Aplikacija za pretvorbu i razmjenu slikovnih datoteka

Kristman, Denis

Master's thesis / Diplomski rad

2018

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:607860

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
ELEKTROTEHNIČKI FAKULTET

Sveučilišni studij

APLIKACIJA ZA PRETVORBU I RAZMJENU SLIKOVNIH DATOTEKA

Diplomski rad

Denis Kristman

Osijek, 2018.

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac D1: Obrazac za imenovanje Povjerenstva za obranu diplomskog rada

Osijek, 06.09.2018.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za obranu diplomskog rada

Ime i prezime studenta:	Denis Kristman
Studij, smjer:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D 782 R, 26.09.2017.
OIB studenta:	32166009801
Mentor:	Doc.dr.sc. Mirko Kohler
Sumentor:	
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	Doc.dr.sc. Ivica Lukić
Član Povjerenstva:	Krešimir Vdovjak
Naslov diplomskog rada:	Aplikacija za pretvorbu i razmjenu slikovnih datoteka
Znanstvena grana rada:	**Obradba informacija (zn. polje računarstvo)**
Zadatak diplomskog rada:	Zadatak rada je napraviti aplikaciju za izmjenu slikovnih datoteka između korisnika. Potrebno je da program prihvati bilo koji tip slikovne datoteke (bmp, jpg, ...) i pretvori ga u png format. Na strani primatelja ista aplikacija treba pretvoriti png format u izvorni tip. Također je potrebno napraviti analizu razlika između originalnih tipova podataka.
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 2 razina
Datum prijedloga ocjene mentora:	06.09.2018.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis: Datum:

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

IZJAVA O ORIGINALNOSTI RADA

Osijek, 09.09.2018.

Ime i prezime studenta:	Denis Kristman
Studij:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D 782 R, 26.09.2017.
Ephorus podudaranje [%]:	10

Ovom izjavom izjavljujem da je rad pod nazivom**: Aplikacija za pretvorbu i razmjenu slikovnih datoteka**

izrađen pod vodstvom mentora Doc.dr.sc. Mirko Kohler

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

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

IZJAVA

Ja, Denis Kristman, OIB: 32166009801, student/ica na studiju: Diplomski sveučilišni studij Računarstvo, dajem suglasnost Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek da pohrani i javno objavi moj **diplomski rad**:

Aplikacija za pretvorbu i razmjenu slikovnih datoteka

u javno dostupnom fakultetskom, sveučilišnom i nacionalnom repozitoriju.

Osijek, 19.09.2018.

____

potpis

SADRŽAJ

1. UVOD

U 21. stoljeću teško je zamisliti život bez interneta. Internetom se danas služimo za obavljanje svih osnovnih potreba, tako da danas možemo npr. kupovati preko interneta, plaćati račune, pronaći odgovore na gotovo sva pitanja, razmjenjivati informacije, fotografije itd. Najčešća metoda prijenosa informacija na webu je HTTP protokol (engl. *Hyper Text Transfer Protocol*). Osnovna namjena ovog protokola je omogućavanje objavljivanja i prezentacije HTML (engl. *Hyper Text Markup Language*) dokumenata, tj. web stranica. [1]

Glavna ideja ovog diplomskog rada je prikaz problematike prijenosa slikovnih datoteka između dva računala preko mreže, te će se dati rješenje za izvedbu web aplikacije za prijenos slikovnih datoteka. Posebno treba istaknuti prijenos BMP (eng. *Bitmap*) slikovnih datoteka koje imaju veću veličinu nego PNG (eng. *Portable Network Graphics*) slikovne datoteke, te se zbog toga svaka BMP slikovna datoteka prije slanja drugom korisniku treba pretvoriti u PNG oblik, što pridonosi manjem zauzimanju prostora na serveru, a to je izuzetno bitno u slučaju velikog broja korisnika i slikovnih datoteka, jer time se štedi slobodni prostor. U tu svrhu izrađuje se web aplikacija bazirana na PHP (eng. *Hypertext Preprocessor*) radnom okviru Laravel. Laravel je razvio Taylor Otwell s ciljem razvoja web aplikacija prateći MVC (engl. *Model-View-Controller*) arhitekturu. Takva arhitektura dijeli web aplikaciju u tri osnovne komponente : Model, Pogled(eng. *View*) i Kontroler (eng. *Controller*). Time se omogućuje razdvajanje programiranja obrade podataka od programiranja sučelja i zahtjeva korisnika.

1.1 Zadatak rada

Zadatak rada je napraviti aplikaciju za izmjenu slikovnih datoteka između korisnika. Potrebno je da program prihvati bilo koji tip slikovne datoteke (BMP, JPG, ...) i pretvori ga u PNG (eng. *Portable Network Graphics*) format. Na strani primatelja ista aplikacija treba pretvoriti PNG format u izvorni tip. Također je potrebno napraviti analizu razlika između originalnih tipova podataka.

2. KORIŠTENE TEHNOLOGIJE

Kako bi se aplikacija uspješno izradila koristeći MVC (eng. *Model, View, Controller*) arhitekturu potrebno je koristiti Laravel PHP (eng. *HyperText Preprocessor*) radni okvir (eng. *framework*). Koristeći Laravel postiže se konzistentnija i stabilnija aplikacija te je potrebno imati znanje u PHP skriptnom programskom jeziku jer je on osnova za rad Laravel aplikacije. Radni okvir je nakupina paketa ili modula kojima se omogućuje razvoj mrežnih aplikacija bez potrebe za upravljanjem detaljima niske razine, što pojednostavljeno znači da mrežni programski okvir sadrži ugrađene klase i funkcije koje pojednostavljuju izradu web aplikacije.

Za definiranje strukture aplikacije odnosno web sučelja koristi se HTML (eng. *Hyper Text Markup Language*) jezik, a za definiranje stilova HTML elemenata koristi se stilski jezik CSS (eng. *Cascading Style Sheets*). Za upravljanje navedenim HTML elementima i web preglednikom koristi se skriptni programski jezik JavaScript, te pripadajuća Javascript biblioteka JQuery. U svrhu olakšavanja izrade izgleda web sučelja, uz sve prethodno navedene tehnologije koristi se i Bootstrap radni okvir baziran na HTML, CSS i Javascript tehnologijama.

Sve što je prethodno navedeno odnosi se na izgled i ponašanje web sučelja (eng. *frontend*), odnosno svega onoga s čime se korisnik susreće kada otvori web aplikaciju. Za pozadinu aplikacije, tj. poslovnu logiku aplikacije (eng. *backend*) koristi se PHP skriptni programski jezik. Za spremanje podataka aplikacije koristi se sustav baze podataka MySQL.

Korištene tehnologije:

- HTML5
- CSS3
- Javascript (Jquery)
- Bootstrap 3
- Laravel 5.6 (PHP 7.2, MySQL)

2.1 Html

HTML (eng. *Hyper Text Markup Language*) što u prijevodu znači prezentacijski jezik za izradu web stranica. HTML jezikom oblikuje se sadržaj web stranice te prema tome HTML nije programski jezik. Njime ne možemo izvršiti nikakvu zadaću, pa niti najjednostavnije operacije zbrajanja ili oduzimanja cijelih brojeva. On služi samo za opis hipertekstualnih dokumenata, što mu je i primarna zadaća. Svaki HTML dokument sastoji se od osnovnih građevnih blokova HTML elemenata, dok se svaki element sastoji od para oznaka (eng. *tag*). HTML se prvi puta spominje krajem 1991. godine na internetu od strane Tim-Beerners-Leeja, dok je prva verzija HTML-a objavljena 1993. godine. [2]

```
<!DOCTYPE html>
<html>
<head>
    <title>Web stranica</title>
</head>
<body>
    <h1>Naslov Web stranice</h1>
</body>
</html>
```

Sl. 2.1. Primjer osnovnog HTML dokumenta

Prema slici 2.1. prikazan je primjer izgleda osnovnog HTML dokumenta. Sastoji od <*html*> tagova koji označavaju HTML dokument, unutar *head* sekcije nalazi se naslov dokumenta, dok *body* sekcija označava tijelo dokumenta, odnosno sve ono što se prikazuje u web pregledniku. Element *h1* označava veličinu i naslovni sadržaj dokumenta.

2.2 Css

CSS (eng. *Cascading Style Sheets*) je stilski jezik koji služi za opis prezentacije dokumenta napisanog pomoću HTML jezika. Kako se web razvijao stvorila se i potreba za oblikovanjem HTML elemenata, te prema tome CSS stilski jezik određuje stilove za HTML elemente. Njime se omogućava definiranje načina prikaza teksta ili drugih elemenata u HTML dokumentu, odnosno definira se stil koji će se koristiti. Napravljen je od strane W3C institucije 1996. godine. Glavni razlog nastanka bilo je izbjegavanje velike količine HTML tagova za uređivanje unutar dokumenta, te da bi se HTML dokument mogao orijentirati samo na sadržaj [3].

CSS stilovi mogu se povezati sa HTML elementom na tri načina [4]:

- Unutar jednog HTML elementa (eng. *inline style*)
- Unutar <*head*> elementa (eng. *internal style sheet*)

- U posebnom CSS dokumentu (eng. *external style sheet*)

CSS koristi selektore (eng. *selector*) za označavanje HTML elemenata na koje se odnosi definirani stil. CSS selektor može se odnositi na:

- Svi elementi istog tipa (jednostavni selektor)
- Element definiran klasom (klasni selektor) ili id-em (Id selektor)
- Elementi u odnosu na druge elemente

```
/*

selektor{
    svojstvo: vrijednost;
}

*/

p{
    color: blue;
}

.klasa1{
    font-family: Arial;
}

#id1{
    font-size: 12px;
}
```

Sl. 2.2. Primjer CSS selektora

Prema slici 2.2. prikazan je način definiranja CSS selektora. Unutar /* */ znakova nalazi se komentar koji može pomoći u olakšavanju orijentacije unutar dokumenta. U ovom primjeru komentarom je prikazana sintaksa pisanja CSS selektora. Svakom selektoru dodjeljuje se svojstvo koje se želi stilizirati te njegova vrijednost. Prvi prikazani selektor je jednostavni selektor koji označava sve *p* elemente odnosno sve odlomke na web stranici i dodjeljuje im plavu boju teksta. Drugi selektor je klasni selektor i označava se sa znakom točka „.“ ispred imena klase te u ovom primjeru svim elementima klase „klasa1“ dodjeljuje vrstu teksta „Arial“. Posljednji selektor na slici je Id selektor. Takav selektor odnosi se samo na jedan element u

HTML elementu kojemu je dodjeljen pripadajući Id. Prema tome, jednom elementu sa pripadajućim Id-em dodjeljuje se veličina fonta od 12px.

2.3 Javascript i JQuery

Javascript je skriptni programski jezik koji se izvršava u web pregledniku na strani korisnika, što znači da se Javascript ne mora instalirati na računalo nego je već ugrađen u web preglednik. Izvorno je razvijen od strane Netscape kompanije 1995. godine. Omogućuje izradu dinamičkih web stranica u kombinaciji sa HTML i CSS tehnologijama. Korištenjem Javascript možemo mjenjati sadržaj na stranici ovisno o načinu interakcije korisnika sa stranicom. [5]

Osnovni dijelovi Javascript skripte su sljedeći:

- Varijabla (riječ koja označava neku vrijednost – tekstualnu, brojčanu, logičku ili objektnu)
- Literal (sama vrijednost varijable)
- Objekt (skup međusobno povezanih varijabli)
- Operator (označava operacije nad varijablama)
- Kontrolna struktura (regulira tijek izvođenja skripte, može biti u obliku funkcije ili metode)
- Događaj (služi za detektiranje korisničkih akcija, kao što su pomicanje miša, pritisak tipke i sl.)
- Referenca (veza na objekt ili događaj)

Javascript skripta umeće se u HTML dokument korištenjem *<script>* elementa, te se može uključiti bilo gdje unutar *head* ili *body* sekcija dokumenta, međutim postoje neke preporuke u vezi toga [6]:

- Većina skripti može se umetnuti u *head* sekciju dokumenta čime se njihov sadržaj odvaja od glavnog sadržaja dokumenta
- Ako je potrebno osigurati izvođenje skripte u nekom trenutku prilikom ispisivanja stranice tada ju je potrebno umetnuti u *body* sekciju dokumenta. Ukoliko se radi o većoj skripti onda ju je ipak preporučljivo umetnuti u *head* sekciju unutar neke funkcije, a pozive funkcije umetnuti na željena mjesta unutar *body* sekcije dokumenta.

- Ako se skripta koristi nekoliko puta unutar dokumenta ili je jako velika, tada ju je preporučljivo pohraniti u zasebnu datoteku koja će se učitati unutar *head* sekcije dokumenta. Na taj način ostvaruje se bolja preglednost glavnog sadržaja HTML dokumenta, te se tako korištena skripta može upotrebljavati na više stranica, što uz korištenje privremene memorije web preglednika pridonosi na uštedi količine prenesnih podataka.

