

MATEMATIČKI FAKULTET  
UNIVERZITETA U BEOGRADU



# MASTER RAD

*Korišćenje radnog okruženja Laravel za izradu  
veb aplikacija i obradu podataka*

Mentor:

Prof. dr Dušan Tošić

Student:

Ivana Dražić

**Beograd, 2016.**

*Korišćenje radnog okruženja Laravel za razvoj veb aplikacija i obradu podataka*

*Apstrakt*

Cilj izrade ovog master rada bio je da se pokaže kako je moguće pomoću Laravela lako napraviti veb aplikaciju za obradu podataka. Laravel je korišćen kao besplatno okruženje otvorenog koda programskog jezika PHP dizajnirano za razvoj MPK (engl. MVC, Model-View-Controller) veb aplikacija. Laravel je poslednjih nekoliko godina postao veoma popularan, pre svega zbog svojih velikih mogućnosti. Aplikacija koja je obrađena u radu prikazuje samo mali deo mogućnosti samog razvojnog okruženja Laravel. Takođe, prepravkom aplikacije ona se može koristiti i za razne druge obrade podataka.

*The use of the operating environment Laravel for web application development and data processing*

*Abstract*

The goal of this master study was to show that it is possible, using Laravel, to easely make a web application for data processing. Laravel is used as a free and open source framework for PHP programming language, designed for developing MVC web application. Laravel has become very popular in recent years because of its great features. The application discussed in this paper demonstrates only a small part of the possibilities available in Laravel. Also, by altering the application, it can be used for a variety of other data processing.

## **Sadržaj**

1	Uvod.....	3
1.1	O aplikaciji .....	3
1.2	O Laravelu .....	3
2	MPK (engl. MVC, Model-View-Controller) .....	5
2.1	MPK za veb aplikacije.....	5
2.1.1	Model .....	6
2.1.2	Pogled (eng. View).....	6
2.1.3	Kontroler (eng. Controller) .....	7
2.1.4	Međudelovanje komponenata .....	7
3	Način izvršenja programa u Laravelu .....	9
3.1	Sintaksa.....	9
4	Blejd (engl. Blade) .....	10
5	Artisan (engl. Artisan) .....	11
6	Opis implementacije aplikacije .....	13
6.1	Opis aplikacije .....	13
6.2	Instalacija razvojnog okruženja .....	13
6.2.1	Instalacija LAMP servera.....	13
6.2.2	Instalacija Laravela .....	14
6.2.3	Izrada baze podataka.....	15
6.3	Putanje (Rute).....	18
7	Pogledi i šabloni (izgled strana aplikacije) .....	21
8	Modeli .....	31
9	Kontroleri .....	33
10	Proširenje aplikacije .....	36
11	Zaključak.....	37
12	Spisak slika i tabela.....	38
	Reference .....	39

## **1 Uvod**

### **1.1 O aplikaciji**

Danas se internet koristi svakodnevno, samim tim i veliki broj veb aplikacija. Kako se povećavala potreba za veb aplikacijama, tako je dolazilo i do razvoja velikog broja programskih jezika, kao i raznih alata i okruženja programskih jezika za razvoj veb aplikacija. Jedan od tih programskih jezika je PHP, a jedno od njegovih radnih okruženja je Laravel[1]. Ideja da se koristi Laravel u izradi veb aplikacije je proizišla zbog njegove velike popularnosti i ogromnih mogućnosti koje pruža.

Laravel se može upotrebiti za izradu raznih aplikacija vezanih za školstvo. Jedna od korisnih bi bila aplikacija vezana za obradu podataka u vezi sa završnim ispitom posle završetka osnovne škole. Svršeni učenici osmog razreda polažu završni ispit na kraju osnovnog obrazovanja. Završni ispit se sastoji iz tri dela: test iz srpskog (maternjeg) jezika, test iz matematike i kombinovani test (fizika, hemija, biologija, istorija, geografija). Na osnovu bodova osvojenih na završnom ispitu i bodova koje nosi uspeh učenika iz šestog, sedmog i osmog razreda, učenici popunjavaju listu želja za upis u srednju školu. Pri izboru srednje škole treba voditi računa o konkurenciji. Često je potrebno znati koliko su drugi učenici imali bodova, kao i kakvo je rangiranje škole.

U aplikaciji će se, na osnovu unetih podataka za svakog učenika, računati koji je najbolje urađen zadatak, koji učenik ima najveći broj bodova i koje je odeljenje u proseku najbolje uradilo test iz matematike. Unose se sledeći podaci o učeniku: ime i prezime, šifra i odeljenje. Zatim se za svakog učenika popunjava lista sa zadacima: zadatak bez odgovora, pogrešno urađen zadatak ili tačno urađen zadatak. Po završetku unošenja podataka za svakog učenika, aplikacija ispisuje statistički obrađene podatke.

Postoji mogućnost proširenja aplikacije, o čemu će biti reči u poglavlju broj 10.

### **1.2 O Laravelu**

Razvojem interneta i veb aplikacija, mnogi programski jezici nastaju, unapređuju se i nadgrađuju. Tako postoji nekoliko verzija programskog jezika PHP, kao i mnoga njegova razvojna okruženja.[4]

Laravel je besplatno razvojno okruženje otvorenog koda programskog jezika PHP dizajnirano za razvoj MPK (engl. MVC, Model-View-Controller) veb aplikacija. U poslednjih nekoliko godina je postao veoma popularan među programerima. Objavljen je pod MIT licencom i njegov izvorni kôd se nalazi na GitHubu[2].Njegove glavne karakteristike su stabilnost, optimizacija resursa i brzina učitavanja.

U avgustu 2009. godine PHP 5.3[3] je zvanično objavljen. Nove mogućnosti su dozvoljavale programerima da pišu bolje objektno orjentisane PHP aplikacije. Razvojna okruženja programskog jezika PHP u to vreme su bila: Symfony, Zend, Slim micro framework, Kohana, Lithium i CodeIgniter. CodeIgniter je bio najpopularniji, a programeri ga vole zbog sveobuhvatne dokumentacije i jednostavnosti. Nedostajalo mu je malo funkcionalnosti koju je Tejlor Otvel (engl. Taylor Otwell), tvorac Laravela, smatrao veoma važnom u izradi veb aplikacija.

Probna verzija Laravel 1 se pojavljuje u junu 2011[4]. Trebalo je da popuni

funkcionalnosti koje su nedostajale i da se pomoću njega jednostavno reše rastući problemi korišćenja CodeIgnitera.

### *Laravel 1*

Počevši od prvog objavljivanja, Laravel se istakao u proveru identiteta, elokventnom ORM (Objektno Relaciono Mapiranje) za operacije sa bazama podataka, lokalizaciji, modelima i odnosima, jednostavnošću mehanizama za usmeravanje, keširanje, formama u HTMLu i drugom. Čak i prvo izdanje je imalo veoma impresivnu funkcionalnost. U tom trenutku Laravel još nije podržavao MPK okvir, jer nije posedovao kontrolere, ali su programeri odmah zavoleli njegovu čistu sintaksu i potencijal. Za manje od šest meseci Laravel je iz verzije 1 prešao u verziju 2.

### *Laravel 2*

Verzija 2 je objavljena samo za programere u novembru 2011. godine. Implementirane su sledeće funkcije: podrška kontrolerima, Blejd (engl. Blade) šablonski motor, upotreba inverzije kontrolnog kontejnera. Zahvaljujući novim kontrolerima, okruženje je postalo potpuno kvalifikovani MPK okvir.

### *Laravel 3*

U februaru 2012. godine objavljena je Laravel verzija 3 uz novi sajt sa dokumentacijom i sa velikim brojem novih karakteristika. Ovo izdanje je fokusirano na test integracije, Artisan komandnu liniju, baze podataka i još mnogo toga. Laravel 3 je bio najstabilniji i dovoljno snažan da se koristi za mnoge vrste veb aplikacija, nudeći jednostavnost uz kratko vreme za učenje.

### *Laravel 4*

Laravel 4 je zvanično objavljen u maju 2013. godine. Iako je objavljivanje nove verzije često znak da se okvir razvija, ovde se kredibilitet smanjuje. Neki programeri su ga okarakterisali kao suviše brz i nestabilan. Oni su morali da se prebacuju na novu verziju, a to je ponekad nemoguće sa velikim aplikacijama već izgrađenim na prethodnoj arhitekturi. Laravel 4 je ponovo napisan od početka kao kolekcija komponenata (ili paketa) koje, međusobno integrisane, čine okvir. Rukovođenje ovim komponentama vrši se preko menadžera programskog jezika PHP dostupnog pod nazivom Kompozer (engl. Composer).

### *Laravel 5*

Laravel 5 je zvanično objavljen u januaru 2015. godine. Ova verzija sadrži više od 22 nove mogućnosti u odnosu na prethodne verzije.

Laravel je prešao dug put za nešto manje od dve godine i privlači sve više programera širom sveta. Vizionar Tejlor Otvel i zajednica okruženja Laravel su napravili ogroman napredak stvorivši arhitekturu za PHP veb aplikacije u kratkom vremenskom roku. Broj korisnika i saradnika stalno raste što znači da će se Laravel još dugo koristiti.

## **2 MPK (engl. MVC, Model-View-Controller)**

U poslednjih nekoliko godina Model-Pogled-Kontroler [5] (MPK) je najčešće korišćen obrazac u svetu veb programiranja.

MPK je obrazac softverske arhitekture koji opisuje način kako da se struktura aplikacija. MPK se koristi za odvajanje pojedinih delova aplikacije i njenih komponenata u zavisnosti od njihove namene. Svaki deo aplikacije ima svoju odgovornost, nalazi se u interakciji sa preostala dva dela i pri tom se rešava problem u organizaciji koda. Ovakva arhitektura olakšava nezavisan razvoj, testiranje i održavanje aplikacije. Na taj način se olakšava posao programerima, jer menjanjem jednog sloja aplikacije, ne moraju se nužno promeniti i ostala dva sloja. Takođe, olakšava se raspodela posla među programerima u timu.

