketa s web stranice *Apache Friends* i pokretanje instalacijskog programa. Nakon pokretanja, korisnik slijedi upute na zaslonu za instalaciju softvera na lokalno računalo. Kada se instalacija završi, XAMPP se može pokrenuti putem kontrolne ploče, gdje korisnik može upravljati servisima poput Apache servera, *MySQL* baze podataka i PHP-a za skriptiranje. [11]

1.5 MYSQL

MySQL, popularni relacijski sustav upravljanja bazama podataka, često se koristi u svijetu razvoja softvera. Sposobnost rukovanja podacima, bez obzira na veličinu baze ili složenost upita, čini ga popularnim izborom za različite vrste aplikacija. Osnovne relacijske operacije poput projekcije, selekcije i spoja koriste se za manipulaciju podacima u tim tablicama. Definiranje odnosa između entiteta u relacijskom modelu je putem primarnih i stranih ključeva, omogućujući povezivanje podataka između različitih tablica. [12]

1.6 HTML5 i CSS3

HTML (eng. *HyperText Markup Language*) i CSS (eng. *Cascading Style Sheets*) koriste se za izradu web stranica. HTML se upotrebljava za strukturiranje sadržaja, dok se CSS koristi za stiliziranje tog sadržaja. Oznake (tagovi) u HTML-u su za definiranje dijelova stranice poput zaglavlja, odlomaka, slika i linkova. Svaka oznaka ima početni i završni tag koji obuhvaća sadržaj koji označava. Na primjer,<*h1*> označava naslove, dok <*p*> označava odlomke. CSS omogućava dodavanje stilova HTML elementima, poput boja, fontova, razmaka i rasporeda. Stilovi se mogu definirati unutar HTML dokumenta ili u vanjskim CSS datotekama. CSS pravila sastoje se od selektora i deklaracija. Selektor određuje koji HTML elementi će biti stilizirani, dok deklaracije specificiraju stilove koji će se primijeniti na te elemente. [13] [14]

1.7 JavaScript i jQuery

JavaScript i *jQuery* su alati u web developmentu za dodavanje dinamičnosti i interaktivnosti na web stranice. JavaScript se koristi kao jezik za programiranje web aplikacija. Omogućava manipulaciju HTML-om i CSS-om kako bi se stvorila interaktivna iskustva za korisnike. JavaScript se može koristiti za validaciju formi, animacije, dodavanje i uklanjanje elemenata iz HTML-a, komunikaciju sa serverom (AJAX) i dr. *jQuery* je JavaScript biblioteka koja dozvoljava interakciju sa HTML elementima, manipulaciju DOM-om (eng. *Document Object Model*) i upravljanje događajima. [15] [16]

1.8 Bootstrap

Bootstrap i *Bootswatch* su alati koji se koriste za kreiranje modernih i responzivnih web stranica. *Bootstrap* je *front-end* okvir koji može olakšati dizajniranje web stranica, dok je *Bootswatch* kolekcija besplatnih tematskih stilova koji se mogu primijeniti na *Bootstrap* kako bi se postigao željeni izgled. *Bootstrap*, razvijen od strane Twittera, nudi set gotovih CSS stilova, JavaScript komponenti i ikona koje su potrebne za izradu web stranica. [17]

2. Kreacija aplikacije

2.1 Instalacija projekta

Za instalaciju je potreban *Composer* koji upravlja ovisnostima PHP-a. *Composer* se instalira sa stranice *https://getcomposer.org/download/* gdje se preuzme *exe* datoteka koja se instalira. Nakon instalacije kreira se prazan direktorij gdje se otvara terminal. Nakon otvaranja terminala pomoću naredbe prikazane u Kod 1. instalira se projekt.

```
composer create-project laravel/laravel recepti
```

Kod 1. Kreacija Laravel projekta

Izvor: Autor

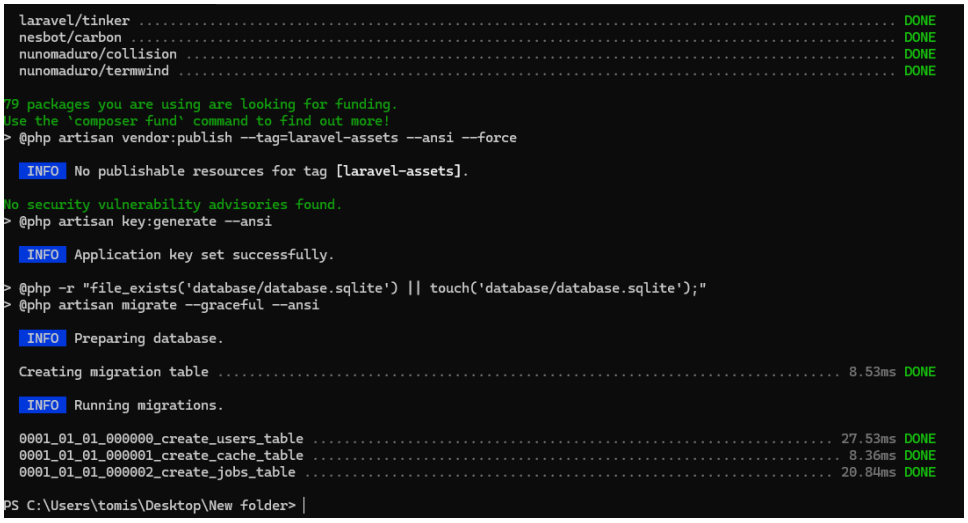
Nakon instalacije otvara se projekt *Visual Studio Code*-u. Zatim se dodatno instalira paket *Livewire* pomoću naredbe prikazane u Kod 2. Nakon instalacije cijeli projekt je spreman za izradu stranice. Slika 1. prikazuje uspješnu instalaciju *Laravel* projekta.

```
composer require livewire/livewire
```

Kod 2. Kod za instalaciju Livewire-a

Izvor: https://laravel-livewire.com/docs/2.x/installation (10.3.2024.)

Slika 1. Primjer instalacije Laravel projekta



Izvor: Autor

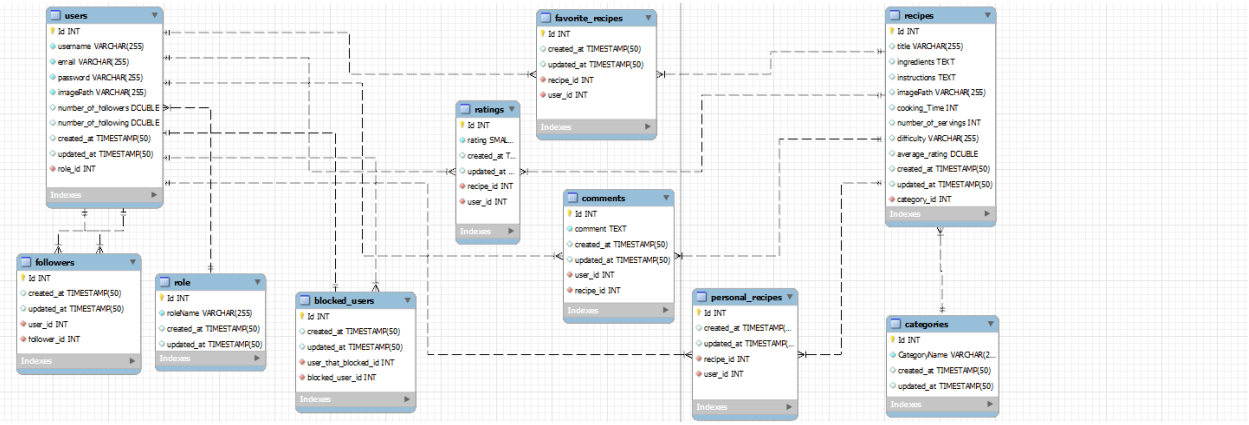
2.2 Baza podataka

Baza podataka je potrebna za spremanje podataka i omogućavanja efikasnog pretraživanja, manipulacije i organizacije istih. Program *MySQL*, *Workbench* nudi mogućnost vizualnog kreiranja tablica za bazu podataka, uključujući i relacije između tablica.

2.2.1 ER Dijagram

Pomoću ER dijagrama mogu se vidjeti sve tablice i relacije koje baza podataka za aplikaciju sadrži.

Slika 2. ER Dijagram baze podataka



Izvor: Autor

2.2.2 Opis tablica

Tablica 1. role

Naziv Stupca	Tip podatka	Osobine	Opis
Id	Integer	Primarni ključ	
roleName	Varchar	Ne smije biti prazno	Sadrži ime uloge
Created_at	Timestamp		Vrijeme kada se uloga kreirala
Updated_at	Timestamp		Vrijeme kada se uloga zadnje promijenila

Izvor: Autor

Tablica 1. predstavlja sve uloge koje korisnik može imati, a to su *users* ili *admin*. To su predefinirane uloge koje svaki korisnik mora imati. Povezana je sa tablicom *users* gdje je veza između njih 1 naprema više (1:N) što znači da jedan korisnik može imati samo jednu ulogu, dok više korisnika mogu imati istu ulogu.

Tablica 2. categories

Naziv Stupca	Tip podatka	Osobine	Opis
Id	Integer	Primarni ključ	
CategoryName	Varchar	Ne smije biti prazno	sadrži ime kategorije
Created_at	Timestamp		Vrijeme kada se kategorija kreirala
Updated_at	Timestamp		Vrijeme kada se kategorija zadnje promijenila

Izvor: Autor

Tablica 2. sadrži imena kategorija koje svaki recept treba imati. Povezana je sa tablicom r*ecipes* gdje je relacija između njih jedan naprema više (1:N). Relacijom se pokazuje da svaki recept može imati samo jednu kategoriju, ali da je svaki recept dodijeljen samo jednoj kategoriji.