Neke od najčešćih primjena Javascript jezika odnose se na povećanje interaktivnosti web stranica dodavanjem dinamički promjenjivih sadržaja, kao što su: promjene izgleda slika prilikom prelaska mišem preko njih, validacija web formi, kreiranje dinamičkih web formi, dinamička promjena sadržaja web stranice ovisno o nekom parametru, otvaranje novog prozora uz kontrolu izgleda prozora, i sl.

```
<!DOCTYPE html>
<html>
<body>

<button onclick="mojaFunkcija()">Pritisni</button>

<p id="odlomak"></p>

<script>
function mojaFunkcija() {
  document.getElementById("odlomak").innerHTML = "Pritisnuli ste tipku.";
}
</script>

</body>
</html>
```

Sl. 2.3. Primjer Javascript koda u HTML dokumentu

Prema slici 2.3. prikazana je Javascript skripta koja se nalazi unutar *body* sekcije HTML dokumenta. Korištenjem HTML elementa *button* definirana je tipka kojoj je pridružen događaj „onclick“ koji označava pritisak tipke miša. Nakon što se klikne na tipku „Pritisni“ pokreće se funkcija „mojaFunkcija()“ koja se nalazi unutar skripte. Pristupa se Id identifikatoru koju je pridružen „p“ elementu te se pomoću Javascript atributa „innerHTML“ pristupa tekstu odlomka i dodjeljuje mu se vrijednost „Pritisnuli ste tipku“.

Kao što je već prethodno spomenuto JQuery je Javascript biblioteka čija je osnovna zadaća da pojednostavi upotrebu Javascript-a na web stranicama. Prvi puta je objavljen od strane Johna Resiga 2006. godine te je ubrzo stekao visoku populanost. JQuery obuhvaća mnoge zadatke koji zahtijevaju veliki broj linija koda Javascript-a te ih oblikuje u metode koje se mogu pozvati samo u jednoj liniji koda. [7]

JQuery biblioteka je samo jedan Javascript dokument koji se poziva unutar *head* sekcije HTML dokumenta. U odnosu na prethodni primjer primjene Javascript programskog jezika, koristeći JQuery može se dohvatiti HTML element također preko identifikatora ali na puno jednostavniji način. (Sl. 2.4.)

```
<script>
function mojaFunkcija() {
  $.("#odlomak").text('Pritisnuli ste tipku.');
}
</script>
```

Sl. 2.4. Dohvaćanje elementa pomoću JQuery Biblioteke

2.4 Bootstrap

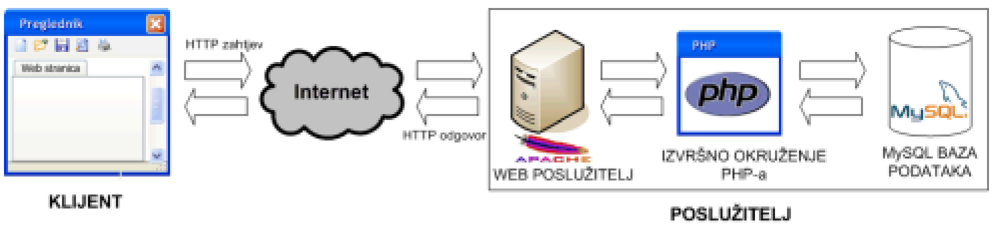
Bootstrap je besplatan radni okvir (eng. *framework*) za razvoj web stranica. Napravljen je od strane Twitter kompanije sredinom 2010. godine.[8] Može se definirati kao skup alata baziran na HTML, CSS i Javascript tehnologijama, koji omogućuje lakšu izradu web stranica. Bootstrap koristi globalne CSS postavke i HTML elemente, stilizirane i poboljšane proširenim klasama te naprednim mrežnim sustavom (eng. *grid system*), koji se skalira i prilagođava web stranicu s obzirom na uređaj na kojem se ona pokreće. Web stranica je prilagođena različitim uređajima od mobitela i tableta pa sve do klasičnih stolnih računala.

Bootstrap mrežni sustav koristi kontenjere (eng. *containers*), stupce (eng. *columns*) i retke (eng. *rows*) za raspoređivanje i usklađivanje sadržaja na web stranici. Svaki definirani redak na stranici sastoji se od 12 stupaca koji se mogu koristiti, ukoliko ne želimo koristiti svih 12 stupaca, oni se mogu međusobno grupirati te tvoriti veći stupac. Bootstrap mrežni sustav je prilagodljiv te se može prilagoditi različitim veličinama zaslona [9].

2.5 PHP i MySQL

PHP (rekurzivna skraćenica za PHP: *HyperText Preprocessor*) je popularan skriptni jezik otvorenog koda (eng. *open-source*) namjenjen za programiranje web stranica. Prve verzije zvale su se PHP/FI (eng. *Personal Home Page Tools/Forms Interpreter*) koje je razvio Rasmus Lerdorf oko 1995. godine. Danas je PHP jedan od najzastupljenijih programskih jezika za programiranje web aplikacija. Vrlo lako ga je spojiti sa HTML-om te time dobiti dinamički kreirane web stranice. Razlikuje se od klijentskih skriptnih jezika poput Javascript-a zbog toga što se izvršava na poslužitelju, a ne na klijentu. Rezultat izvršavanja je HTML kod koji se šalje pregledniku kojeg on razumije bez potrebe za bilo kakvim nadogradnjama. [10]

Podaci koje obrađuje PHP programski jezik trebaju se negdje spremiti, a tu nastupa MySQL koji je besplatan sustav otvorenog koda koji služi za upravljanje bazom podataka. Neke od najpoznatijih svjetskih kompanija koje koriste MySQL baze podataka su: Facebook, Nasa, Youtube, Netflix i dr. [11]. Prema slici 2.5. ispod, prikazan je odnos između klijenta i poslužitelja, te je vidljivo da nakon svakog zahtjeva od strane klijenta poslužitelj će izvršavati PHP skripte koje komuniciraju sa bazom podataka i vraćaju rezultat klijentu.



Sl 2.5. Odnos klijent poslužitelj [12]

U izradi kompleksnijih web aplikacija programeri najčešće koriste PHP radne okvire kao što su: Laravel, CodelGniter, CakePHP, Simfony, Zend, Phalcon, Yii.

Karakteristike PHP-a [13]:

- Vrlo popularan u upotrebi, alternativa je svom glavnom konkurentu – Microsoft's ASP-u
- Besplatan i otvorenog koda
- Pogodan za razvoj web stranica i može se direktno upisati u HTML kod
- Sintaksa je slična C programskom jeziku, te se vrlo često koristi uz Apache poslužitelj (engl. *web server*), na različitim operacijskim sustavima.
- Može se koristiti i na Microsoft's Internet Informations Server-u (IIS), pod Windows operacijskim sustavom
- Podržava rad s raznim bazama podataka: MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL i sl.

Za upotrebu PHP-a na računalu potrebno je imati zadovoljene sljedeće uvjete:

- Instaliran Apache server na Windows ili Unix/Linux poslužitelju
- Instaliran PHP jezik na Windows ili Unix/Linux poslužitelju
- Instaliranu MySQL bazu podataka na Windows ili Unix/Linux poslužitelju

PHP skripte se nalaze unutar tagova *<?php* i *?>* te se mogu postaviti bilo gdje unutar HTML dokumenta. PHP datoteke su datoteke sa ekstenzijom *.php*, te mogu sadržavati HTML kod.

```
<html>
<body>
    <?php
        echo "Ovo je php skripta.";
    ?>
</body>
</html>
```

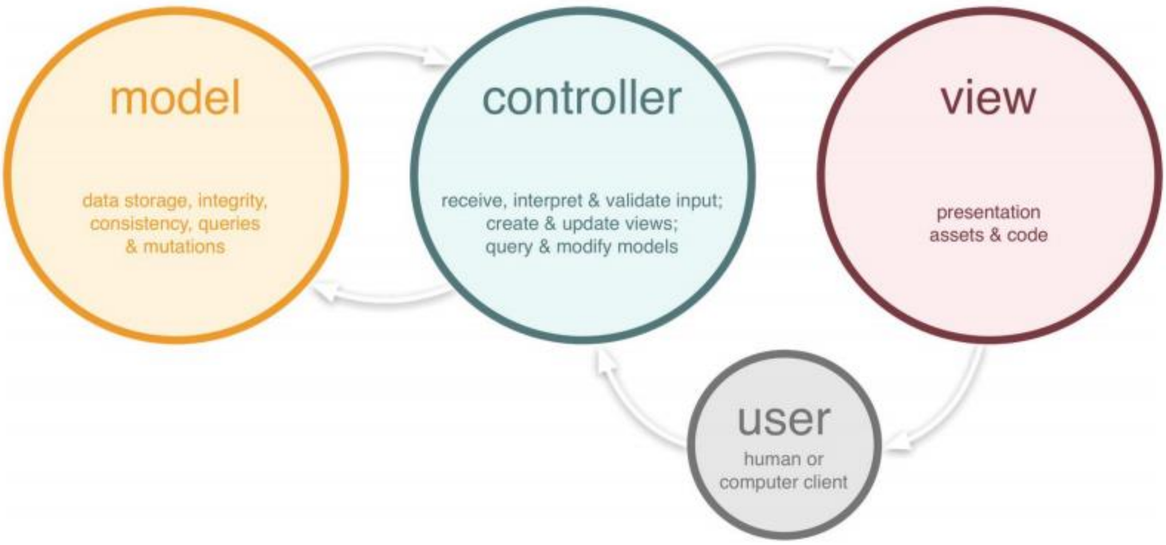
Sl. 2.6. Primjer PHP skripte

Prema slici 2.6. prikazana je PHP skripta. Unutar *<?php* i *?>* tagova nalazi se kod PHP skripte. Naredbom *echo* ispisuje se tekst na zaslon. Prethodno je naglašeno da je PHP programski jezik koji se izvršava na poslužitelja pa prema tome ovaj kod se ne može izvršiti na lokalnom računalu nego se treba postaviti na poslužitelj gdje je instalirana podrška za PHP.

2.6 Laravel

Laravel je radni okvir otvorenog koda koji je baziran na PHP-u. Razvio ga je Taylor Otwell s ciljem razvoja web aplikacija koristeći MVC arhitekturu (eng. *Model-View-Controller*). Prva beta verzija pojavila 9. lipnja 2011. godine, dok je prva stabilna verzija izašla mjesec dana kasnije[14]. Verzija Laravela koja se koristi u ovom diplomskom radu je 5.6.

Kao što je navedeno Laravel koristi MVC arhitekturu. Takva arhitektura sastoji se od 3 komponente: Model, Pogled i Kontroler. Model je dio u kojem se drži poslovna logika aplikacije. Model je zadužen za rad sa bazom podataka, odnosno u modelu se nalazi kod za rad sa bazom podataka a to se odnosi na dohvaćanje ili spremanje podataka u bazu. Sama riječ „Pogled“ (engl. *view*) govori da se ovaj dio MVC arhitekture odnosi na sam izgled web aplikacije. Sve ono što korisnik vidi u web pregledniku može se definirati kao pogled. Kontroler odvaja poslovnu logiku iz modela od korisničkog sučelja te određuje kako će aplikacija odgovoriti na korisničke zahtjeve. Prva komponenta aplikacije koja se poziva je kontroler te on nakon toga povezuje poglede i modele (Sl. 2.7.).



Sl. 2.7. Prikaz MVC arhitekture [15]

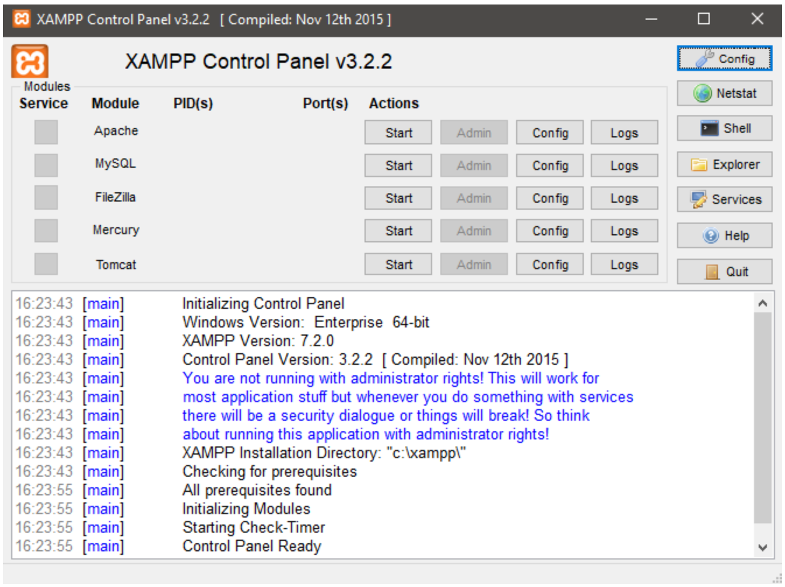
Za korištenje Laravel-a potrebno je ispuniti sljedeće zahtjeve (odnosi se na posljednju verziju Laravel-a 5.6):

- PHP >= 7.1.3
- OpenSSL PHP Extension
- PDO PHP Extension
- Mbstring PHP Extension
- Tokenizer PHP Extension
- XML PHP Extension
- Ctype PHP Extension
- JSON PHP Extension

Ukoliko se zadovoljavaju svi zahtjevi, prvo što je potrebno napraviti je instalirati Composer alat. Laravel koristi Composer za upravljanje svojim ekstenzijama. Nakon instalacije Composer-a potrebno ga je pokrenuti, te koristeći naredbu *composer global require laravel/installer* instalirati će se posljednja verzija Laravel-a na računalo. Koristeći naredbu *laravel new ime_projekta* u komandnoj liniji kreirati će se novi Laravel projekt na računalo [16].