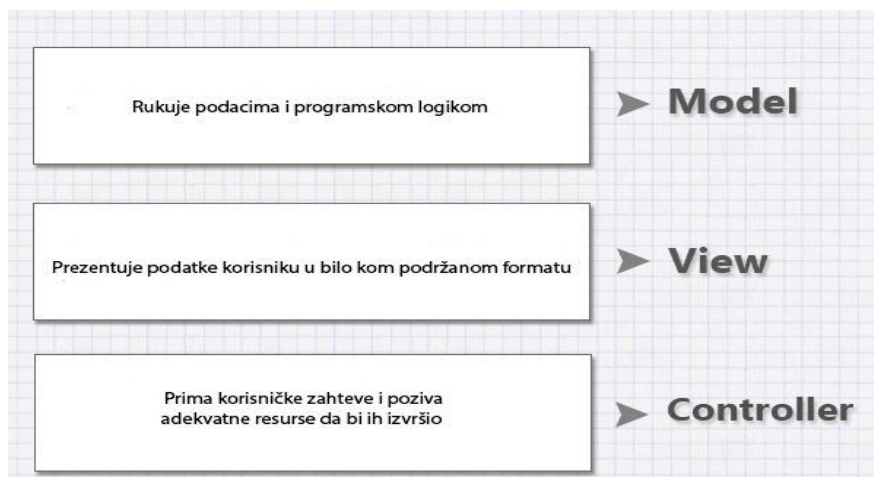
MPK arhitektura je prvobitno formulisao krajem sedamdesetih godina prošlog veka norveški informatičar Trigvi Riensku (norv. Trygve Reenskauga), kao deo *Smalltalk* programskog jezika[6]. Danas se MPK koristi u velikom broju modernih veb i *GUI* razvojnih okruženja kao što su: *CodeIgniter*, *Ruby on Rails*, *Apple Cocoa*, *Apache Struts*... U osnovi ove arhitekture nalaze se dve centralne ideje: ponovno korišćenje već postojećeg koda i jasna raspodela zaduženja među različitim delovima sistema. Prva ideja omogućava da se jednom napisan kôd uz minimalne ili čak nikakve izmene može koristiti u više različitih projekata. Druga ideja je podela sistema na više međusobno nezavisnih celina od kojih svaka ima svoj zadatak i koje se pojedinačno mogu modifikovati, bez bojazni da će se te promene odraziti na ostatak sistema.

MPK je prvi put predstavljen 1979. godine, ali mu je koncept bio drugačiji. U to vreme koncept veb aplikacija nije postojao, tako da je šablon, koji danas koristimo u veb programiranju adaptacija prvobitne arhitekture. Do popularizacije ovog šablona u veb programiranju dovela su dva najkorišćenija razvojna okruženja *Struts* i *Ruby on Rails*. Ova dva razvojna okruženja su utabala put hiljadama drugih razvojnih okruženja koja se danas koriste u razvoju veb aplikacija.

### **2.1 MPK za veb aplikacije**

Sušтина MPK obrasca je razdvajanje prezentacijskog sloja od same logike aplikacije. Prezentacijski sloj ili pogled (engl. *View*) je, jednostavno rečeno, interfejs koji korisnik vidi i koji obezbeđuje interakciju sa korisnikom. Model (engl. *Model*) predstavlja logiku same aplikacije, odnosno podatke i strukture pomoću kojih se operiše. Logika obrasca je takva da model "ne zna" za postojanje pogleda i obrnuto. Kontroler (engl. *Controller*) ima ulogu koordinatora. Poštovanjem logike MPK aplikacije, svaki veb zahtev dolazi direktno do odgovarajućeg kontrolera, koji pomoću modela generiše potrebne podatke, prosleđuje ih pogledu i na kraju se odgovarajući pogled vraća klijentu, odnosno veb pretraživaču.

Ideja iza MPK arhitekture je jednostavna: podelićemo odgovornosti između različitih slojeva aplikacije. Na *Slici 1* prikazani su slojevi MPK arhitekture.



*Slika 1. – Slojevi MPK arhitekture*

Aplikacija je podeljena na tri glavne komponente, a svaka od njih obavlja različite zadatke.

### 2.1.1 Model

Model [7] sadrži glavne programske podatke kao što su informacije o objektima iz baze podataka, kao i SQL upite. Sva poslovna logika aplikacije se nalazi u modelu. Svi podaci se dobijaju od modela, ali se on ne može direktno pozvati, već je kontroler taj koji od modela zahteva određene podatke. Model zatim obrađuje zahteve i vraća podatke kontroleru, a ovaj ih dalje prosleđuje pogledu koji ih interpretira krajnjem korisniku.

Model predstavlja jednu ili više klasa sa svojim stanjima koja se mogu prikazivati na zahtev pogleda, a kontroler ih može menjati.

Postoje aktivan i pasivan model. Pasivan model je model čije se stanje menja sa učešćem kontrolera, a aktivan model je model čije se stanje menja bez učešća kontrolera.

Međutim, u praksi je češći scenario u kojem model ima aktivnu ulogu i predstavlja deo poslovne logike. U tom slučaju, model obezbeđuje metode koje će se koristiti kada se ažurira trenutno stanje objekta. Model takođe može obavestiti pogled da je došlo do promene, da bi se osvežio trenutni prikaz.

### 2.1.2 Pogled (eng. View)

Pogled [7] je poslednji sloj MPK arhitekture koji sadrži korisničko okruženje aplikacije, odnosno obezbeđuje različite načine za prezentovanje podataka koje dobija od modela. U veb aplikacijama pogled sadrži: HTML, CSS, JavaScript, XML ili JSON, itd. Ne preporučuje se korišćenje HTML, CSS ili JavaScript u kontroleru. Korisnik može videti ono što pogled prikazuje, dok su model i kontroler skriveni. Pogled je vizuelni prikaz modela koji sadrži metode za prikazivanje i omogućava korisniku da menja podatke. On svakako ne treba da bude nadležan za skladištenje podataka, osim kada se koristi keširanje kao mehanizam poboljšanja performansi. Može biti odgovoran za prikaz jednog ili više objekata iz modela. Sam pogled nikada ne obrađuje podatke – njegov posao je gotov kada su podaci prikazani.

### 2.1.3 Kontroler (eng. Controller)

Kontroler [7] sadrži glavne mehanizme za kontrolu programa i odgovoran je za njegov tok. U veb aplikacijama on predstavlja prvi sloj koji se poziva kada pretraživač pozove URL adresu. Takođe, kontroler upravlja korisničkim zahtevima (HTTP GET ili POST zahtevima kada korisnik klikne na neki GUI element). Obično kontroler poziva određeni model za zadatak i zatim bira odgovarajući pogled. Kontroler interpretira ulazne podatke korisnika i prosleđuje ih do modela ili pogleda. On odlučuje kako model treba da se promeni kao rezultat korisničkih zahteva i koji pogled treba da se koristi. On u stvari obezbeđuje vezu između modela i pogleda.

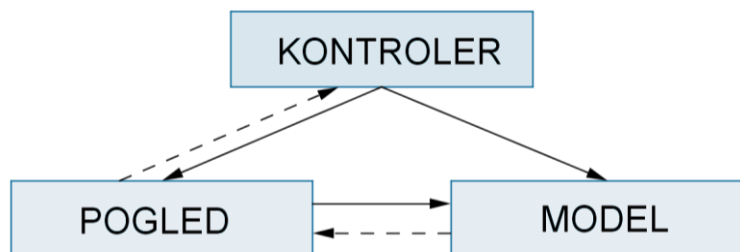
### 2.1.4 Međudelovanje komponentata

Osim podele aplikacije u tri odvojene komponente, MPK arhitektura omogućava i interakciju između njih.

Model je centralna komponenta. On direktno upravlja podacima, logikom i pravilima aplikacije. Takođe, on skladišti podake koji se preuzimaju od strane kontrolera i prikazuju se u pogledu. Kad god postoji promena, kontroler ažurira model. Model ne mora da zna ko su kontroler i pogled.

Kontroler prihvata zahteve od klijenata za izvršenje operacije. Nakon toga poziva operaciju koja je definisana u modelu i, ukoliko model promeni stanje, obavešava pogled o promeni stanja modela. Kontroler može da šalje komande modelu za ažuriranje njegovog stanja.

Pogled od modela traži informacije koje koristi za generisanje izlaznih podataka. On takođe obezbeđuje korisniku interfejs pomoću koga unosi podatke i poziva odgovarajuće operacije koje je potrebno izvršiti nad modelom. On, u stvari, korisniku prikazuje stanje modela. Na *Slici 2* je prikazana međusobna interakcija između modela, kontrolera i pogleda.



*Slika 2. – Međudelovanje komponentata*

MPK obrazac ima nekoliko pozitivnih osobina: model se može prikazati na više načina; lakše je dodati novi pogled (na primer: novu internet stranicu koja prikazuje postojeće podatke ili deo njih); interakcija sa korisnikom se lako menja; više programera mogu raditi istovremeno i paralelno na različitim slojevima aplikacije (posebno važno u timovima specijalizovanim za pojedine slojeve, na primer, tim zadužen za implementaciju i održavanje korisničkog interfejsa, poslovne logike...)

Najvažnije prednosti MPK arhitekture su ponovno korišćenje kôda i podela sistema na međusobno nezavisne celine, odakle proizilaze i ostale prednosti. Projekti su mnogo sistematičniji, njihovi pojedinačni delovi se mogu lako menjati i poboljšavati, kôd pisan na ovaj način je neuporedivo lakše testirati, paralelni razvoj aplikacije se može lako organizovati i samim tim se povećava produktivnost.

MPK ima, naravno, i svoje mane. Previše je kompleksan za implementaciju kod razvoja



### ***Korišćenje radnog okruženja Laravel za razvoj veb aplikacija i obradu podataka***

manjih aplikacija i njegovo korišćenje u tim slučajevima dovodi do pogoršanja kako dizajna tako i performansi. Ponekad se može desiti da, usled čestih promena modela, pogled bude preplavljen zahtevima za izmenu. Ukoliko on služi za prikazivanje sadržaja, kojem je potrebno određeno vreme za stvaranje slike koji prikazuje, česti zahtevi za izmenu mogu dovesti do kašnjenja.