Tablica 3. users

Naziv Stupca	Tip podatka	Osobine	Opis
Id	Integer	Primarni ključ	

Username	Varchar	Ne smije biti prazno	Sadrži korisničko ime
Email	Varchar	Ne smije biti prazno	Sadrži email od korisnika
Password	Varchar	Ne smije biti prazno	Sadrži kriptiranu lozinku korisnika
imagePath	Varchar	Ne smije biti prazno	Sadrži putanju do korisničke slike
Number_of_followers	Double		Sadrži broj pratitelja određenog korisnika
Number_of_following	Double		Sadrži broj ljudi koje određeni korisnik prati
Role_id	Integer	Ne smije biti prazno	Sadrži referencu na primarni ključ tablice „Role“
Created_at	Varchar		Vrijeme kada je korisnik kreiran
Updated_at	Varchar		Zadnje vrijeme kada se dogodila promjena kod korisnika

Izvor: Autor

Tablica 3. sadrži sve podatke o korisnicima. Relacije postoje sa tablicama role, *blocked_users*, *followers*, *comments*, *favorite_recipes*, *personal_recipes*, *ratings*.

Tablica 4. blocked_users

Naziv Stupca	Tip podatka	Osobine	Opis
Id	Integer	Primarni ključ	

Blocked_user_id	Integer	Strani ključ	Sadrži Id od korisnika koji je blokiran
User_that_blocked_id	Integer	Strani ključ	Sadrži Id od korisnika koji je blokirao tog korisnika
Created_at	Timestamp		Vrijeme kada je korisnik bio blokiran
Updated_at	Timestamp		Vrijeme kada se dogodila promjena

Izvor: Autor

Tablica 4. sadrži korisnike koji su blokirani i tko je blokirao određenog korisnika. Povezana je sa tablicom *users* pomoću dviju relacija. Prva relacija je *blocked_user_id* ima vezu jedan naprema jedan (1:1) sa *users* tablicom. Druga relacija je *user_that_blocked_id* ima vezu jedan naprema više (1:N) sa *users* tablicom. Kombinacija između *blocked_user_id* i *user_that_blocked_id* mora biti jedinstvena tj. ne smije biti ista, kako bi se spriječilo ponovo blokiranje istog korisnika.

Tablica 5. followers

Naziv Stupca	Tip podatka	Osobine	Opis
Id	Integer	Primarni ključ	
User_id	Integer	Strani ključ	sadrži Id od korisnika kojeg ga prate
Follower_id	Integer	Strani ključ	Sadrži Id od korisnika koji prati određenog korisnika
Created_at	Timestamp		Vrijeme kada se kreirala nova vrijednost u tablici
Updated_at	Timestamp		Vrijeme kada se dogodila promjena

Izvor: Autor

Tablica 5. sadrži podatke o pratiteljima i pratiocima. Tablica ima relaciju sa *users* tablicom preko *user_id* i *follower_id*. Kod relacije *user_id* veza između tablica je jedan naprema više (1:N) jer više zapisa može biti povezano sa jednim zapisom u tablici *users*. Relacija *follower_id* ima istu vezu kao i *user_id*. Kombinacija dviju relacija mora biti jedinstvena tj. ne smije se ponavljati.

Tablica 6. recipes

Naziv Stupca	Tip podatka	Osobine	Opis
Id	Integer	Primarni ključ	
Title	Varchar		Sadrži naslov recepta
Ingredients	Text		Sadrži sastojke recepta
Instructions	Text		Sadrži instrukcije za spremanje recepta
imagePath	Varchar		Putanja slike recepta
Cooking_time	Int		Sadrži dužinu kuhanja recepta
Number_of_servings	Int		Sadrži broj porcija recepta
Difficutly	Varchar		Sadrži težinu recepta
Average_rating	Double		Sadrži prosječnu ocjenu koju recept ima
Category_id	Int	Strani ključ	Sadrži id od kategorije
Created_at	Timestamp		Vrijeme kada je recept kreiran
Updated_at	Timestamp		Vrijeme kada je recept ažuriran

Izvor: Autor

Tablica 6. sadrži sve atribute koji su potrebni za spremanje recepta. Postoji relacija sa tablicama *categories*, *ratings*, *comments*, *favorite_recipes* i *personal_recipes*.

Tablica 7. comments

Naziv Stupca	Tip podatka	Osobine	Opis
Id	Integer	Primarni ključ	
Comment	Text		Sadrži komentar od korisnika
User_id	Integer	Strani ključ	Sadrži Id od korisnika koji je ostavio određeni komentar
Recipe_id	Integer	Strani ključ	Sadrži id od recepta na kojem je određeni korisnik ostavio komentar
Created_at	Timestamp		Vrijeme kreiranja komentara
Updated_at	Timestamp		Zadnje ažuriran komentar

Izvor: Autor

Tablica 6. sadrži komentare od korisnika, koji korisnik je komentirao i na koji recept. Postoji relacija sa tablicom *users* preko *user_id* veze i sa tablicom *recipes* preko *recipe_id* veze. Tip veze sa *users* tablicom je jedan naprema više (1:N) kao i veza sa *recipes* tablicom. Kombinacija tih veza mora biti jedinstvena, što znači da korisnik može samo komentirati jednom na recept.

Tablica 8. ratings

Naziv Stupca	Tip podatka	Osobine	Opis
Id	Integer	Primarni ključ	

Rating	Small Integer		Sadrži ocjenu od korisnika
User_id	Integer	Dio kompozitnog primarnog ključa, strani ključ	Sadrži Id od korisnika koji je ostavio ocjenu
Recipe_id	Integer	Dio kompozitnog primarnog ključa, strani ključ	Sadrži id od recepta na kojem je određeni korisnik ostavio ocjenu
Created_at	Timestamp		Vrijeme kreiranja ocjene
Updated_at	Timestamp		Zadnje ažurirana ocjena

Izvor: Autor

Tablica 8. sadrži ocjene koje je korisnik ostavio na određenom receptu. Postoji relacija sa tablicom *users* i *recipes* preko *user_id* i *recipe_id* veze. Veza sa *users* tablicom je jedan naprema više (1:N), isto kao i veza sa *recipes* tablicom.

Tablica 9. personal_recipes

Naziv Stupca	Tip podatka	Osobine	Opis
Id	Integer	Primarni ključ	
User_id	Integer	Strani ključ	Sadrži Id od korisnika koji je ostavio ocjenu
Recipe_id	Integer	Strani ključ	Sadrži id od recepta na kojem je određeni korisnik ostavio ocjenu
Created_at	Timestamp		Vrijeme kreiranja ocjene
Updated_at	Timestamp		Zadnje ažurirana ocjena

Izvor: Autor

Tablica 9. sadrži sve recepte koji su korisnici kreirali. Postoji relacija sa *users* i *recipes* tablicama. Veza sa tablicom *users* je jedan naprema više (1:N), dok je veza sa *recipes* tablicom jedan naprema više (1:1), što znači da korisnik može kreirati više recepata, dok jedan recept može imati samo jednog kreatora.

Tablica 10. favorite_recipes

Naziv Stupca	Tip podatka	Osobine	Opis
Id	Integer	Primarni ključ	
User_id	Integer	Dio kompozitnog primarnog ključa, strani ključ	Sadrži Id od korisnika koji je ostavio ocjenu
Recipe_id	Integer	Dio kompozitnog primarnog ključa, strani ključ	Sadrži id od recepta na kojem je određeni korisnik ostavio ocjenu
Created_at	Timestamp		Vrijeme kreiranja ocjene
Updated_at	Timestamp		Zadnje ažurirana ocjena

Izvor: Autor

Tablica 10. sadrži recepte koje su korisnici označili da su im omiljeni. Tablica ima relaciju jedan naprema više (1:N) sa *users* tablicom jer jedan korisnik može imati više omiljenih recepata, a veza sa *recipes* tablicom je ista. Kombinacija *user_id* i *recipe_id* veza se ne smije ponavljati kako korisnik ne bi mogao stavljati jedan recept više puta kao omiljen.

2.3 Migracije

Migracije u *Laravel*-u omogućuju definiranje strukture baze podataka kroz PHP kod. Umjesto pisanja SQL skripti, koriste se PHP datoteke koje opisuju tablice, stupce i njihove veze. Pomoću *Laravel*-ovog alata za migracije, *Artisan*-a, migracije se mogu izvršiti i promjene se mogu

primijeniti na bazu podataka. Također, migracije se mogu poništiti i baza podataka se može vratiti na prethodno stanje.

2.3.1 Povezivanje na bazu

Konfiguracijski parametri za povezivanje s bazom podataka u *Laravel*-u nalaze se u *.env* datoteci i može se vidjeti na Kod-u 3.

```
DB_CONNECTION="mysql"
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE="recipeDatabase"
DB_USERNAME="root"
DB_PASSWORD=
```

Kod 3. Primjer spajanje na bazu podataka u *Laravel*-u

Izvor: Autor

Ime baze podataka unosi se pod DB_DATABASE. Ako baza već postoji, tablice će biti kreirane unutar nje. Ako baza ne postoji, bit će prikazan upit za kreiranje baze podataka s imenom navedenim u DB_DATABASE. Korisničko ime se postavlja pod DB_USERNAME na *root* ako već nije specificirano, dok se lozinka ostavlja prazna jer se lokalno spaja, čime se izbjegava potreba za unosom lozinke. Konfiguracija vrijedi samo za spajanje na lokalni server, poput XAMPP-a.