2.7 Alati

Kako bi se omogućio Apache server sa PHP programskim jezikom i MySQL bazom podataka na računalu, potrebno je koristiti Xampp alat. Xampp je multi-platforma otvorenog koda koja omogućuje Apache poslužitelj, PHP, MySQL, MariaDB i druge aplikacije na lokalnom računalu. Prema tome Xampp platforma je izuzetno pogodna za razvijanje i testiranje aplikacija na lokalnom računalu bez potrebe za postavljanjem aplikacije na web poslužitelj. Prema slici 2.8. ispod, prikazana je kontrolna ploča Xampp alata te su vidljivi moduli koji se mogu uključiti unutar aplikacije.



Sl. 2.8. Prikaz kontrolne ploče Xampp alata

Za pisanje i strukturiranje koda koristi se razvojno okruženje Sublime Text 3, a za njegovo održavanje koristi se Git sustav u kombinaciji sa servisom Github. Git sustav omogućuje korisniku vraćanje na prethodnu verziju koda u slučaju pogreške na trenutnoj verziji. Testiranje koda se izvodi pomoću Xampp multi-platforme i web preglednika Google Chrome.

3. ARHITEKTURA APLIKACIJE

Kao što je već prethodno navedeno za izradu aplikacije koristi se PHP radni okvir Laravel. Laravel se temelji na MVC arhitekturi te se prema tome aplikacija može podijeliti na tri dijela: Modeli, Kontroleri i Pogledi. Sama izrada se ne odvija takvim redosljedom nego se svi dijelovi izrađuju paralelno jer su međusobno povezani.

3.1 Baza podataka

Temelji se na MySQL sustavu baze podataka kojeg omogućuje Xampp platforma. Prema tome, potrebno je kreirati bazu podataka korištenjem navedene platforme. Za potrebu aplikacije za prijenos i pretvornu slikovnih datoteka kreirana je baza podataka *images_database* te se konfiguracija za spajanje na navedenu bazu podataka treba postaviti u Laravel aplikaciji unutar *.env* datoteke koja se nalazi u početnom direktoriju. Prema slici 3.1 ispod, prikazane su vrijednosti u konfiguracijskog datoteci potrebne za spajanje na bazu podataka.

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

Sl. 3.1. Konfiguracija za spajanje na bazu podataka

3.1.1 Migracije

Nakon spajanja sa bazom podataka potrebno je kreirati tablice za registrirane korisnike, vrste korisnika, učitane slike i tablica za poslane slike. Za kreiranje tablice koriste se Laravel migracije (eng. *migrations*) koje omogućuju jednostavno kreiranje strukture baze podataka [17].

```
Denis@DESKTOP-PTVO1MT MINGW64 ~/Desktop/diplomski_rad (master)
$ php artisan make:migration create_users_table
```

Sl. 3.2. Primjer naredbe za kreiranje migracije za tablicu korisnika

Prema slici 3.2. vidljivo je da Laravel koristi svoj vlastiti „Artisan“ CLI (engl. *Command-Line Interface*) koji omogućuje velik broj naredbi za olakšavanje izrade aplikacije, što se može provjeriti korištenjem naredbe *php artisan list*. Sve kreirane migracije za bazu podataka spremaju se u mapu *database/migrations* (Sl. 3.3.).

- migrations
  - 2014_10_12_000000_create_users_table.php
  - 2014_10_12_100000_create_password_resets_table.php
  - 2018_05_16_143937_create_roles_table.php
  - 2018_05_16_145134_create_images_table.php
  - 2018_06_12_123937_create_send_images_table.php

Sl. 3.3. Kreirane migracije za bazu podataka

Prema slici 3.3. može se zaključiti da se svaka kreirana migracija odnosi na jednu tablicu u bazi podataka. Migracije za tablicu korisnika i resetiranje lozinke se kreiraju same prilikom stvaranja Laravel okruženja, dok su ostale migracije ručno kreirane za potrebe aplikacije. Primjer strukture migracije za kreiranje tablice korisnika može se vidjeti na slici 3.4.

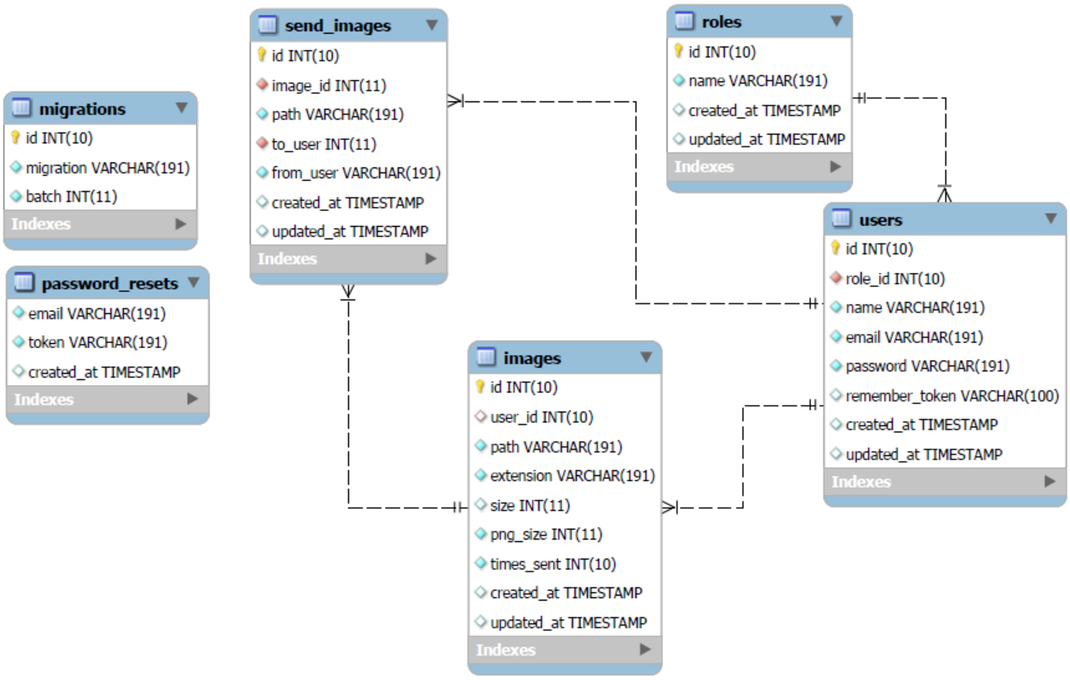
```php
class CreateUsersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('users', function (Blueprint $table) {
            $table->increments('id');
            $table->integer('role_id')->default(2)->unsigned();
            $table->string('name');
            $table->string('email')->unique();
            $table->string('password');
            $table->rememberToken();
            $table->timestamps();
        });
    }

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

Sl. 3.4. Struktura *create_users_table* migracije

Prema slici 3.4. iznad vidljivo je da svaka migracija ima dvije metode. *Up()* metoda koristi se za kreiranje tablice, a *down()* metoda skida migraciju za baze podataka, odnosno uklanja tablicu. Koristi se *Schema* okruženje koje omogućuje kreiranje i manipuliranje tablicama baze podataka. Sve dostupne vrste tipova podataka za tablice koji se mogu koristiti unutar *Schema* okruženja mogu se pogledati u dokumentaciji. [17]

Nakon kreiranja migracija za bazu podataka koristi se naredba *php artisan:migrate* kojom se migracije postavljaju na bazu podataka i kreiraju se tablice, dok se za uklanjanje migracija koristi naredba *php artisan migrate:rollback.*

Dizajn baze podataka aplikacije za pretvorbu i prijenos slikovnih datoteka je kreiran koristeći *MySQL Workbench* alat, te se može pogledati na slici 3.5. ispod.



Sl. 3.5. Grafički prikaz baze podataka korištenjem programa *MySQL Workbench*

Prema slici 3.5. prikazan je grafički prikaz baze podataka sa prethodno navedenim pripadajućim tablicama. Više o relacijama između navedenih tablica biti će objašnjeno u sljedećem poglavlju.

3.2 Modeli

Model je dio u kojem se drži poslovna logika aplikacije. Pomoću modela može se spremati i dohvaćati podatke iz baze podataka. Kada se određeni zahtjev iz pogleda prosljedi kontroleru tada ga on prosljeđuje odgovarajućem modelu koji se odnosi na određenu tablicu u bazi podataka, te model obavlja zadanu zadaću i vraća kontroleru obrađene podatke. Za svaku kreiranu tablicu u bazi podataka potrebno je kreirati model, te se pomoću njega mogu dohvaćati podaci iz te tablice, ili spremati u nju.

Model se kreira korištenjem „Artisan“ *Command Line Interface* (CLI) naredbe *php artisan make:model naziv_modela*. Kada se ne odredi u modelu na koju se tablicu u bazi podataka on odnosi tada naziv modela mora biti jednak jednini imena tablice u bazi podataka. Primjerice, za naziv tablice *Users* model se mora nazvati *User*. Svi modeli se spremaju u *app* direktorij [18].

Kao što je prethodno navedeno, pomoću modela može se pristupati podacima u tablici na koju se on odnosi te ih obrađivati i upravljati njima. Kako bi se moglo pristupati tim podacima u bazi, svaki model nasljeđuje roditeljsku klasu *Model* koja ima ugrađene statične metode koje možemo pozivati i slati upite na bazu podataka.

Tablice u bazi podataka često su međusobno povezane. Primjerice, svaki korisnik može učitati sliku u aplikaciju te je on vlasnik te učitane slike, u tu svrhu koriste se Laravel relacije. Relacije su statične metode ugrađene u klasi *Model* kojima se mogu određivati odnosi između tablica. Relacije koje se mogu koristiti su: *One To One, One To Many, Many To Many* itd. [19].

3.2.1 User

Prvi kreirani model u aplikaciji korištenjem „Artisan“ CLI je *User* model koji se odnosi na tablicu *users* u bazi podataka. Definirane relacije i funkcije u *User* modelu mogu se vidjeti na slici 3.6. ispod.

```php
public function role(){

    return $this->belongsTo('App\Role');
}

public function images(){

    return $this->hasMany('App\Image');
}

public function received_images(){

    return $this->hasMany('App\SendImage', 'to_user');
}

//Provjera da li je user admin (Admin Middleware)
public function isAdmin(){

    return ($this->role->name == "Administrator") ? true : false;
}
```

Sl. 3.6. Isječak koda iz *User.php* datoteke

Prema slici 3.6. vidljivo je da su u *User* modelu definirane relacije sa drugim tablicama. Prva metoda *role()* odnosi se na vrstu korisnika. Tablica *users* povezana je sa tablicom *roles* u kojoj se nalaze uloge korisnika kao što su *Administrator* i *User*. Kako bi se tablica *users* povezala sa tablicom *roles* koristi se relacija *hasMany* kojom je definirano da se svaka uloga može dodjeliti većem broju korisnika, ali svaki korisnik može imati samo jednu ulogu. Kako bi se odredila takva relacija koristi se metoda *belongsTo()* koja se dodjeljuje modelu koji se odnosi na tablicu koja sadrži strani ključ (engl. *foreign key*) iz druge tablice. Metodom *images()* određena je relacija *hasMany* sa tablicom *images*, prema kojoj svakom korisniku može pripadati veći broj slika iz te tablice, a svaka slika može pripadati samo jednom korisniku, onome koji ju je učitao u aplikaciju. Metodom *received_images()* određeno je da korisnik može imati veći broj primljenih slika, te se preko te metode mogu dohvaćati sve korisnikove primljene slike. Pozivanjem metode *isAdmin()* može se provjeriti kojoj ulozi pripada navedeni korisnik. Kao rezultat dobiva se vrijednost istina ili laž ovisno o tome da li je korisnik administrator ili nije.

3.2.2 Role

Model *Role* odnosi se na tablicu *roles* iz baze podataka. U toj tablici nalaze se imena uloga koje se mogu pridružiti korisniku. Korisnik može biti običan korisnik ili mu se može dodijeliti administratorska uloga. Tablica *roles* u relaciji je sa tablicom *users* što je prethodno već navedeno, pozivanjem metode *belongsTo()* u *User* modelu. Time je određeno da se svakom korisniku može dodjeliti jedna uloga, dok svakoj ulozi može pripasti više korisnika.

3.2.3 Image

Model *Image* odnosi se na tablicu *images* iz baze podataka. Tablica *images* u relaciji sa tablicom *users*. Prema toj relaciji svaki korisnik može imati više slika, dok jedna slika može pripadati samo jednom korisniku. U modelu *Image* poziva se metoda *belongsTo()* kojom se određuje da slika pripada korisniku (Sl. 3.7).

```
class Image extends Model
{
    public function user(){

        return $this->belongsTo('App\User');
    }
}
```

Sl.3.7. Isječak koda iz *Image.php* datoteke

3.2.4 SendImage

Posljednji kreirani model je *SendImage* model koji se odnosi na *sendImages* tablicu u bazi podataka. Ovaj model koristi se za spremanje i dohvaćanje podataka o poslanim slikama u tablicu *sendImages*. Za svaku poslanu sliku drugom korisniku spremaju se ime pošiljatelja, „id" od slike iz tablice *images*, te „id" od korisnika primatelja.