### **3 Način izvršenja programa u Laravelu**

Program koji se napiše u Laravelu ne zahteva prevođenje [8] (kompajliranje), nego se interpretira [9] pri svakom izvršavanju.

#### **3.1 Sintaksa**

Laravel ne poseduje početnu (glavnu) funkciju, već jednostavno sadrži skup naredbi, koje se izvršavaju jedna za drugom, od prve do poslednje, gde se poslednja ujedno smatra i krajem programa.

Laravel kôd može biti organizovan u funkcije i klase, pomoću kojih može biti smešten u više datoteka. Kao početna datoteka, tj. datoteka čije naredbe se izvršavaju prve, uzima se ona koja se daje interpretatoru na izvršavanje. Ukoliko datoteka nije navedena, server će automatski pokušati da pokrene datoteku *index.php*.

Da bi se napravila kompleksna Laravel aplikacija, potrebno je najpre napraviti rutu (putanju) koja povezuje model, kontroler i pogled.

Sama sintaksa Laravela je ista kao kod programskog jezika PHP. Sintaksa programskog jezika PHP se može naći na raznim sajtovima, kao i u velikom broju knjiga.[10, 11]

## 4 Blejd (engl. Blade)

Blejd je jednostavan, ali veoma jak alat Laravela i služi za pravljenje interfejs šablona. Za razliku od drugih sličnih alata, Blejd nema ograničenja u korišćenju PHP koda u svom fajlu. Blejd dokumenti imaju ekstenziju *.blade.php*. Oni se nalaze u datoteci *resources/views*.<sup>[12]</sup>

Dve najveće prednosti korišćenja Blejda su nasleđivanje šablona i određenih sekcija. Direktiva *@section* definiše deo (poglavlje) sa sadržajem, dok *@yield* služi za prikazivanje sadržaja svakog poglavlja.

Sledi deo koda koji se nalazi u *resources/views/layouts/nasledna\_strana.blade.php*, a gde vidimo primenu direktive *@section*:

```
<html>
  <head>
    <title>Naziv aplikacije - @yield('title')</title>
  </head>
  <body>
    @section('sidebar')
      Ovo je glavna traka.
    @show

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

Direktiva *@extends* nam govori koji šablon ćemo naslediti.

Navodimo deo koda koji se nalazi u *resources/views/layouts/nasledna\_strana.blade.php*, a koji nam govori koji deo nasleđujemo:

```
@extends('layouts.glavna_strana')
@section('title', 'Naziv strane')

@section('sidebar')
  @parent
  <p>Ovo je dodatak na glavnu traku.</p>
@endsection

@section('content')
  <p>Telo dokumenta.</p>
@endsection
```

Deo u kom dolazi do nasleđivnja se mora odvojiti u sekciju.

Blejd šablon se može pozvati i preko putanja korišćenjem funkciju *view*:

```
Route::get('blade', function () {
    return view('ime_strane');
});
```

## **5 Artisan (engl. Artisan)**

Artisan je ime komandne linije u Laravelu. Ona nam pruža mnogo korisnih komandi dok razvijamo našu aplikaciju. [13]

Da bismo mogli da vidimo listu komandi koje nam pruža Artisan, koristićemo sledeću komandu koju ćemo kucati u terminalu:

```
php artisan list
```

Na *Slici 3* se nalazi detaljan prikaz terminala sa listom komandi koje nam pruža Artisan. Svaka komanda ima opciju za pomoć.

```
php artisan help ime_komande
```

Pozivom komande za pomoć, na terminalu nam se ispisuje komanda sa svojim atributima, a zatim njen opis, kao i opis atributa te komande.

Pored komandi koje idu uz Artisan komandnu liniju, postoji mogućnost da sami napravimo svoje komande. Preporuka je da se te komande čuvaju u datoteci *app/Console/Commands*. Međutim, komande se mogu čuvati i u nekoj datoteci po izboru programera, sve dok se one automatski učitavaju na osnovu podešavanja u fajlu *composer.json*.

```
ivana@ivana-LIFEB00K-A544:~/www/ivana/laravel$ php artisan list
Laravel Framework version 5.0.18

Usage:
  command [options] [arguments]

Options:
  --help (-h)            Display this help message
  --quiet (-q)           Do not output any message
  --verbose (-v|vv|vvv) Increase the verbosity of messages: 1 for normal output,
  2 for more verbose output and 3 for debug
  --version (-V)         Display this application version
  --ansi                 Force ANSI output
  --no-ansi              Disable ANSI output
  --no-interaction (-n) Do not ask any interactive question
  --env                  The environment the command should run under.

Available commands:
  clear-compiled          Remove the compiled class file
  down                   Put the application into maintenance mode
  env                    Display the current framework environment
  fresh                  Remove the scaffolding included with the framework
  help                   Displays help for a command
  inspire               Display an inspiring quote
  list                   Lists commands
  migrate                Run the database migrations
  optimize               Optimize the framework for better performance
  serve                  Serve the application on the PHP development server
  tinker                 Interact with your application
  up                    Bring the application out of maintenance mode
  app
  app:name               Set the application namespace
  auth
  auth:clear-resets     Flush expired password reset tokens
  cache
  cache:clear           Flush the application cache
  cache:table           Create a migration for the cache database table
  config
  config:cache          Create a cache file for faster configuration loading
  config:clear          Remove the configuration cache file
  db
  db:seed               Seed the database with records
  event
  event:generate        Generate the missing events and handlers based on registra
tion
  handler
  handler:command       Create a new command handler class
  handler:event         Create a new event handler class
  key
  key:generate          Set the application key
  make
  make:command          Create a new command class
  make:console          Create a new Artisan command
  make:controller       Create a new resource controller class
  make:event            Create a new event class
  make:middleware       Create a new middleware class
  make:migration        Create a new migration file
  make:model            Create a new Eloquent model class
  make:provider         Create a new service provider class
  make:request          Create a new form request class
  migrate
  migrate:install       Create the migration repository
  migrate:refresh       Reset and re-run all migrations
  migrate:reset         Rollback all database migrations
  migrate:rollback      Rollback the last database migration
  migrate:status        Show a list of migrations up/down
  queue
  queue:failed          List all of the failed queue jobs
  queue:failed-table   Create a migration for the failed queue jobs database tabl
e
  queue:flush           Flush all of the failed queue jobs
  queue:forget         Delete a failed queue job
  queue:listen         Listen to a given queue
  queue:restart        Restart queue worker daemons after their current job
  queue:retry          Retry a failed queue job
  queue:subscribe      Subscribe a URL to an Iron.io push queue
  queue:table           Create a migration for the queue jobs database table
  queue:work           Process the next job on a queue
  route
  route:cache           Create a route cache file for faster route registration
  route:clear          Remove the route cache file
  route:list           List all registered routes
  schedule
  schedule:run         Run the scheduled commands
  session
  session:table        Create a migration for the session database table
  vendor
  vendor:publish        Publish any publishable assets from vendor packages
ivana@ivana-LIFEB00K-A544:~/www/ivana/laravel$
```

Slika 3. – Lista komandi Artisan komandne linije

## **6 Opis implementacije aplikacije**

### **6.1 Opis aplikacije**

Aplikacija se sastoji iz dva dela, administratorskog dela (deo koji nije vidljiv korisniku, već pristup njemu imaju samo administratori koji mogu da kontrolišu podatke) i korisničkog dela (deo koji je vidljiv svima i gde se nalaze već obrađeni podaci, kao i statistički deo aplikacije). Administratorski deo se takođe sastoji iz dva dela: deo za korisnike sa administratorskim pravima i deo za korisnike bez administratorskih prava. Korisnički deo se sastoji iz tri dela: statistika zadataka, statistika učenika i statistika odeljenja.

Prijavljivanje u administratorski deo se vrši pomoću imejl adrese i šifre korisnika. Posle prijavljivanja u administratorski deo, pojavljuju se još dve kartice: lista učenika i administrator. U delu administrator imamo mogućnost dodavanja novog korisnika, odeljenja i učenika, kao i mogućnost promene postojećih podataka. Svaki korisnik sa privilegijama administratora može dodati novog korisnika koji može biti takođe administrator ili korisnik kome će biti dozvoljeno samo da unosi podatke u vezi testa tako što će dodati njegovo korisničko ime, imejl adresu, šifru za prijavljivanje i privilegije. Ako nekome dodeljujemo privilegiju administratora, u polje privilegije kucamo *admin*. U suprotnom korisnik nema privilegije administratora, već samo mogućnost unošenja i menjanja podataka sa testa.

Korisnički deo se sastoji od tri kartice na kojima se nalaze obrađeni podaci i kartice za prijavljivanje u administratorski deo aplikacije. Kartica, na kojoj se nalazi statistika zadataka, sadrži i stubičasti dijagram koji pokazuje koliko je učenika tačno uradilo određeni zadatak. Takođe, ispisuje se rečenica u kojoj se kazuje koji zadatak je tačno urađen najviše puta, kao i koliko puta. Na kartici na kojoj se ispisuje statistika učenika (koja sadrži ispisane statističke podatke učenika) se nalaze podaci o maksimalnom broju bodova, kao i ime i šifra učenika sa najvećim brojem bodova. Na kartici, sa statističkim podacima odeljenja, nalaze se podaci o tome koje odeljenje je najbolje uradilo test, kao i koji je prosečan broj bodova za to odeljenje.

Izrada aplikacije se može podeliti u tri celine:

- Instalacija razvojnog okruženja,
- Izrada baze podataka, konekcija sa bazom podataka,
- Obrada podataka.

### **6.2 Instalacija razvojnog okruženja**

#### 6.2.1 Instalacija LAMP servera

Čitava aplikacija je rađena na lokalnom nivou.

Aplikacija je rađena pod operativnim sistemom *Linux Ubuntu 14.04*. [14]

Linuks (engl. Linux) je operativni sistem slobodnog softvera i otvorenog koda. Može se besplatno skinuti sa zvaničnog sajta [15] i svako ima pravo da ga slobodno koristi i menja njegov kôd. Ubuntu Linuks je operativni sistem koji je baziran na Debian Linuks distribuciji. Regularno izdaje nove verzije svakih šest meseci. Ima lakoću korišćenja i sigurnost.

Da bismo mogli da radimo sa radnim okruženjem Laravel u offline modu, potrebno je da instaliramo LAMP server.

LAMP je akronim za skup paketa slobodnog softvera [16] otvorenog koda [17] koji se odnosi na prva slova:

**L** – Linux (Linuks operativni sistem)

**A** – Apache (Apači server – platforma veb servera)

**M** – MySQL (sistem za upravljanje bazama podataka)

**P** – PHP (softver za veb programiranje - ponekad Perl ili Python).

LAMP komponente su u velikoj meri zamenljive i ne ograničavaju se na početni izbor. Njegov ekvivalent u Microsoft Windows operativnom sistemu je WAMP (Windows, Apache, MySQL, PHP).

Instalacija:

1. Instalacija Apache servera