2.3.2 Način kreiranja tablice

Kroz terminal kreiraju se migracije pomoću naredbe koja je prikazana u Kod 4. dobije se prazna tablica koja sadrži samo *id* tablice i vremensku oznaku. Zatim su, prema prethodno

definiranoj strukturi tablice, ispunjeni potrebni atributi za tablicu *users*. Za korisničko ime i email je postavljeno da moraju biti unikatni kako bi se spriječilo da se korisnici s istim korisničkim imenom ili email-om registriraju. Broj pratitelja i broj ljudi koje korisnik prati, postavljeni su kao opcionalni jer na početku registracije korisnik nema pratitelja i ne prati nikoga. Zatim je napravljena referenca na tablicu *roles* kako bi se za svakog korisnika mogla postaviti određena uloga. Na kraju, vremenska oznaka označena kao *timestamp* automatski kreira atribute *created_at* i *updated_at* na tablici.

```
php artisan make: migration naziv_migracije
```

Kod 4. Kreacija Tablica u Laravel-u

Izvor: Autor

Slika 3. Primjer kreiranja tablice u Laravel-u

```
public function up(): void
    {
        Schema::create('users', function (Blueprint $table) {
            $table->id();
            $table->string('username')->unique();
            $table->string('email')->unique();
            $table->string('password');
            $table->string('imagePath')->nullable();
            $table->float("number_of_followers")->nullable()->default(0);
            $table->float("number_of_following")->nullable()->default(0);
            $table->unsignedBigInteger('role_id');
            $table->foreign('role_id')->references('id')->on('role');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('users');
    }
```

Izvor: Autor

3. Izrada stranice

3.1 Modeli

Kada su sve potrebne tablice već kreirane u bazi podataka, modeli za sve tablice trebaju biti kreirani kako bi se omogućilo upisivanje, čitanje i brisanje podataka iz baze pomoću

Eloquent ORM-a. Kreacija modela izvršava se pomoću terminala kako bi se modeli implementirali. Naredbom prikazanom na Kod 5. model se kreira i sprema u direktorij *App/Models*.

```
php artisan create model imeModela
```

Kod 5. Kreacija Modela

Izvor: Autor

Opis modela

Model *BlockedUser*

Pomoću naredbe prikazane na Kod 6. dobije se prazna klasa koja može biti uređena prema potrebama.

```
php artisan create model BlockedUser
```

Kod 6. Kreacija BlockedUser Modela

Izvor: Autor

Slika 4. Model *BlockedUser*

```
class BlockedUser extends Model
{
    use HasFactory;
    protected $table = 'blocked_users';
    protected $fillable = [
        'user_that_blocked_id',
        'blocked_user_id'
    ];
    public function userThatBlocked(){
        return $this->belongsTo(User::class);
    }
    public function blockedUser(){
        return $this->belongsTo(User::class);
    }
}
```

Izvor: Autor

Slika 4. prikazuje kod za BlockedUser model, gdje je potrebno definirati naziv tablice kako bi Laravel mogao pronaći tablicu. Nadalje, potrebno je odrediti koje atribute tablice je moguće masovno popuniti, u ovom slučaju to su id-evi od user_that_blocked_id i blocked_user_id. Na kraju, postoje dvije funkcije koje uspostavljaju relaciju između User i BlockedUser, dozvoljavaju pristup tom korisniku.

Model Category

Slika 5. prikazuje kod za model kategorije. Definira se ime tablice i određuje koji atribut se može masovno popuniti. Funkcija recipes() uspostavlja relaciju "ima mnogo" između Recipe i Category klasa, što daje pristup svim receptima u određenoj kategoriji. Slično, model Role sadrži atribut roleName, ali ne sadrži funkciju za dohvaćanje korisnika.

Slika 5. Model *Category*

```
class Category extends Model
{
    use HasFactory;
    protected $table = 'categories';
    protected $fillable=["CategoryName"];
    public function recipes(){
        return $this->hasMany(Recipe::class);
    }
}
```

Izvor: Autor

Model Comments

Slika 6. prikazuje kod za definiranje modela Comments, specifikacije imena tablice i atributa za masovno punjenje su određene. Dvije funkcije pružaju informacije o korisniku i receptu na kojem je korisnik ostavio komentar, što omogućava pronalaženje autora komentara. Slični modeli ovom modelu su Ratings, koji umjesto komentara ima atribut za ocjenu, a ostali atributi su isti. Favorite_Recipe, koji ima samo user_id i recipe_id kao punjive atribute, s funkcijama za dohvaćanje identičnim ovima. Posljednji model koji je sličan je Personal_Recipe modelu je Favorite_Recipe model.

Slika 6. Model *Comments*

```
class Comments extends Model
{
    use HasFactory;
    protected $table = 'comments';
    protected $fillable=["comment", "user_id", "recipe_id"];
    public function user(){
        return $this->belongsTo(User::class);
    }
    public function recipe(){
        return $this->belongsTo(Recipe::class);
    }
}
```

Izvor: Autor

Model Follower

Slika 7. prikazuje model Follower atributi *user_id* i *follower_id* definirani su kao punjivi. Funkcije vraćaju korisnika u oba slučaja, ali ovisno o kontekstu, može se dobiti pratitelj ili osoba koju netko prati.

Slika 7. Model *Follower*

```
class Follower extends Model
{
    use HasFactory;
    protected $table="followers";
    protected $fillable=[
        "user_id",
        "follower_id"
    ];
    public function user(){
        return $this->belongsTo(User::class);
    }
    public function follower(){
        return $this->belongsTo(User::class);
    }
}
```

Izvor: Autor

Model User

Slika 8. prikazuje model User koji sadrži više atributa koje treba definirati kako bi se omogućilo upisivanje korisnika. Ti atributi uključuju korisničko ime, email, lozinku, putanju do slike, ID uloge iz tablice role, te broj pratitelja i broj korisnika koje korisnik prati. U modelu se nalazi jedna funkcija koja vraća ulogu kojoj korisnik pripada.

Slika 8. Model *User*

```
class User extends Authenticatable
{
    use HasFactory, Notifiable;


    protected $table="users";
    protected $fillable = [
        'username',
        'email',
        'password',
        "imagePath",
        "role_id",
        "number_of_followers",
        "number_of_following",
    ];
    public function role()
    {
        return $this->belongsTo(Role::class);
    }
}
```

Izvor: Autor

Model Recipe

Slika 9. prikazuje kod za definiranje *Recipe* model. Atributi uključuju naslov recepta, sastojke, instrukcije, sliku recepta, vrijeme kuhanja, broj porcija, težinu (koja nema posebnu tablicu) i ID kategorije. Osim toga, model sadrži funkciju koja vraća kategoriju kojoj određeni recept pripada.

Slika 9. Model *Recipe*

```
class Recipe extends Model
{
    use HasFactory;
    protected $table = 'recipes';
    protected $fillable=[
        "title",
        "ingredients",
        "instructions",
        "imagePath",
        "cooking_time",
        "number_of_servings",
        "difficulty",
        "category_id",

    ];
    public function category(){
        return $this->belongsTo(Category::class);
    }
}
```

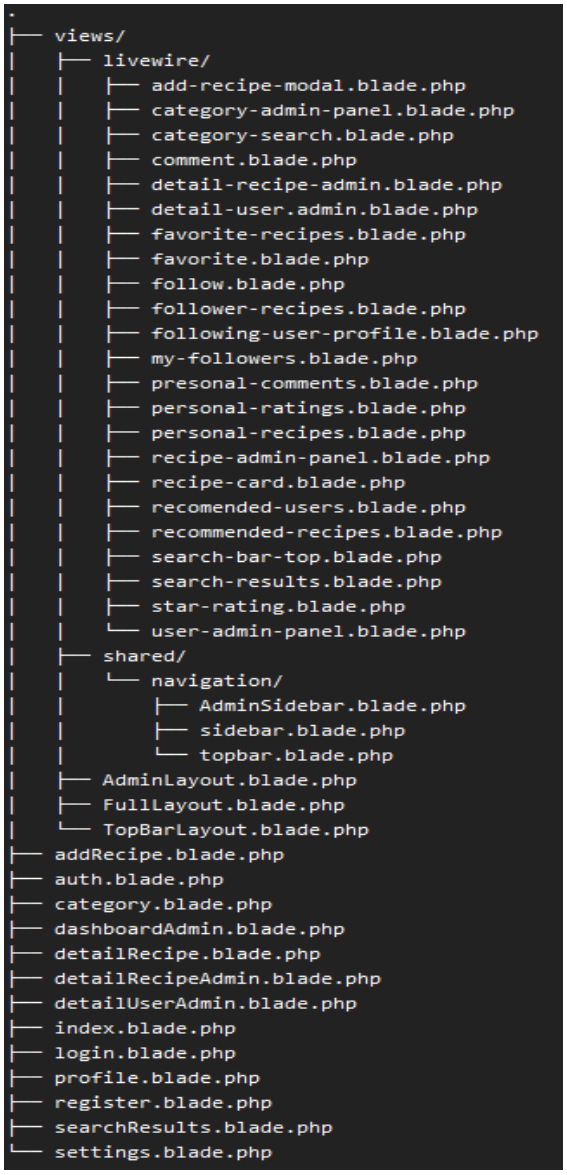
Izvor: Autor

3.2 Pogledi

3.2.1 Struktura direktorija pogleda

Slika prikazuje svaki pogled koji aplikacija ima, pri čemu mapa *shared* sadrži sve zajedničke poglede koji se koriste u ostalim pogledima. Također, u mapi *Livewire* se smještaju sve komponente koje su izrađene pomoću *Livewire*-a.

Slika 10. Struktura pogleda u aplikaciji



Izvor: Autor

3.2.2 Kreiranje pogleda i implementacija

Kreacija

Kreiranje dinamičkih web aplikacija u *Laravel*-u zahtijeva razumijevanje korištenja pogleda i *Blade*-a. Struktura i prezentacija aplikacije definiraju se kroz poglede, dok se

Blade koristi za uključivanje PHP logike unutar HTML-a. Pogledi se nalaze u direktoriju *resources/views*. Novi pogledi mogu biti kreirani ručno u tom direktoriju ili se može koristiti *Laravel*-ov CLI alat, *Artisan*, za generiranje novih pogleda. Pogledi mogu biti naslijeđeni kako bi se omogućila ponovna upotreba koda i definiranje zajedničkih dijelova sučelja. Važno je napomenuti da pogledi uvijek završavaju s *.blade.php* ekstenzijom jer koriste *Blade*. Za nasljeđivanje i uključivanje drugih stranica koriste se *@yield*, *@extends* i *@section* direktive kako bi se uspješno uključio pogled u druge poglede.

Slika 11. Primjer direktive *yield*

```
<body>
    <div class="container">
        @yield('content')
    </div>
</body>
```

Izvor: https://dcblog.dev/creating-flexible-layouts-in-laravel-with-yields-includes-and-slots (5.6.2024.)