```
public function toUser(){

    return $this->belongsTo('App\User', 'to_user');
}

public function imageData(){

    return $this->belongsTo('App\Image', 'image_id');
}
```

Sl. 3.8. Isječak koda iz *SendImage.php* datoteke

Prema slici 3.8. može se vidjeti da su u *sendImage* modelu definirane dvije relacije. Kako se u *sendImages* tablici nalaze „id" od korisnika primatelja i slike, potrebno je definirati relacije prema kojima poslana slika pripada primatelju i originalnoj slici. Metodom *toUser()* mogu se dohvatiti podaci o korisniku koji prima sliku, dok se metodom *imageData()* mogu dohvatiti detaljni podaci o slici koja je poslana. Vidljivo je da u obje *belongsTo()* metode osim prvog parametra koji se odnosi na model sa kojim su u relaciji, primaju i drugi parametar. Drugim parametrom određen je strani ključ (engl. *foreign key*) iz druge tablice koji se nalazi u *sendImages* tablici te je pomoću njega određena relacija.

3.3 Kontroleri

Kontroler je komponenta koja povezuje poglede i modele. Odvaja poslovnu logiku iz modela od korisničkog sučelja. Kontroleri određuju kako će aplikacija odgovoriti na korisničke zahtjeve (eng. *requests*), takvi ulazni podaci prema kojima se poziva kontroler u web aplikaciji su HTTP GET i POST zahtjevi. Zbog toga je kontroler uvijek prva komponenta koja se poziva u aplikaciji, jer svaki zahtjev se prvo prosljeđuje kontroleru koji nakon toga povezuje pogled sa odgovarajućim modelom [20].

Kontroler se kreira naredbom *php artisan make:controller naziv_kontrolera*, te je svaki kontroler u aplikaciji smješten u direktoriju *app/http/controllers*. Kako bi se moglo definirati koji će se kontroler povezati sa kojim korisničkim zahtjevom potrebno je definirati rute u *routes/web.php* datoteci. Svakom URL-u (engl. *Uniform Resource Locator*) dodjeljuje se metoda koja će se pozvati iz određenog kontrolera. Također je potrebno odrediti koji je tip korisničkog zahtjeva. Za automatsko kreiranje ruta odnosno URL-ova korisi se *resource()* metoda koja automatski kreira rute za određeni kontroler i povezuje ih sa metodama. Na slikama 3.9. i 3.10. ispod prikazani su isječci koda iz *routes/web.php* datoteke sa kreiranim rutama korištenjem *resource()* i spajanjem jednog URL-a sa metodom iz kontrolera.

```
Route::resource('/images', 'UserImagesController');
```

Sl. 3.9. Korištenje *resource()* metode za automatsko kreiranje putanja

```
Route::get('download/{image_id}/{isoriginal?}', 'DownloadController@download_user_image');
```

Sl. 3.10. Povezivanje jednog URL-a sa metodom iz kontrolera

Prema slici 3.9. vidljivo je korištenje *resource()* metode za automatsko kreiranje ruta i njihovo povezivanje sa metodama u kontroleru. Rute koje se pri tome kreiraju odnose se na osnovne CRUD (engl. *Create, Read, Update, Delete*) operacije, te se mogu vidjeti u literaturi [21].

Prema slici 3.10. vidljivo je povezivanje jednog URL-a koji se odnosi na putanju za skidanje slike na računalo. Vidljivo je da se pri tome koristi HTTP GET zahtjev pozivanjem metode *get()*. Prvi parametar je URL u kojem se nalaze dva parametra *image_id* i *is*_original, te se preko njih dohvaća „id“ od slike iz tablice te parametar koji govori da li se slika treba prebaciti u prvobitni oblik ili ne. Drugi parametar *get()* metode odnosi se na kontroler i njegovu metodu koja će se pozvati. Poziva se *DownloadController* i njegova pripadajuća metoda *download_user_image()* koja sadrži logiku za skidanje slike na računalo.

U direktoriju *app/http/controllers/auth* nalaze se kontroleri za logiranje, registraciju i oporavak zaporke. Takvi kontroleru kreiraju se automatski prilikom incijalizacije Laravel okruženja.

3.3.1 AdminUsersController

AdminUsersController odnosi se na obavljanje operacija povezanih sa administratorskom ulogom. Prema tome, svim URL-ovima koji su povezani sa metodama iz ovog kontrolera, može pristupiti samo korisnik koji ima ulogu administratora. U tu svrhu potrebna je autorizacija korisnikovih zahtjeva, treba se osigurati da samo korisnik sa ulogom administratora može pristupiti ovom kontroleru. Kako bi se ograničili korisnici potrebno je koristiti *middleware*.

Middleware je mehanizam koji omogućuje filtriranje HTTP zahtjeva. Potrebno je osigurati da svim operacijama pristupa logirani korisnik koji mora biti administrator. U tu svrhu koristi se *middleware* za autentifikacijuk koji osigurava da korisnik mora biti logiran u aplikaciju, i ručno kreirani *AdminMiddleware* korištenjem „Artisan“ CLI naredbe *php artisan make:middleware*. *AdminMiddleware* osigurava da svim operacijama u ovom kontroleru pristupa samo korisnik sa ulogom administratora. Svaki *middleware* spremljen je u direktorij *app/Http/Middleware*. Prije upotrebe potrebno ga je registrirati u datoteci *app/Kernel.php* [22].

```php
class AdminMiddleware
{

    //Provjerava da li je korisnik administrator, ako nije vraća ga nazad
    public function handle($request, Closure $next)
    {
        $user = Auth::user();

        if(!$user->isAdmin())
        {
            return redirect()->back();
        }
        return $next($request);
    }
}
```

Sl. 3.11. Isječak koda iz datoteke *AdminMiddleware.php*

Prema slici 3.11. vidljiva je klasa *AdminMiddleware* koja sadrži metodu *handle()* za rukovanje sa HTTP zahtjevom. Prvo se dohvaća autentificirani korisnik u aplikaciji te se poziva metoda *isAdmin()* koja je prethodno definirana u *User* modelu. Ukoliko navedena metoda vraća laž (engl. *false*) vratiti će se rezultat klijentu i korisnik će biti vraćen na prethodnu putanju gdje se nalazio te će mu pristup biti onemogućen. Ukoliko metoda kao rezultat vrati istinu (engl. *true*) tada će se korisnički zahtjev prosljediti dalje u aplikaciju i preći će se na sljedeći zahtjev. Kako bi se *middleware* dodijelio kontroleru potrebno ga je registrirati unutar *__construct()* metode kontrolera (Sl. 3.12).

```php
public function __construct(){
    return $this->middleware('admin');
}
```

Sl. 3.12. Dodavanje *middleware-a* kontroleru

Kao što samo ime kontrolera govori, njime se omogućuje upravljanje korisnicima u aplikaciji od strane administratora. Administrator može imati uvid u sve registrirane korisnike, klikom na korisnika može vidjeti podatke o njemu kao i njegove učitane slike. Također administrator može registrirati nove korisnike u aplikaciju, ali i brisati postojeće. Prikaz isječka koda za registraciju novih korisnika može se vidjeti na slici 3.13. ispod.

```php
public function store(AdminNewUserRequest $request)
{
    $user = new User();
    $user->name = $request['name'];
    $user->email = $request['email'];
    $user->password = Hash::make($request['password']);
    $user->save();

    Session::flash('Success', 'New user registered!');
    return redirect('admin/users');
}
```

Sl. 3.13. Isječak koda za registraciju novih korisnika od strane administratora u *AdminUsersController.php* datoteci

Prema slici 3.13. prikazana je metoda za registraciju novih korisnika od strane administratora, u kojoj se kreira novi objekt klase *User* te mu se dodjeljuju vrijednosti tekstualnih polja iz poslane forme od strane administratora u HTTP zahtjevu. Takav zahtjev se prethodno prosljeđuje putanji (engl. *route*) koja je povezana sa *store()* metodom u kontroleru. Kako bi se pristupilo korisničkom zahtjevu, *Store()* metoda kao parametar prima objekt klase *AdminNewUserRequest*. Klasa *AdminNewUserRequest* je ručno kreiran zahtjev koji nasljeđuje klasu *FormRequest* te sadrži logiku za validaciju unešenih podataka o novom korisniku od strane administratora aplikacije. Svi ručno kreirani zahtjevi spremaju se u direktorij *app/Http/Requests*. Nakon što administrator pošalje korisnički zahtjev slanjem forme iz preglednika, prije pozivanja *Store()* metode iz kontrolera izvodi se validacija podataka u *AdminNewUserRequest* datoteci, ako je validacija podataka uspješna poziva se navedena *Store()* metoda, dok u suprotnom korisniku se vraća odgovor sa pogreškom u validaciji podataka. Isječak koda za validaciju može se pogledati na slici 3.14. ispod.

```php
public function rules()
{
    return [
        'name' => 'required|string|max:255',
        'email'=> 'required|string|email|max:255|unique:users',
        'password' => 'required|string|min:6|confirmed'
    ];
}
```

Sl. 3.14. Isječak koda za validaciju podataka novog korisnika u datoteci *AdminNewUserRequest.php*

Prema slici 3.14. prikazana je metoda *rules()* u kojoj se nalazi logika za validaciju unešenih podataka za novog korisnika. Funkcija vraća asocijativno polje podataka u kojemu se ključevi (engl. keys) odnose na imena tekstualnih polja u korisničkoj formi koja je poslana, a vrijednosti (engl. values) koje se dodjeljuju ključevima su pravila koja se trebaju zadovoljiti kako bi validacija podataka bila uspješna. Primjerice, *required* pravilo označava da tekstualno polje iz forme ne smije biti prazno, *string* označava da podatak mora biti niz znakova, *unique:users* označava da navedeni email za novog korisnika mora biti jedinstven i ne smije već postojati u tablici *users*, itd. Sva dostupna pravila za validaciju podataka mogu se pogledati u Laravel dokumentaciji [23].

Nakon uspješne validacije podataka, kreiranja objekta klase *User* i spremanja podataka u tablicu *users*. Kreira se poruka o uspješnoj registraciji korisnika, koja se prikazuje administratoru nakon preusmjeravanja na drugu putanju korištenjem metode *redirect()*. Takva metoda je globalna *helper* metoda kojom se inicijalizira objekt klase *RedirectResponse*, koja sadrži pripadajuća zaglavlja potrebna za prebacivanje korisnika na drugu putanju [24].

3.3.2 AdminImagesController

AdminImagesController je kontroler koji se odnosi na obavljanje operacija povezanih sa administratorskom ulogom. Ovaj kontroler odnosi se na sve operacije koje su povezane sa korisničkim slikama učitanim u aplikaciju. Administrator ima uvid u sve učitane slike i poslane slike u aplikaciji. Također, administrator može vidjeti učitane i primljene slike po svakom korisniku klikom na njega. Klikom na određenu sliku, administrator ima uvid u detaljne podatke o slici. Ukoliko slika nije prikladna za aplikaciju administrator zadržava mogućnost za brisanje takve slike. Kao i prethodni, *AdminUsersController*, i ovaj kontroler sadrži *middleware* kojim se osigurava da sve operacije pripadaju samo administratorskoj ulozi. Na slici 3.15. ispod prikazan je isječak koda za brisanje korisnikove slike.

```php
public function destroy($id)
{

    $image = Image::findOrFail($id);
    Storage::delete('public/images/'.$image->user_id.'/uploaded/'.$image->path);

    if($image->times_sent == 0)
    {
        $image->delete();
    }
    else
    {
        $image->user_id = 0;
        $image->save();
    }

    Session::flash('success', 'Image successfully deleted!');
    return redirect('/admin/images');
}
```

Sl. 3.15. Isječak koda za brisanje korisnikove slike od strane administratora iz *AdminImagesController.php*

Prema slici 3.15. prikazan je isječak koda za brisanje korisnikove slike od strane administratora. Nakon klika na sliku, administratoru se otvara prikaz slike sa detaljima. Klikom na tipku za brisanje, poziva se putanja za brisanje slike i u njoj se prosljeđuje parametar koji se odnosi na „id“ slike u tablici *images*. Metoda *destroy()* prima taj parametar, te se u njoj dohvaća slika iz tablice korištenjem statičke metode *findOrFail()* iz klase *Image*. Korištenjem takve metode osigurava se hvatanje iznimke ukoliko rezultat nije pronađen u tablici baze podataka i korisniku se vraća HTTP 404 pogreška koja označava da tražena stranica ne postoji.

Za upravljanje učitanim datotekama u Laravel aplikaciji koristi se Laravel datotečni sustav (eng. *filesystem*). Sve datoteke spremaju se u *storage/app/public* direktorij kojem se može pristupiti korištenjem *Storage* okruženja [25]. Statičkoj metodi *delete()* potrebno je prosljediti putanju do datoteke koja se želi obrisati. U navedenom primjeru briše se korisnikova učitana slika koja je smještena u korisnički direktorij *uploads*. Nakon što je slika obrisana sa poslužitelja, potrebno je podatke o njoj obrisati iz baze podataka. Ukoliko slika nije poslana niti jednom korisniku odmah se briše iz baze podataka pozivanjem metode *delete()* na objektu *$image*. Ukoliko je slika poslana drugom korisniku tada je potrebno sačuvati podatke o njoj jer svaka poslana slika dohvaća podatke o sebi iz tablice *images*, te će se sa slike ukloniti samo podatak o vlasniku, a iz baze podataka će biti obrisana tek prilikom brisanja posljednje instance poslane slike.

Navedena metoda *destroy()* odnosi se samo na brisanje korisnikove učitane slike, a u kontroleru se nalazi još i metoda za brisanje primljene slike, kao i metode za prikaz svih učitanih slika u aplikaciji i metode za prikaz detalja o slikama.

3.3.3 UserImagesController

UserImagesController je kontroler povezan sa korisničkim akcijama za učitavanje te prikaz učitanih i primljenih slika. U odnosu na prethodne administratorske kontrolere, u ovom kontroleru se nalazi *UserMiddleware* koji osigurava da svim akcijama pristupa samo obični korisnik ali ne i administrator. Kao što je prethodno navedeno u opisu aplikacije, korisniku je potrebno omogućiti učitavanje slike u aplikaciju koja se nakon toga prebacuje u PNG (engl. *Portable Network Graphics*) oblik. Kako bi se osiguralo učitavanje ispravnog oblika datoteke, ručno se kreira zahtjev *UploadRequest* koji sadrži logiku za validaciju učitane datoteke (Sl. 3.16).