```
$ sudo apt-get install apache2
```

2. Instalacija MySQLa

```
$ sudo apt-get install mysql-server
```

3. Instalacija PHP – a

```
$ sudo apt-get install php5 libapache2-mod-php5
```

4. Restartujemo server

```
$ sudo /etc/init.d/apache2 restart
```

### 6.2.2 Instalacija Laravela

Navedena instalacija se odnosi na operativni system Linux Ubuntu 14.04.

Pokrećemo komandni prozor.

Da bismo instalirali Laravel, potrebno je da prethodno instaliramo PHP menadžer Kompozer (engl. Composer).

Komande idu sledećim redosledom:

1. Pravimo folder gde će se nalaziti instalacioni fajl

```
$ mkdir instalacija
```

2. Dolazimo do foldera gde se nalazi instalacioni fajl

```
$ cd instalacija
```

3. Preuzimamo instalacioni fajl i smeštamo ga u folder instalacija

```
$ curl -sS https://getcomposer.org/installer | php
```

```
$ sudo cp composer.phar /usr/local/bin/composer
```

4. Instaliramo Kompozer

```
$ php composer.phar
```

Kompozer [18] je sada instaliran globalno. Kompozer je alat za upravljanje bibliotekama i paketima u PHP programskom jeziku. On nije klasičan alat za upravljanje, već pomoću njega se upravlja tačno određenim bibliotekama i paketima koji su potrebni za naš projekat, u određenom direktorijumu. Podrazumeva se da pomoću njega ne instaliramo ništa globalno, već se instaliranje vrši samo u određenom direktorijumu našeg projekta. Zato se i naziva menadžer zavisnosti.

1. Dolazimo do foldera gde će nam se nalaziti Laravel

```
$ cd /home/ivana/www/master
```

2. Pozivamo Kompozer da nam instalira Laravel

```
$ composer create-project laravel/laravel --prefer-dist  
Laravel projekat može da počne!
```

### 6.2.3 Izrada baze podataka

Treba napomenuti da Laravel 5.0 trenutno podržava četiri sistema za upravljanje bazama podataka i to:

- MySQL (korišćen u radu)
- Postgres
- SQLite
- SQL Server

Kod Laravela je povezivanje sa bazama podataka i izvršenje upita krajnje jednostavno. Konfiguracija baza podataka za aplikaciju se nalazi u dokumentu *config/database.php*. U ovom dokumentu se mogu definisati sve veze baza podataka kao i odrediti koju vezu treba koristiti kao primarnu. Primeri za sve podržane baze podataka se nalaze u ovom fajlu, pa programer može da izabere sa kojom želi da radi.

Pri instalaciji LAMP servera, instalirali smo i alat za upravljanje bazama podataka *phpmyadmin*. *PhpMyAdmin* [20] predstavlja besplatan alat otvorenog koda napisan u PHP programskom jeziku, koji služi za upravljanje bazama podataka putem veba. *PhpMyAdmin* je grafički korisnički interfejs (GUI) koji se koristi za upravljanje podacima MySQL baze podataka.

U *PhpMyAdminu* kreiramo novu bazu podataka pomoću funkcije *create database*. Kada smo kreirali našu bazu u datoteci *laravel*, prikažemo skrivena dokumenta i zatim u dokumentu *.env* izmenimo podatke i to na sledeći način:

```
APP_ENV=local  
APP_DEBUG=true  
APP_KEY=52gx7C8Pc9h5v69ekf3BBFeXyFg6WHyH  
  
DB_HOST=127.0.0.1  
DB_DATABASE=ime_baze_podataka  
DB_USERNAME=korisničko_ime  
DB_PASSWORD=korisnička_šifra  
  
CACHE_DRIVER=file  
SESSION_DRIVER=file  
QUEUE_DRIVER=sync  
  
MAIL_DRIVER=smtp  
MAIL_HOST=mailtrap.io  
MAIL_PORT=2525  
MAIL_USERNAME=null  
MAIL_PASSWORD=null
```

Sada je naš projekat povezan sa našom bazom podataka u koju možemo ubacivati tabele i kolone koje su nam potrebne. Na *Slici 4* se nalazi izgled *PhpMyAdmina* koji se odnosi na našu aplikaciju.



Otvaramo konzolu i prelazimo u datoteku *laravel*.

Prvo ćemo u bazi napraviti tabelu *migrations*. U tabeli koja sadrži migracije će se nalaziti sve migracije koje su izvršene nad našom bazom podataka i to onim redosledom kojim su izvršavane. To ćemo uraditi pomoću sledeće komande:

```
php artisan migrate:install
```

Sada imamo tabelu *migracija*.

Migracije su kontrolne strukture za baze podataka koje nam omogućavaju lako i jednostavno kontrolisanje baze za našu aplikaciju. Laravel pruža jednostavan način definisanja i modifikovanja struktura tabela baze podataka koristeći PHP kôd. Umesto da kopiramo ručno deo po deo baze podataka na našu lokalnu mašinu, možemo samo izvršiti komandu migracije naše baze podatka u Artisan komandnoj liniji i naša baza će doslovce biti preneti. Artisan može da se koristi za generisanje različitih klasa. Na taj način ćemo uštedeti mnogo vremena, kao i smanjiti količinu kucanja pri izradi projekta.

Da bismo kreirali tabelu u bazi, koristićemo sledeću komandu:

```
php artisan make:migration create_ime_tabele_table --  
create=ime_tabele
```

```
php artisan migrate
```

Sada u dokumentu, koji se nalazi u datoteci *laravel/database/migration* pod nazivom *datum\_vreme\_create\_ime\_tabele\_table*, koristimo *mysql* komande za upravljanje tabelom. Kreiramo potrebne kolone. Migracije automatski dodeljuju svakoj svojoj komandi datum i vreme da bi Laravel mogao da odredi redosled migracija.

Kolone možemo dodati i naknadno na sledeći način:

```
php artisan make:migration  
add_ime_kolone_to_ime_tabele_table -table=ime_tabele
```

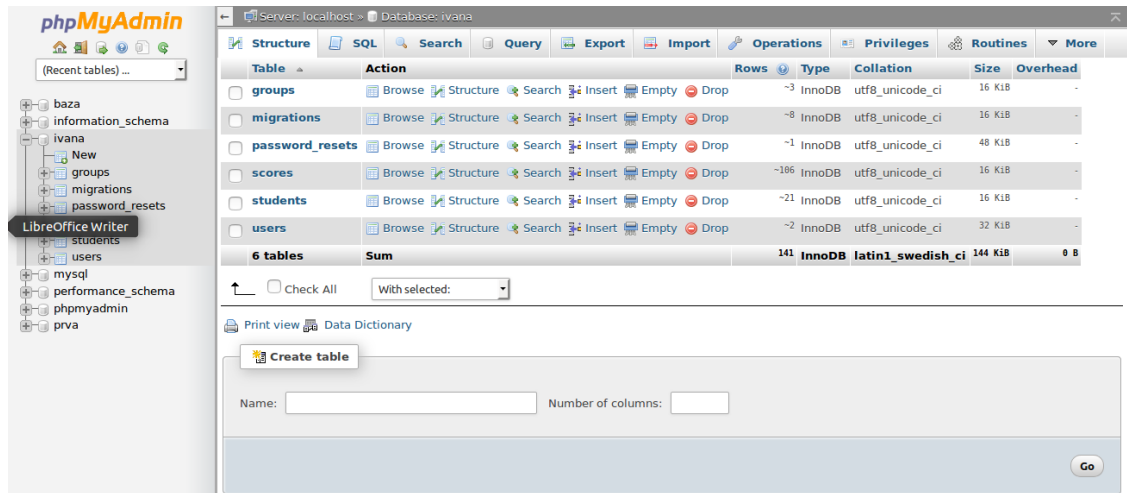
```
php artisan migrate
```

Kolone se takođe mogu modifikovati. U *Tabeli 1* se nalaze komande za to.

Komande za modifikovanje kolona	
->first()	Postavi kolonu na prvo mesto u tabeli (MySQL)
->after('column')	Postavi kolonu posle neke druge kolone (MySQL)
->nullable()	Dozvoli da NULL vrednosti budu u koloni
->default(\$value)	Postavi difoltnu vrednost kolone
>unsigned()	Postavi integer kolone u UNSIGNED

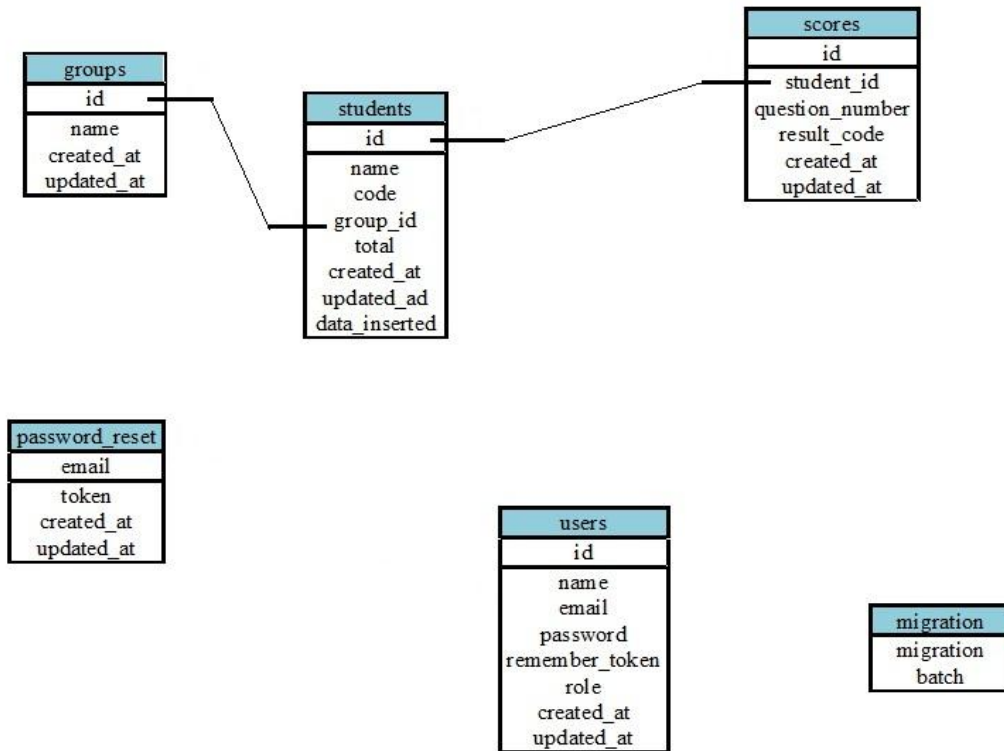
*Tabela 1. – Komande za modifikovanje kolona u tabeli baze podataka*

Izgled phpMyAdmina:



Slika 4. – Izgled phpMyAdmina

Baza podataka naše aplikacije i relacije među tabelama izgledaju ovako:



Slika 5. – Izgled baze podataka sa relacijama među tabelama

Zbog konvencije kod *elokvent orm* (objektno relaciono mapiranje), nazivi modela su uvek u jednini, dok odgovarajuća tabela za dati model ima naziv u množini sa istim imenom. Ne mora biti eksplicitno deklarirano šta je primarni, a šta strani ključ, već to *elokvent orm* radi sam na osnovu konvencije.

### 6.3 Putanje (Route)

Posle konekcije sa bazom podatka, potrebno je napraviti rute (putanje). Putanje služe da uspostave komunikaciju između URL adrese i kontrolera ili određenih funkcija koje treba da se izvrše kada korisnik pristupi datoj stranici.

Putanje se nalaze u dokumentu *routes.php* koji se nalazi u datoteci *app/Http/* i koji mora biti uključen u svaki Laravel projekat. Svaka ruta različitom metodom bira određenu URL adresu. Različitim metodama sa istom putanjom mogu se dobiti različiti rezultati, odnosno, može se doći do različitih kontrolera u zavisnosti od metode. Veb pretraživač prepoznaje samo *get* i *post* metode i Laravel ih implementira tako što u formi skrivenog polja zadaje odgovarajuću metodu u formi *imemetode\_metoda*. On prepoznaje različite rute na osnovu *http* metode.

Ovde su neke *http* metode:

GET metoda (Get metode služe za prikupljanje podataka):

```
Route::get('/', function () {  
    return 'Hello World';  
});
```

POST metoda (Post metoda služi za kreiranje novih podataka):

```
Route::post('foo/bar', function () {  
    return 'Hello World';  
});
```

PUT metoda (Put metoda služi za ažuriranje postojećih podataka):

```
Route::put('foo/bar', function () {  
    //  
});
```

DELETE metoda (Delete metoda služi za brisanje):

```
Route::delete('foo/bar', function () {  
    //  
});
```

Metoda s parametrima:

```
Route::get('user/{id}', function($id = null)  
{  
    return 'User '.$id;  
});
```

Metoda s ograničenjima:

```
Route::get('user/{id}/{name}', function($id, $name)  
{  
    //
```

```
})  
->where(array('id' => '[0-9]+', 'name' => '[a-z]+'));
```

Ukoliko naša verzija Laravela sadrži *RouteServiceProvider*, koji uključuje podrazumevane rute, ne moramo ih ručno dodavati u naš *routes.php* dokument.

U našoj aplikaciji, koja služi za obradu podataka o završnom ispitu iz matematike na kraju osnovne škole, bilo je potrebno šest putanja i to:

Osnovna putanja koja povezuje početnu stranu sa kontrolerom *HomeController*:

Ovo je ruta koja služi za prikaz početne strane, njen izgled se nalazi u dokumentu *home.blade.php*

```
Route::get('/', array(  
    'as' => 'home',  
    'uses' => 'HomeController@index'  
));
```

Putanja koja povezuje početnu stranu sa kontrolerom *StudentController* (metodom *GET* vršimo prikupljenje podataka):

```
Route::get('/home/{page?}', array(  
    'as' => 'student-list',  
    'uses' => 'StudentController@index'  
));
```

Putanja koja povezuje početnu stranu sa kontrolerom *StudentController* (metodom *POST* vršimo prikaz podataka):

```
Route::post('/student/find', array(  
    'as' => 'student-find',  
    'uses' => 'StudentController@find'  
));
```

Putanja koja povezuje početnu stranu sa kontrolerom *StudentController* (metodom *GET* ovde vršimo prikupljenje i izmenu podataka):

```
Route::get('/student/edit/{code}', array(  
    'as' => 'student-edit',  
    'uses' => 'StudentController@edit'  
));
```

Putanja koja povezuje početnu stranu sa kontrolerom *StudentController* (metodom *POST* ovde vršimo prikaz izmenjenih podataka):

```
Route::post('/student/save', array(  
    'as' => 'student-update',
```

### *Korišćenje radnog okruženja Laravel za razvoj veb aplikacija i obradu podataka*

```
'uses' => 'StudentController@update'  
));
```

Putanja kojom se kontroliše autorizacija i autentifikacija korisnika (u aplikaciji korišćena samo autentifikacija):

```
Route::controllers([  
    'auth' => 'Auth\AuthController',  
    'password' => 'Auth>PasswordController',  
    'stats' => 'StatsController'  
]);
```

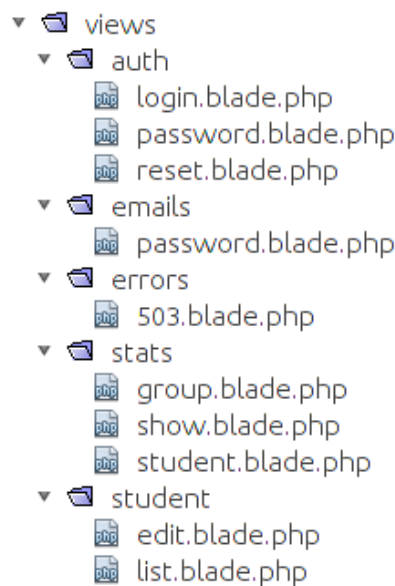
Treba napomenuti da se kod putanje za autorizaciju i autentifikaciju korisnika deklarišu kontroleri i tako deklarirani ne funkcionišu kao putanja. Laravel standardno dolazi sa sistemom za autorizaciju i autentifikaciju korisnika.

## 7 Pogledi i šabloni (izgled strana aplikacije)

Posle uspostavljanja putanja i povezivanja kontrolera sa URL adresama, moramo napraviti poglede. To znači da je potrebno napraviti izgled strana onako kako to treba da vide korisnici.

Sve strane aplikacije, koje su dostupne samom korisniku, imaju sličan izgled. Ova aplikacija ima na svakoj svojoj strani navigacioni bar na samom vrhu strane. Laravel daje mogućnost da lako podelimo zajedničke karakteristike svake strane pomoću Blade šablona. Kao što samo pomenuli ranije, svi pogledi (HTML šabloni) u Laravelu su smešteni u direktorijumu *resources/views*. Možemo koristiti jednostavne PHP šablone u Laravelu. Međutim, Blade sadrži praktične prečice za pisanje jasnog i kratkog koda.

U ovim pogledima su prikazani izgledi forme koji se koriste za čuvanje, resetovanje i operacije sa šiframa, a sve ostalo se nalazi (logika) u kontrolerima i modelima. Za izradu ove aplikacije bili su potrebni pogledi koji se nalaze na *Slici 6*.



Slika 6. – Spisak pogleda korišćenih u aplikaciji

Sada ćemo malo detaljnije proći kroz svaki od pogleda.

U dokumentu *app.blade.php* je definisan osnovni izgled strana aplikacije korišćenjem Blade šablona. Sve ostale strane će nekim delom naslediti ovaj pogled (šablon). Sledi kôd “roditelja” šablona, koji će naslediti ostali pogledi:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>Laravel</title>

<link href="{{ asset('/css/app.css') }}" rel="stylesheet">

<!-- Fonts -->
<link href="//fonts.googleapis.com/css?family=Roboto:400,300"
rel='stylesheet' type='text/css'>
```