Slika 12. Primjer direktive *extends* i *section*

```
@extends('layouts.app')

@section('title', 'Page Title')

@section('content')
    <p>This is the content of the page.</p>
@endsection
```

Izvor: https://dcblog.dev/creating-flexible-layouts-in-laravel-with-yields-includes-and-slots (5.6.2024.)

Slike 11. i 12. prikazuju upotrebu direktiva @yield, @section i @extends. Na slici 11. je prikazan glavni pogled, dok je na slici 12. prikazan sporedni pogled koji se uklapa u glavnu stranicu. Naziv @yield direktive u glavnoj stranici mora se poklapati s nazivom @section direktive. Nadalje, potrebno je odrediti iz kojeg pogleda se proširuje, pri čemu se u ovom slučaju glavni pogled naziva *layout.app*.

Implementacija u aplikaciji

Aplikacija sadrži tri glavna pogleda s različitim dizajnom: *TopBarLayout*, *FullLayout* i *AdminLayout*. Iz tih pogleda, svi ostali pogledi se uklapaju, što olakšava izbjegavanje ponavljanja istog koda za svaku stranicu. U ovom isječku koda prikazan je način na koji je implementirano uključivanje pogleda. Koristi se @include direktiva, kako bi se uključila navigacija za pogled. Razlika između @include i @yield direktiva je u tome što @include direktno uključuje sadržaj unutar trenutnog pogleda, dok @yield rezervira mjesto unutar pogleda gdje se dinamički uključuje sadržaj. Na slici 13. je prikazan *TopBarLayout* pogled koji uključuje samo gornji dio navigacije, dok je na slici 14. prikazan pogled *Auth* koji sadrži forme za prijavu i registraciju korisnika.

Slika 13. Isječak koda zajedničkog pogleda

```
<body class="css-selector">
    <div class="vh-100 d-flex flex-column">
        <div class="container-fluid">
            <div class="row fixed-top">
                @include('shared.navigation.topbar')
            </div>
            <div class="mt-5">
            </div>
            <br>
            <div class="container-fluid" id="">
                <div class="">
                    @yield('contentHalf')
                </div>
            </div>
        </div>
</body>
```

Izvor: Autor

Slika 14. Primjer koda za autentifikaciju

```
@extends('shared.TopBarLayout')
@section('contentHalf')
    <script> ···
    </script>

    <div class="row">
        <div class="d-flex justify-content-center mt-4">
            <div class="btn text-white prijavaButton" id="PrijavaBtn" onclick="Login()">
                Prijava
            </div>
            <div class="btn text-white registerButton " id="RegisterBtn" onclick="Register()">
                Registracija
            </div>
        </div>
    </div>
    <div class="">
        @include('login')
    </div>
    <div>
        @include('register')
    </div>
    @if (session()->has('regError'))
        <script>
            Register();
        </script>
    @else
        <script>
            Login();
        </script>
    @endif
@endsection
```

Izvor: Autor

3.3 *Livewire* komponente

3.3.1 Kreacija komponenti

Svaki pogled sadrži jednu *Livewire* komponentu kako bi se osvježavao samo dio koda na stranici, a ne cijela stranicu kod promjena. Kreacija *Liwewire* komponente se izvršava pomoću terminala i sljedeće naredbe prikazane na Kod 7. Nakon izvršavanja naredbe, stvaraju se dvije datoteke. Prva je klasa kojom se upravlja podacima, dok je druga datoteka pogled kojim se vizualno predstavljaju podaci. Datoteka sadrži klasu, a nalazi se u direktoriju *app/livewire* i sadrži samo jednu metodu, *render*, koja vraća novostvoreni pogled. Na slici 15. se vidi novo stvorena klasa koja vraća pogled, dok se lokacija tog pogleda nalazi u direktoriju *resources/views/livewire*. Taj pogled sadrži jedan div element u kojem se piše kod za prikazivanje podataka iz klase.

```
php artisan make: livewire ImeKomponente
```

Kod 7. Kreacija Livewire komponente

Izvor: Autor

Slika 15. *Livewire* klasa

```php
<?php

namespace App\Http\Livewire;

use Livewire\Component;

class MyComponent extends Component
{
    public function render()
    {
        return view('livewire.my-component');
    }
}
```

Izvor: Autor

Slika 16. Uključivanje komponente u pogledu

```
<div>
    <h1>Welcome to My Website</h1>

    <!-- Include the Livewire component -->
    <livewire:my-component />
</div>
```

Izvor: Autor

Na slici 16. je prikazano uključivanje kreirane komponente u pogled na način gdje "my-component" predstavlja ime komponente koje je navedeno tijekom kreacije komponente.

3.3.2 Implementacija u aplikaciji

Slika 17. prikazuje klasu za komponentu nazvanu *RecipeCard* koja je namijenjena za ispis svih recepata koji postoje u bazi podataka. Kako bi se izbjeglo preopterećenje baze podataka prilikom osvježavanja stranice i dohvata svih recepata, korištena je paginacija koja omogućava dohvaćanje samo određenog broja recepata. Korisnik ima mogućnost učitavanja dodatnih recepata po potrebi, što smanjuje opterećenje baze podataka i servera. Na slici su prikazane dvije definirane varijable, pri čemu je prva varijabla povezana s brojem recepata koji će biti dohvaćeni, dok je druga varijabla namijenjena za prikaz učitavanja tih recepata. Metoda *render* dohvaća sve recepte pomoću *Eloquent ORM-a*, gdje kod dohvaćanja recepata se specificira koliko recepata treba uzeti. Nadalje, dohvaćaju se svi osobni recepti kako bi se ispisali korisnici koji su kreirali te recepte, te se obje

varijable šalju pogledu. Na kraju, u metodi *loadMore* dodaje se broj recepata koji treba biti dohvaćen, dok se ne dohvate svi recepti.

Slika 17. Klasa *RecipeCard*

```
class RecipeCard extends Component
{
    use WithPagination;

    3 references
    public $items = 12;
    1 reference
    public $loading=true;
    0 references | 0 overrides
    public function render()
    {
        $recipes = Recipe::paginate($this->items);
        $personalRecipes=Personal_Recipe::all();
        if($recipes->count() < $this->items){
            $this->loading = false;
        }

        return view('livewire.recipe-card', ['recipes' => $recipes,'personalRecipes'=>$personalRecipes]);
    }
    0 references | 0 overrides
    public function loadMore()
    {
        sleep(1);
        $this->items += 5;
    }
}
```

Izvor: Autor

Slika 18. prikazuje isječak koda za ispis svih recepata i njihovih autora. Varijable koje su poslane u klasi mogu se ovdje koristiti, te pomoću *@foreach* direktive u *Blade*-u ispisuju se svi recepti. Za pristup atributima kao što su naslov recepta, prosječna ocjena i ostali, koristi se "->" operator za pristup objektu, što omogućava ispis svih potrebnih podataka o receptu na vizualno prihvatljiv način. Važno je napomenuti da *Livewire* pogledi i pogledi mogu koristiti iste *Blade* direktive.

Slika 18. *Livewire* pogled

```
@foreach ($recipes as $recipe)
    <div class="col">
        <div class="card cardEffect rounded-5">
            <a href="{{ route('recipe', ['id' => $recipe->id]) }}" class=""><img
                    src="{{ $recipe->imagePath }}" class="card-img-top img-fluid rounded-5"
                    style="height:200px" alt="" /></a>
            <div class="card-body changeColorOnHover rounded-5">
                <h5 class="card-title fs-4" style="color: white">{{ $recipe->title }}</h5>
                <p class="card-text">
                    @for ($i = 1; $i <= 5; $i++)
                        @if ($i <= $recipe->average_rating)
                            <span class="star" href="#"><i class="fa-solid fa-star fa-xl"
                                    style="color: #FFC72C;"></i></span>
                        @else
                            <span class="star" href="#"><i class="fa-regular fa-star fa-xl"
                                    style="color: #FFC72C;"></i></span>
                        @endif
                    @endfor

                </p>
                @foreach ($personalRecipes as $ps)
                    @if ($ps->recipe_id == $recipe->id)
                        <div>
                            <img src="{{ $ps->user->imagePath }}" alt="" width="50"
                                height="50">
                            <a class="btn btn-link text-decoration-none" style="color: #FFC72C " href="{{route("profile",["id"=>$p
                        </div>
                    @endif
                @endforeach

            </div>
        </div>
    </div>
@endforeach
```

Izvor: Autor

3.4 Kontroleri

3.4.1 Izrada kontrolera

Pomoću kontrolera komunicira se između modela i pogleda, a oni omogućuju prikazivanje podataka i pogleda. Za kreiranje kontrolera u *Laravel* okviru, u terminalu mora se upisati naredba prikazana u Kod 8, gdje je konvencija završiti ime s nazivom *Controller*. Nakon izvršavanja naredbe, kreira se datoteka na lokaciji *app/http/Controllers* gdje su locirani svi kontroleri. Nakon

kreacije klasa kontrolera je prazna, odnosno ne sadrži nikakve metode. Metodu index se napiše kako bi mogla vraćati pogled index kao što je prikazano na slici 19.

```
php artisan make: controller TestController
```

Kod 8. Kreacija Kontrolera u Laravel-u

Izvor: Autor

Slika 19. Metoda Kontrolera

```
public function index()
    {
        return view('test.index');
    }
```

Izvor: Autor

3.4.2 Implementacija kontrolera u aplikaciji

Aplikacija sadrži tri kontrolera: *AuthController*, koji je zadužen za korisnike, *RecipeController*, koji se bavi receptima, te *IndexController*, koji vraća index pogled kao početni pogled aplikacije koji korisnici vide. Kontroleri se koriste za operacije poput prijave, odjave, dodavanja, brisanja, ažuriranja i validacije podataka. Slika 20. prikazuje implementaciju validacije podataka prilikom registracije korisnika. Validacija provjerava različite uvjete koji su određeni, kao što su *required* koji označava obavezna polja, te *mimes* koji određuje tipove prihvaćenih datoteka. Ako neki od tih uvjeta nije zadovoljen, greška specificirana u drugom dijelu bit će vraćena od strane aplikacije.