```
public function rules(Request $request)
{
    return [
        'file' => 'bail|required|mimes:jpeg,jpg,png,bmp,gif|max:5000'
    ];
}
```

Sl. 3.16. Prikaz pravila za validaciju učitane datoteke

Prema slici 3.16. vidljivo je da su određeni tipovi datoteke koja se može učitati, kao i maksimalna veličina. Nakon što je validacija učitane datoteke uspješno izvršena, poziva se *store()* metoda iz kontrolera. Ta metoda osigurava konverziju slike u PNG oblik te spremanje slike u korisnički direktorij kao i bazu podataka. Kako bi se izvršila konverzija slike u PNG oblik poziva se metoda *check_extension_and_convert()* (Sl. 3.17).

```
function check_extension_and_convert($image, $user, $time){

    //Ukloni ekstenziju
    $image_name_without_extension = pathinfo($image->getClientOriginalName(),
        PATHINFO_FILENAME);
    $image_name_png = $time . $image_name_without_extension .'.png';
    $extension = $image->getClientOriginalExtension();

    switch($extension){

        //Bmp baca gresku za monokromatski bitmap
        case 'bmp':
            try {
                $original_image = imagecreatefrombmp($image->path()); break;
                }
            catch (\Exception $e){

                return 'Invalid type, 1-bit Bitmap is not allowed.'; break;
            }
        case 'jpg': $original_image = imagecreatefromjpeg($image->path()); break;
        case 'gif': $original_image = imagecreatefromgif($image->path()); break;

        default: return 'Invalid type of file.';
    }

    //Provjeri da li postoji direktorij, ako ne postoji kreiraj ga
    if (!file_exists(storage_path().'/app/public/images/'.$user->id.'/uploaded/')) {

        mkdir(storage_path().'/app/public/images/'.$user->id.'/uploaded/', 0777, true);
    }

    $is_converted_and_saved = imagepng($original_image, storage_path().'/app/public/images/'.$
        user->id.'/uploaded/'.$image_name_png);

    return $is_converted_and_saved;
```

Sl. 3.17. Prikaz metode za konverziju slike iz *UserImagesController.php* datoteke

Prema slici 3.17. prikazana je metoda za konverziju slike. Naredbom *switch-case* provjerava se ekstenzija slike i ovisno o njoj pozivaju se ugrađene PHP funkcije za kreiranje slike. Takve funkcije primaju putanju do slike i kreiraju novu sliku koja se sprema u varijablu. Nakon što se kreira slika poziva se PHP funkcija *imagepng()* kojoj se prosljeđuje učitana slika i putanja na koju se sprema slika. Funkcija vraća istinu i laž ovisno o tome da li je slika prebačena u PNG oblik i spremljena na poslužitelj. Nakon što je slika prebačena u PNG oblik i spremljena na server *store()* metoda iz kontrolera sprema sliku u tablicu *images* u bazi podataka.

Nakon što je slika učitana u aplikaciju, korisnik se prebacuje na putanju za prikaz svojih slika u aplikaciji. Korisnik ima uvid u svoje učitane slike, kao i u slike koje je primio od drugih korisnika (Sl. 3.18).

```
public function index(){

    $user = Auth::user();
    $images = $user->images;

    if($received_images = $user->received_images)
    {
        $all_images = collect();

        foreach($images as $image)
        {
            $all_images->push($image);
        }

        foreach($received_images as $image)
        {
            $all_images->push($image);
        }

        $images = $all_images->sortBy(function($image){

            return $image->created_at;
        });
    }


    return view('user.images', compact('images'));
}
```

Sl. 3.18. Prikaz metode za dohvaćanje korisnikovih slika iz baze podataka i prosljeđivanje slika odgovarajućem pogledu

Prema slici 3.18. vidljiva je metoda za dohvaćanje i prikaz korisnikovih slika. Korištenjem *Auth* okruženja kreira se instanca identificiranog korisnika u aplikaciji. Pozivanjem metoda *images()* i *received_images()* koje su prethodno definirane u *User* modelu, dohvaćaju se korisnikove učitane i primljene slike iz tablica u bazi podataka. Korištenjem *helper* metode *collect()* kreira se instanca klase *Collection*. Polja objekata korisnikovih učitanih i primljenih slika učitavaju se u kreiranu kolekciju. Nakon toga slike se svrstavaju prema vremenu kreiranja. Korištenjem metode *view()* dohvaća se pogled za prikaz slika i prosljeđuju mu se slike za prikaz.

3.3.4 SendReceiveImageController

SendReceiveImageController služi za upravljanje operacijama povezanim sa slanjem i primanjem slika. U njemu se nalaze metode za prikaz i brisanje primljenih slika te metoda za slanje slike drugom korisniku. Sve učitane slike u aplikaciji su u PNG obliku, prema tome korisnik šalje PNG sliku drugom korisniku. Korisnik upisuje *email* adresu drugog registriranog korisnika kojem želi poslati sliku. Provjerava se da li unešena adresa postoji u bazi podataka, te ukoliko postoji slika se šalje (Sl. 3.19).

```
$validator = Validator::make($request->all(), [
'email' => [
    'required',
    Rule::exists('users')->where(function ($query) {

        $query->where([
            ['id', '!=', Auth::user()->id],
            ['role_id', 2]]);
        }),
    ],
]);

if($validator->fails())
{
    return redirect()->back()->withErrors($validator);
}
```

Sl. 3.19. Isječak koda za provjeru *email* adrese drugog korisnika

Prema slici 3.19. prikazan je isječak koda za provjeru *email* adrese korisnika primatelja u tablici baze podataka. Vidljivo je da se koristi pravilo prema kojem adresa mora postojati u tablici *users* u bazi podataka, te se koristi *where* uvjet prema kojem taj korisnik ne smije biti administrator. Ukoliko validacija email adrese ne prođe, korisniku se vraća pogreška da navedena adresa nije ispravna. Ako je validacija uspješna, na temelju Id od slike iz tablice *images* u bazi podataka, slika se šalje drugom korisniku i sprema u tablicu poslanih slika *SendImages* u bazi podataka. Na slici 3.20. prikazan je isječak koda za kopiranje slike u direktorij korisnika primatelja.

```
Storage::copy('public/images/'.$image->user_id.'/uploaded/'.$image->path, 'public/
images/'.$receiver->id.'/received/'.$image->path))
```

Sl. 3.20. Isječak koda za kopiranje slike u direktorij korisnika primatelja

Prema slici 3.20. prikazan je isječak koda za kopiranje slike korisniku primatelju. Statičkoj metodi *copy()* prosljeđuju se 2 parametra koji se odnose na putanju slike koja se kopira, i odredišnu putanju na koju će se slika kopirati. Nakon što je slika primljena korisnik u aplikaciji može vidjeti primljenu sliku, klikom na nju moguće je vidjeti koji je korisnik poslao sliku, te u kojem obliku i kojoj veličini.

3.3.5 DownloadController

Za svaku učitanu sliku u aplikaciji postoji mogućnost skidanja na računalo. Slika se može skinuti u učitanom PNG obliku ili u originalnom obliku koji je bio prije konverzije slike u PNG oblik. U tu svrhu koristi se *DownloadController* koji sadrži logiku za skidanje slike na računalo i po potrebi konverziju slike u njen originalni oblik. *DownloadController* sadrži samo *middleware* za autentifikaciju korisnika što znači da je dostupan običnim korisnicima i administratoru.

DownloadController sadrži metode za skidanje korisnikove učitane slike i primljene slike. Uz te dvije metode kreirana je i *helper* metoda za konverziju slike koja se poziva po potrebi. Obje metode primaju parametar koji definira da li se slika treba skinuti u PNG ili ju je potrebno prebaciti u originalni oblik i omogućiti skidanje takve slike na računalo. Na slici 3.21. ispod prikazan je isječak koda iz metode za konverziju slike u prvobitni oblik.

```
$png_image = imagecreatefrompng($image_full_path);

switch($extension){

    case 'bmp': $is_converted = imagebmp($png_image, $store_path.$image_realname.'bmp');
        break;
    case 'jpg': $is_converted = imagejpeg($png_image, $store_path.$image_realname.'jpg');
        break;
    case 'gif': $is_converted = imagegif($png_image, $store_path.$image_realname.'gif');
        break;

    default: return false;
}

return $is_converted;
```

Sl. 3.21. Isječak koda za konverziju slike u originalni oblik

Prema slici 3.21. iznad prikazan je isječak koda za konverziju slike pomoću *helper* metode *convert_image()*. Ukoliko korisnik želi skinuti sliku na računalo u prvobitnom obliku, funkcija za skidanje slike poziva metodu *convert_image()*. Kreira se PNG slika korištenjem PHP funkcije *imagecreatefrompng()* kojoj se prosljeđuje putanja do PNG slike koja je spremljena na poslužitelj u korisnikov direktorij. Koristi se *switch-case* naredba koja prima podatak iz baze podataka o originalnom obliku slike, te se prema tome izvodi jedan od slučajeva za prebacivanje slike u originalan oblik. Slika koja je prebačena u originalan oblik spremljena je u korisnikov direktorij na poslužitelj, te će se nakon skidanja na računalo obrisati. Ukoliko je slika uspješno prebačena u originalan oblik i spremljena, funkcija vraća logičku istinu i izvodi se naredba za skidanje slike na računalo (Sl. 3.22).

```
return response()->download('storage/images/'. $image->user_id .'/uploaded
    /'. $image_real_name, $image_real_name)->deleteFileAfterSend(true);
```

Sl. 3.22. Isječak koda za skidanje slike na računalo

Prema slici 3.22. prikazan je isječak koda za skidanje slike na računalo. *Download()* metoda govori web pregledniku da treba na računalo skinuti datoteku sa prosljeđene putanje, što je ujedno i prvi parametar koji se prosljeđuje toj metodi. Drugi parametar koji metoda *download()* prima je pa naziv koji će se dodijeliti datoteci prilikom skidanja na računalo. Nakon što se slika skine na računalo poziva se metoda za brisanje datoteke sa poslužitelja.

3.4 Pogledi

Pogledi (eng. *views*) se mogu definirati kao sam izgled web aplikacije, odnosno sve ono sa čime korisnik komunicira. Od tri glavne komponente MVC arhitekture, korisnik vidi samo pogled. Odgovoran je za prikaz podataka iz modela te omogućuje korisniku mjenjanje istih po potrebi, ali ih ne može spremati u bazu. Sve ono što korisnik vidi u web pregledniku je pogled, prema tome sastoji se od HTML, CSS, Javascript i drugih tehnologija za definiranje izgleda web sučelja, više o pogledima može se pogledati u Laravel dokumentaciji [26].

U Laravel pogledima koristi se jednostavan, ali moćan Blade predložak koji olakšava pisanje PHP koda unutar pogleda, ali pri tome korisnik nije ograničen te može i dalje pisati običan PHP kod. Blade pogledi unutar Laravel aplikacije koriste ekstenziju *.blade.php* te su pohranjeni unutar *resources/views* direktorija [27].

Jedna od glavnih prednosti korištenja Blade predloška je nasljeđivanje pogleda i definiranje sekcija unutar pogleda. Glavni i osnovni pogled aplikacije smješten je unutar *resources/views/layouts/app.blade.php* datoteke, te ga ostali pogledi nasljeđuju. Kako bi pogled nasljedio roditeljski, odnosno glavni pogled, koristi se sljedeća sintaksa prikazana na slici 3.23. ispod.

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

@section('content')

@endsection
```

Sl. 3.23. Prikaz sintakse za nasljeđivanje pogleda

Prema slici 3.23. mogu se vidjeti direktive koje se koriste za nasljeđivanje pogleda korištenjem Blade predloška. Naredbi *@extends* prosljeđuje se putanja do pogleda koji se nasljeđuje, dok *@section* direktiva određuje naziv sekcije koja će se ubaciti u glavni pogled unutar njegove *@yield* direktive.

Kao što je prethodno navedeno unutar pogleda se koriste tehnologije za dizajniranje web sučelja kao što su HTML, CSS, Bootstrap i Javascript tehnologija za dinamičnost web stranice. CSS i Javascript datoteke smještene su unutar *public/css* i *public/js* direktorija, te se uključuju unutar *head* sekcije glavnog pogleda *app.blade.php*. Primjer strukture pogleda za prikaz učitanih korisnikovih slika u aplikaciji može se vidjeti na slici 3.24. ispod.