## Korišćenje radnog okruženja Laravel za razvoj veb aplikacija i obradu podataka

```
    <!-- HTML5 shim and Respond.js for IE8 support of HTML5 elements and
media queries -->
    <!-- WARNING: Respond.js doesn't work if you view the page via file:// --
>

    <!--[if lt IE 9]>
    <script
src="https://oss.maxcdn.com/html5shiv/3.7.2/html5shiv.min.js"></script>
    <script
src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script>
    <![endif]-->
</head>
<body>
<nav class="navbar navbar-default">
<div class="container-fluid">
<div class="navbar-header">
<button type="button" class="navbar-toggle collapsed" data-
toggle="collapse" data-target="#bs-example-navbar-collapse-1">
<span class="sr-only">Toggle Navigation</span>
<span class="icon-bar"></span>
<span class="icon-bar"></span>
<span class="icon-bar"></span>
</button>
<a class="navbar-brand" href="/">Početna</a>
</div>

<div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
<ul class="nav navbar-nav">
<li><a href="{{ url('/stats/questions') }}">Statistika zadatka</a></li>
<li><a
href="{{ url('/stats/students') }}">Statistika učenika</a></li>
<li><a
href="{{ url('/stats/class') }}">Statistika odeljenja</a></li>
<li><a
href="{{ url('/home') }}">Lista učenika</a></li>
<li><a
href="{{ url('/admin') }}">Administrator</a></li>
</ul>

<ul class="nav navbar-nav navbar-right">
<li><a href="{{ url('/auth/login') }}">Login</a></li>
<li class="dropdown">
<a href="#" class="dropdown-toggle" data-toggle="dropdown" role="button"
aria-expanded="false">{{ Auth::user()->name }} <span class="caret"></span></a>
<ul class="dropdown-menu" role="menu">
<li><a href="{{ url('/auth/logout') }}">Logout</a></li>
</ul>
</li>
</ul>
```

```
</div>
</div>
</nav>

@yield('content')

<!-- Scripts -->
<script
src="//cdnjs.cloudflare.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
<script src="//cdnjs.cloudflare.com/ajax/libs/twitter-
bootstrap/3.3.1/js/bootstrap.min.js"></script>

@yield('scripts')
</body>
</html>
```

Definisali smo osnovni šablon aplikacije.

Pomoću rute koja se nalazi u direktorijumu *app/Http/routes.php* :

```
Route::get('/', function () {
    return view('tasks');
});
```

dobijamo izgled našeg pogleda.

Sada možemo definisati “decu” poglede, odnosno poglede koji nasleđuju osnovni pogled *app.blade.php*. Pomoću komandi Blejda možemo iskoristiti delove osnovnog pogleda koje želimo.

Naredbom *@extends* obaveštavamo Blejda da koristimo šablon koji je definisan u *app.blade.php*. Čitav sadržaj između naredbi *@section('content')* i *@endsection* biće ubačen na lokaciju koja je označena naredbom *@yield('content')*, a nalazi se u *app.blade.php*. Naredbom *@include('common.errors')* biće učitana šablon koji se nalazi u dokumentu *resource/views/common/errors.blade.php*.

```
<!-- Forma izgleda dela za proveru grešaka -->
@if (count($errors) > 0)
<div class="alert alert-danger">
<strong>Greška!</strong> Postoje problemi sa Vašim
podacima.<br><br>
<ul>
@foreach ($errors->all() as $error)
<li>{{ $error }}</li>
@endforeach
</ul>
</div>
@endif
```

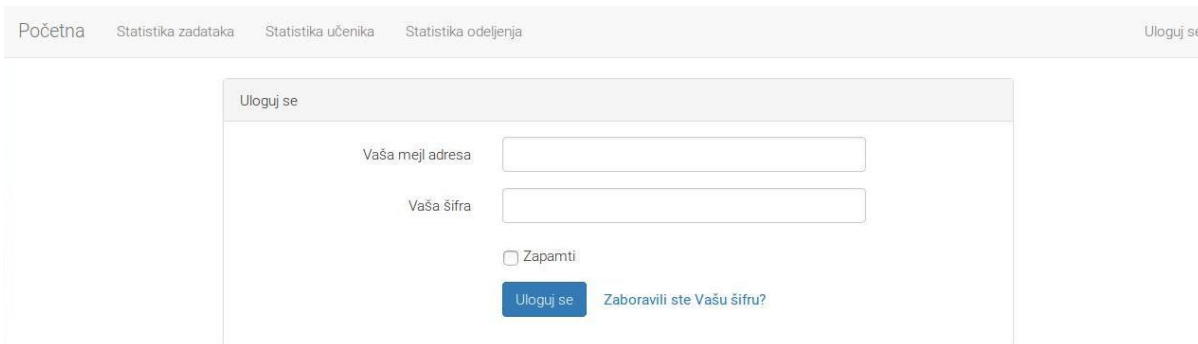
Promenljive *\$errors* se nalaze u svakom pogledu u Laravelu. Ukoliko ne dođe do greške, nema vraćanja vrednosti promenljive, odnosno vraća se vrednost *NULL*.

Kada smo napravili šablon osnovnog pogleda, dodajemo stranicu za logovanje korisnika



(administratora). Logovanje se vrši preko imejl adrese korisnika i šifre korisnika (šifra korisnika ne sme imati više od 255 karaktera, što smo mi ograničili). U datoteci `vendor/laravel/framework/src/Illuminate/Auth` se nalazi kôd paketa koji dolazi sa Laravelom, a koji smo koristili za autentifikaciju i autorizaciju (u aplikaciji je korišćen samo deo za autentifikaciju).

Na *Slici 7* se nalazi izgled strane na kojoj se korisnici loguju u administratorski deo aplikacije.



*Slika 7. – Izgled strane za logovanje korisnika*

Ukoliko dođe do greške pri logovanju, obavestićemo korisnika o tome. U dokumentu `views/auth/login.blade.php` se nalazi forma za logovanje korisnika, dok se u dokumentima `views/auth/password.blade.php` i `views/auth/reset.blade.php` (postupak resetovanja šifre) nalazi provera u vezi sa šifrom korisnika.

U direktorijumu `vendor/laravel/framework/src/Illuminate/Auth/Password` se nalaze dokumenta u kojima Laravel poseduje automatsku obradu grešaka pri logovanju.

Sada ćemo kratko proći kroz izgled i novine za svaku stranicu aplikacije.

### 1. Statistika zadataka / `show.blade.php`

Na slici ispod (*Slika 8.*) se nalazi izgled strane aplikacije na kojoj se nalazi grafički prikaz (dijagram) analize urađenih zadataka, koji predstavlja statistiku tačnosti urađenih zadataka.



*Slika 8. – Izgled strane sa statistikom zadataka*

Na dijagramu je za svaki zadatak prikazano koliko je učenika tačno uradilo taj zadatak. Za prikaz dijagrama korišćen je *jQuery*. *jQuery* je veoma moćna *JavaScript* biblioteka koja pojednostavljuje samu sintaksu *JavaScripta* i omogućava njegovu lakšu interakciju sa programskim

jezicima. Ovde je korišćeno gotovo rešenje. Kroz ovaj pogled prikazana je obrada podataka koji se nalaze u kontroleru *StatsController.php*. Obrada podataka i sam ispis su smešeni u kontroleru, dok se u pogledu samo poziva ispis i prikaz dijagrama. Sledi deo koda pomoću koga se vrši prikazivanje dijagrama:

```
@extends('app')

@section('content')

<div class="container">

    <h1>Statistika zadataka</h1>

    <p>{{ $stats }}</p>

    <div id="chartContainer" style="height: 300px; width: 100%;"></div>

</div>

@endsection

@section('scripts')

<script type="text/javascript"
src="{{ asset('js/jquery.canvasjs.min.js') }}"></script>

<script type="text/javascript">

window.onload = function () {

var options = {

title: {

text: "Statistika zadataka"

},

                    animationEnabled: true,

data: [

{

type: "column",

dataPoints: [

                                @foreach($rowData as $question => $score)

                                { x: {{$question}}, y:{{$score}} },

                                @endforeach

]

}

],

                    axisY:{

                        maximum: {{$totalStudents}},

                    },

};

jQuery("#chartContainer").CanvasJSChart(options);
```

```
}  
</script>  
@endsection
```

## 2. Statistika učenika / student.blade.php

Na *Slici 9* se nalazi izgled strane gde se ispisuju podaci o tome koliki je maksimalan broj poena koji je neko od učenika postigao, kao i koji učenik ima najveći broj poena.



*Slika 9. – Izgled strane sa statistikom učenika*

Ispisuju se ime i šifra učenika i njegov broj bodova. Podaci su prikupljeni i obrađeni u kontroleru *StatsController.php*, dok je u pogledu izvršeno njihovo ispisivanje i formatiranje. Sva poslovna logika mora biti u kontroleru koji poziva model. Sledi deo koda pomoću kog se prikazuje izgled strane:

```
@extends('app')  
@section('content')  
<div class="container">  
    <h1>Statistika učenika</h1>  
  
    <p>Najveći broj poena je {{$max}}.</p>  
    @foreach($students as $element)  
  
        <p>Učenik <strong>{{$element->name}</strong>  
            sa šifrom <strong>{{$element->code}</strong>  
            ima <strong>{{$element->total}</strong> bodova.</p>  
    @endforeach  
</div>  
@endsection
```

## 3. Statistika odeljenja / group.blade.php

Ovo je stranica gde se ispisuju podaci o tome koje je odeljenje najbolje uradilo test, kao i koji je prosečan broj bodova za to odeljenje. Ovaj pogled ispisuje podatke koji su obrađeni u kontroleru *StatsController.php*. Na *Slici 10* se nalazi izgled strane sa statistikom odeljenja.



## Statistika odeljenja

Odeljenja VIII 2 imaju najbolji prosečni rezultat. Prosečni rezultat je 19.