Slika 20. Validacija podataka kod registracije

```
public function register(Request $request)
{
    session()->flash("regError", true);

    $request->validate([
        "file" => "image|mimes:jpg,jpeg,png|max:2048",
        "username" => "required|min:4|max:50",
        "email" => "required|email",
        "password" => "required|min:6|max:25",
        "passwordR" => "required|same:password"
    ], [
        "file.image" => "Morate unijeti sliku",
        "file.mimes" => "Slika mora biti u formatu jpg,jpeg,png",
        "file.max" => "Slika ne smije biti veća od 2MB",
        "username.required" => "Korisničko ime je obavezno",
        "username.min" => "Korisničko ime mora imati najmanje 4 karaktera",
        "username.max" => "Korisničko ime može imati najviše 50 karaktera",
        "email.required" => "Email je obavezan",
        "email.email" => "Email nije u dobrom formatu",
        "password.required" => "Lozinka je obavezna",
        "password.min" => "Lozinka mora imati najmanje 6 karaktera",
        "password.max" => "Lozinka može imati najviše 25 karaktera",
        "passwordR.required" => "Ponovljena lozinka je obavezna",
        "passwordR.same" => "Lozinke se ne poklapaju"

    ]);
```

Izvor: Autor

Za upis podataka preko kontrolera nakon validacije korišten je *Eloquent ORM* kako bi se zapisali podaci u bazu. Na slici se može vidjeti primjer korištenja funkcije *insertGetId*, kojom se upisuju podaci, a funkcija vraća *Id* zadnjeg upisanog korisnika, što je potrebno za kasniju upotrebu. Imena stupaca u tablici *users* su navedena na lijevoj strani, dok su podaci koje se upisuju prikazani na desnoj strani. Za kriptiranje lozinke korisnika zbog sigurnosti korištena je funkcija *bcrypt.* Za stupac uloga postavljeno je da svaki registrirani korisnik ima ulogu 1, što znači da nema ovlasti administratora.

Slika 21. Upis korisnika

```php
$lastId = User::insertGetId([
    "username" => $request->username,
    "email" => $request->email,
    "password" => bcrypt($request->password),
    "role_id" => 1,
    "created_at" => date("Y-m-d H:i:s"),
    "updated_at" => date("Y-m-d H:i:s")
]);
```

Izvor: Autor

Slika 22. Prijava Korisnika

```php
$user = User::where("username", $request->loginName)->orWhere("email", $request->loginName)->first();
if ($user) {
    if (Auth::attempt(["username" => $request->loginName, "password" => $request->password]) ||
    Auth::attempt(["email" => $request->loginName, "password" => $request->password])) {
        session()->flash("login", true);
        return redirect()->route("index");
    } else {
        $errors = "Pogrešna lozinka";
        return redirect()->back()->withErrors($errors)->withInput(["loginName" => $request->loginName]);
    }
} else {
    $errors = "Korisnik ne postoji";
    return redirect()->back()->withErrors($errors)->withInput(["loginName" => $request->loginName]);
}
```

Izvor: Autor

Slika 22. prikazuje način prijave korisnika, gdje prije ovog dijela koda je validacija podataka koja provjera podatke koje je korisnik unio. Ako je validacija uspješna, provjerava se postoji li korisnik s istim korisničkim imenom ili e-poštom, a u slučaju da ne postoji, vraća se greška. Ako korisnik

postoji, pokušava se prijaviti korisnika pomoću klase *Auth*, koja je ugrađena u osnovni *Laravel* paket, korištenjem funkcija *attempt*. Ako funkcija uspješno prijavi korisnika, korisnik se preusmjerava na naslovnu stranicu s porukom o uspješnoj prijavi. U slučaju krive lozinke, vraća se greška korisniku.

3.5 Rute

3.5.1 Izrada ruta

Rute koje omogućuju pristup određenim pogledima ili akcijama definiraju se u datotekama *routes/web.php.* Način definiranja ruta se može vidjeti u Kod 9. primjeru. Iz primjera funkcija sadrži nekoliko parametara. Prvi parametar definira URL putanju preko koje će ta ruta pozvati funkciju u kontroleru. Zatim, drugi parametar je polje koje sadrži ime kontrolera i ime funkcije koja će biti pozvana. Osim funkcije *get*, postoje i druge kao što su *post*, *put*, *delete* i mnoge druge.

```
Route::get('/', [HomeController::class, 'index']);
```

Kod 9. Kreacija ruta

Izvor: Autor

3.5.2 Implementacija ruta

Slika 23. prikazuje nekoliko ruta i način implementacije istih. Nadalje nalazi se ruta index koja se poziva u korijenu aplikacije, odnosno na početnoj stranici aplikacije. Zatim su određeni kontroler i ime funkcije koja će biti izvršena kada korisnik posjeti tu stranicu. Nakon toga, određeno je ime te rute kako bi se mogla pozvati unutar pogleda, što je vidljivo na slici 24.

Slika 23. Primjer implementacije rute u aplikaciji

```
Route::get("/",[IndexController::class,"index"])->name("index");
Route::get("/auth",[AuthController::class,"auth"])->name("auth")->middleware("guest");
```

Izvor: Autor

Slika 24. Pozivanje rute unutar pogleda

```
<a href="{{ route('index') }}"><img src="/Slike/icons8-recipe-96.png" alt="" width="60"
    height="60"></a>
```

Izvor: Autor

3.6 *Middleware*

3.6.1 Kreacija *middleware*

Middleware u *Laravel*-u predstavlja sloj između HTTP zahtjeva i aplikacije. Omogućuju filtriranje i manipulaciju zahtjevima prije nego što dođu do rute. Aplikacija koristi *middleware* za provjeru administrativnih dozvola korisnika radi pristupa određenim stranicama. *Middleware* se kreira pomoću naredbe napisane u Kod 10. Nakon izvršavanja naredbe, unutar dobivene funkcije upiše se kod koji provjerava da li korisnik ima ulogu administratora. Ako korisnik nema potrebne dozvole, bit će vraćen na naslovnicu. Ako dozvole postoje, pristup željenoj stranici bit će omogućen. Implementacija je prikazana na slici 25. radi jasnijeg uvida.

```
php artisan make:middleware CheckAdminRole
```

Kod 10. Kreacija Middleware-a

Izvor: Autor

Slika 25. Middleware za provjeru uloge korisnika

```
public function handle(Request $request, Closure $next): Response
 {
     if($request->user()->role->roleName=="admin"){
         return $next($request);
     }
     return redirect()->route("index");

 }
```

Izvor: Autor

Middleware-a je potrebno registrirati u *Kernel* datoteci, koja se nalazi u *Http/Kernel.php*, pod $*middlewareAliases*. Na slici dolje se može vidjeti gdje je taj *middleware* definiran. Dodijeljen mu je alias *admin* radi mogućnosti poziva u rutama. Ostali *middleware*-i prikazani dolaze s osnovnim paketom *Laravel*-a. Aplikacija koristi *auth*, koji osigurava da je korisnik prijavljen, i *guest*, koji osigurava da korisnik nije prijavljen.

Slika 26. Registracija *Middleware*-a

```
protected $middlewareAliases = [
    "admin"=> \App\Http\Middleware\CheckAdminRole::class,
    'auth' => \App\Http\Middleware\Authenticate::class,
    'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,
    'auth.session' => \Illuminate\Session\Middleware\AuthenticateSession::class,
    'cache.headers' => \Illuminate\Http\Middleware\SetCacheHeaders::class,
    'can' => \Illuminate\Auth\Middleware\Authorize::class,
    'guest' => \App\Http\Middleware\RedirectIfAuthenticated::class,
    'password.confirm' => \Illuminate\Auth\Middleware\RequirePassword::class,
    'precognitive' => \Illuminate\Foundation\Http\Middleware\HandlePrecognitiveRequests::class,
    'signed' => \App\Http\Middleware\ValidateSignature::class,
    'throttle' => \Illuminate\Routing\Middleware\ThrottleRequests::class,
    'verified' => \Illuminate\Auth\Middleware\EnsureEmailIsVerified::class,
];
```

Izvor: Autor

3.6.2 Implementacija sa rutama

Middleware se kreira na ruti dodavanjem istoimene funkcije na kraj rute i upisivanjem imena koje je registrirano. Kao što je moguće vidjeti na slici 27., primjer implementacije *middleware*-a u aplikaciji je prikazan. Važno je napomenuti da rute mogu istovremeno imati više *middleware*-a.

To omogućuje ograničavanje pristupa određenim rutama samo na nekoliko korisnika radi osiguranja od neautoriziranog pristupa.