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

@section('main-content')

<h2 align="center"> My Images </h2>

<hr>

    @if(Session::has('success'))
    <div class="row">
        <div class="col-md-4 alert alert-success ml-auto mr-auto text-center">
            {{Session::get('success')}}
        </div>
    </div>
    @endif

<div class="images-container">
@if(count($images) > 0)
@foreach($images as $image)
<div class="images-container-item hoverable">
    @if($image->to_user)
        <a href="/received/{{$image->id}}"><img id="myImg" src="/storage/
        images/{{Auth::user()->id}}/received/{{$image->path}}" height=100 width=150 alt="
        {{$image->path}}"></img></a>
        <div> received </div>
    @else
        <a href="/images/{{$image->id}}"><img src="/storage/
        images/{{Auth::user()->id}}/uploaded/{{$image->path}}" height=100 width=150></img></a>
    @endif
</div>
@endforeach
@endif
</div>

@endsection
```

Sl. 3.24. Primjer strukture pogleda za prikaz korisnikovih učitanih i primljenih slika

Prema slici 3.24. vidljivo jekorištenje HTML elemenata i Blade predloška na primjeru *if-else* grananja kojim se provjerava da li postoje korisnikove slike te se dohvaća svaka slika pojedinačno korištenjem *@foreach* petlje.

Ukoliko postoje korisnikove slike, stavljaju se u HTML *<div>* elemente sa dodjeljenom CSS klasom *images-container-item.* Definicija te klase može se vidjeti unutar *public/css/styles.css* datoteke (Sl. 3.25).

```
.images-container-item{
    margin: 10px;
    padding: 5px;
    background-color: white;
    border: 1px solid #ddd;
    border-radius: 5px;
    min-height: 100px;
    width: 150px;
    position: relative;
}
```

Sl. 3.25. Prikaz CSS klase za smještaj slike u pogledu

Drugi primjer na kojem se može vidjeti struktura isječka koda jednog od pogleda aplikacije je web forma za učitavanje slike sa računala (Sl. 3.26).

```
<form method="post" action="{{action('UserImagesController@store')}}" enctype="multipart/form-data">
    {{csrf_field()}}
    <label class="btn btn-secondary ml-auto" for="inputImage">Select Image</label>
   <input type="file" class="form-control-file" id="inputImage" name="file"></input>
   <p class="selected-file-name"></p>
   @if ($errors->any())
        <div class="alert alert-danger" onclick="remove(this)">
            <ul>
            @foreach ($errors->all() as $error)
                <li>{{ $error }}</li>
            @endforeach
            </ul>
        </div>
    @endif

   <div class="form-button-wrapper">
   <button class="btn btn-primary" value="submit" onclick="buttonSubmit(this)"> Upload </button>

   </div>
   <div class="upload-file-types-text">Only PNG, BMP, JPG and GIF file types are allowed</div>
</form>
```

Sl. 3.26. Isječak koda iz pogleda za učitavanje slike u aplikaciju

Prema slici 3.26. za učitavanje slike sa računala koristi se forma u kojoj se može vidjeti primjer korištenja Blade predloška na atributu *action* gdje se koriste vitičaste zagrade „{{}}“ koje označavaju PHP *echo* naredbu za ispis sadržaja. Koristi se Laravel *helper* metoda *action()* koja generira URL koji je spojen sa zadanom metodom u kontroleru koja obavlja posao spremanja slike u aplikaciju. Više Laravel *helper* metoda može se pogledati u Laravel dokumentaciji [28].

Primjer korištenja Javascript tehnologije u pogledu može se vidjeti na korisnikov klik može se vidjeti prilikom slanja spomenute forme kontroleru gdje je HTML elementu *<button>* pridružena Javascript funkcija funkcija *buttonSubmit()* kojom se onemogućuje korisnika da više puta stisne na gumb za slanje forme (Sl. 3.27).