Slika 10. – Izgled strane sa statistikom odeljenja

Kao i kod prethodnog pogleda, ovde se izgled stranice ispisuje u samom pogledu uz pozivanje na podatke iz kontrolera. Sledi kôd koji to omogućava:

```
@extends('app')
@section('content')
<div class="container">
    <h1>Statistika odeljenja</h1>

    <p>{{ $result }}</p>
</div>
@endsection
```

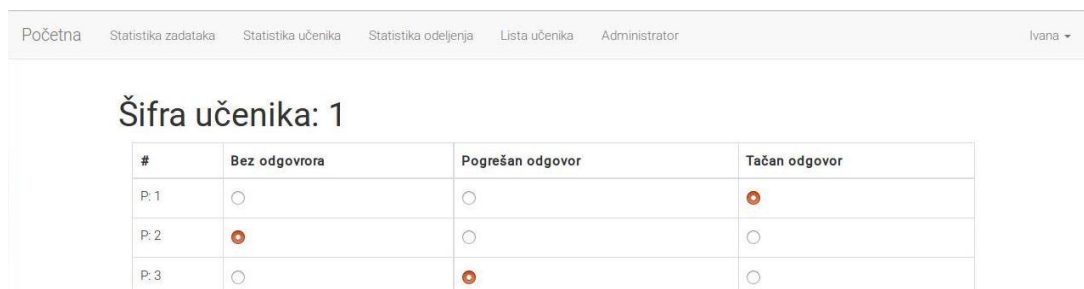
#### 4. Administratorske stranice / list.blade.php / edit.blade.php

Na *Slici 11* i *Slici 12* se nalaze izgledi strana koje se odnose na administraciju podataka o urađenim zadacima za datog učenika:



Slika 11. – Izgled strane na kojoj se nalazi spisak učenika

Na ovoj stranici se nalazi spisak učenika. Klikom na ikonicu *uredi*, ulazimo na stranicu preko koje unosimo u bazu podataka podatke o tačnosti urađenog zadatka.



Slika 12. – Izgled strane na kojoj se nalazi spisak zadataka - bodovanje

## 5. Administratorki deo / frozennode

*Frozennode* je paket koji ne dolazi sa Laravelom, već se naknadno može dodati (instalirati), napravljen je od strane neke treće osobe. Administrator je deo sajta koji je stvoren da olakša dodeljivanje korisnika i njegovih prava. Administrator je administrativni interfejs u Laravelu. Kao i Laravel, nalazi se na *GitHubu* i takođe se distribuira pod MIT licencom, što znači da ga slobodno možemo koristiti. Pomoću administratora vizuelno upravljamo elokventnim modelom i odnosima unutar njega. Takođe, omogućava samostalne postavke za skladištenje podataka i obavljanje zadataka koje je potrebno ispuniti.

Za svaki elokvent model možemo da definišemo koja polja administrativni korisnik može menjati, koje kolone iz tabele treba prikazati, podesiti dugmad za razne akcije, kao i filtere koje će moći da koristi. Ova polja mogu imati veze tipa “Pripada” i “PripadaMnogima”, ali ne i “ImaJedno” ili “ImaViše”. Na ovaj način lako se može odrediti u kakvoj su vezi podaci sa našeg sajta.

Instalacija se vrši pomoću Kompozera. U dokument *composer.json* dodajemo sledeći kôd:

```
"require": {
    "laravel/framework": "4.0.*",
    "frozennode/administrator": "dev-master"}
```

Zatim pokrenemo komandu *composer update*.

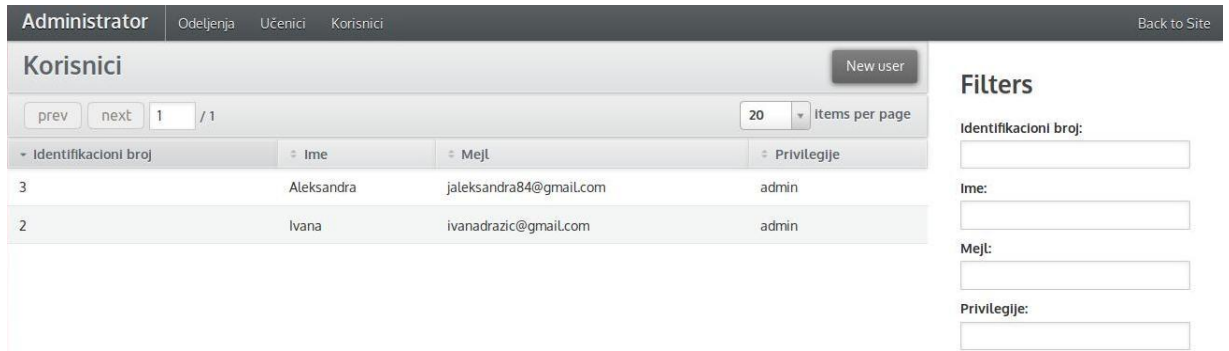
Na sledećim slikama se nalaze izgledi strana administratorskog dela.

Identifikacioni broj	Razred i odeljenje
6	VIII
3	VIII 3
2	VIII 2

Slika 13. – Izgled administratorskog dela za dodavanje novog odeljenja

Identifikacioni broj	Ime učenika	Šifra	Odeljenje
24	Milica	2356	VIII
23	Aleksandar	3001995	VIII 3
22	Mika	20	VIII 3
21	Zika	18	VIII 2
20	Pera	17	VIII 3
19	Ivana	16	VIII 2

Slika 14. – Izgled administratorskog dela za dodavanje novog učenika



Slika 15. – Izgled administratorskog dela za dodavanje novog korisnika

Kada smo instalirali administratorski deo, potrebno je napraviti modele administratora pomoću kojih ćemo ubaciti podatke o korisnicima, odeljenjima i učenicima (*users.php*, *group.php*, *students.php*). Sledi deo koda pomoću kojeg ćemo podesiti naše podatke za odeljenje u administratorskom delu:

```
<?php

/*Konfiguracija modela Odeljenje.*/
return array(
    'title' => 'Odeljenja',
    'single' => 'group',
    'model' => 'App\Group',
    /*Kolone koje se prikazuju. */
    'columns' => array(
        'id' => array (
            'title' => 'Identifikacioni broj'
        ),
        'name' => array(
            'title' => 'Razred i odeljenje',
        )
    ),
    /* Filteri na osnovu kojih je odeljenje određeno.*/
    'filters' => array(
        'Identifikacioni broj',
        'name' => array(
            'title' => 'Razred i odeljenje',
        )
    ),
    /*Polja koja se mogu menjati.*/
    'edit_fields' => array(
```

## ***Korišćenje radnog okruženja Laravel za razvoj veb aplikacija i obradu podataka***

```
'name' => array(  
'title' => 'Razred i odeljenje',  
'type' => 'text',  
)  
,  
);
```

Ovaj model u stvari nasleđuje osobine modela *Group.php*, koji se nalazi u datoteci *App*. Slično je i za preostala dva: *students.php* nasleđuje *Student.php*; *users.php* nasleđuje *User.php*.

## 8 Modeli

Ranije smo već rekli šta su modeli u MPK arhitekturi i čemu služe. Modeli koji su korišćeni u ovoj aplikaciji su: *Group.php*, *Score.php*, *Student.php*, *User.php*.

### 1. *User.php*

Ovaj model nam služi za korisnike administrativnog dela. On koristi podatke iz tabele *users*. *User.php* dolazi sa svakom instalacijom Laravela kao i *Auth.php* koji omogućava rad sa *userom* (korisnikom).

### 2. *Student.php*

Model *Student.php* koristi tabelu *student* iz naše baze podataka. Ovaj model nam omogućava da operišemo podacima o učenicima. U njega vršimo upis podataka o tačnim zadacima.

```
/* Ovom metodom dodeljujemo učenika odgovarajućem odeljenju ( jedan
učenik pripada jednom odeljenju )*/

public function group()
{
    return $this->belongsTo('App\Group');
}

/* Ovom metodom dodeljujemo zadatke odgovarajućem učeniku ( jedan
učenik može da reši više zadataka )*/

public function scores()
{
    return $this->hasMany('App\Score');
}

/* Metoda za upis podataka o zadacima. */

public function getTotalScore()
{
    if($this->scores->count()) {
        if($this->scores->count() < 20) {
            return 'Podaci nisu kompletni.';
        }
        return intval($this->total);
    }
    return 'Podatak nije unet.';
}
```

### 3. *Group.php*

Model *Group.php* je vezan za odeljenje učenika. Koristi tabelu *groups*. U ovom modelu se računa i prosečan broj bodova u odeljenju.

```
/* Ovim modelom dodeljujemo učenike odgovarajućem odeljenju ( jedno
odeljenje može imati više učenika )*/
```



## ***Korišćenje radnog okruženja Laravel za razvoj veb aplikacija i obradu podataka***

```
public function students()
{
    return $this->hasMany('App\Student');
}
/* Metoda koja računa prosečan broj bodova u datom odeljenju */
public function getAverage(){
    $count = $this->students->count();
    if($count != 0) {
        $sum = 0;
        foreach ($this->students as $student) {
            $sum += $student->total;
        }
        return $sum / $count;
    }
    return 0;
}
```

### ***4. Score.php***

Model koji smo nazvali *Score.php* nam koristi za upis podataka o tačnosti urađenih zadataka u bazu podataka. Ovaj model koristi tabelu *scores*.

```
<?php namespace App;
use Illuminate\Database\Eloquent\Model;
class Score extends Model {
    protected $table = 'scores';
    protected $fillable = ['student_id', 'question_number',
'result_code'];
    public static $rules = array(
        'student_id' => 'required',
        'question_number' => 'required',
        'result_code' => 'required',
    );
    public function student()
    {
        return $this->belongsTo('App\Student');
    }
}
```

## 9 Kontroleri

Kontroleri služe za komunikaciju između modela i pogleda. U njima se, takođe, može izvršiti neki deo obrade podataka. U ovoj aplikaciji su korišćeni sledeći kontroleri: *AuthController.php*, *PasswordController.php* (dolaze uz Laravel), *HomeController.php*, *StatsController.php*, *StudentController.php*, *WelcomeController.php*.

### 1. *AuthController.php*, *PasswordController.php*

Uz Laravel dolazi i sistem za autorizaciju i autentifikaciju korisnika. Oba ova kontrolera su sastavni deo istog sistema.

### 2. *WelcomeController.php*

Kontroler koji upućuje na prikaz početne strane.

```
/* Metoda kojom upućujemo na početnu stranu za korisnika */
public function index()
{
    return view('welcome');
}
```

### 3. *StudentController.php*

Ovaj kontroler obavlja komunikaciju između modela *Student.php*, *Score.php* i pogleda *list.blade.php* / *edit.blade.php*. Osim što obavlja komunikaciju, u njemu se nalaze i tri metode:

1. Metoda za indeksiranje i listanje strana na kojima se nalazi spisak učenika.

Ova metoda ograničava broj učenika na 10 po jednoj strani koja se prikazuje pomoću pogleda *list.blade.php*.