Slika 27. Implementacija *middleware*-a

```
Route::get("/detailUser/{id}",[AuthController::class,"detailUser"])->name("detailUser")->middleware("auth","admin");
Route::get("/detailRecipe/{id}",[RecipeController::class,"detailRecipe"])->name("detailRecipe")->middleware("auth","admin");
Route::get("/deleteUser/{id}",[AuthController::class,"deleteUser"])->name("deleteUser")->middleware("admin");
```

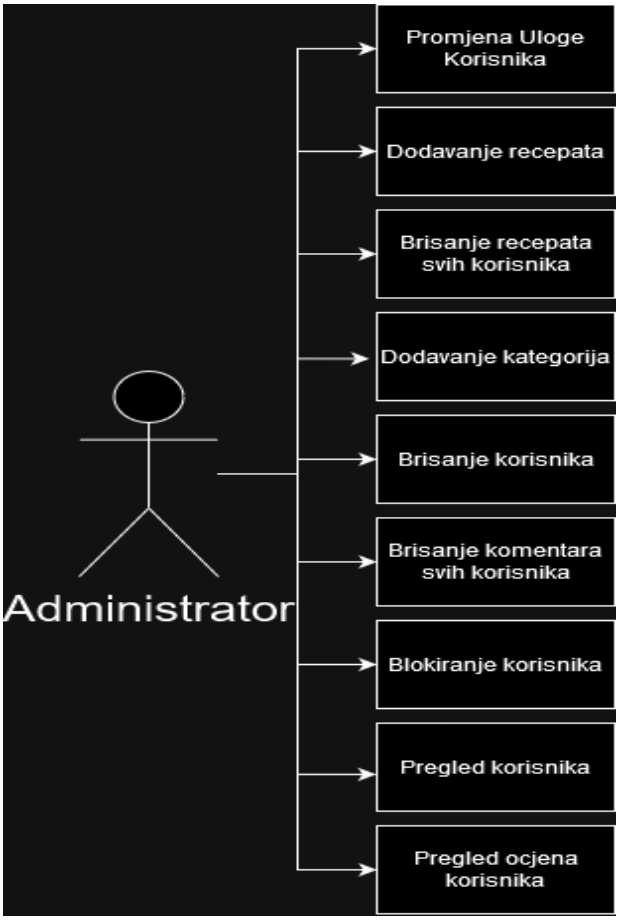
Izvor: Autor

4. Slučajevi upotrebe

4.1 Administrator

Dijagram na slici 28. prikazuje ovlasti administrator na stranici. Administrator ima mogućnost promijeniti ulogu korisnika primjerice iz uloge *users* u ulogu *admin* ili obrnuto. Shodno tome, može dodati recept na dva načina, prvi je sa naslovne stranice kao i svi ostali korisnici, dok je drugi način dodavanje kroz upravljačku ploču. Administrator ima uvid u sve recepte stranice i dodatne detalje o receptu, kao što su ocjene i komentari korisnika, osim toga ima mogućnost obrisati recept. Istodobno ima slične mogućnosti kod upravljanja korisnicima. Na kraju ima mogućnost upravljanja kategorijama stranice kako bi mogao dodavati, brisati ili uređivati kategorije stranice, kao i sve ostale mogućnosti registriranog korisnika.

Slika 28. Mogućnosti Administratora

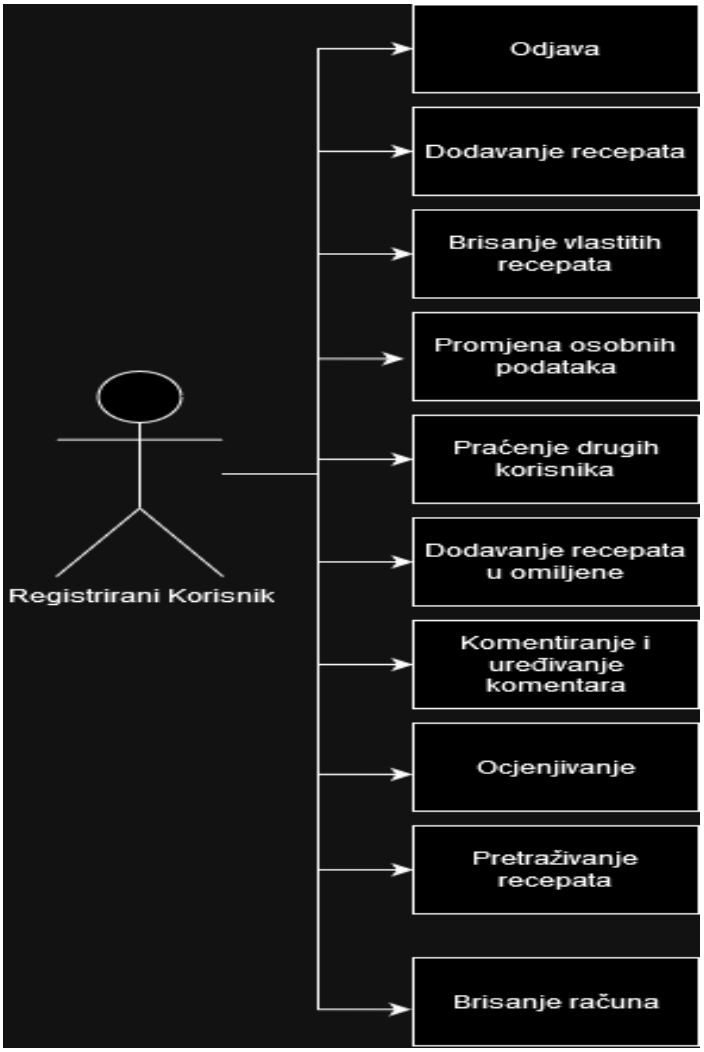


Izvor: Autor

4.2 Registrirani korisnik

Dijagram na slici 29. prikazuje mogućnosti registriranog i prijavljenog korisnika. Korisnik može dodavati i brisati vlastite recepte. Nadalje, može promijeniti osobne podatke to se odnosi na promjenu lozinke, korisničkog imena, email-a i korisničke slike. Može početi pratiti druge korisnike tako i može prestati pratiti druge korisnike. Korisnik može dodati recepte u omiljene, komentirati na recepte te ocjenjivati recepte. Komentare može uređivati i brisati. Naposljetku može pretraživati sve recepte pomoću trake za pretraživanje i obrisati račun kao i ostale mogućnosti ne registriranog korisnika.

Slika 29. Mogućnosti Registriranog Korisnika

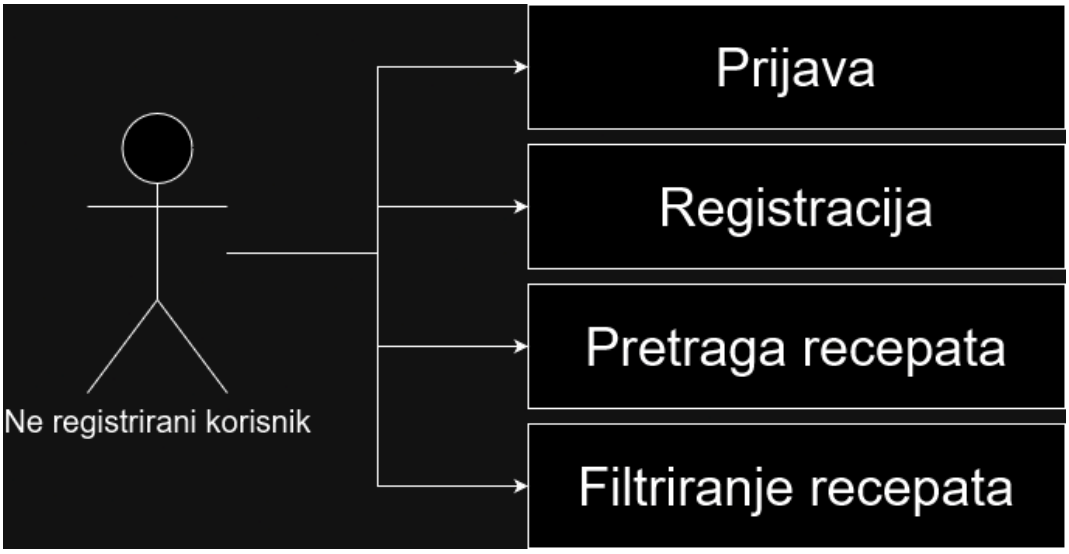


Izvor: Autor

4.3 Neregistrirani korisnik

Dijagram na slici 30. prikazuje mogućnosti ne registriranog korisnika. Ne registriran korisnik može pretraživati sve recepte i filtrirati ih po kategorijama i ostalim atributima. Naposljetku ima mogućnost prijave i registracije.

Slika 30. Mogućnosti ne registriranog korisnika

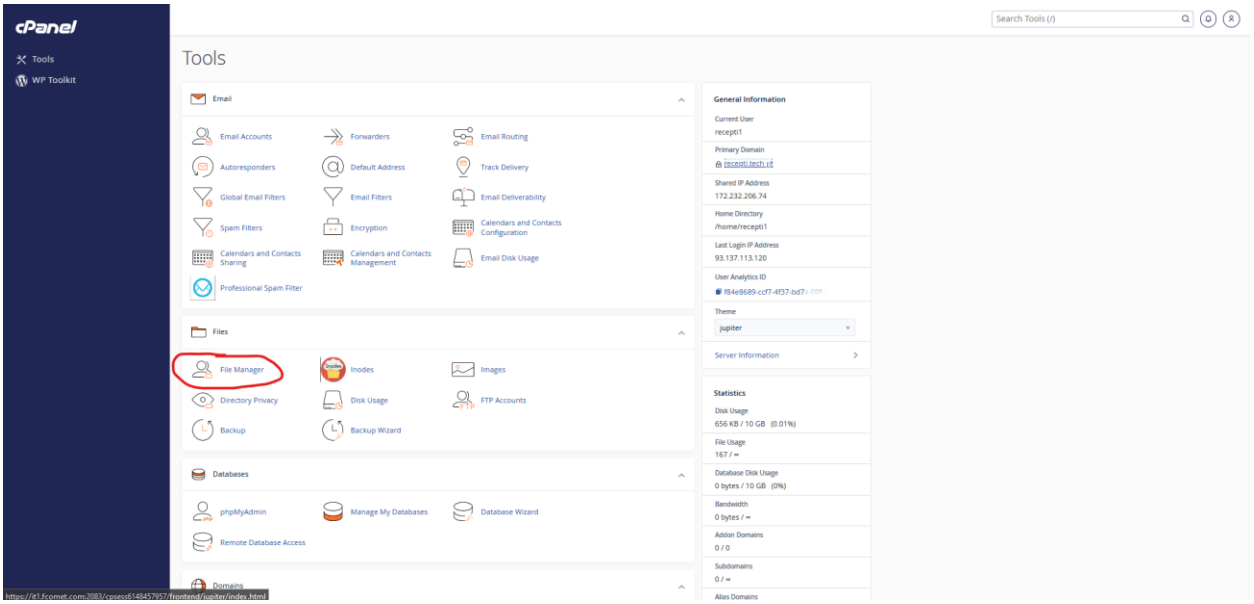


Izvor: Autor

5. Spajanje na udaljeni server

Za spajanje na server potrebne su usluge poslužitelja, korištene su usluge *FastComet* gdje je dobivena domena *https://www.recepti.tech*. Početno je otvoren *Cpanel* i *File Manageru* radi prebacivanja projekta na server. Cijeli *Laravel* projekt je komprimiran u zip datoteku kako se može učitati na stranicu. Nakon toga, zip datoteka se izdvaja u *root* direktoriju, a mapa *public* će biti preimenovana u *public_html* kako bi server prepoznao iz kojeg direktorija treba dohvatiti stranicu.