```
function buttonSubmit(btn){
    btn.disabled = true;
    btn.form.submit();
};
```

Sl. 3.27. Prikaz Javascript funkcije koja se pokreće nakon klika na gumb za slanje forme

Prema slici 3.27. vidljivo je da nakon klika na gumb, isti se onemogućuje za ponovni klik te se nakon toga forma šalje pripadajućem kontroleru. Time se onemogućuje korisnika da više puta pošalje zahtjev za učitavanjem slike u aplikaciju.

3.5 Usporedba slikovnih datoteka

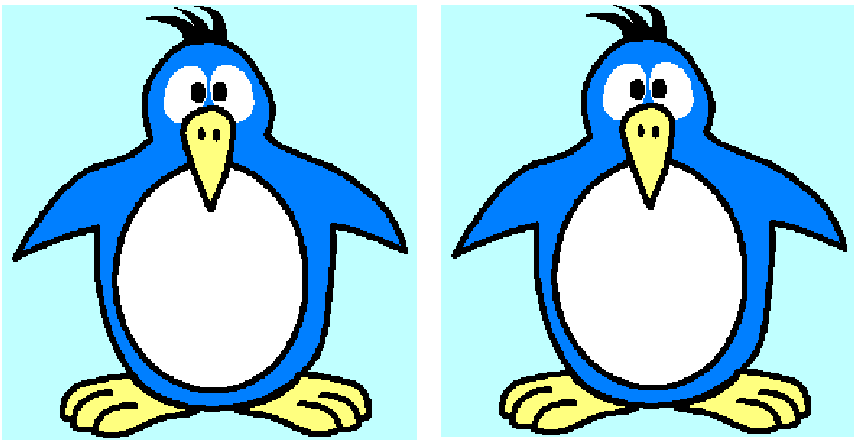
Za usporedbu kvalitete slikovnih datoteka koristi se PSNR (eng. *Peak Signal To Noise Ratio*) metoda [29]. PSNR je logaritamski omjer između maksimalne moguće snage signala i snage šuma koji utječu na kvalitetu slike, te se izražava sljedećom relacijom (3-1):

$$PSNR_{db} = 10 \times \log_{10}\left(\frac{MAX_I^2}{MSE}\right) \qquad (3\text{-}1)$$

Prema formuli (3-1), MAX_I predstavlja maksimalnu vrijednost piksela slikovne datoteke, što za 8-bitne slikovne datoteke iznosi 255. *MSE* (eng. *Mean Square Error*) predstavlja iznos srednje kvadratne pogreške između elemenata originalne i izobličene slikovne datoteke, te se izražava sljedećom relacijom (3-2):

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2 \qquad (3\text{-}2)$$

Kao što je prethodno navedeno aplikacija se posebno orijentira na prijenos BMP slikovnih datoteka. Zbog toga je izuzetno bitno da slikovna datoteka nakon pretvorbi u PNG oblik i obrnuto, bude jednake kvalitete kao i original. Na slici 3.28. ispod nalaze se primjeri BMP slikovnih datoteka korištenih za usporedbu.



a) penguin_original.bmp   b) penguin_conv.bmp

Sl. 3.28. MSE = 0, PSNR = 100dB

Prema slici 3.28. prikazane su BMP slikovne datoteke korištene za usporedbu kvalitete. Nakon korištenja PSNR metode, srednja kvadratna pogreška *MSE* je jednaka nuli. Prema tome, PSNR je jednak 100dB, što označava da su obje slike jednake kvalitete. Na slici 3.29. ispod može se vidjeti primjer JPEG slikovnih datoteka korištenih za usporedbu.

a) sea_original.jpg b) sea_conv.jpg

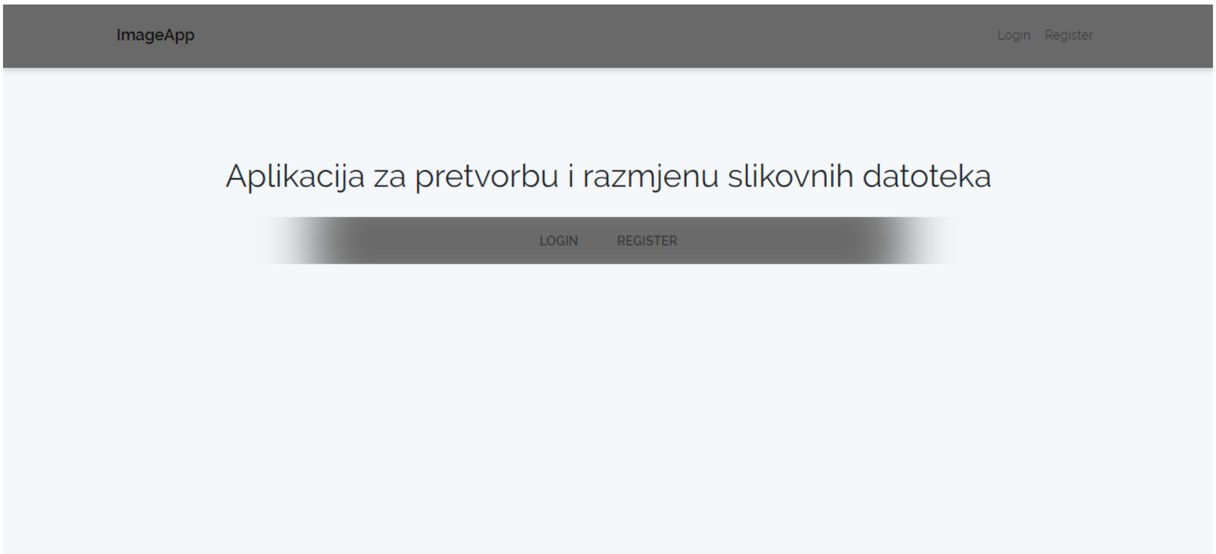
Sl. 3.29. PSNR = 38.77dB

Prema slici 3.29. prikazane su JPEG slikovne datoteke korištene za usporedbu kvalitete. Može se zaključiti da nisu jednake kvalitete jer je PSNR jednak 38.77db, kako je PSNR bliži razlika između slika je veća. Kako se veličina JPEG slikovnih datoteka višestruko povećava nakon konverzije u PNG oblik, takve datoteke nisu pogodne za aplikaciju pa nije izuzetno bitno da budu jednake kvalitete bez gubitaka, jer se aplikacija posebno orijentira na prijenos BMP slikovnih datoteka.

Za izvođenje PSNR metode koristi se Python programski jezik, te se pripadajuća skripta može pogledati u prilozima.

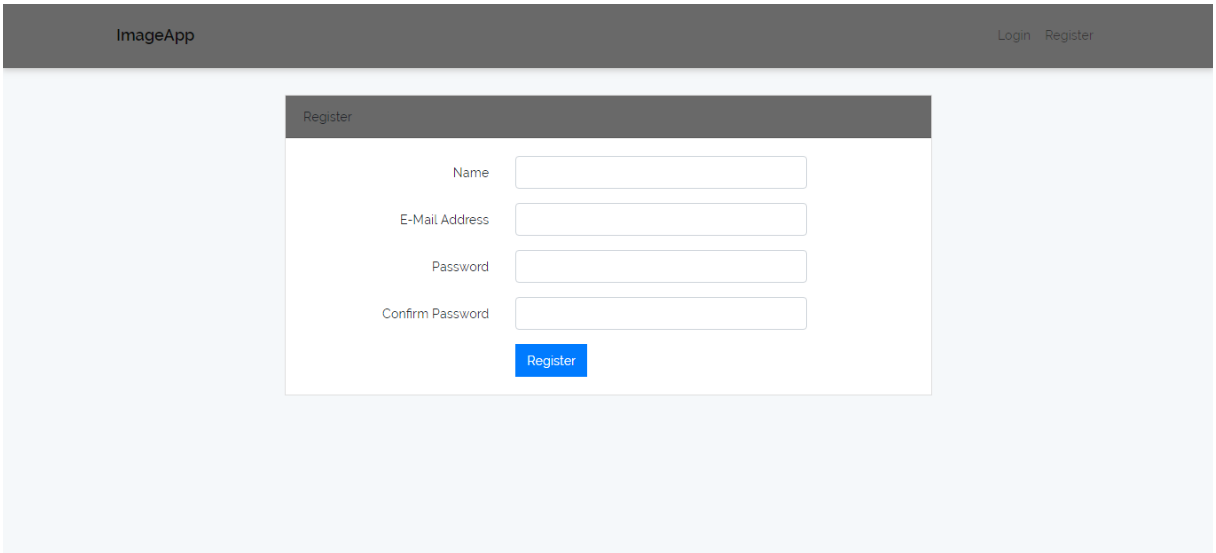
4. KORIŠTENJE APLIKACIJE

Na navedenim primjerima biti će prikazan način korištenja aplikacije za pretvorbu i prijenos slikovnih datoteka. Nakon pokretanja aplikacije korisniku se otvara početni zaslon za logiranje sa postojećim ili registraciju novog računa unutar aplikacije (Sl. 4.1.).



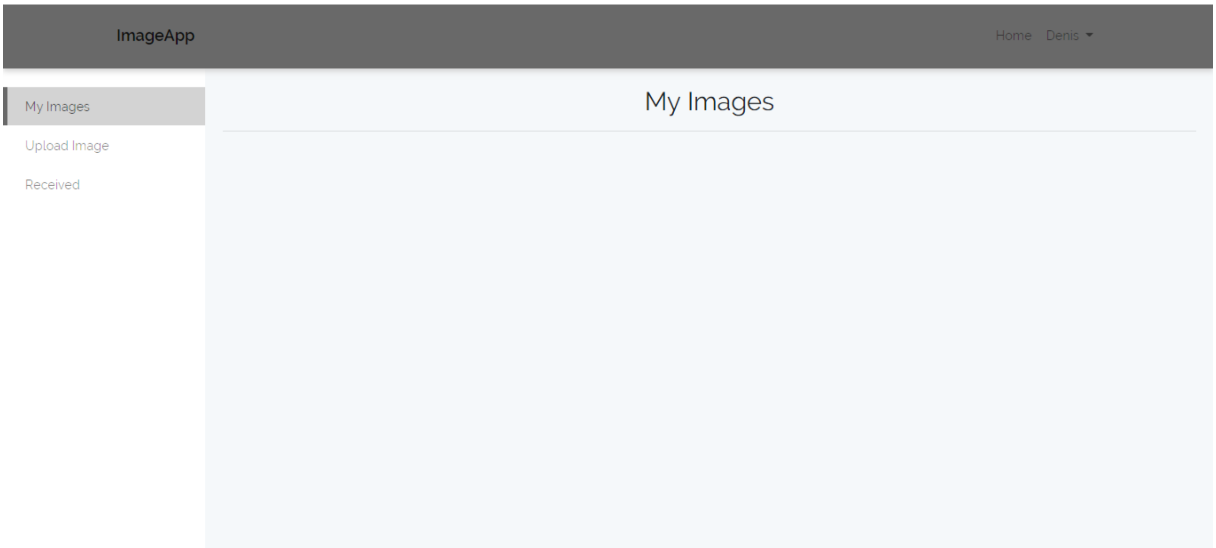
Sl. 4.1. Početni zaslon za logiranje ili registraciju

Klikom na tipku *Register* korisnik se prebacuje na pogled za registraciju u aplikaciju gdje je potrebno unijeti podatke kao što su ime, adresa, lozinka. *Email* adresa mora biti jedinstvena dok ime ne mora. Lozinka mora sadržavati minimalno 6 znakova. Pogled za registraciju u aplikaciju može se vidjeti na slici 4.2 ispod.



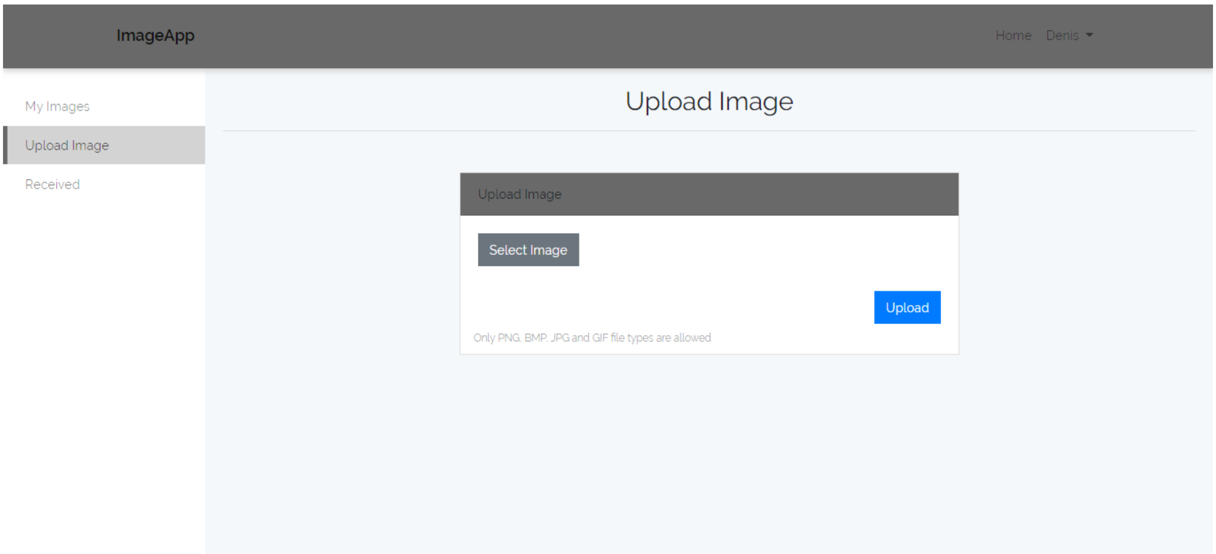
Sl. 4.2. Pogled za registraciju u aplikaciju

Nakon registracije u aplikaciju korisnik se prebacuje na osnovni pogled gdje će se moći prikazati učitane i primljene slike. Sa lijeve strane se nalazi izbornik u kojem korisnik može odabrati pogled *Upload Image* za učitavanje slike i pogled *Received* za prikaz samo primljenih slika. Izgled aplikacije nakon registracije može se vidjeti na slici 4.3 ispod.



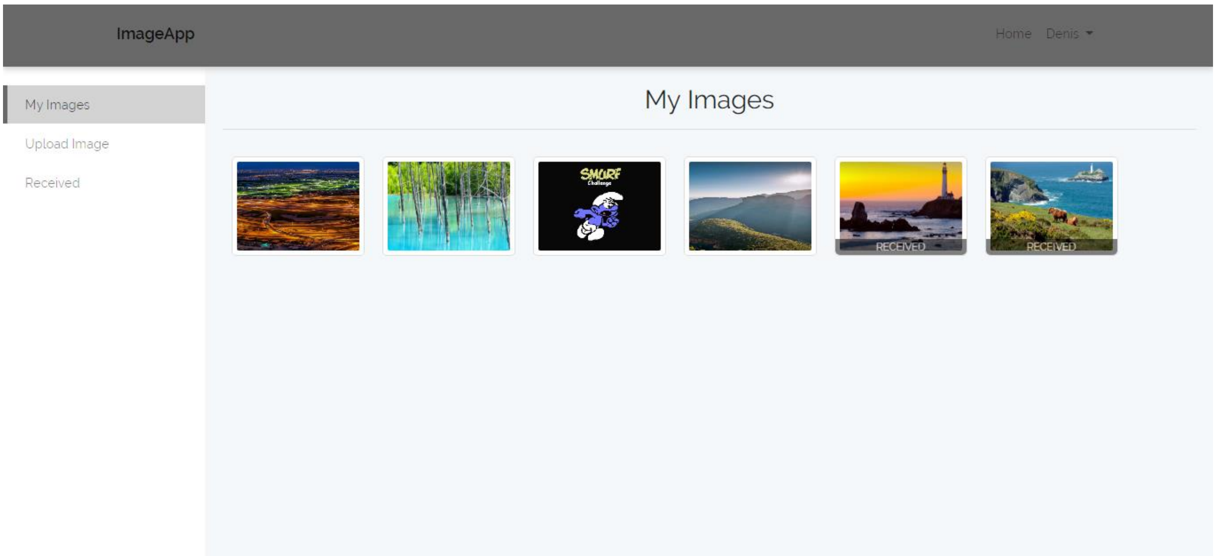
Sl. 4.3. Osnovni pogled nakon logiranja u aplikaciju

Kada je korisnik logiran u aplikaciji, na lijevom izborniku se može prebaciti *Upload Image* pogled, koji mu omogućuje prikaz forme za odabir slike sa računala i spremanje u aplikaciju. Tipovi slika koji se mogu učitati su: JPG, PNG, BMP, GIF. Na slici 4.4. ispod se može vidjeti pogled sa formom za učitavanje slike sa računala.



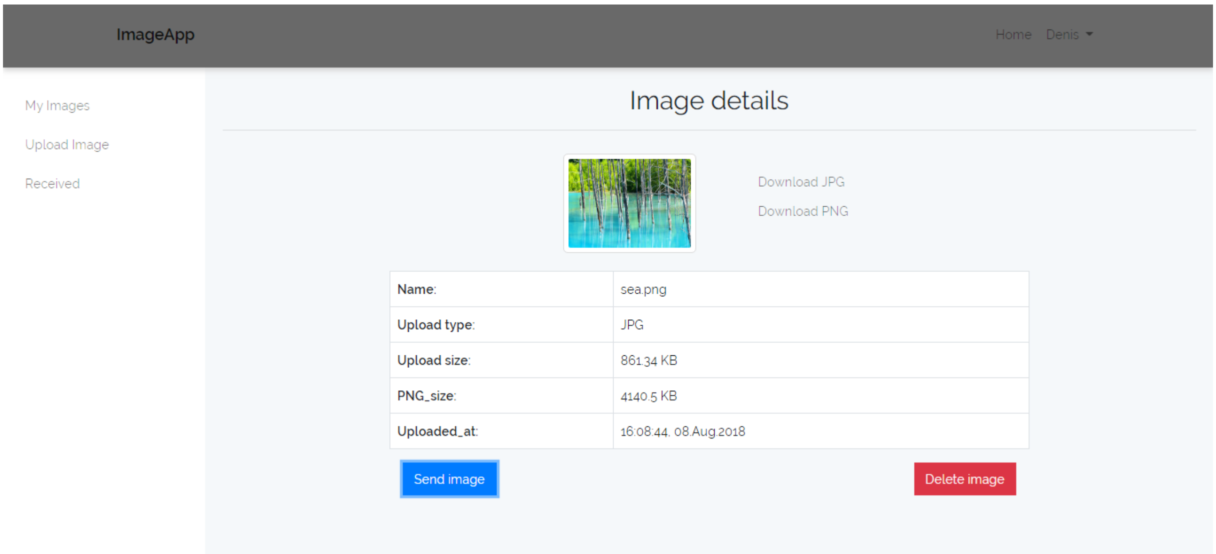
Sl. 4.4. Pogled sa formom za učitavanje slike sa računala

Kada korisnik odabere sliku sa računala i klikne na gumb, slika će se prebaciti u PNG oblik te spremiti u bazu podataka i na poslužitelj. Korisnik se preusmjerava na *My Images* pogled na kojemu se nalaze učitane i primljene slike od drugih korisnika. Kada korisnik želi vidjeti samo primljene slike tada će na lijevom izborniku odabrati pogled *Received*. Izgled pogleda *My Images* sa prikazom učitanih i primljenih slika može se vidjeti na slici 4.5. ispod.



Sl. 4.5. Pogled sa učitanim i primljenim slikama u aplikaciji

Korisnik ima mogućnost pogledati detalje o slici u svom profilu, klikom na sliku otvara se pogled sa detaljima o slici. Na slici 4.6. ispod prikazani su detalji o korisnikovoj učitanoj slici nakon što je korisnik kliknuo na nju, te mogućnost skidanja slike na računalo kao i slanje drugom korisniku.

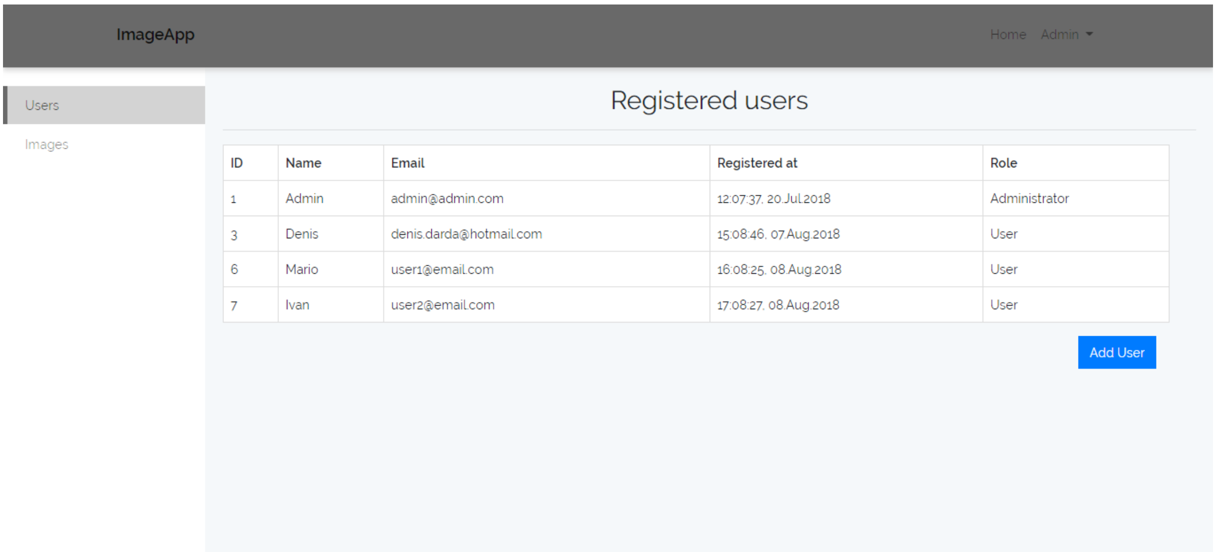


Sl. 4.6. Pogled sa detaljima korisnikove učitane slike i mogućnost slanja

Prema slici 4.6. može se vidjeti mogućnost korisnika da svoju učitanu slika pošalje drugom registriranom korisniku, klikom na gumb *Send Image* kojim se otvara forma za upis *email* adrese drugog korisnika. Ukoliko adresa u bazi podataka ne postoji vratiti će se obavijest korisniku.

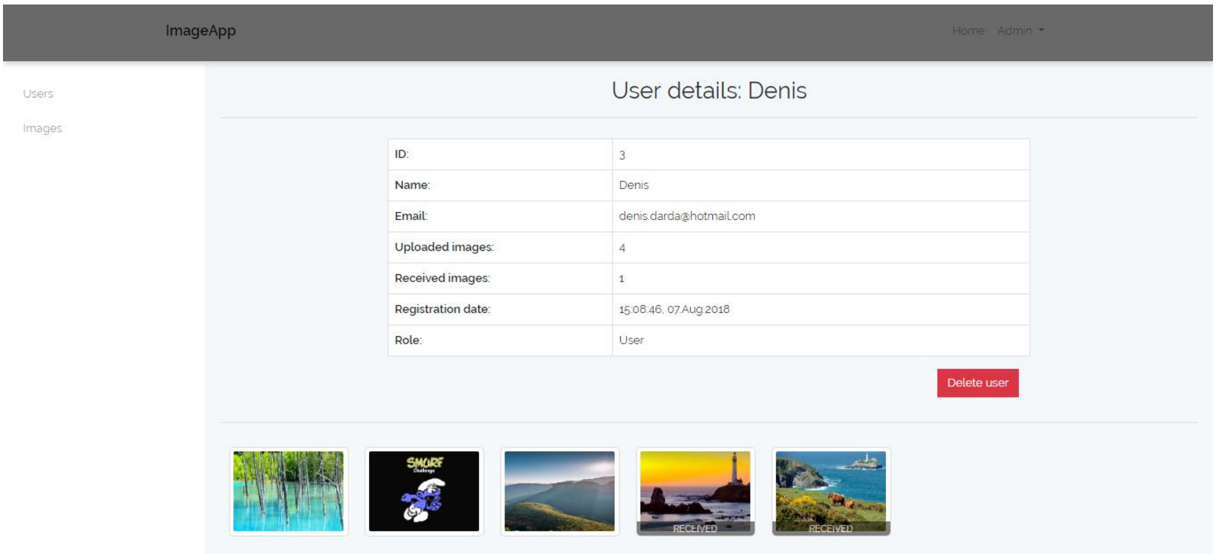
Kada korisnik klikne na jednu od primljenih slika pojaviti će se podaci o slici kao i o imenu korisnika koji je poslao sliku. Klikom na bilo koju sliku u aplikaciji korisnik ima mogućnost skinuti ju u originalnom obliku kao i u PNG obliku koji se nalazi na poslužitelju, što je također vidljivo na slici 4.6. iznad.

Sve prethodno navedeno u vezi korištenja aplikacije odnosi se na korisničku ulogu. Administrator ima mogućnost vidjeti sve registrirane korisnike u aplikaciji kao i njihove učitane i primljene slike. Administrator može dodavati i brisati korisnike. Na slici 4.7. ispod prikazan je pogled sa svim registriranim korisnicima.



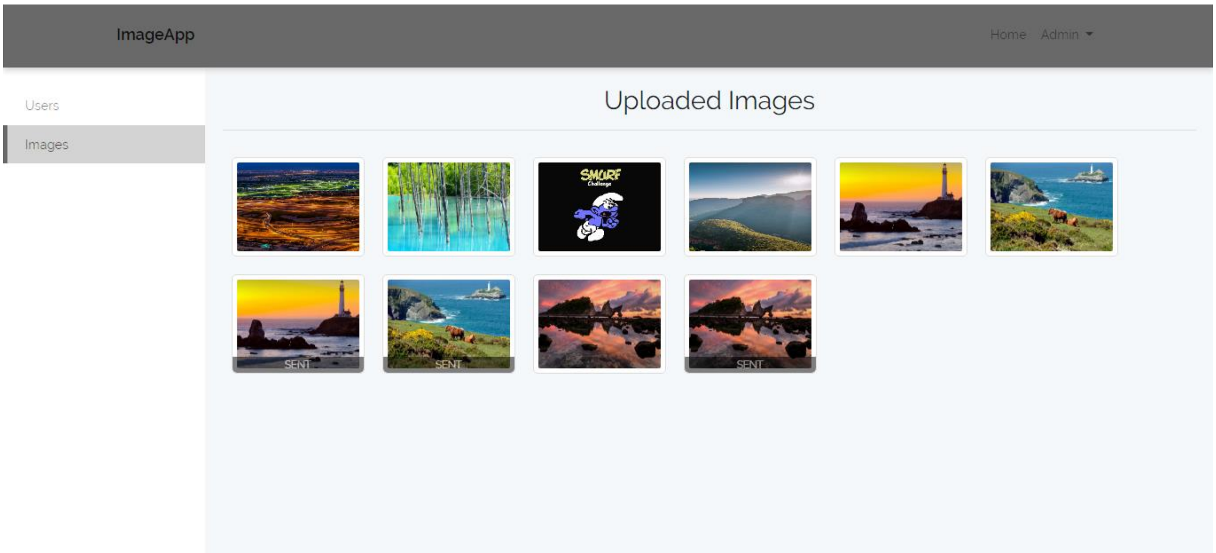
Sl. 4.7. Prikaz svih registriranih korisnika u aplikaciji

Klikom na korisnika otvara se pogled sa detaljima o korisniku sa svim njegovim učitanim i primljenima slikama (Sl.4.8.).



Sl. 4.8. Prikaz korisnika zajedno sa njegovim učitanim i primljenim slikama

Administrator ima uvid u sve učitane slike u aplikaciju zajedno, a ne samo pojedinačno po korisnicima, što se može vidjeti klikom na *Images* u lijevom izborniku. Pogled sa svim učitanim i poslanim slikama u aplikaciji može se vidjeti na slici 4.9 ispod.



Sl. 4.9. Prikaz učitanih i poslanih slika od strane svih korisnika aplikacije

Administrator može pogledati detalje o slici klikom na nju, također može skinuti sliku kao i svi ostali korisnici, ali može i brisati slike od drugih korisnika ukoliko one nisu prikladne za aplikaciju.

ZAKLJUČAK