2. Metoda kojom proveravamo da li su uneti svi zadaci za datog studenta.

Ovo se vrši uz pomoć modela *Student.php*.

3. Metoda za upis i ispravku podataka o urađenim zadacima.

Ovde se, takođe, komunikacija vrši sa modelom *Student.php*. Omogućava se upis i ispravak zadataka. Pogled koji se koristi je *edit.blade.php*.

### 4. *StatsController.php*

U ovom kontroleru postoje metode koje se bave statističkom obradom i prikazivanjem podataka. Modeli koji su uključeni su *Score.php*, *Student.php* i *Group.php*.

Model *StatsController.php* sadrži metodu kojom se proverava i ispisuje na koje zadatke je odgovorio najveći broj učenika, kao i koliko je to. Pogled pomoću koga se ovo prikazuje je *show.blade.php*.

```
class StatsController extends Controller {
    public function getQuestions(){
        $data = array();
    }
}
```

## Korišćenje radnog okruženja Laravel za razvoj veb aplikacija i obradu podataka

```
for($i =1; $i <= 20; $i++) {
    $score = Score::where('question_number', $i)-
>where('result_code', '1')->count();
    $data[$score][] = $i;
    $rowData[$i] = $score;
}
$maxKey = $this->_maxKey($data);
if(count($data[$maxKey]) > 1) {
    $result = "Na zadatke sa rednim brojevima " . implode(',
', $data[$maxKey]) ." tačno je odgovorio najveći broj učenika. Ove zadatke tačno
je uradilo ".$maxKey." učenika.";
} else {
    $result = "Na zadatke sa rednim brojevima ".
$data[$maxKey][0]. " tačno je odgovorio najveći broj učenika. Ove zadatke tačno
je uradilo ". $maxKey ." učenika.";
}

return view('stats.show', array('rowData' => $rowData,
'stats' => $result, 'totalStudents' => Student::count()));
}
```

Metoda kojom se proveravaju i ispisuju statistički podaci odeljnja, to jest koje odeljenje je najbolje uradilo test i koji je prosečan broj bodova. Pogled pomoću koga se ovo prikazuje je *group.blade.php*.

```
public function getClass(){
    $groups = Group::with(array('students' => function($query){
        $query->where('data_inserted', 1);
    }->get());
    foreach ($groups as $key => $group) {
        $groupsAverage[$group->getAverage()][] = $key;
    }
    $maxKey = $this->_maxKey($groupsAverage);
    if(count($groupsAverage[$maxKey]) > 1) {
        $groupNames = array();
        foreach ($groupsAverage[$maxKey] as $id) {
            $groupNames[] = $groups[$id]->name;
        }
        $result = "Odeljenja " . implode(', ', $groupNames) ."
imaju najbolji prosečni rezultat. Prosečni rezultat je ".$maxKey.".";
    }
}
```

## ***Korišćenje radnog okruženja Laravel za razvoj veb aplikacija i obradu podataka***

```
    } else {  
        $result = "Odeljenja " .  
$groups[$groupsAverage[$maxKey][0]]->name ." imaju najbolji prosečni rezultat.  
Prosečni rezultat je ".$maxKey."";  
    }  
    return view('stats.group', array('result' => $result));  
}
```

### ***5. HomeController.php***

*HomeController.php* služi da se korisniku prikaže početna strana aplikacije. Ovaj kontroler možemo menjati ili ukloniti ukoliko to želimo. On nam je potreban da bi se naša aplikacija pokrenula i dolazi uz Laravel, kao podrazumevani.

## **10 Proširenje aplikacije**

Za aplikaciju koja je opisana u prethodnim poglavljima može se reći da predstavlja osnovne podatke za jednu školu gde smo pokazali korišćenje malog dela mogućnosti Laravela za izradu veb aplikacija. Da bi ova aplikacija bila komercijalno isplativa, moguće je uraditi razna proširenja.

Moguća proširenja aplikacije su:

- analiza različitih vrsta testiranja i velikih takmičenja,
- povezivanje više škola u jednoj opštini, pa onda na više opština u jednoj regiji, pa na više regija u državi,
- primena na dva ili više predmeta (ovde je urađeno samo za matematiku) koji se polažu kao predmeti na završnom ispitu,
- ispisivanje dodatnih podataka o učeniku, predmetu, broju bodova.

## **11 Zaključak**

Laravel se konstantno razvija i svakodnevno pruža niz mogućnosti za korišćenje pri izradi velikih i zahtevnih veb aplikacija. U odnosu na ostala okruženja programskog jezika PHP, Laravel ima elegantan, razumljiv i lak kôd. Svakodnevno se radi na otklanjanju nedostataka Laravela i veoma često (na svakih nekoliko meseci) pojavljuje se nova poboljšana verzija. Laravel poseduje veliku zajednicu, koja se stalno razvija i povećava. Dostupan je svima i ima dobru podršku. Povećavanjem broja korisnika Laravela, lakše se može učiti i vrlo lako saznati koje su njegove mogućnosti.

Laravel je besplatan i njegova najnovija verzija se uvek može preuzeti sa zvaničnog sajta [www.laravel.com](http://www.laravel.com). Na istom sajtu se nalaze sva potrebna uputstva kao i obimna dokumentacija. Korisne informacije se mogu pronaći na sajtu zajednice [www.github.com](http://www.github.com), kao i na [www.laracasts.com](http://www.laracasts.com).

## **12 Spisak slika i tabela**

Spisak slika:

- Slika 1. Slojevi MPK arhitekture
- Slika 2. Međudelovanje komponenata
- Slika 3. Lista komandi Artisan komandne linije
- Slika 4. Izgled phpMyAdmina
- Slika 5. Izgled baze podataka sa relacijama među tabelama
- Slika 6. Spisak pogleda korišćenih u aplikaciji
- Slika 7. Izgled strane za logovanje korisnika
- Slika 8. Izgled strane sa statistikom zadataka
- Slika 9. Izgled strane sa statistikom učenika
- Slika 10. Izgled strane sa statistikom odeljenja
- Slika 11. Izgled strane na kojoj se nalazi spisak učenika
- Slika 12. Izgled strane na kojoj se nalazi spisak zadataka - bodovanje
- Slika 13. Izgled administatorskog dela za dodavanje novog odeljenja
- Slika 14. Izgled administatorskog dela za dodavanje novog učenika
- Slika 15. Izgled administatorskog dela za dodavanje novog korisnika

Spisak tabela:

- Tabela 1. Komande za modifikovanje kolona u tabeli baze podataka

## Reference

- [1] “Laravel – The PHP Framework For Web Artisans”, [www.laravel.com](http://www.laravel.com), Taylor Otwell, posećeno:05.08.2015.
- [2] “How people build software GitHub”, [www.github.com](http://www.github.com), posećeno:16.11.2014.
- [3] “PHP: Releases”, <http://php.net/releases/>, posećeno:16.01.2015.
- [4] "History of Laravel PHP framework, Eloquence emerging".[www.maxoffsky.com](http://www.maxoffsky.com), Maks Surguy, posećeno:16.01.2015.
- [5] “Smalltalk-Vikipedija, slobodna enciklopedija”, <https://sr.wikipedia.org/sr/Smalltalk>, posećeno:12.06.2016.
- [6] “MVC arhitektura-Vikipedija, slobodna enciklopedija”, [https://sr.wikipedia.org/wiki/MVC\\_arhitektura](https://sr.wikipedia.org/wiki/MVC_arhitektura), posećeno:12.06.2016.
- [7] “Šta je i čemu služi MVC|Učim programiranje”, <http://www.ucim-programiranje.com/2013/02/mvc-model-view-controller/>, posećeno:12.06.2016.
- [8] “Kompilator-Vikipedija, slobodna enciklopedija”, <https://sr.wikipedia.org/wiki/Компилятор>, posećeno:12.06.2016.
- [9] “Interpretator(računarstvo)-Vikipedija, slobodna enciklopedija”, [https://sr.wikipedia.org/wiki/Интерпретатор\\_\(рачунарство\)](https://sr.wikipedia.org/wiki/Интерпретатор_(рачунарство)), posećeno:12.06.2016.
- [10] “PHP-Vikipedija, slobodna enciklopedija”, <https://sr.wikipedia.org/wiki/PHP>, posećeno: 12.06.2016.
- [11] “Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License”, <http://phpsrbija.github.io/php-the-right-way/>, Josh Lockhart, posećeno: 17.03.2016.
- [12] “Templates-Laravel–The PHP Framework For Web Artisans”, <https://laravel.com/docs/5.0/templates>, Taylor Otwell, posećeno:05.08.2015.
- [13] “Artisan-Laravel–The PHP Framework For Web Artisans”, <https://laravel.com/docs/5.0/artisan>, Taylor Otwell, posećeno:05.08.2015.
- [14] “Ubuntu 14.04 LTS (Trusty Tahr)”, <http://releases.ubuntu.com/14.04/>, posećeno: 12.05.2014.
- [15] “Get Ubuntu|Download|Ubuntu“, <http://www.ubuntu.com/download>, posećeno: 12.05.2104.
- [16] “Slobodni softver-Vikipedija, slobodna enciklopedija”, [https://sr.wikipedia.org/wiki/Слободни\\_софтвер](https://sr.wikipedia.org/wiki/Слободни_софтвер), posećeno: 18.06.2016.
- [17] “Otvoreni kod-Vikipedija, slobodna enciklopedija”, [https://sr.wikipedia.org/wiki/Отворени\\_код](https://sr.wikipedia.org/wiki/Отворени_код), posećeno: 18.06.2016.
- [18] “Composer”, [www.getcomposer.org](http://www.getcomposer.org), posećeno 03.03.2015.
- [19] “phpMyAdmin”, <https://www.phpmyadmin.net/>, posećeno 03.03.2015.