Slika 31. Izgled C-Panel-a



Izvor: Autor

Zatim je potrebno kreirati bazu podataka i korisnika preko kojeg će se pristupati toj bazi. Nakon kreacije baze na serveru treba izvesti bazu iz lokalnog servera za prebacivanje baze sa podacima i tablicama. Nakon prebacivanja potrebno je promijeniti naziv baze u *.env* datoteci. Nakon čega je stranica spremna za korištenje.

Slika 32. Spajanje baze na serveru

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=recepti1_bazarecepti
DB_USERNAME=recepti1_user1
DB_PASSWORD=K-yxFLN]RN*0
```

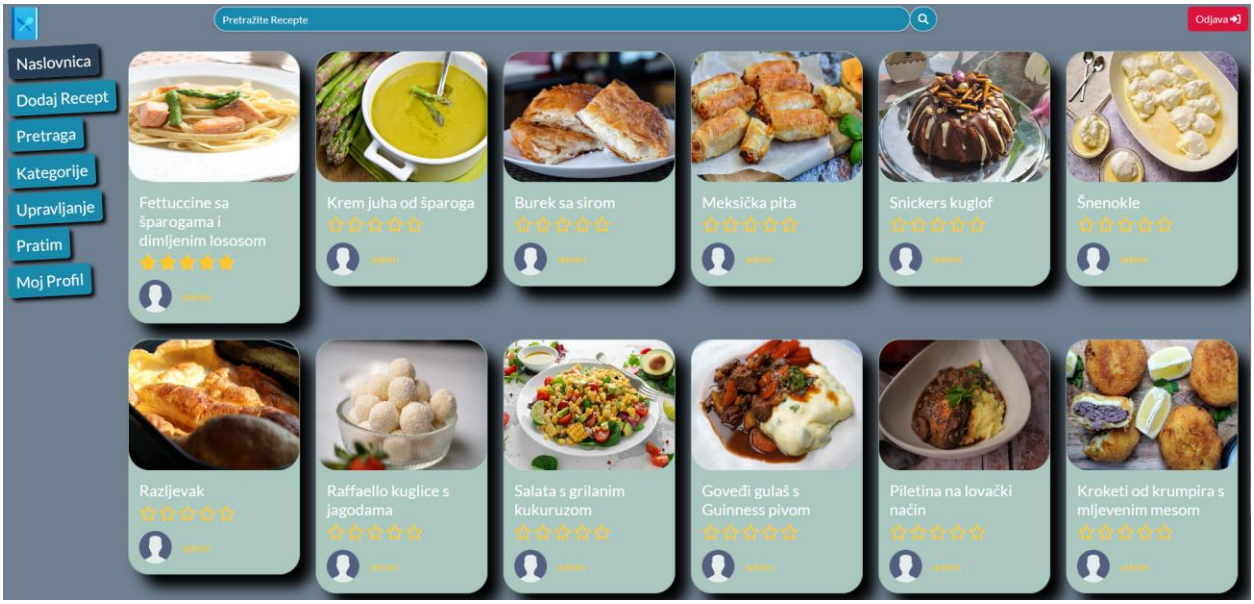
Izvor: Autor

6. Izgled stranice

6.1 Naslovnica

Slika 33. prikazuje gotovu naslovnicu aplikacije, gdje korisnici mogu pregledavati i pretraživati sve recepte. Sa lijeve strane na slici se može vidjeti navigacija pomoću koje korisnici mogu dodavati recepte, provjeriti recepte korisnika koje prate i posjetiti vlastiti profil.

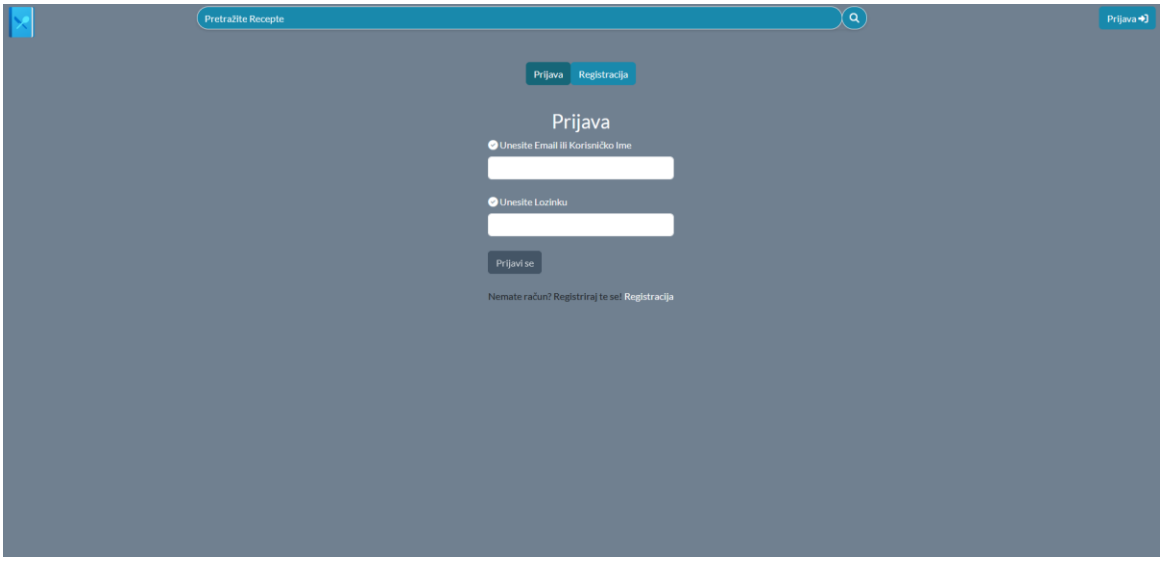
Slika 33. Naslovnica Stranice



Izvor: Autor

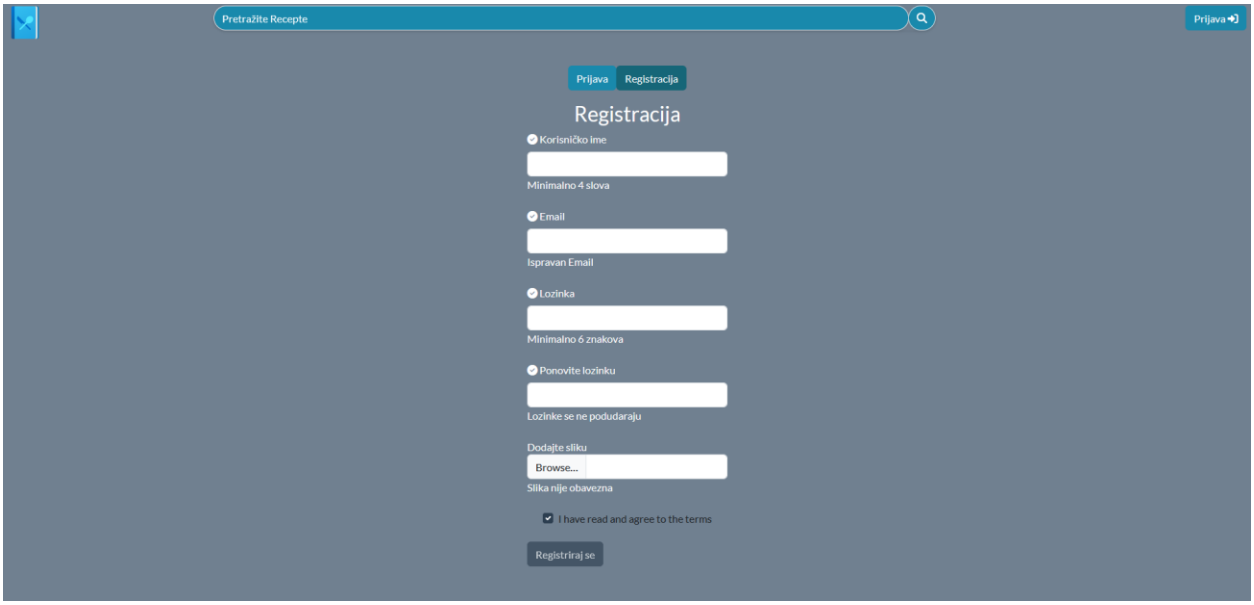
6.2 Prijava i Registracija

Slika 34. Prijava korisnika



Izvor: Autor

Slika 35. Registracija korisnika



Izvor: Autor

6.3 Pretraga

Slika 36. prikazuje stranicu za pretragu gdje korisnici mogu pretraživati i filtrirati recepte po različitim kriterijima. Neki od mogućih filtera su vrijeme kuhanja, broj porcija koji funkcioniraju

na način da pretražuju sve recepte sa jednakim ili manjim iznosom. Filtrirani se recepti mogu sortirati kao najnoviji ili najbolji.

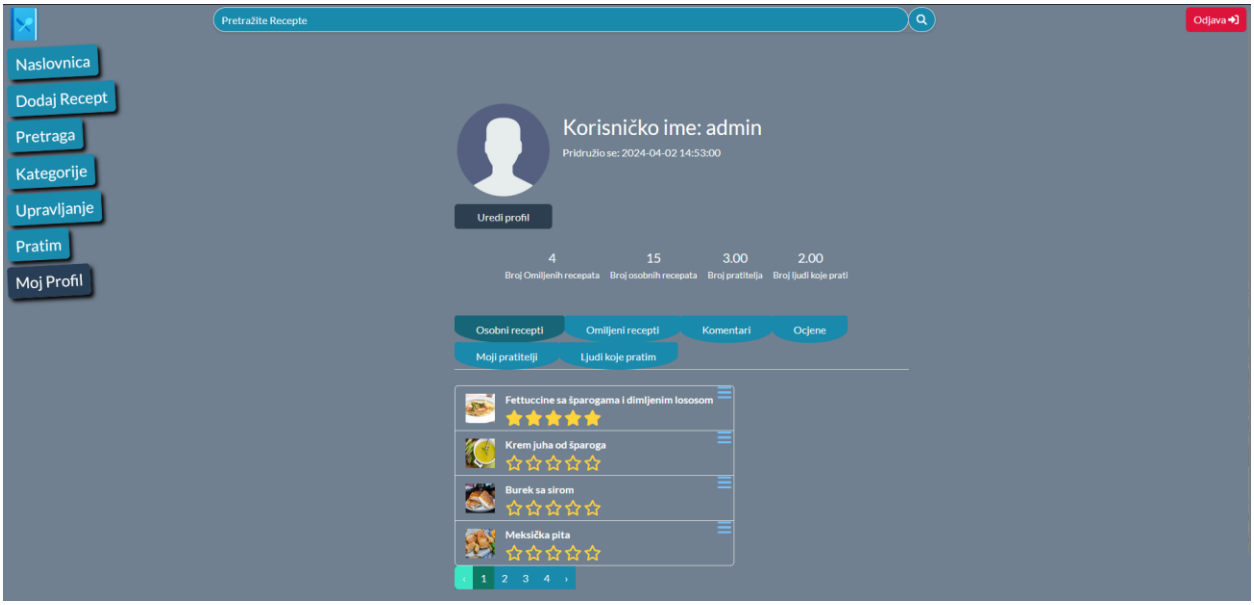
Slika 36. Pretraga recepata

Izvor: Autor

6.4 Profil

Slika 37. prikazuje korisnički profil korisnika *admin*. Kod profila korisnici mogu pregledati ili obrisati sve osobne recepte, omiljene recepte, komentare koje su ostavili, ocjene i pratioce. Nadalje korisnici mogu urediti osobne podatke pritiskom na gumb uredi profil.

Slika 37. Profil korisnika

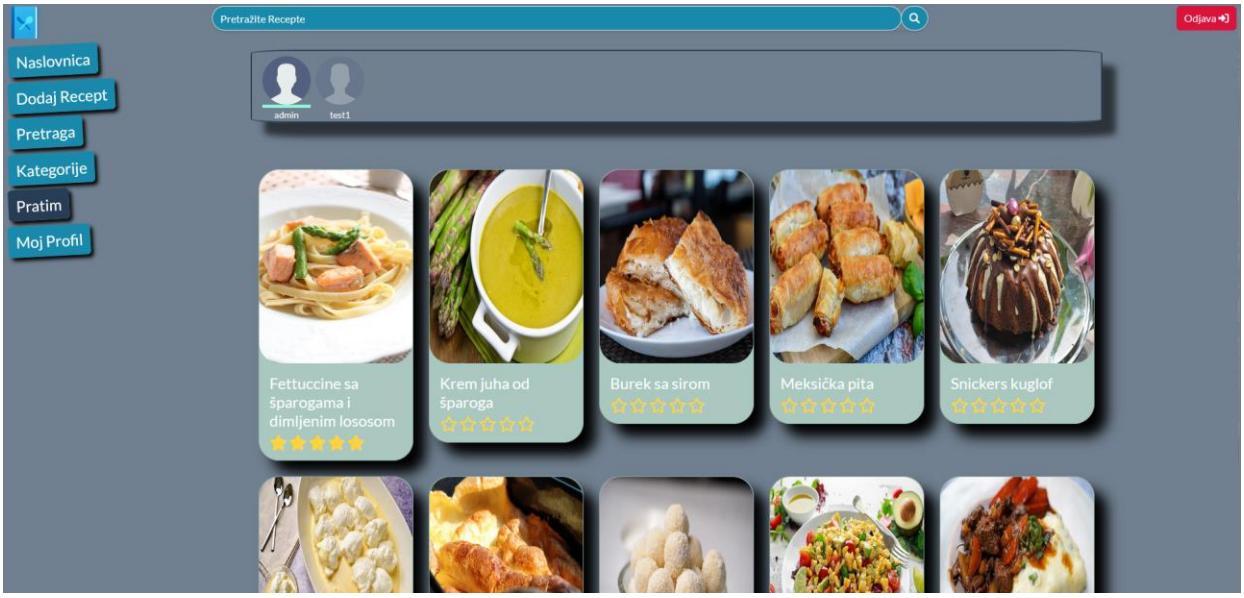


Izvor: Autor

6.5 Pratim

Slika 38. prikazuje stranicu pratim gdje korisnici mogu vidjeti recepte korisnika koje prate. Ukoliko se pritisne na sliku korisnika stranica prikazuje samo recepte tih korisnika.

Slika 38. Stranica pratim



7. Zaključak

Rad istražuje različite aspekte korištenja Laravel-a kao okvira za razvoj web aplikacija s ciljem stvaranja funkcionalnih i atraktivnih korisničkih iskustava. Kod procesa izrade aplikacije

uspostavljena je jasna arhitektura i struktura koja omogućuje upravljanje i održavanje aplikacije. Kako bi aplikacija bila intuitivna za korisnike, analizirani su primjeri sličnih aplikacija za objavu recepata. *Laravel* se ističe u svojoj primjeni zbog MVC arhitekture koja je ključna za organizaciju koda i održavanje projekta. Korištenje *Artisan*-a olakšava upravljanje aplikacijom, dok *Blade* omogućuje dinamičko generiranje HTML-a. *Eloquent ORM* omogućuje interakciju s bazom podataka, dok je korištenje *Middleware*-a pružilo dodatnu sigurnost stranice. *Livewire* komponente aplikaciji dodaju interaktivnost, omogućujući dinamičko osvježavanje dijelova stranice. Kroz analizu procesa izrade aplikacije, potrebno je naročito obratiti pažnju na migracije za održavanje strukture baze podataka te pravilno povezivanje s bazom putem *Laravel*-ovih funkcionalnosti. Razrada izgleda stranice, naslovnica, prijava i registracija, pretraga, profil i praćenje izrađeni su kao ključni dijelovi koji su implementirani kroz strukturirane modele, poglede i kontrolere. Rute su definirane kako bi osigurale pravilno upravljanje zahtjevima korisnika. Slučajevi upotrebe za administratora, registriranog korisnika i neregistriranog korisnika pružaju dublji uvid u funkcionalnosti aplikacije i različite načine interakcije korisnika s istom.

MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU

Bana Josipa Jelačića 22/a, Čakovec

IZJAVA O AUTORSTVU

Završni/diplomski rad isključivo je autorsko djelo studenta te student odgovara za istinitost, izvornost i ispravnost teksta rada. U radu se ne smiju koristiti dijelovi tuđih radova (knjiga, članaka, doktorskih disertacija, magistarskih radova, internetskih i drugih izvora) bez pravilnog citiranja. Dijelovi tuđih radova koji nisu pravilno citirani, smatraju se plagijatom i nezakonitim prisvajanjem tuđeg znanstvenog ili stručnoga rada. Sukladno navedenom studenti su dužni potpisati izjavu o autorstvu rada.

Ja, Tomislav Sedlarević (ime i prezime studenta) pod punom moralnom, materijalnom i kaznenom odgovornošću, izjavljujem da sam isključivi autor/ica završnog/diplomskog rada pod naslovom

Mrežna stranica za recepte izrađena u PHP-u pomoću Laravel programskog okvira

te da u navedenom radu nisu na nedozvoljeni način (bez pravilnog citiranja) korišteni dijelovi tuđih radova.

Student/ica:

Sedlarević T.

(vlastoručni potpis)

8. Popis Slika

9. Popis tablica

10. Popis kodova

11. Bibliografija

[1] [Mrežno]. Available: https://w3techs.com/technologies/comparison/pl-aspnet,pl-php. [Pokušaj pristupa 21. 6. 2024.].

[2] R. Awati. [Mrežno]. Available: https://www.techtarget.com/searchbusinessanalytics/definition/Google-Analytics. [Pokušaj pristupa 21. 6. 2024.].

[3] 10 May 2024. [Mrežno]. Available: https://www.geeksforgeeks.org/php-tutorial/. [Pokušaj pristupa 15 3 2024].

[4] [Mrežno]. Available: https://en.wikipedia.org/wiki/Laravel. [Pokušaj pristupa 20. 2. 2024.].

[5] [Mrežno]. Available: https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller. [Pokušaj pristupa 25. 2. 2024.].

[6] [Mrežno]. Available: https://laravel.com/docs/11.x/artisan#tinker. [Pokušaj pristupa 2. 3. 2024.].

[7] [Mrežno]. Available: https://laravel.com/docs/11.x/blade. [Pokušaj pristupa 3. 3. 2024.].

[8] [Mrežno]. Available: https://laravel.com/docs/11.x/eloquent-serialization#main-content. [Pokušaj pristupa 20. 2. 2024.].

[9] [Mrežno]. Available: https://laravel.com/docs/11.x/middleware. [Pokušaj pristupa 21. 2. 2024.].

[10] [Mrežno]. Available: https://medium.com/@developer.olly/an-overview-of-how-livewire-works-85395746d10a. [Pokušaj pristupa 10. 3. 2024.].

[11] [Mrežno]. Available: https://en.wikipedia.org/wiki/XAMPP. [Pokušaj pristupa 10. 3. 2024.].

[12] [Mrežno]. Available: https://www.w3schools.com/MySQL/mysql_intro.asp. [Pokušaj pristupa 3. 3. 2024.].

[13] [Mrežno]. Available: https://en.wikipedia.org/wiki/HTML. [Pokušaj pristupa 2. 3. 2024.].

[14] [Mrežno]. Available: https://en.wikipedia.org/wiki/CSS. [Pokušaj pristupa 3. 3. 2024.].

[15] [Mrežno]. Available: https://www.w3schools.com/js/js_intro.asp. [Pokušaj pristupa 2. 3. 2024.].

[16] [Mrežno]. Available: https://jquery.com/. [Pokušaj pristupa 4. 3. 2024.].

[17] [Mrežno]. Available: https://getbootstrap.com/. [Pokušaj pristupa 1. 3. 2024.].