U radu je opisana izrada aplikacije za pretvorbu i prijenos slikovnih datoteka. Prikazano je korištenje popularnih web tehnologija koje se koriste u izradi modernih web aplikacija. Tema je izabrana kako bi se nadogradila znanja u području web tehnologija i prikazala problematika spremanja, pretvorbe, i razmjene slikovnih datoteka između korisnika. Aplikacija je bazirana na najpopularnijem PHP radnom okviru, Laravel.

Glavni zadatak rada bio je omogućiti pretvorbu BMP slikovnih datoteka u PNG oblik prilikom spremanja slike na poslužitelj te omogućiti njeno slanje drugim korisnicima, što je uspješno odrađeno. Spremanjem BMP slike u PNG oblik štedi se slobodni prostor poslužitelja jer se takvoj slici smanjuje veličina, a ne gubi se na kvaliteti. Nakon pretvorbe slike u prvobitni oblik i skidanja na računalo, korištenjem PSNR metode testirana je njena kvaliteta te je utvrđeno da u odnosu na originalnu BMP sliku nema gubitaka. Također je omogućeno učitavanje drugih tipova slikovnih datoteka u aplikaciju. Treba naglasiti da aplikacija nije isplativa za druge tipove slikovnih datoteka, što je prikazano na JPG slikovnim datotekama, čijom se pretvorbom u PNG oblik povećava veličina slike i gubi na slobodnom prostoru poslužitelja. Prilikom izrade aplikacije pažnja je usmjerena i na prilagodljivost aplikacije različitim veličinama zaslona, što je riješeno korištenjem popularnog Bootstrap radnog okvira. Izradom aplikacije stečena su dodatna znanja u području web programiranja, što se posebno odnosi na korištenje Laravel PHP radnog okvira.

LITERATURA

[1] HTTP – izvor: https://hr.wikipedia.org/wiki/HTTP - datum zadnje posjete: 17.7.2018.

[2] HTML – izvor: https://hr.wikipedia.org/wiki/HTML – datum zadnje posjete: 17.7.2018.

[3] CSS – izvor: http://www.mathos.unios.hr/wp/wp2009-10/P4_CSS.pdf - datum zadnje posjete: 17.7.2018.

[4] dipl. ing. Alen Šimec: Uvod u XHTML, HTML i CSS – izvor: https://bib.irb.hr/datoteka/532594.Skripta_-_Uvod_u_xhtml_html_i_css.pdf - datum zadnje posjete: 17.7.2018.

[5] Saša Fajković: Uvod u Javascript – izvor: http://carpediem.hr/PublikacijeCarpeDiem/Publikacije/JavaScript%20skripta.pdf – datum zadnje posjete: 17.7.2018.

[6] CCERT-PUBDOC-2007-10-206: Sigurnost Javascript programskog jezika – izvor: https://www.cis.hr/www.edicija/LinkedDocuments/CCERT-PUBDOC-2007-10-206.pdf - datum zadnje posjete: 17.7.2018.

[7] Jquery – izvor: https://www.popwebdesign.net/popart_blog/2014/02/sta-je-jquery/ - datum zadnje posjete: 18.7.2018.

[8] Bootstrap – izvor: https://getbootstrap.com/docs/4.1/about/overview/ - datum zadnje posjete: 18.7.2018.

[9] Bootstrap mrežni sistem – izvor: https://getbootstrap.com/docs/4.1/layout/grid/ - datum zadnje posjete: 18.7.2018.

[10] PHP – izvor: https://hr.wikipedia.org/wiki/PHP - datum zadnje posjete: 18.7.2018.

[11] MySQL korisnici – izvor: https://www.mysql.com/customers/ - datum zadnje posjete: 18.7.2018.

[12] https://www.srce.unizg.hr/files/srce/docs/edu/osnovni-tecajevi/d350_polaznik.pdf - datum zadnje posjete: 18.7.2018.

[13] Edin Mujadžević, Srce (Sveučilišni Računski Centar Zagreb) - 2007. – izvor: http://www.mathos.unios.hr/wp/wp2009-10/P10_PHP1.pdf - datum zadnje posjete: 18.7.2018.

[14] Laravel – izvor: https://hr.wikipedia.org/wiki/Laravel - datum zadnje posjete: 21.7.2018.

[15] Napredno web programiranje kolegij, FERIT Osijek – izvor: https://loomen.carnet.hr/pluginfile.php/762136/mod_resource/content/2/Predavanje%203.pdf - datum zadnje posjete: 21.7.2018.

[16] Laravel instalacija – izvor: https://laravel.com/docs/5.6/installation - datum zadnje posjete: 7.8.2018.

[17] Laravel migracije – izvor: https://laravel.com/docs/5.6/migrations - datum zadnje posjete: 7.8.2018.

[18] Laravel Eloquent – izvor: https://laravel.com/docs/5.6/eloquent - datum zadnje posjete: 7.8.2018.

[19] Laravel relacije – izvor: https://laravel.com/docs/5.6/eloquent-relationships - datum zadnje posjete: 7.8.2018.

[20] Laravel kontroleri – izvor: https://laravel.com/docs/5.6/controllers - datum zadnje posjete: 7.8.2018.

[21] Laravel *resources* – izvor: https://laravel.com/docs/5.6/eloquent-resources - datum zadnje posjete: 7.8.2018.

[22] Laravel *middleware* – izvor: https://laravel.com/docs/5.6/middleware - datum zadnje posjete: 7.8.2018.

[23] Laravel validacija – izvor: https://laravel.com/docs/5.6/validation#available-validation-rules - datum zadnje posjete: 7.8.2018.

[24] Laravel *redirects* – izvor: https://laravel.com/docs/5.6/redirects - datum zadnje posjete: 8.8.2018.

[25] Laravel *filesystem* – izvor: https://laravel.com/docs/5.6/filesystem - datum zadnje posjete: 8.8.2018.

[26] Laravel pogledi – izvor: https://laravel.com/docs/5.6/views - datum zadnje posjete: 8.8.2018.

[27] Laravel *blade* predložak – izvor: https://laravel.com/docs/5.6/blade - datum zadnje posjete: 8.8.2018.

[28] Laravel *helpers* metode – izvor: https://laravel.com/docs/5.6/helpers - datum zadnje posjete: 8.8.2018.

[29] PSNR metoda za usporedbu slike – izvor: https://en.wikipedia.org/wiki/Peak_signal-to-noise_ratio - datum zadnje posjete: 14.8.2018.

SAŽETAK

U ovom diplomskom radu opisana je problematika pretvorbe i prijenosa slikovnih datoteka između korisnika. Aplikacija korisniku omogućuje učitavanje slike na poslužitelj i razmjenu iste sa drugim korisnicima. Izrada aplikacije temeljila se na Laravel radnom okviru kao i na popularnim tehnologijama koje se koriste za izradu modernih web aplikacija. U radu je prikazana funkcionalnost aplikacije te su dani primjeri programskih kodova korištenih u njenoj izradi.

Ključne riječi: Bitmap, Laravel, MVC, pretvorba formata, web aplikacija.

ABSTRACT

Title: Application for conversion and exchange image files

This graduation thesis discusses the problem of converting and transferring images between users. Application allows users to upload and exchange images. Application development is based on Laravel PHP framework and popular web technologies for develop modern web applications. Application is thoroughly described and source code examples are given.

Keywords: Bitmap, format conversion, Laravel, MVC, web application.

ŽIVOTOPIS

Denis Kristman rođen je 23.10.1991. godine u Zagrebu. Osnovnu školu pohađao je u Dardi nakon koje se upisuje u Elektrotehničku i prometnu školu Osijek, smjer Tehničar za mehatroniku. Srednjoškolsko obrazovanje završava 2009. godine i iste godine upisuje se na Elektrotehnički fakultet u Osijeku, Stručni studij smjer informatika koji završava 2014. godine. Trenutno je student 2. godine diplomskog studija računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. Od stranih jezika koristi se engleskim.

Potpis: ______________________

PRILOZI

Prilog 1. Na priloženom optičkom disku uz završni rad nalazi se .doc i .pdf verzija rada te kod aplikacije.

Prilog 2. Python skripta za usporedbu kvalitete slikovnih datoteka:

```
import numpy

import math

import cv2


image = cv2.imread("original_image.jpg")

imageconv = cv2.imread("converted_image.jpg")

def psnr(img1, img2):

    mse = numpy.mean( (img1 - img2) ** 2 )

    if mse == 0:

        return 100

    PIXEL_MAX = 255.0

    return 20 * math.log10(PIXEL_MAX / math.sqrt(mse))


result=psnr(image,imageconv)

print(result)
```